UNIVERZITET U NIŠU
PRIRODNO-MATEMATIČKI FAKULTET
DEPARTMAN ZA RAČUNARSKE NAUKE

Sead H. Mašović

ALGORITMI ZA TRIANGULACIJU POLIGONA I NJIHOVA IMPLEMENTACIJA U VEB OKRUŽENJU

- Doktorska disertacija -

Mentor:

Prof. dr Predrag S. Stanimirović

Niš, 2022

University of Niš
Faculty of Science and Mathematics
Department of Computer Science

Sead H. Mašović

ALGORITHMS FOR POLYGON TRIANGULATION AND THEIR IMPLEMENTATION IN A WEB ENVIRONMENT

- PhD thesis -

Niš, 2022

Imam posebnu čast i zadovoljstvo da se zahvalim mentoru prof. dr Predragu Stanimiroviću, redovnom profesoru Prirodno-matematičkog fakulteta u Nišu, na nesebičnoj pomoći i brojnim korisnim savetima, kojima je doprineo boljem kvalitetu ove disertacije, članovima komisije, kao i svim profesorima PMF-a, na korektnoj saradnji u toku doktorskih studija.

Veliku zahvalnost dugujem: prof. dr Predragu Krtolici, na iskustvu stečenom prilikom izrade naučnih radova i uspešnosti istih; prof. dr Muzaferu Saračeviću, na iskrenoj podršci, dobrim idejama i savetima; dr Islamu A. Elshaarawy, profesoru Tehničkog fakulteta Shoubra, Benha Univerziteta u Egiptu, na stručnim savetima koji se odnose na programiranje i implementaciju metoda.

I naravno, posebno se zahvaljujem članovima svoje porodice na nesebičnoj podršci, razumevanju i ohrabrenjima, koji su bili glavni razlog moje istrajnosti i uspeha.

Podaci o doktorskoj disertaciji

Mentor:	Prof. Dr Predrag S. Stanimirović, redovni profesor, Univerzitet u Nišu, Prirodno-matematički fakultet
Naslov:	Algoritmi za triangulaciju poligona i njihova implementacija u veb okruženju
Rezime:	Doktorska disertacija predstavlja nove algoritme za rešavanje problema triangulacije konveksnog poligona, u cilju unapređenja brzine pri generisanju triangulacija. Unapređenja, koja disertacija predstavlja, odnose se na konstrukciju i skladištenja triangulacija konveksnog poligona, koje se generišu u veb okruženju primenom baza podataka. Ideja je, da se jednom obrađeni rezultati koriste iščitavanjem iz baze podataka, kako bi se vreme obrade svelo na najmanje moguće. Polazeći od definicije problema, predmet istraživanja disertacije je primena veb tehnologija (PHP/MySQL) u funkciji povećanja efikasnosti pri generisanju triangulacija. U disertaciji je, takođe, predstavljen i novi pristup u rešavanju problema optimalnih triangulacija. Tema je multidisciplinarna jer dotiče oblasti računarske grafike, geometrije, internet tehnologija, veb programiranja i primenu troslojne arhitekture.
Naučna oblast:	Računarske nauke
Naučna disciplina:	Računarska grafika, Računarska geometrija, Veb programiranje i Baze podataka
Ključne reči:	Triangulacija poligona, Katalanov trougao, Troslojna arhitektura
UDK:	004.42:004.92
CERIF klasifikacija:	P 150 Geometrija, algebarska topologija P 170 Računarstvo, numerička analiza, sistemi, kontrola
Tip licence Kreativne zajednice:	**CC BY-NC-ND**

Data on Doctoral Dissertation

Doctoral Supervisor:	Predrag S. Stanimirović, Ph.D., full professor at Faculty of Sciences andMathematics, University of Niš
Title:	Algorithms for polygon triangulation and their implementation in a web environment
Abstract:	The doctoral dissertation presents new algorithms for solving the problem of convex polygon triangulation in order to improve the speed of triangulation generation. The improvements, which the dissertation presents, refer to the construction and storage of convex polygon triangulations, which are generated in a web environment using databases. The idea is that once processed results are used by reading them from the database, in order to reduce the processing time to the minimum possible. Starting from the definition of the problem, the subject of the dissertation research is the application of web technologies (PHP/MySQL) in the function of increasing efficiency when generating triangulations. The dissertation also presents a new approach to solving the problem of optimal triangulation. The topic is multidisciplinary because it touches the fields of computer graphics, geometry, internet technology, web programming and the application of three-layer architecture.
Scientific Field:	Computer science
Scientific Discipline:	Computer graphics, Computer geometry, Web programming and Databases
Key Words:	Polygon triangulation, Catalan's triangle, Three-layer architecture
UDC:	004.42:004.92
CERIF Classification:	P150 Geometry, algebraic topology P170 Computer science, numerical analysis, systems, control
Creative Commons License Type:	**CC BY-NC-ND**

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		монографска
Тип записа, **ТЗ**:		текстуални / графички
Врста рада, **ВР**:		докторска дисертација
Аутор, **АУ**:		**Сеад Х. Машовић**
Ментор, **МН**:		Предраг С. Станимировић
Наслов рада, **НР**:		**АЛГОРИТМИ ЗА ТРИАНГУЛАЦИЈУ ПОЛИГОНА И ЊИХОВА ИМПЛЕМЕНТАЦИЈА У ВЕБ ОКРУЖЕЊУ**
Језик публикације, **ЈП**:		српски
Језик извода, **ЈИ**:		енглески
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		Србија
Година, **ГО**:		2022.
Издавач, **ИЗ**:		ауторски репринт
Место и адреса, **МА**:		Ниш, Вишеградска 33.
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		5 поглавља / 89 страна / 49 референци / 24 табеле / 25 слика / 1 прилог
Научна област, **НО**:		Рачунарске науке
Научна дисциплина, **НД**:		Рачунарска графика, Рачунарска геометрија, Веб програмирање и Базе података
Предметна одредница/Кључне речи, **ПО**:		Триангулација полигона, Каталанов троугао, Трослојна архитектура
УДК		004.42:004.92
Чува се, **ЧУ**:		библиотека
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Докторска дисертација представља нове алгоритме за решавање проблема триангулације конвексног полигона, у циљу унапређења брзине при генерисању триангулација. Унапређења, која дисертација представља, односе се на конструкцију и складиштења триангулација конвексног полигона, које се генеришу у веб окружењу применом база података. Идеја је, да се једном обрађени резултати користе ишчитавањем из базе података, како би се време обраде свело на најмање могуће. Полазећи од дефиниције проблема, предмет истраживања дисертације је примена веб технологија (PHP/MySQL) у функцији повећања ефикасности при генерисању триангулација. У дисертацији је, такође, представљен и нови приступ у решавању проблема оптималних триангулација. Тема је мултидисциплинарна јер дотиче области рачунарске графике, геометрије, интернет технологија, веб програмирања и примену трослојне архитектуре.
Датум прихватања теме, **ДП**:		8.6.2020.
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	
	Члан:	
	Члан, ментор:	

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		monograph
Type of record, **TR**:		textual / graphic
Contents code, **CC**:		doctoral dissertation
Author, **AU**:		**Sead H. Mašović**
Mentor, **MN**:		Predrag S. Stanimirović
Title, **TI**:		**ALGORITHMS FOR POLYGON TRIANGULATION AND THEIR IMPLEMENTATION IN A WEB ENVIRONMENT**
Language of text, **LT**:		Serbian
Language of abstract, **LA**:		English
Country of publication, **CP**:		Serbia
Locality of publication, **LP**:		Serbia
Publication year, **PY**:		2022.
Publisher, **PB**:		author's reprint
Publication place, **PP**:		Niš, Višegradska 33.
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		5 chapters / 89 pages / 49 references / 24 tables / 25 pictures / 1 apendix
Scientific field, **SF**:		Computer science
Scientific discipline, **SD**:		Computer graphics, Computer geometry, Web programming and Databases
Subject/Key words, **S/KW**:		Polygon triangulation, Catalan's triangle, Three-layer architecture
UC		004.42:004.92
Holding data, **HD**:		library
Note, **N**:		
Abstract, **AB**:		The doctoral dissertation presents new algorithms for solving the problem of convex polygon triangulation in order to improve the speed of triangulation generation. The improvements, which the dissertation presents, refer to the construction and storage of convex polygon triangulations, which are generated in a web environment using databases. The idea is that once processed results are used by reading them from the database, in order to reduce the processing time to the minimum possible. Starting from the definition of the problem, the subject of the dissertation research is the application of web technologies (PHP/MySQL) in the function of increasing efficiency when generating triangulations. The dissertation also presents a new approach to solving the problem of optimal triangulation. The topic is multidisciplinary because it touches the fields of computer graphics, geometry, internet technology, web programming and the application of three-layer architecture.
Accepted by the Scientific Board on, **ASB**:		8.6.2020.
Defended on, **DE**:		
Defended Board, **DB**:	President:	
	Member:	
	Member, Mentor:	

Sadržaj

Predgovor

Doktorska disertacija je iz oblasti algoritama računarske geometrije u kombinaciji sa naprednim internet tehnologijama i korišćenja baza podataka. Na osnovu *MSC2010* klasifikacije, klasifikuje se u više oblasti *68U05: (computer graphics; computational geometry), 32B25: (triangulation and related questions), 68N15: (programming languages), 68P15: (database theory)*, kao i *68P20: (information storage and retrieval)*.

Ova disertacija će predstaviti nove algoritme za rešavanje problema triangulacije konveksnog poligona u cilju unapređenja brzine pri generisanju triangulacija. Unapređenja koja će disertacija predstaviti se odnose na konstrukciju i skladištenja triangulacija konveksnog poligona, koje se generišu u veb okruženju, optimizovanih po pitanju brzine i mogućnosti skladištenja. Ideja je da se jednom dobijeni rezultati koriste iščitavanjem iz baze podataka, kako bi se vreme obrade svelo na najmanje moguće. U disertaciji će takođe biti predstavljen i novi pristup u rešavanju optimalnih triangulacija.

Disertacija se sastoji od pet poglavlja:

1. **Prvo poglavlje** sadrži osnovne podatke o metodološkom okviru istraživanja (problemu, predmetu, ciljevima i tehnikama istraživanja). Zatim su date osnovne postavke Katalanovih brojeva, Katalanovog trougla i triangulacije konveksnih poligona. U nastavku je predstavljen i jedan postojeći algoritam za generisanje triangulacija poligona (*Hurtado-Noy*), koji se koristi za komparativnu analizu sa novim algoritmima, koji predstavljaju rezultat ove disertacije.

2. **U drugom poglavlju** su predstavljena dva nova algoritma za generisanje tringulacija konveksnog poligona. Oba algoritma se zasnivaju na korišćenju skupa triangulacija poligona P_{n-1} za generisanje triangulacija poligona P_n. Kod prvog algoritma ovo svojstvo proizilazi iz tehnike obilaska poligona P_n gde se generiše uređena lista (`OT-lista`), koja se koristi za generisanje triangulacija poligona P_n. Prvi algoritam naziva se *Orbiting triangle metoda - OTM metoda.* Glavna karakteristika OTM metode je postupak mapiranja triangulacija kroz dve operacije, komplementiranja i rotiranja, kao i upotrebu Katalanovog trougla za identifikaciju važećih triangulacija. Kod drugog algoritma svojstvo dobijanja triangulacija se ogleda u tome da poligon P_n može nastati iz poligona P_{n-1} dodavanjem temena n. Drugi algoritam naziva se *Modifikovana Blok metoda.* Modifikovana Blok metoda

teme n posmatra kroz dva slučaja: slučaj kada je n uvo poligona i slučaj kada je n teme unutrašnje dijagonale. Glavna karakteristika ove metode se ogleda u korišćenju vrednosti iz Katalanovog trougla kako bi se izbeglo pojavljivanje duplih triangulacija. Za oba algoritma urađena je komparativna analiza sa *Hurtado-Noy* algoritmom i Blok metodom, gde su predočene prednosti novih predloženih algoritama. Za potrebe dobijanja eksperimentalnih rezultata, odrađena je implementacija u *Java* programskom jeziku.

Objavljeni naučni rad autora disertacije na kojem se bazira drugo poglavlje je [40].

3. **Treće poglavlje** tretira problem pronalaženja optimalnih (minimalnih težinskih) triangulacija. U prvom delu poglavlja predstavljena je nova metoda za pronalaženje i skladištenje optimalne triangulacije. Metoda se zasniva na funkcionalnosti kvadratne matrice (SM), u kojoj se skladište svi rezultati triangulacija za dati n. Glavni akcenat metode se ogleda na brzini generisanja optimalne triangulacije i uštedi memorijskog prostora prilikom izračunavanja velikog broja triangulacija. U drugom delu poglavlja, predstavljen je postupak pronalaženja optimalne triangulacije koji se zasniva na Blok metodi za generisanje triangulacija. Analizirana su dva slučaja izračunavanja težina triangulacije (klasični slučaj i služaj zasnovan na Blok metodi). Značaj prikazanog postupka se ogleda u činjenici da korišćenje skladištenih vrednosti može biti veoma efikasan način za pronalaženje optimalne triangulacije. Osnovni cilj predloženog postupka je brzina dobijanja optimalne triangulacije. Za obe metode odrađena je implementacija u *Java* programskom jeziku i predstavljeni su eksperimentalni rezultati.

   Naučni radovi na kojima se bazira treće poglavlje su [29, 38].

4. **Četvrto poglavlje** prikazuje mogućnosti implementacije generisanja triangulacija u veb okruženju, optimizovanih po pitanju brzine i mogućnosti skladištenja. Ideja je da se jednom obrađeni rezultati koriste iščitavanjem iz baze podataka, kako bi se vreme obrade svelo na najmanje moguće. U prvom delu poglavlja predstavljena je implementacija Blok metode [33] za triangulaciju konveksnog poligona u PHP/MySQL okruženju. U drugom delu poglavlja predstavljene su prednosti uskladištenih procedura gde je proces generisanja triangulacija sa aplikativnog sloja spušten na sloj baze podataka u cilju dobijanja boljih rezultata. Na kraju poglavlja su prestavljeni eksperimentalni rezultati testiranja koji idu u korist polazne pretpostavke.

   Objavljeni naučni rad na kojem se bazira ovo poglavlje je [41].

5. U poslednjem poglavlju su data zaključna razmatranja i kratak pregled rezultata predloženih algoritama za generisanje triangulacija konveksnih poligona, mogućnosti njihove implementacije u veb okruženju, kao i budući pravci istraživanja.

Poglavlje 1

Uvodna razmatranja

Računarska geometrija je oblast računarske nauke koja se bavi proučavanjem algoritama i struktura podataka za izvođenje geometrijskih proračuna. Termin "*računarska geometrija*" prvi put se spominje u ranim sedamdesetim godinama prošlog veka. Od svog početka, računarska geometrija je privukla interesovanje velikog broja istraživača, zbog brojnih i jakih interakcija sa različitim oblastima nauke i inženjerstva, kao što su algoritmi i strukture podataka, kombinatorna matematika, euklidska geometrija i optimizacija [1, 2, 25].

Predmet istraživanja računarske geometrije je rešavanje geometrijskih zadataka uz pomoć računara, pri čemu dobijeni rezultati geometrijskih algoritama nalaze široku primenu u različitim oblastima. U središtu razvoja računarske geometrije, kao discipline, su problemi geometrije koji su bili prisutni u računarskoj grafici, računarskom projektovanju i proizvodnji (CAD/CAM). Računarska grafika obuhvata sve aspekte sinteze, prikazivanja i manipulacije slikovnim prikazima pomoću računarskih mašina, zajedno sa njihovim predstavljanjem vizuelnom sistemu čoveka.

Druge važne primene računarske geometrije mogu se naći u: robotici (planiranje kretanja i problemi sa vidljivošću); geografskim informacionim sistemima (geometrijske lokacije i pretraga, planiranje kretanja); dizajnu integrisanih kola (dizajn i verifikacija IC geometrije); računarsko inženjerstvo (CAE) (mrežna generacija); računarskoj viziji (3D rekonstrukcija) itd.

Računarska složenost je presudna za računarsku geometriju i od velikog je praktičnog značaja ako se algoritmi koriste na veoma velikim skupovima podataka koji sadrže desetine ili stotine miliona tačaka. Za takve skupove, razlika između $O(n^2)$ i $(n \log n)$ može biti razlika između dana i sekundi računanja. Cilj svih istraživanja u računarskoj geometriji je da se minimizira vrednost ove razlike.

Jedan od važnijih problema računarske geometrije je triangulacija poligona. U računarskoj geometriji, triangulacija poligona predstavlja dekompoziciju poligonalne površine (prostog poligona) P na skup trouglova, tj. pronalaženje skupa trouglova međusobno nepresecajućim unutrašnjim dijagonalama čija je unija P [18, 19, 30].

Jedan od osnovnih izazova u formiranju trodimenzionalnih prikaza objekata iz skupa tačaka jeste brzina pronalaženja triangulacija poligona, jer se sa povećanjem broja temena

poligona drastično povećava i broj različitih triangulacija.

Predmet istraživanja ove disertacije predstavlja analizu i testiranje algoritama za rešavanje problema triangulacije konveksnog poligona, prvenstveno sa aspekta generisanja i skladištenja triangulacija (korišćenjem baza podataka).

Ova disertacija će predstaviti nove algoritme za rešavanje problema triangulacije poligona u cilju unapređenja brzine pri generisanju triangulacija. O važnosti ove oblasti i aktuelnosti teme koja će biti obrađena u ovoj disertaciji govori veliki broj radova koji su objavljeni u naučnim i stručnim časopisima. U disertaciji su predstavljeni novi algoritmi koji se mogu primenjivati u ovoj oblasti kao i prednosti njihove implementacije u veb okruženje primenom baza podataka. U disertaciji će takođe biti predstavljen i novi pristup u rešavanju optimalnih triangulacija.

Disertacija će prikazati mogućnosti implementacije generisanja triangulacija u veb okruženju, optimizovanih po pitanju brzine i mogućnosti skladištenja. Ideja je da se jednom obrađeni rezultati koriste iščitavanjem iz baze podataka, kako bi se vreme obrade svelo na najmanje moguće. Polazeći od definicije problema, predmet istraživanja disertacije je primena veb tehnologija (PHP/MySQL) u funkciji povećanja efikasnosti pri generisanju triangulacija. Tema je multidisciplinarna jer dotiče oblasti računarske grafike, geometrije, internet tehnologije, veb programiranja i primenu troslojne arhitekture.

Napredna razvojna okruženja i tehnike, koje se primenjuju u praktičnom delu disertacije (PHP/MySQL), omogućavaju implementaciju algoritama u cilju dobijanja eksperimentalnih rezultata. PHP i MySQL su veoma popularne tehnologije koje su otvorenog koda, savršene za brz i efikasan razvoj veb aplikacija koje su prilagođene za rad sa bazama podataka. Istraživanje u disertaciji ima empirijski i eksperimentalni karakter.

Na osnovu postavljenih ciljeva, definisani su sledeći zadaci istraživanja:

1. Napraviti komparativnu analizu datih novih algoritama sa postojećim algoritmima za generisanje triangulacija, i predstaviti dobijene rezultate. Takođe, utvrditi procenat uštede memorije u primenih novih metoda u odnosu na postjeće kao i prednosti korišćenja baza podataka za skladištenje rezultata generisanja triangulacija i njihovog ponovnog korišćenja.
2. Ispitati efekat primene PHP/MySQL razvojnog okruženja na brzini generisanja triangulacija poligona, odnosno na primeni klasične troslojne veb arihtekture i korišćenju naprednih tehnika PHP/MySQL-a.
3. Uporediti vremena dobijanja rezultata generisanja triangulacija poligona, išitavanjem rezultata iz baze podataka i novog pozivanja algoritma za generisanje triangulacija.

Na osnovu predmeta istraživanja, postavljenog cilja i zadataka istraživanja, potrebno je utvrditi da li optimalnim korišćenjem i alociranjem hardverskih resursa klijent-server aplikacije, bazirane na MySQL sistemu za skladištenje, omogućava uštedu CPU vremena pri generisanju triangulacija.

Da bi se realizovali ciljevi i zadaci istraživanja, korišćene su sledeće naučne metode i postupci:

- Metodom teorijske analize su ispitivana dosadašnja teorijska znanja o problemu triangulisanja poligona i analizirane su postojeće metode i algoritmi.
- Metdom komparacije izvršeno je poređenje rezultata implementacije postojećih algoritama i novih predloženih algoritama za rešavanje problema.
- Eksperimentalnom primenom razvijenih aplikacija utvrdili smo efikasnost novih algoritama u brzini generisanja triangulacija, ili uštedi memorijskog prostora pri korišćenju baza podataka za skladištenje.
- Metodom indukcije, u toku eksperimentalnog ispitivanja, formiran je zaključak o implementiranim algoritmima i metodama i njihovoj primeni u oblasti računarske geometrije.
- Metoda generalizacije se koristi u analizi određenog broja slučajeva, a na osnovu matematičkih metoda, gde se dolazi do uopštene tvrdnje koja važi za sve slučajeve.
- Metod specijalizacije, u obliku konkretnog primera koji prati korake algoritma (metoda), biće primenjen u predstavljanju određenog slučaja.
- Metodom apstrakcije i konkretizacije, ispitivane su osobine objekata, pri čemu su izostavljene one koje nisu relevantne.

Na osnovu neposredne implementacije stečenih saznanja, može se govoriti o stepenu naučnog otkrića i naučnog objašnjenja rezultata istraživanja, čime se direktno ostvaruje i društveni značaj rada, a time i kompatibilnost sa svetskim standardima i trendovima u oblastima koje predstavljaju predmet istraživanja, a sve u cilju uštede CPU vremena pri generisanju triangulacija poligona. Naučni doprinos disertacije se ogleda u rezultatima istraživanja publikovanim u međunarodnim naučnim časopisima [28, 34, 39, 40, 41].

Doktorska disertacija donosi novine u odnosu na postojeće stanje i otvara prostor za dalja istraživanja praktične primene složenih algoritama iz računarske geometrije u veb okruženju i korišćenju baza podataka za skladištenje dobijenihh rezultata.

Rezultati disertacije se mogu svrstati u tri kategorije. Prvu kategoriju čine rezultati, koji se odnose na nove algoritme, koji se primenjuju u generisanju triangulacija konveksnog poligona. Drugu kategoriju čine rezultati, koji se odnose na nove metode u pronalaženju optimalnih triangulacija. Treću kategoriju čine rezultati, koji se odnose na primenu PHP/MySQL okruženja za generisanje triangulacija konveksnih poligona.

1.1 Pojam Katalanovih brojeva

Katalanovi brojevi (C_n) predstavljaju niz prirodnih brojeva koji se pojavljuju kao rešenje velikog broja kombinatornih problema. Ime su dobili po belgijskom matematičaru Ežen Šarl Katalanu (*Eugène Charles Catalan*, 1814–1894), koji je izveo i dokazao mnoga svojstva i identitete ovih brojeva. Na Katalanove brojeve prvi je naišao Leonard Ojler (*Leonhard Euler*, 1707-1783) tražeći opšte rešenje za broj različitih načina deljenja mnogougla na trouglove, vodeći računa da se ne računaju dijagonale mnogougla koje se međusobno seku.

Ono što je zanimljivo je da je ove brojeve potpuno nezavisno otkrio i kineski matematičar Ming Antu (*Ming'antu*, 1692-1763), 1730-ih godina [16], ali njegovi radovi su zapadnom svetu postali poznati tek posle 1839. godine. 1730. godine on je napisao knjigu "Quick Methods for Accurate Values of Circle Segments" koja je uključivala brojne trigonometrijske identitete i nizove stepena, od kojih su neki uključivali Katalanove brojeve(1.1):

$$\sin(2\alpha) = 2\sin\alpha - \sum_{n=4}^{\infty} \frac{C_{n-1}}{4^{n-1}} \sin^{2n+1}\alpha = 2\sin\alpha - \sin^3\alpha - \frac{1}{4}\sin^5\alpha - \frac{1}{8}\sin^7\alpha - \dots \quad (1.1)$$

Kratku istoriju Katalanovih brojeva, od njihovog prvog otkrića u 18. veku do modernog doba, predstavio je Igor Pak[1], koja se pojavljuje u knjizi [37].

Katalanovi brojevi [45] se izračunavaju po sledećoj formuli:

$$\begin{aligned} C_n &= \frac{(2n)!}{(n+1)!n!} \\ &= \frac{1}{n+1}\begin{pmatrix} 2n \\ n \end{pmatrix}, n \geq 0 \end{aligned} \quad (1.2)$$

gde je n broj trouglova na koliko se može podeliti dati poligon.

Drugi način za definisanje Katalanovih brojeva je:

$$\begin{pmatrix} 2n \\ n \end{pmatrix} - \begin{pmatrix} 2n \\ n+1 \end{pmatrix} = \frac{(2n)!}{(n!)^2} - \frac{(2n)!}{(n-1)!(n+1)!} = \frac{(2n)!}{n!(n+1)!} = C_n \quad (1.3)$$

Katalanovi brojevi se takođe mogu izraziti i Segnerovom rekurzivnom formulom [46], kao i Bertrandovom *ballot* teoremom [49]. Tabela 1.1 sadrži Katalanove brojeve za vrednosti $n \in \{1, 2, ..., 30\}$, koji se izračunavaju po formuli (1.2) ili (1.3).

[1] https://www.math.ucla.edu/ pak/lectures/Cat/pakcat.htm

Tabela 1.1: Neke vrednosti Katalanovih brojeva

n	C_n	n	C_n	n	C_n
1	1	**11**	58,786	**21**	24,466,267,020
2	2	**12**	208,012	**22**	91,482,563,640
3	5	**13**	742,900	**23**	343,059,613,650
4	14	**14**	2,674,440	**24**	1,289,904,147,324
5	42	**15**	9,694,845	**25**	4,861,946,401,452
6	132	**16**	35,357,670	**26**	18,367,353,072,152
7	429	**17**	129,644,790	**27**	69,533,550,916,004
8	1,430	**18**	477,638,700	**28**	263,747,951,750,360
9	4,862	**19**	1,767,263,190	**29**	1,002,242,216,651,360
10	16,796	**20**	6,564,120,420	**30**	3,814,986,502,092,300

Katalanovi brojevi $(C_n)_{n\in N_0}$ predstavljaju i rešenja rekurzivne relacije

$$C_n = C_0C_{n-1} + C_1C_{n-2} + \ldots + C_{n-1}C_0, \qquad C_0 = C_1 = 1.$$

Katalanovi brojevi nalaze svoju primenu kod mnogih kombinatornih problema, kao što su deljenje poligona na trouglove, problem uparenih zagrada, šetnja po celobrojnoj mreži, problemi puteva u mreži, problem binarnih stabala itd. Praktična primena Katalanovih brojeva u rešavanju određenih kombinatornih problema i problema u računarskoj geometriji može se naći u radovima [2, 3, 6, 8, 32, 44]. Ričard Stenli (Richard Stanley) je u svojoj knjizi [36] naveo preko 60 različitih interpretacija Katalanovih brojeva, a osnovni pojmovi i konkretna upotreba, i probleme koji se zasnivaju na Katalanovim brojevima možemo pronaći u [45].

1.2 Pojam Katalanovog trougla

Katalanov trougao je trougao brojeva koji za ulazne parametre $C(n, k)$ daje niz brojeva koji se sastoji od n vrednosti po X osi i k vrednosti po Y osi, tako da nijedan početni segment niza nema više vrednosti po Y nego po X osi. Katalanov trougao predstavlja generalizaciju Katalanovih brojeva. Katalanov trougao $C(n, k)$ zadovoljava sledeće osobine [7]:

1. $C(n, 0) = 1$ za $n \geq 0$
2. $C(n, 1) = n$ za $n \geq 1$
3. $C(n + 1, k) = C(n + 1, k - 1) + C(n, k)$ za $1 < k < n + 1$
4. $C(n + 1, n + 1) = C(n + 1, n)$ za $n \geq 1$

Osnovno pravilo koje se koristi za konstrukciju Katalanovog trougla dato je u sledećoj formuli:

$$C(n,k) = \frac{(n+k)!(n-k+1)}{k!(n+1)!}, \tag{1.4}$$

gde je n nenegativni ceo broj i $k = 0, \ldots, n$. Prema tome, za $k > 0$ važi

$$C(n,k) = \binom{n+k}{k} - \binom{n+k}{k+1}$$

Primera radi, 8×8 Katalanov trougao je predstavljen u Tabeli 1.2. Za više možete videti u radovima [7, 23].

Tabela 1.2: Neke vrednosti Katalanovog trougla.

n\k	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	1								
1	1	1							
2	1	2	2						
3	1	3	5	5					
4	1	4	9	14	14				
5	1	5	14	28	42	42			
6	1	6	20	48	90	132	132		
7	1	7	27	75	165	297	429	429	
8	1	8	35	110	275	572	1001	1430	1430

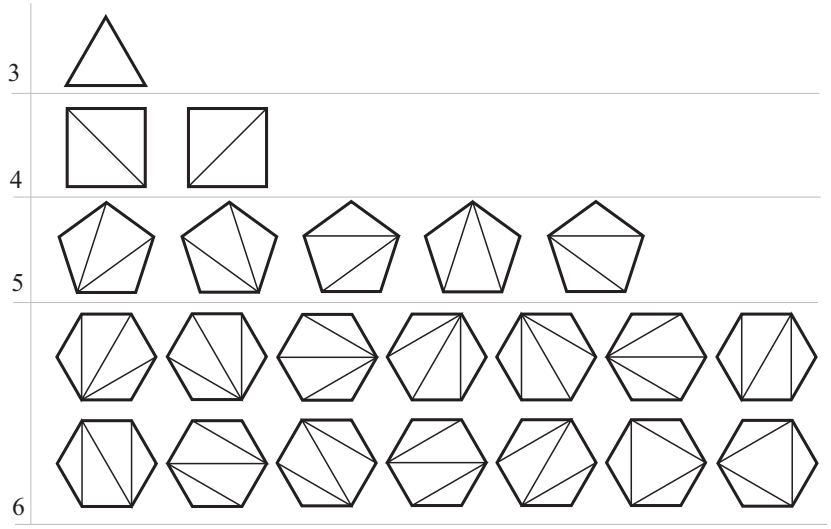
Kombinatorna interpretacija $(n, k-1)$–te vrednosti je broj neopadajućih particija sa tačno n delova sa maksimalnim delom k tako da je svaki deo manji ili jednak njegovom indeksu. Tako se, na primer, računa $(4, 2) = 9$.

1.3 Pojam triangulacije poligona

U računarskoj geometriji, triangulacija poligona predstavlja podelu poligonalne površine (jednostavnog poligona) P na skup trouglova [24]. U ovoj disertaciji je obrađen pojam triangulacije konveksnog poligona. Za konveksne poligone je karakteristično da su sve dijagonale unutrašnje.

Triangulacija konveksnog poligona podrazumeva dekompoziciju unutrašnjosti poligona na trouglove, međusobno nepresecajućim unutrašnjim dijagonalama. Kod ovog problema se zapravo razmatra broj triangulacija, gde je moguća maksimalna dekompozicija konveksnog poligona na $n-2$ trougla. U radovima [18, 19, 30] su predstavljeni neki načini rešavanja ovog problema.

Triangulacija poligona za skup $n \in \{3, ..., 6\}$, gde n predstavlja broj temena (Slika 1.1).



Slika 1.1: Triangulacija konveksnog poligona za $n \in \{3, ..., 6\}$

Konveksan poligon sa n temena označiti ćemo sa P_n. On je opisan nizom temena $v_1, v_2, ..., v_n$. Unutrašnja dijagonala koja povezuje temena v_i i v_j je označena sa $\delta_{p,q}$. Takođe i stranice poligona se smatraju dijagonalama pa tako $\delta_{i,i+1}$ predstavlja ivicu $v_i v_{i+1}$. Skup triangulacija poligona P_n je obeležen sa $\mathcal{T}_n$ a sa τ_n označićemo određenu triangulaciju iz skupa $\mathcal{T}_n$. Koristićemo oznaku $\deg(i)$ za stepen temena i u triangulaciji. Sa T_i ćemo označiti ukupan broj triangulacija u skupu $\mathcal{T}_n$.

U nastavku ćemo uspostaviti relaciju između triangulacija poligona i Katalanovih brojeva. Na osnovu Ojlerove postavke problema triangulacije poligona [45], važi sledeća jednakost:

$$T_n = C_{n-2}, n \geq 3 \tag{1.5}$$

gde je n broj temena poligona.

Problem triangulacije poligona počinje sa trouglom. Pošto je trougao već triangulisan, postoji samo jedan način triangulacije i stoga je $T_3 = C_{3-2} = C_1 = 1$. Za kvadrat ($n = 4$) možemo nacrtati jednu dijagonalu. Ovo se može uraditi na dva načina (jer kvadrat ima dve dijagonale) pa je $T_4 = C_{4-2} = C_2 = 2$. Broj triangulacija petougla (T_5), na osnovu Katalanovog broja iznosi $C_3 = 5$, za šestougao (T_6) važi $C_4 = 14$ itd. (videti Tabelu 1.1).

Prema tome, vrednost Katalanovog broja C_{n-2} određuje broj triangulacija koji odgovara poligonu P_n. Na osnovu formule za dobijanje vrednosti Katalanovih brojeva (1.2) možemo definisati vrednost T_n, za $n \geq 3$:

$$T_n = \frac{1}{n-1} \begin{pmatrix} 2n-4 \\ n-2 \end{pmatrix} = \frac{(2n-4)!}{(n-1)!(n-2)!} \tag{1.6}$$

Sledi da vrednost T_{n+2} možemo izraziti kao:

$$\begin{aligned} T_{n+2} &= \frac{2}{n+1}\begin{pmatrix} 2n-1 \\ n \end{pmatrix} \\ &= \frac{1}{n+1}\begin{pmatrix} 2n \\ n \end{pmatrix} = C_n \end{aligned} \tag{1.7}$$

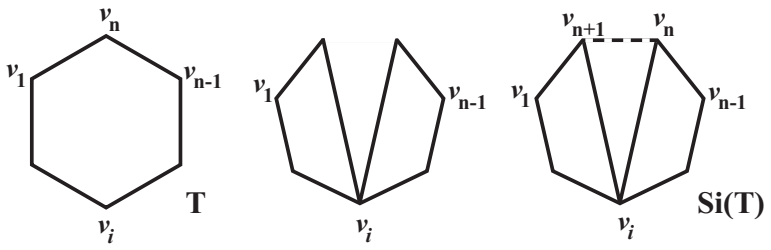
što je ekvivalentno sa (1.5).

Triangulacija konveksnih poligona je jedan od aktuelnih problema koji se pojavljuje u dvodimenzionalnoj računarskoj geometriji. Tehnika triangulacije u računarskoj geometriji predstavlja jednu od najčešće korišćenih metoda panelizacije. Aproksimacija glatke površine trouglastim elementima je najstariji i još uvek najpopularniji način panelizacije. Prednost trougla kao geometrijskog tela je što je njegova površina uvek ravna. Prednosti ovakve primene su: mala odstupanja od originalnog oblika, dobra strukturalna svojstva i mogućnost oblaganja složenih slobodnih formi. Triangulacija poligona zbog svoje karakterističnosti nalazi svoju primenu i u geografskim informacionim sistemima, kao i u procesu digitalnog modeliranja terena [31].

Triangulacija jednostavnog poligona je jedan od najistaknutijih otvorenih problema u dvodimenzionalnoj računarskoj geometriji. Ona predstavlja uopšteni primitiv u računarskoj grafici i, uopšteno gledano, čini se prirodnim korakom preprocesiranja za većinu netrivijalnih operacija na jednostavnim poligonima [17, 30]. Triangulacija omogućava da se iz skupa tačaka dobije trodimenzionalni prikaz objekata. Ona, takođe, omogućava mehanizam za tzv. *glačanje trodimenzionalnih figura*, što je jako važno za brzinu i kvalitet (rezoluciju) prikaza objekata. Pa tako, triangulacija poligona nalazi svoju primenu u modelovanju 3D objekata. U urbanizmu se koristi pri određivanju polažaja objekata. U godeziji za premer, određivanje geodetske mreže, modelovanju terena i izradi tačnih karata.

1.4 Jedan algoritam za triangulaciju konveksnih poligona

Algoritam autora Hurtada i Noja [11] (u nastavku *Hurtado-Noy* algoritam) se bazira na korišćenju skupa triangulacija poligona P_{n-1} za generisanje triangulacija poligona P_n. Autori su ustanovili stablo triangulacija sa tačno uređenom hijerarhijom. Stablo hijerarhije na n-tom nivou uređuje sve triangulacije poligona P_n iz skupa $\mathcal{T}_n$. Iz toga sledi da svaka triangulacija na n-tom nivou ima "roditelja" u skupu $\mathcal{T}_{n-1}$ i dva ili više "potomaka" u skupu $\mathcal{T}_{n+1}$. Potomci istog roditelja se tretiraju kao "braća". Postoji određeni redosled (uređenje) među potomcima jedne triangulacije, pa je na osnovu toga definisano uređenje koje tretira sve triangulacije u skupu $\mathcal{T}_n$.



Slika 1.2: Postupak "cepanja" dijagonale i konstrukcija potomka

Za generisanje poligona P_n, autori su predstavili postupak "cepanja" dijagonala poligona P_{n-1} (kako unutrašnjih dijagonala, tako i spoljašnjih ivica poligona), kao što je prikazano na Slici 1.2. Ako se izvrši cepanje dijagonala $\delta_{i,n-1}, i \in \{1, 2, \ldots, n-2\}$ u rastućem redosledu od i, dobićemo uređene triangulacije za P_n.

Označimo jednu triangulaciju $\tau_{n-1} \in \mathcal{T}_{n-1}$ koja na osnovu τ_{n-1}^l zadovoljava $\deg(n-1) = l$. Pretpostavimo da su sortirane dijagonale incidentne za $n-1$, počevši od $\delta_{1,n-1}$, po sistemu suprotno kretanju kazaljke na satu. Označimo k-tu dijagonalu u pomenutom redosledu sa $\delta_{i_k,n-1}$, gde k uzima vrednost iz skupa $k \in \{1, \ldots, l\}$. Prema tome, sledi da je broj incidentnih dijagonala za $n-1$ koje prethode $\delta_{i_k,n-1}$ jednak $k-1$.

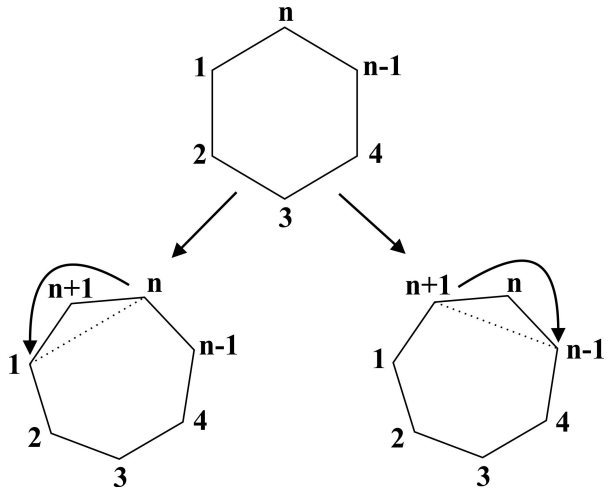
Potomci za τ_{n-1}^l izvode se tako što se "cepaju" incidentne dijagonale za teme $n-1$

$$\delta_{i_k,n-1},\ k = 1, \ldots, l,\ i_k \in \{1, \ldots, n-2\},\ 1 = i_1 < i_2 < \cdots < i_l = n-2.$$

Ako se vrši cepanje dijagonale $\delta_{i_k,n-1}$ onda dobijamo potomka $S^{i_k}(\tau_{n-1}^l)$. Tada sledi da su potomci triangulacije τ_{n-1}^l

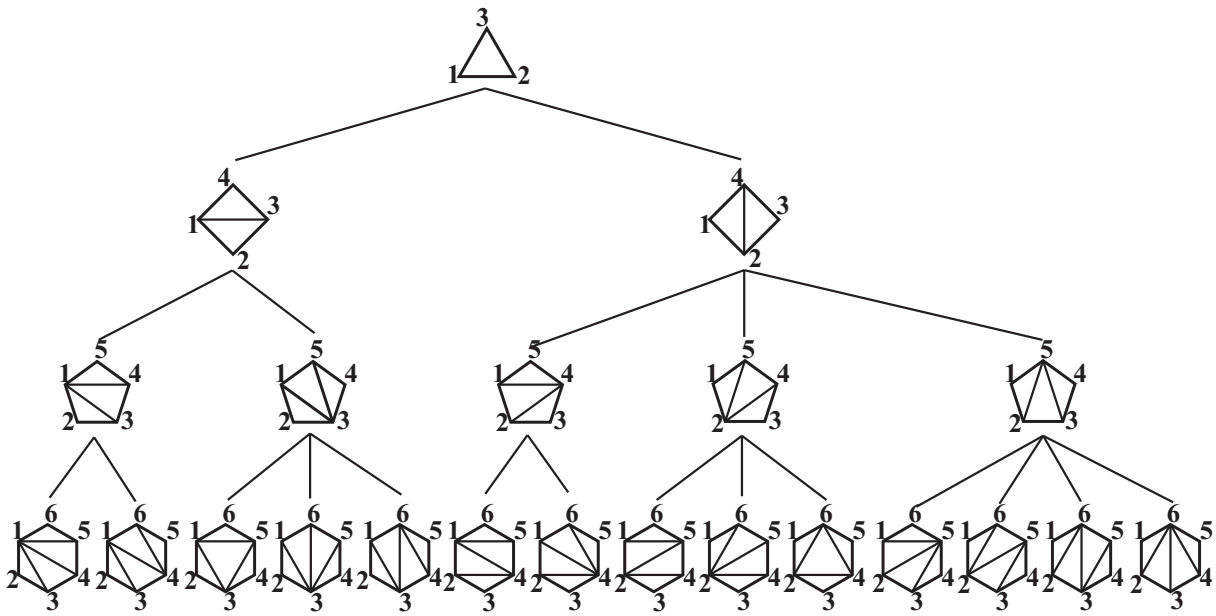
$$S^{i_1}(\tau_{n-1}^l), S^{i_2}(\tau_{n-1}^l), \ldots, S^{i_l}(\tau_{n-1}^l).$$

Cepanje dijagonale $\delta_{i_k,n-1}$ proizvodi potomka $S^{i_k}(\tau_{n-1}^l)$ sa $\deg(n) = 2 + k - 1$.



Slika 1.3: Ilustracija kreiranja potomaka triangulacije poligona P_n

Na Slici 1.4 je prikazana hijerarhija na stablu triangulacija od $n = 3$ do $n = 6$.



Slika 1.4: *Hurtado-Noy* hijerarhija

U nastavku je predstavljen Algoritam za generisanje triangulacija poligona iz rada [11], a isti se koristi za komparativnu analizu sa Algritmima koje smo razvili. Algoritam 1.4.1, po svojoj stukturi, na ulazu očekuje prirodan broj n i skup triangulacija poligona P_{n-1}. Triangulacije poligona su prikazane kao strukture koje sadrže $2n - 5$ parova temena koji predstavljaju dijagonale poligona P_{n-1} (dijagonale predstavljaju unutrašnje dijagonale i spoljašnje ivice poligona).

Algoritam 1.4.1 Hurtado-Noy algoritam

Ulaz: Pozitivni ceo broj n i skup triangulacija $\mathcal{T}_{n-1}$ poligona P_{n-1}.

1: Proverava strukturu koja sadrži $2n - 5$ parova temena koje odgovaraju određenoj triangulaciji τ_{n-1}, gde se traže parovi $(i_k, n-1)$, $i_k \in \{1, 2, \ldots, n-2\}$ (gornja granica za k je između 2 i $n - 2$), tj. dijagonale koje su incidente u odnosu na teme $n - 1$.

2: Za svako i_k generiše potomka $S^{i_k}(\tau_{n-1})$ tako što se vrši transformacija $\delta_{i_l,n-1} \to \delta_{i_l,n}$, $i_l < i_k,\ 0 \leq l \leq n-3$, i uvodi novi par temena $\delta_{i_k,n}$ i $\delta_{n-1,n}$.

3: Uzima sledeći i_k, ako postoji, i ide na Korak 2.

4: Nastavlja gore navedenu proceduru za sledeću triangulaciju poligona P_{n-1} (tj. strukturu sa $2n - 5$ parova temena), ako postoji. U suprotnom se završava.

Poglavlje 2

Algoritmi za generisanje triangulacija konveksnih poligona

Ovo poglavlje sadrži analizu novih algoritama za generisanje triangulacija konveksnih poligona.

Prvi algoritam za generisanje triangulacija poligona se naziva *Orbiting triangle metoda - OTM metoda*. Metoda se zasniva na korišćenju skupa triangulacija poligona P_{n-1} za generisanje triangulacija poligona P_n. Osnovno svojstvo ove metode proizilazi iz tehnike obilaska poligona P_n gde se generiše uređena lista (`OT-lista`), koja se koristi za generisanje triangulacija poligona P_n. Glavna karakteristika OTM metode je postupak mapiranja triangulacija kroz dve operacije, komplementiranja i rotiranja, kao i upotrebu Katalanovog trougla za identifikaciju važećih triangulacija. Naučni rad na kojem se zasniva ova metoda je [40]

Drugi algoritam za generisanje triangulacija poligona naziva se *Modifikovana Blok metoda*. Ova metoda predstavlja modifikaciju originalne Blok metode koja takođe koristi triangulacije poligona P_{n-1} za generisanje triangulacija poligona P_n. Obzirom da poligon P_n može nastati iz poligona P_{n-1} dodavanjem temena n, Modifikovana Blok metoda teme n posmatra kroz dva slučaja: slučaj kada je n uvo poligona; i slučaj kada je n teme unutrašnje dijagonale. Glavno svojstvo ove metode se ogleda u korišćenju vrednosti iz Katalanovog trougla kako bi se izbeglo pojavljivanje duplih triangulacija.

U cilju dobijanja eksperimentalnih rezultata, za svaki od pomenutih algoritama je odrađena implementacija u Java programskom jeziku. Urađena je i komparativna analiza, i to: u prvom slučaju OTM metode [40] u odnosu na Hurtado-Noy metodu [11] i Blok metodu [33]; i u drugom slučaju Originalne Blok metode [33] i Modifikovane Blok metode.

2.1 OTM metoda za generisanje triangulacija poligona

OTM metoda ima za cilj da ubrza proces generisanja triangulacija konveksnog poligona. Osnovna strategija OTM metode je korišćenje triangulacija iz skupa $\mathcal{T}_{n-1}$ da bi se generisale triangulacije $\mathcal{T}_n$. Glavna karakteristika OTM metode je upotreba Katalanovog trougla za identifikaciju važećih triangulacija, tako da algritam ne troši skoro nikakvo veme za eliminaciju duplikata. Na ovaj način, algoritam poseduje malu složenost i štedi vreme potrebno za otkrivanje i eliminaciju duplikata.

2.1.1 Uvodne napomene i osnovne postavke

U slučaju konveksnih poligona, triangulacija predstavlja dekompoziciju poligona u skup generisanih trouglova sa nepresecajućim unutrašnjim dijagonalama.

Konveksni poligon P_n je određen svojim temenima označenim $(v_1, v_2, \ldots, v_n)$. Poligon P_n se može dekomponovati u $n-2$ trouglova sa svojim $n-3$ unutrašnjim dijagonalama. Unutrašnja dijagonala poligona P_n je definisanja svojim krajnim tačkama. Pretpostavimo da je dijagonala koja spaja tačke v_i i v_j označena sa $\delta_{v_i,v_j} = (v_i, v_j)$.

Skup triangulacija poligona P_n je označen sa

$$\mathcal{T}_n = \{t_n^p \mid p = 1, \ldots, C_{n-2}\}, \tag{2.1}$$

gde

$$C_{n-2} = \frac{1}{n-1}\binom{2n-4}{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}, \quad n \geq 3 \tag{2.2}$$

označava kardinalnost skupa $\mathcal{T}_n$ i C_n predstavlja n-ti Katalanov broj (više za ovu temu pogledati rad [45])

Osnovno pravilo za konstrukciju Katalanovog trougla predstavljeno je sledećom formulom:

$$C(n,k) = \frac{(n+k)!(n-k+1)}{k!(n+1)!}, \tag{2.3}$$

gde je n nenegativni ceo broj i $k = 0, \ldots, n$.

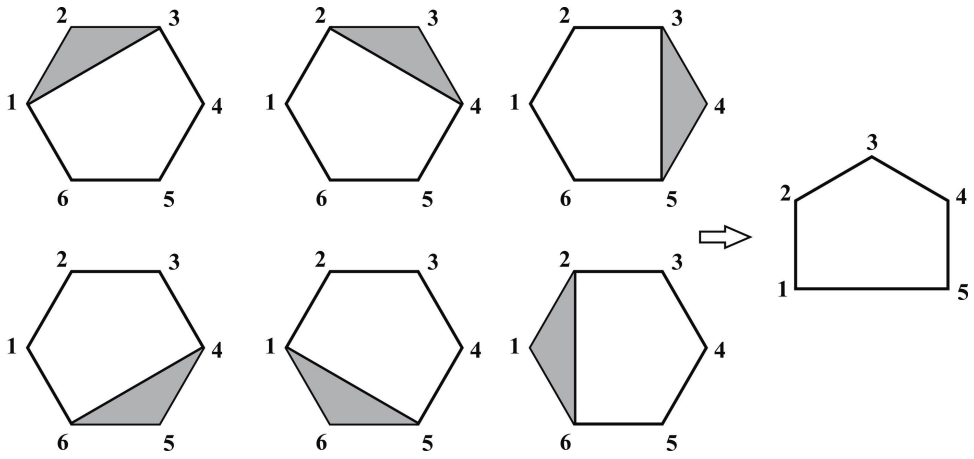
Vrednosti iz Katalanovog trougla su date u Poglavlju 1 (Tabela 1.1), koje se koriste za efikasno detektovanje i eliminisanje duplih triangulacija u OTM metodi.

2.1.2 Algoritam za OTM metodu

Uvo e poligona P je trougao formiran od tri uzastopna temena, tako da je jedna od njegovih stranica dijagonala poligona P, a preostale dve stranice su spoljašnje stranice poligona P.

Prema definiciji uva u radu [10] ili [17], uvo u temenu v_2 konveksnog poligona predstavlja trougao (v_1, v_2, v_3) formiran od tri uzastopna temena v_1, v_2, v_3.

Tada linijski segment $\delta_{v_1,v_3} = (v_1, v_3)$ između v_1 i v_3 predstavlja dijagonalu poligona. Ako krenemo od inicijalnog uva $(1, 2, 3)$ i obiđemo oko datog poligona P_n u smeru kazaljke na satu (sive površine na Slici 2.1), dobijamo sva moguća uva poligona. Broj triangulacija konveksnog poligona sa propisanim brojem ušiju je definisan u radu [10]. Odsecanjem bilo kojeg uva od poligona P_n dobijamo poligon P_{n-1}, kao što je prikazano na Slici 2.1.



Slika 2.1: Obilazak poligona P_6 i postupak odsecanja uva.

Uva dobijena obilaskom poligona daju pozadinu OTM metode. Očigledno je da su sva uva poligona P_n definisana skupom uređenih trojki temena, kao što sledi:

$$\mathcal{E} = \{(1,2,3), (2,3,4), \ldots, (n-2, n-1, n), (n-1, n, 1), (n, 1, 2)\}. \tag{2.4}$$

Pošto je svako uvo (i, j, k) iz skupa $\mathcal{E}$ jedinstveno određeno unutrašnjim dijagonalama $\delta_{i,k} = (i, k)$ poligona P_n, kao rezultat obilaska poligona P_n dobija se skup unutrašnjih dijagonala $\delta_{i,k}$ dobijenih od generisanih ušiju iz skupa $\mathcal{E}$, koji se naziva `OT-lista`. Preciznije, `OT-lista` poligona P_n je jednaka sledećoj uređenoj listi:

$$\{\delta_{1,3}, \delta_{2,4}, \ldots, \delta_{n-3,n-1}, \delta_{n-2,n}, \delta_{n-1,1}, \delta_{n,2}\}.$$

OTM metoda koristi samo prva $n-2$ elementa iz `OT-liste`. Lista ovih elemenata je označena sa ℓ_n i poseduje formu

$$\ell_n = \{\delta_{1,3}, \delta_{2,4}, \ldots, \delta_{n-3,n-1}, \delta_{n-2,n}\} = \{\delta_{k,k+2} \mid k = 1, \ldots, n-2\}. \tag{2.5}$$

Veoma je važno sačuvati redosled triangulacija iz skupa triangulacija $\mathcal{T}_{n-1}$ prema redosledu unutrašnjih dijagonala ℓ_n u postupku generisanja triangulacija $\mathcal{T}_n$. Preciznije, triangulacija iz skupa triangulacija $\mathcal{T}_{n-1}$ označena sa $\delta_{1,3}$ se koristi prva, zatim triangulacija označena sa $\delta_{2,4}$ i na kraju triangulacija označena sa $\delta_{n-3,n-1}$.

Prva operacija koja se koristi u OTM metodi je *komplementiranje triangulacija*, koja se koristi za mapiranje unutrašnjih dijagonala poligona P_{n-1} na osnovu pravila

$$\mathfrak{C}_{n-1}(\delta_{i,j}) = \delta_{n-i,n-j}, \quad i,j \in \{1,\ldots,n-1\},\ |i-j| > 1. \tag{2.6}$$

Komplementiranje triangulacije (2.6) mapira skup $\mathcal{T}_{n-1}$ u skup označen sa ${\mathcal{T}_{n-1}}^c = \mathfrak{C}_{n-1}(\mathcal{T}_{n-1})$, gde je mapiranje $\mathfrak{C}_{n-1}$ elementarno primenjeno na svaku unutrašnju dijagonalu uključenu u

$$\mathcal{T}_{n-1} = \{t^p_{n-1} \mid\ p = 1,\ldots,C_{n-3}\}. \tag{2.7}$$

Kako bi ovo detaljnije objasnili, skup unutrašnjih dijagonala koje definišu p-tu triangulaciju t^p_{n-1} je označen sa

$$t^p_{n-1} = \left\{\delta^p_{i_1,j_1},\ldots,\delta^p_{i_{n-4},j_{n-4}}\right\}. \tag{2.8}$$

Na osnovu (2.7) i (2.8), skup unutrašnjih dijagonala koje definišu sve triangulacije u $\mathcal{T}_{n-1}$ su definisane sa uređenim $(n-4)$-torkama unutrašnjih dijagonala, kao što sledi

$$\mathcal{T}_{n-1} = \left\{\left\{\delta^p_{i_1,j_1},\ldots,\delta^p_{i_{n-4},j_{n-4}}\right\} \mid\ p = 1,\ldots,C_{n-3}\right\}.$$

Iz čega sledi

$$\begin{aligned}
{\mathcal{T}_{n-1}}^c &= \mathfrak{C}_{n-1}(\mathcal{T}_{n-1}) \\
&= \left\{\left(t^p_{n-1}\right)^c \mid p = 1,\ldots,C_{n-3}\right\} \\
&= \left\{\left\{\left(\delta^p_{i_1,j_1}\right)^c,\ldots,\left(\delta^p_{i_{n-4},j_{n-4}}\right)^c\right\} \mid p = 1,\ldots,C_{n-3}\right\}.
\end{aligned}$$

Drugo važno mapiranje koje se koristi u OTM metodi se naziva *rotiranje triangulacija*. Ovo mapiranje definiše pomeranje svih dijagonala tj. njegovih krajnjih temena (koje opisuju određenu triangulaciju) za $l \geq 1$ pozicija u smeru kazaljke na satu. Rotiranje dijagonale $\delta_{i,j} \in \mathcal{T}_{n-1}$ je definisano kao transformacija na osnovu pravila

$$\mathfrak{R}_{l,n-1}(\delta_{i,j}) = \delta_{1+(i+l) \bmod n,1+(j+l) \bmod n} \in \mathcal{T}_n, \tag{2.9}$$

gde se vrednosti za l uzimaju iz skupa $\{1,\ldots,n-2\}$. Elementarno rotiranje skupa $\mathcal{T}_{n-1}$ za l pozicija je označeno sa $\mathfrak{R}_{l,n-1}(\mathcal{T}_{n-1}) = {\mathcal{T}_{n-1}}^{r_l}$ i definisano sa

$$\begin{aligned}
{\mathcal{T}_{n-1}}^{r_l} &= \mathfrak{R}_{l,n-1}(\mathcal{T}_{n-1}) \\
&= \left\{\left(t^p_{n-1}\right)^{r_l} \mid p = 1,\ldots,C_{n-3}\right\} \\
&= \left\{\left\{\left(\delta^p_{i_1,j_1}\right)^{r_l},\ldots,\left(\delta^p_{i_{n-4},j_{n-4}}\right)^{r_l}\right\} \mid p = 1,\ldots,C_{n-3}\right\}.
\end{aligned}$$

OTM metoda se zasniva na korišćenje skupa
$\mathfrak{R}_{l,n-1}\left({\mathcal{T}_{n-1}}^c\right) = {\mathcal{T}_{n-1}}^{c,r_l} = \left\{\left\{\left(\delta^p_{i_1,j_1}\right)^{c,r_l},\ldots,\left(\delta^p_{i_{n-4},j_{n-4}}\right)^{c,r_l}\right\} \mid p = 1,\ldots,C_{n-3}\right\}.$

U cilju boljeg pojašnjenja transformacije (2.6), može se zaključiti da svaki poligon P_n može posmatrati u formi $P_n = P_{n-1} \cup (v_{j-1}, v_j, v_{j+1})$, gde je

$$\begin{aligned} P_n &= \{v_1, \ldots, v_{j-1}, v_j, v_{j+1}, \ldots, v_n\}, \\ P_{n-1} &= \{u_1 := v_1, \ldots, u_{j-1} := v_{j-1}, u_j := v_{j+1}, \ldots, u_{n-1} := v_n\}. \end{aligned} \tag{2.10}$$

Na osnovu (2.10), svaka triangulacija iz skupa $\mathfrak{C}_{n-1}(\mathcal{T}_{n-1})$ je uključena u P_n sa odvojenim uvom (v_{j-1}, v_j, v_{j+1}), gde teme v_j ne učestvuje u rotaciji. U suštini, rotacija (2.9) je primenjena na u_i temena i konačni rezultat je dat u skupu v_i temena.

Pored komplementacije i rotacije, neophodno je odraditi konkatenaciju k-tog elementa iz ℓ_n, koji je jednak $\delta_{k,k+2}$, sa komplementiranim i rotiranim triangulacijama uključenim u skupu ${\mathcal{T}_{n-1}}^{c,r_k}$, za svaki $k = 1, \ldots, n-2$. Ova konkatenacija je označena sa $\delta_{k,k+2} \circ {\mathcal{T}_{n-1}}^{c,r_k}$ i definisana sa

$$\delta_{k,k+2} \circ {\mathcal{T}_{n-1}}^{c,r_k} = \left\{ \delta_{k,k+2} \diamond \left(t^p_{n-1}\right)^{c,r_k} \;\middle|\; t^p_{n-1} \in \mathcal{T}_{n-1}, p = 1, \ldots, C_{n-3} \right\},$$

gde $\delta_{k,k+2} \diamond \left(t^p_{n-1}\right)^{c,r_k}$ pretpostavlja umetanje $\delta_{k,k+2}$ na početak triangulacije $\left(t^p_{n-1}\right)^{c,r_k}$. Preciznije, za svaki element iz skupa

$$\left(t^p_{n-1}\right)^{c,r_k} = \left\{ \left(\delta^p_{i_1,j_1}\right)^{c,r_k}, \ldots, \left(\delta^p_{i_{n-4},j_{n-4}}\right)^{c,r_k} \right\},$$

umetanje je definisano kao

$$\delta_{k,k+2} \diamond \left(t^p_{n-1}\right)^{c,r_k} = \left\{ \delta_{k,k+2}, \left(\delta^p_{i_1,j_1}\right)^{c,r_k}, \ldots, \left(\delta^p_{i_{n-4},j_{n-4}}\right)^{c,r_k} \right\}.$$

Na ovaj način dobijamo validne i jedinstvene triangulacije, gde se koriste samo sledeće triangulacije iz skupa ${\mathcal{T}_{n-1}}^{c,r_k}$:

$$\Upsilon^k_{n-1} = \left\{ \left(t^p_{n-1}\right)^{c,r_k} \in {\mathcal{T}_{n-1}}^{c,r_k} \mid p = C_{n-3} - C(n-3, n-k-2) + 1, \ldots, C_{n-3} \right\}. \tag{2.11}$$

U tom smislu, potrebno je generisati skup

$$\delta_{k,k+2} \circ \Upsilon^k_{n-1} = \left\{ \delta_{k,k+2} \diamond \left(t^p_{n-1}\right)^{c,r_k} \mid p = C_{n-3} - C(n-3, n-k-2) + 1, \ldots, C_{n-3} \right\}, \tag{2.12}$$

za svaki $k = 1, \ldots, n-2$.

Prema tome potrebno je rotirati samo određene delove iz skupa $\mathcal{T}_{n-1}$, koji se određuju na osnovu formule (2.11). Na osnovu gore navedenog, dolazimo do ključnog segmenta OTM metode, tehnike za efikasno eliminisanje duplih triangulacija, što rezultira na uštedi računarskih resursa prilikom generisanja triangulacija.

Pseudokod OTM metode je opisan u sledećem Algoritmu 2.1.1.

Algoritam 2.1.1 Orbiting Triangle Method - OTM metoda

Ulaz: Pozitivan ceo broj n i skup triangulacija $\mathcal{T}_{n-1}$ poligona P_{n-1} (opisan unutrašnjim dijagonalama u formi $\delta_{i,j}$).

1: Formira se lista ℓ_n.

2: Sa ulaza se učitava skup triangulacija $\mathcal{T}_{n-1}$.

3: Generisanje triangulacija $\mathcal{T}_n$:

(a) Izvršavanje operacije *komplementiranje triangulacija* nad skupom $\mathcal{T}_{n-1}$ gde se dobija skup $\mathfrak{C}_{n-1}(\mathcal{T}_{n-1})$.

(b) Generisanje skupa triangulacija (2.11) iz skupa ${\mathcal{T}_{n-1}}^{c,r_k}$, za svaki $k = 1, \ldots, n-2$.

(c) Generisanje skupa Υ^k_{n-1} na osovu formule (2.11) i generisanje skupa $\delta_{k,k+2} \circ \Upsilon^k_{n-1}$, za svaki $k = 1, \ldots, n-2$.

4: Vraća se skup triangulacija $\mathcal{T}_n = \left\{\delta_{k,k+2} \circ \Upsilon^k_{n-1},\ k = 1, \ldots, n-2\right\}$.

Napomena 2.1.1. *Lista ℓ_n je uvek oblika* (2.5) *i podrazumevano je dostupna.*

Primer 2.1.1. Ilustrujmo kako radi Algoritam 2.1.1 pri generisanju triangulacija šestougla koristeći poznate triangulacije petougla.
U koraku 1 se formira lista ℓ_6:

$$\ell_6 = (\delta_{1,3}, \delta_{2,4}, \delta_{3,5}, \delta_{4,6}).$$

U koraku 2 se vrši iščitavanje skupa triangulacija petougla $\mathcal{T}_5$

$$\mathcal{T}_5 = \{\{\delta_{1,3}, \delta_{1,4}\}, \{\delta_{1,3}, \delta_{5,3}\}, \{\delta_{2,4}, \delta_{2,5}\}, \{\delta_{2,4}, \delta_{1,4}\}, \{\delta_{3,5}, \delta_{2,5}\}\}.$$

Uređenost triangulacija u skupu $\mathcal{T}_5$ je inicirana sekvencom

$$\delta_{1,3}, \delta_{2,4}, \delta_{3,5}$$

nametnutom od strane prva tri elementa liste ℓ_6.

Na osnovu koraka 3(a), neophodno je odraditi komplementiranje triangulacija (formula (2.6)) iz skupa $\mathcal{T}_5$, gde se dobija skup $\mathfrak{C}_5(\mathcal{T}_5) = {\mathcal{T}_5}^c$. Ceo postupak je predstavljen u Tabeli 2.1.

Tabela 2.1: Skup triangulacija petougla i postupak komplementiranja.

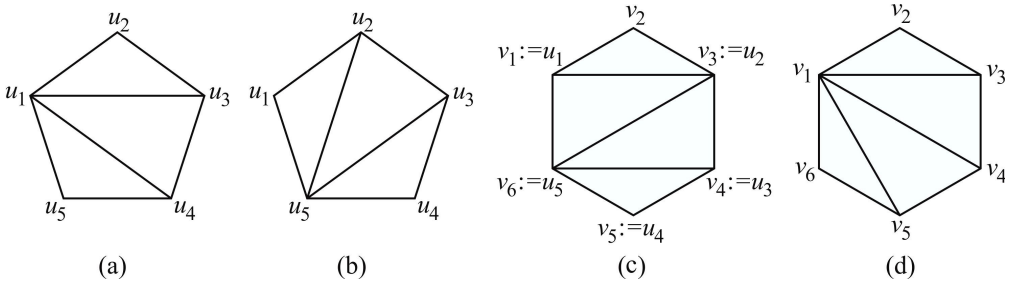
$\mathcal{T}_5$	$\delta_{i,j} \rightarrow \delta_{(6-i),(6-j)}$	$\mathcal{T}_5^c$
$\{\delta_{1,3}, \delta_{1,4}\}$	$\{\delta_{(6-1),(6-3)}, \delta_{(6-1),(6-4)}\}$	$\{\delta_{5,3}, \delta_{5,2}\}$
$\{\delta_{1,3}, \delta_{5,3}\}$	$\{\delta_{(6-1),(6-3)}, \delta_{(6-5),(6-3)}\}$	$\{\delta_{5,3}, \delta_{1,3}\}$
$\{\delta_{2,4}, \delta_{2,5}\}$	$\{\delta_{(6-2),(6-4)}, \delta_{(6-2),(6-5)}\}$	$\{\delta_{4,2}, \delta_{4,1}\}$
$\{\delta_{2,4}, \delta_{1,4}\}$	$\{\delta_{(6-2),(6-4)}, \delta_{(6-1),(6-4)}\}$	$\{\delta_{4,2}, \delta_{5,2}\}$
$\{\delta_{3,5}, \delta_{2,5}\}$	$\{\delta_{(6-3),(6-5)}, \delta_{(6-2),(6-5)}\}$	$\{\delta_{3,1}, \delta_{4,1}\}$

Na osnovu koraka 3(b), nephodno je rotirati triangulacije iz skupa $\mathcal{T}_5^c$ za $1, 2, 3$, i 4 pozicija. Naravno, rotira se samo podskup skupa $\mathcal{T}_5^c$ koji je definisan na osnovu formule (2.11). Posebno, uzima se samo 5, 5, 3 i 1 triangulacija koje su poravnane pri dnu a nalaze se u kolonama 1, 2, 3 i 4, respektivno. Ovaj postupak je ilustrovan u Tabeli 2.2.

Tabela 2.2: Rotiranje triangulacija iz skupa $\mathfrak{C}_{n-1}(\mathcal{T}_5)$ kako bi odgovarale za P_6.

Υ_{n-1}^1	Υ_{n-1}^2	Υ_{n-1}^3	Υ_{n-1}^4
$\{\delta_{5,3}, \delta_{5,2}\} \rightarrow \{\delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{5,3}, \delta_{5,2}\} \rightarrow \{\delta_{2,6}, \delta_{2,5}\}$		
$\{\delta_{5,3}, \delta_{1,3}\} \rightarrow \{\delta_{1,5}, \delta_{3,5}\}$	$\{\delta_{5,3}, \delta_{1,3}\} \rightarrow \{\delta_{2,6}, \delta_{4,6}\}$		
$\{\delta_{4,2}, \delta_{4,1}\} \rightarrow \{\delta_{6,4}, \delta_{6,3}\}$	$\{\delta_{4,2}, \delta_{4,1}\} \rightarrow \{\delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{4,2}, \delta_{4,1}\} \rightarrow \{\delta_{2,6}, \delta_{2,5}\}$	
$\{\delta_{4,2}, \delta_{5,2}\} \rightarrow \{\delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{4,2}, \delta_{5,2}\} \rightarrow \{\delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{4,2}, \delta_{5,2}\} \rightarrow \{\delta_{2,6}, \delta_{3,6}\}$	
$\{\delta_{3,1}, \delta_{4,1}\} \rightarrow \{\delta_{5,3}, \delta_{6,3}\}$	$\{\delta_{3,1}, \delta_{4,1}\} \rightarrow \{\delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{3,1}, \delta_{4,1}\} \rightarrow \{\delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{3,1}, \delta_{4,1}\} \rightarrow \{\delta_{2,6}, \delta_{3,6}\}$

Postupak komplementiranja i rotiranja triangulacija je ilistrovan na Slici 2.2. Slika 2.2(a) prikazuje jednu triangulaciju pentagona $t_5^1 = \{\delta_{1,3}, \delta_{1,4}\}$. Odgovarajuća komplementirana trangulacija $(t_5^1)^c$ je prikazana na Slici 2.2(b). Slika 2.2(c) ilustruje pozicioniranje komplementirane triangulacije $(t_5^1)^c$ u šestougao sa uvom (v_1, v_2, v_3). Na kraju, triangulacija $(t_5^1)^c$ je rotirana za jednu poziciju, označeno sa $(t_5^1)^{c,r_1}$, što je prikazano na Slici 2.2(d).



Slika 2.2: Postupak komplementiranja i rotiranja triangulacije.

Pojasnimo postupak sa Slike 2.2(c) malo detaljnije. Na osnovu formule (2.10), šestougao $P_6 = (v_1, v_2, v_3, v_4, v_5, v_6)$ se može posmatrati kao unija petougla $P_5 = (u_1, u_2, u_3, u_4, u_5)$ i

uva ($v_1 := 1, v_2 := 2, v_3 := 3$). To znači da P_5 može biti izveden iz P_6 na osnovu pravilnosti:

$$P_5 = \{u_1 := v_1, u_2 := v_3, u_3 := v_4, u_4 := v_5, u_5 := v_6\}.$$

Posebno, uvo sa temenom $v_2 := 2$ ne učestvuje u postupak rotiranja. Prema tome, dijagonalna rotacija za $l = 1$ poziciju se izvodi na sledeći način:

$$\delta_{5,2} \to \delta_{\mathbf{1,4}} \Longleftrightarrow \delta_{u_5:=v_6,u_2:=v_3} \to \delta_{u_1:=v_1,u_3:=v_4},$$
$$\delta_{5,3} \to \delta_{\mathbf{1,5}} \Longleftrightarrow \delta_{u_5:=v_6,u_3:=v_4} \to \delta_{u_1:=v_1,u_4:=v_5}.$$

Detalji ove dijagonalne rotacije prikazani su u Tabeli 2.3.

Tabela 2.3: Rotiranje dijagonala na osnovu formule (2.9) za $l = 1$ poziciju.

$\delta_{5,2} \to \delta_{\mathbf{1,4}}$	
$u_5 \to 1 + (5+1) \bmod 6 = \mathbf{1} = v_1$	$u_2 \to 1 + (2+1) \bmod 6 = \mathbf{4} = v_4$
$\delta_{5,3} \to \delta_{\mathbf{1,5}}$	
$u_5 \to 1 + (5+1) \bmod 6 = \mathbf{1} = v_1$	$u_3 \to 1 + (3+1) \bmod 6 = \mathbf{5} = v_5$

Imajući u vidu formulu (2.10), vrši se rotiranje dijagonala u redosledu u kojem se nalaze u poligonu P_{n-1}, samo što ih u ovom slučaju pozicioniramo u poligon P_n sa određenim uvom, preuzimajući oznake temena iz poligona P_n. Kao posledica toga, rotiranje se izvodi jednostavnim rotiranjem temena, tako da, postupak dodatno ne opterećuje resurse računarskog sistema bez obzira na broj pozicija l, za koliko se vrši rotiranje.

Sledeći korak 3(c) ujedinjuje vrednosti iz liste ℓ_6 i Tabele 2.2, kao što je prikazano na Tabeli 2.4.

Tabela 2.4: Ažuriranje blokova za šestougao.

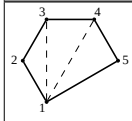
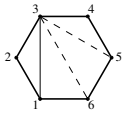
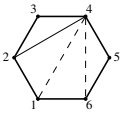
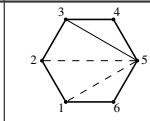
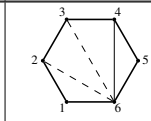
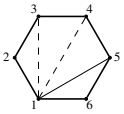
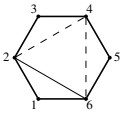
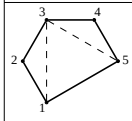
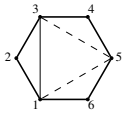
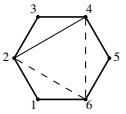
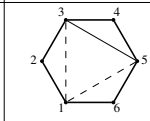
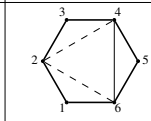
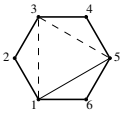
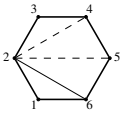
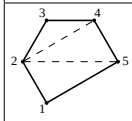
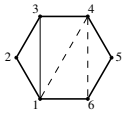
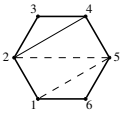
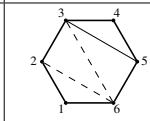
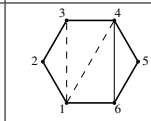
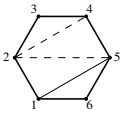
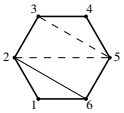
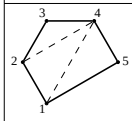
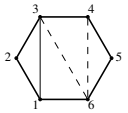
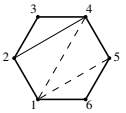
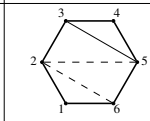
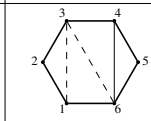
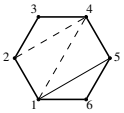
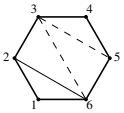
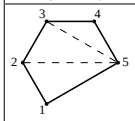
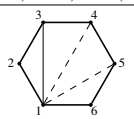
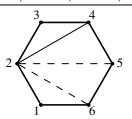
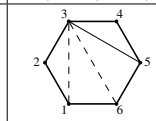
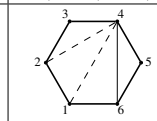
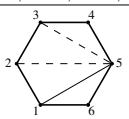
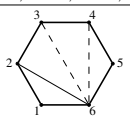
$\delta_{1,3} \circ \Upsilon^1_{n-1}$	$\delta_{2,4} \circ \Upsilon^2_{n-1}$	$\delta_{3,5}\Upsilon^3_{n-1}$	$\delta_{4,6} \circ \Upsilon^4_{n-1}$
$\{\delta_{1,3}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{2,5}\}$		
$\{\delta_{1,3}, \delta_{1,5}, \delta_{3,5}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{4,6}\}$		
$\{\delta_{1,3}, \delta_{6,4}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{2,5}\}$	
$\{\delta_{1,3}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{3,6}\}$	
$\{\delta_{1,3}, \delta_{5,3}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{4,6}, \delta_{2,6}, \delta_{3,6}\}$

Kao rezultat OTM metode, triangulacije šestougla $\mathcal{T}_6$ su definisane kao

$$\begin{aligned}\mathcal{T}_6 &= \left\{\delta_{1,3}\circ\Upsilon^1_{n-1}, \delta_{2,4}\circ\Upsilon^2_{n-1}, \delta_{3,5}\Upsilon^3_{n-1}, \delta_{4,6}\circ\Upsilon^4_{n-1}\right\}\\ &= \big\{\{\delta_{1,3},\delta_{1,5},\delta_{1,4}\},\{\delta_{1,3},\delta_{1,5},\delta_{3,5}\},\{\delta_{1,3},\delta_{6,4},\delta_{6,3}\},\{\delta_{1,3},\delta_{6,4},\delta_{1,4}\},\{\delta_{1,3},\delta_{5,3},\delta_{6,3}\},\\ &\quad \{\delta_{2,4},\delta_{2,6},\delta_{2,5}\},\{\delta_{2,4},\delta_{2,6},\delta_{4,6}\},\{\delta_{2,4},\delta_{1,5},\delta_{1,4}\},\{\delta_{2,4},\delta_{1,5},\delta_{2,5}\},\{\delta_{2,4},\delta_{6,4},\delta_{1,4}\}\\ &\quad \{\delta_{3,5},\delta_{2,6},\delta_{2,5}\},\{\delta_{3,5},\delta_{2,6},\delta_{3,6}\},\{\delta_{3,5},\delta_{1,5},\delta_{2,5}\}\\ &\quad \{\delta_{4,6},\delta_{2,6},\delta_{3,6}\}\big\}.\end{aligned}$$

Primer 2.1.2. Ilustrujmo primenu OTM metode ako izuzmemo operaciju komplementiranja triangulacija.

Tabela 2.5: Generisanje triangulacija šestougla $\mathcal{T}_6$ pomoću OTM metode bez operacije komplementiranja triangulacija

	$\ell_6 = (\delta_{1,3}, \delta_{2,4}, \delta_{3,5}, \delta_{4,6})$					
$\mathcal{T}_5$	$\delta_{1,3}\circ\Upsilon^1_5$	$\delta_{2,4}\circ\Upsilon^2_5$	$\delta_{3,5}\circ\Upsilon^3_5$	$\delta_{4,6}\circ\Upsilon^4_5$		
 $\{\delta_{1,3},\delta_{1,4}\}$	 $\{\delta_{1,3},\delta_{3,5},\delta_{3,6}\}$	 $\{\delta_{2,4},\delta_{4,6},\delta_{4,1}\}$	 $\{\delta_{3,5},\delta_{5,1},\delta_{5,2}\}$	 $\{\delta_{4,6},\delta_{6,2},\delta_{6,3}\}$	 ~~$\{\delta_{5,1},\delta_{1,3},\delta_{1,4}\}$~~	 ~~$\{\delta_{6,2},\delta_{2,4},\delta_{2,5}\}$~~
 $\{\delta_{1,3},\delta_{5,3}\}$	 $\{\delta_{1,3},\delta_{3,5},\delta_{1,5}\}$	 $\{\delta_{2,4},\delta_{4,6},\delta_{2,6}\}$	 ~~$\{\delta_{3,5},\delta_{5,1},\delta_{3,1}\}$~~	 ~~$\{\delta_{4,6},\delta_{6,2},\delta_{4,2}\}$~~	 ~~$\{\delta_{5,1},\delta_{1,3},\delta_{5,3}\}$~~	 ~~$\{\delta_{6,2},\delta_{2,4},\delta_{6,4}\}$~~
 $\{\delta_{2,4},\delta_{2,5}\}$	 $\{\delta_{1,3},\delta_{4,6},\delta_{4,1}\}$	 $\{\delta_{2,4},\delta_{5,1},\delta_{5,2}\}$	 $\{\delta_{3,5},\delta_{6,2},\delta_{6,3}\}$	 ~~$\{\delta_{4,6},\delta_{1,3},\delta_{1,4}\}$~~	 ~~$\{\delta_{5,1},\delta_{2,4},\delta_{2,5}\}$~~	 ~~$\{\delta_{6,2},\delta_{3,5},\delta_{3,6}\}$~~
 $\{\delta_{2,4},\delta_{1,4}\}$	 $\{\delta_{1,3},\delta_{4,6},\delta_{3,6}\}$	 $\{\delta_{2,4},\delta_{5,1},\delta_{4,1}\}$	 $\{\delta_{3,5},\delta_{6,2},\delta_{5,2}\}$	 ~~$\{\delta_{4,6},\delta_{1,3},\delta_{6,3}\}$~~	 ~~$\{\delta_{5,1},\delta_{2,4},\delta_{1,4}\}$~~	 ~~$\{\delta_{6,2},\delta_{3,5},\delta_{2,5}\}$~~
 $\{\delta_{3,5},\delta_{2,5}\}$	 $\{\delta_{1,3},\delta_{5,1},\delta_{4,1}\}$	 $\{\delta_{2,4},\delta_{6,2},\delta_{5,2}\}$	 ~~$\{\delta_{3,5},\delta_{1,3},\delta_{6,3}\}$~~	 ~~$\{\delta_{4,6},\delta_{2,4},\delta_{1,4}\}$~~	 ~~$\{\delta_{5,1},\delta_{3,5},\delta_{2,5}\}$~~	 ~~$\{\delta_{6,2},\delta_{4,6},\delta_{3,6}\}$~~

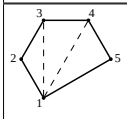
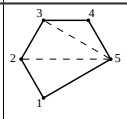
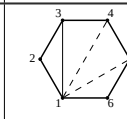
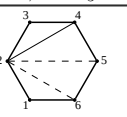
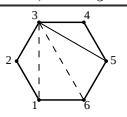
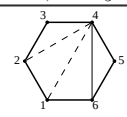
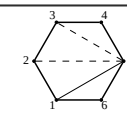
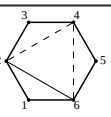
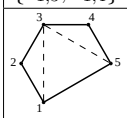
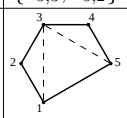
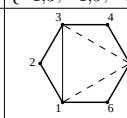
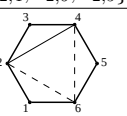
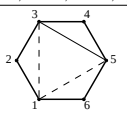
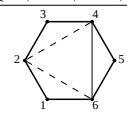
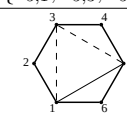
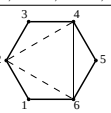
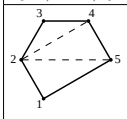
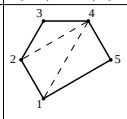
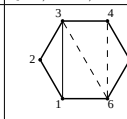
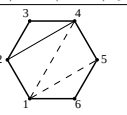
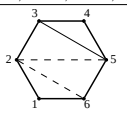
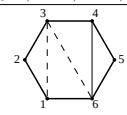
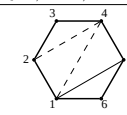
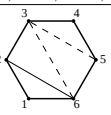
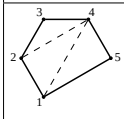
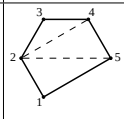
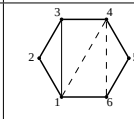
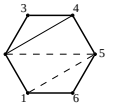
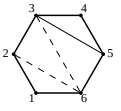
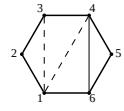
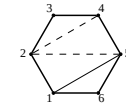
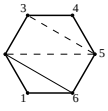
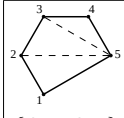
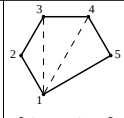
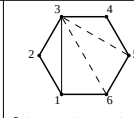
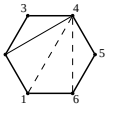
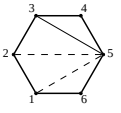
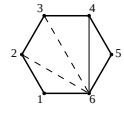
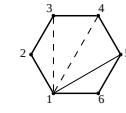
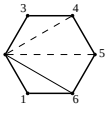
Kao što se može videti u Tabeli 2.5, redosled pojavljivanja duplih triangulacija nije uređen i u ovom slučaju se ne može primeniti svojstvo isključivanja duplih triangulacija na osnovu vrednosti iz Katalanovog trougla.

Iz gore navedenog možemo zaključiti da je primena operacije komplementiranja triangulacija, osnov za efikasno isključivanje duplih triangulacija, što je i jedna od glavnih karakteristika OTM metode.

U sledećem primeru je prikazana glavna karakteristika OTM metode u efiksnom isključivanju duplih triangulacija, tako da se iste i ne obrađuju.

Primer 2.1.3. Ključni deo OTM metode je jeftino isključivanje duplih triangulacija na osnovu vrednosti iz Katalanovog trougla. Za detaljnije objašnjenje, ilustrujmo (u Tabeli 2.8) postupak generisanja triangulacija P_6 primenjujući operacije komplementiranja i rotiranja kao što je navedeno u Algoritmu 2.1.1.

Tabela 2.6: Generisanje triangulacije šestougla $\mathcal{T}_6$ pomoću OTM metode.

		$\ell_6 = (\delta_{1,3}, \delta_{2,4}, \delta_{3,5}, \delta_{4,6})$					
$\mathcal{T}_5$	$\mathfrak{C}_{n-1}(\mathcal{T}_5)$	$\delta_{1,3} \circ \Upsilon_5^1$	$\delta_{2,4} \circ \Upsilon_5^2$	$\delta_{3,5} \circ \Upsilon_5^3$	$\delta_{4,6} \circ \Upsilon_5^4$		
							
$\{\delta_{1,3}, \delta_{1,4}\}$	$\{\delta_{5,3}, \delta_{5,2}\}$	$\{\delta_{1,3}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{2,5}\}$	~~$\{\delta_{3,5}, \delta_{3,1}, \delta_{3,6}\}$~~	~~$\{\delta_{4,6}, \delta_{4,2}, \delta_{4,1}\}$~~	~~$\{\delta_{5,1}, \delta_{5,3}, \delta_{5,2}\}$~~	~~$\{\delta_{6,2}, \delta_{6,4}, \delta_{6,3}\}$~~
							
$\{\delta_{1,3}, \delta_{5,3}\}$	$\{\delta_{5,3}, \delta_{1,3}\}$	$\{\delta_{1,3}, \delta_{1,5}, \delta_{3,5}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{4,6}\}$	~~$\{\delta_{3,5}, \delta_{3,1}, \delta_{5,1}\}$~~	~~$\{\delta_{4,6}, \delta_{4,2}, \delta_{6,2}\}$~~	~~$\{\delta_{5,1}, \delta_{5,3}, \delta_{1,3}\}$~~	~~$\{\delta_{6,2}, \delta_{6,4}, \delta_{2,4}\}$~~
							
$\{\delta_{2,4}, \delta_{2,5}\}$	$\{\delta_{4,2}, \delta_{4,1}\}$	$\{\delta_{1,3}, \delta_{6,4}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{2,5}\}$	~~$\{\delta_{4,6}, \delta_{3,1}, \delta_{3,6}\}$~~	~~$\{\delta_{5,1}, \delta_{4,2}, \delta_{4,1}\}$~~	~~$\{\delta_{6,2}, \delta_{5,3}, \delta_{5,2}\}$~~
							
$\{\delta_{2,4}, \delta_{1,4}\}$	$\{\delta_{4,2}, \delta_{5,2}\}$	$\{\delta_{1,3}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{3,6}\}$	~~$\{\delta_{4,6}, \delta_{3,1}, \delta_{4,1}\}$~~	~~$\{\delta_{5,1}, \delta_{4,2}, \delta_{5,2}\}$~~	~~$\{\delta_{6,2}, \delta_{5,3}, \delta_{6,3}\}$~~
							
$\{\delta_{3,5}, \delta_{2,5}\}$	$\{\delta_{3,1}, \delta_{4,1}\}$	$\{\delta_{1,3}, \delta_{5,3}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{4,6}, \delta_{2,6}, \delta_{3,6}\}$	~~$\{\delta_{5,1}, \delta_{3,1}, \delta_{4,1}\}$~~	~~$\{\delta_{6,2}, \delta_{4,2}, \delta_{5,2}\}$~~

Važno je napomenuti da se duple triangulacije u Tabeli 2.8 pojavljuju sekvencijalno i izlaze na vrhu. U skladu sa ovim zapažanjem, OTM metoda identifikuje jedinstvene triangulacije u tačnom redosledu kao vrednosti iz Katalanovog trougla, što predstavlja pozadinu za eliminisanje duplih triangulacija.

U sledećoj Teoremi 2.1.1 je prikazana ispravnost Algoritma 2.1.1.

Teorema 2.1.1. *Skup triangulacija* $\mathcal{T}_n = \left\{\delta_{k,k+2} \circ \Upsilon_{n-1}^k, k = 1, \ldots, n-2\right\}$ *generisan pomoću Algoritma 2.1.1 je validan.*

Dokaz. Očigledno, kardinalnost skupa triangulacija $\mathcal{T}_n$ je jednak Katalanovom broju C_{n-2}, pošto je broj elemenata skupa $\mathcal{T}_n$ jednak sumi elemenata odgovarajućeg reda $(n-3)$ iz Katalanovog trougla. Kasnije, utvrđuje se da skup triangulacija $\mathcal{T}_n$ ne sadrži duplikate.

Dokaz se nastavlja matematičkom indukcijom. Pretpostavimo da je skup triangulacija $\mathcal{T}_{n-1}$ validan, tj. bez duplikata. Ovo implicira da je skup ${\mathcal{T}_{n-1}}^{c,r_k}$, generisan u koraku 3(b) Algoritma 2.1.1, bez duplikata. Dalje, skup Υ_{n-1}^k je, takođe, bez duplikata i konačno skup $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ je bez duplikata za svaki $k = 1, \ldots, n-2$. Ovo implicira da su svi elementi u proizvoljnoj k-toj koloni bez duplikata.

Pored toga, razmotrimo triangulacije uključene u k-toj koloni, $k \in \{1, \ldots, n-2\}$. Ove triangulacije su forme $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ i počinju sa dijagonalom $\delta_{k,k+2}$, koja je jedinstvena za ovu kolonu. Proizvoljno različita kolona j, $j \neq k$, uključuje triangulacije forme $\delta_{j,j+2} \circ \Upsilon_{n-1}^j$, koja sadrži prvu dijagonalu koja je jednaka $\delta_{j,j+2}, \neq \delta_{k,k+2}$. Na ovaj način, triangulacije $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ i $\delta_{j,j+2} \circ \Upsilon_{n-1}^j$ su sigurno različite za svaku $j \neq k$. Prema tome, elementi iz $j$te i $k$te kolone su međusobno različiti. Na osnovu prethodne izjave, sve triangulacije $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ uključene u proizvoljnu $k$tu kolonu su bez duplikata, tako da je dokaz završen. □

2.1.3 Kompleksnost algoritma

Razmotrimo kompleksnost Algoritma 2.1.1. Drugim rečima, želimo da odredimo koliko operacija je potrebno da bi se generisala jedna triangulacija poligona P_n pomoću Algoritma 2.1.1.

Operacija komplementiranja triangulacija zahteva izvođenje $2(n-4) = 2n-8$ operacija oduzimanja. Zaista, svaka triangulacija iz skupa $\mathcal{T}_{n-1}$ je definisana od $n-4$ dijagonala i potrebna su dva oduzimanja za komplementiranje svake dijagonale u triangulaciji, kao što je definisano u formuli (2.6).

U nastavku, za operacije rotiranja triangulacija, potrebno je da rotiramo temena poligona za k pozicija, $k \in \{1, \ldots, n-2\}$ u smeru kazaljke na satu, kao što je definisano u formuli (2.9). Prema tome, to uvek čini dva dodatka. Obzirom da je broj dijagonala u triangulaciji $n-4$ onda sledi da je i ukupan broj dodataka potrebnih za izvršenje rotacije $2n-8$.

Sada je neophodno izvršiti jednu konkatenaciju da bi se dobio element skupa (2.12). Sveukupno to čini $4n-15$ operacija. Zaključujemo da Algoritam 2.1.1 pokazuje linearnu kompleksnost.

Naravno, generisanje celog skupa $\mathcal{T}_n$ bilo bi složeniji problem zbog nelinearnosti Katalanovih brojeva. Kako generisanje pojedinačne triangulacije ima kompleksnost $\mathcal{O}(n)$, ukupan broj operacija za generisanje triangulacija $\mathcal{T}_n$ je jednak

$$N_{OTM} = (2n-8)C_{n-3} + (2n-8)C_{n-2} + C_{n-2} = (2n-8)C_{n-3} + (2n-7)C_{n-2}.$$

Ovaj broj operacja se može uporediti sa brojem operacija potrebnih kod Hurtado-Noy metode:

$$N_H = (2n-5)C_{n-3} + (2n-3)C_{n-2}$$

Što je lako izračunati

$$N_H - N_{OTM} = 3C_{n-3} + 4C_{n-2} \geq 0, \quad n > 4.$$

2.1.4 Komparativna analiza i eksperimentalni rezultati

U ovoj sekciji je navedena komparativna analiza Blok metode, Hurtado-Noy metode i OTM metode. Za implementaciju smo koristili NetBeans IDE okruženje i Java programski jezik.

Vreme izvršenja sva tri algoritma je prikazano u Tabeli 2.9. Kolona "Ubrzanje" u tabeli je predstavljena u dva oblika, uporedni rezultati izvršavanja Blok metode naspram OTM metode i Hurtado-Noy metode naspram OTM metode.

Testiranje je odrađeno u NetBeans testnom modulu "Profile Main Project / CPU Analyze Performance" na računaru sledećih karakteristika*: *CPU - Inter(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40GHz, RAM memorija 8GB, Grafička karta: NVIDIA GeForce 820M.*

Tabela 2.7: Eksperimentalni rezultati testiranja: Blok metoda vs. OTM metoda & Hurtado-Noy metoda vs. OTM metoda.

n	**Broj triangulacija**	**Blok metoda**	**H.N. metoda**	**OTM metoda**	**Ubrzanje Blok vs OTM**	**Ubrzanje H.N. vs OTM**
5	5	0.000	0.001	0.000	-	-
6	14	0.001	0.002	0.001	1.00	2.00
7	42	0.001	0.002	0.001	1.00	2.00
8	132	0.001	0.002	0.001	1.00	2.00
9	429	0.002	0.003	0.002	1.03	1.50
10	1,430	0.004	0.006	0.003	1.33	2.00
11	4,862	0.010	0.014	0.008	1.25	1.75
12	16,796	0.015	0.033	0.013	1.15	2.54
13	58,786	0.047	0.055	0.038	1.24	1.45
14	208,012	0.140	0.143	0.114	1.23	1.25
15	742,900	0.396	0.512	0.313	1.27	1.64
16	2,674,440	1.761	2.536	1.293	1.36	1.96
17	9,694,845	8.443	10.098	6.728	1.25	1.50
18	35,357,670	106.782	151.273	49.688	2.15	3.04

Rezultati prikazani u Tabeli 2.9 potvrđuju efikasnost OTM metode u generisanju triangulacija i eliminisanju pojavljivanja duplikata. Tačnije, OTM metoda daje bolje rezultate u odnosu na Blok metodu i Hurtado-Noy metodu.

2.2 Modifikovana Blok metoda za generisanje triangulacija poligona

U ovom delu poglavlja je predstavljena modifikovana verzija Blok metode za generisanje triangulacija konveksnih poligona. Modifikovana, kao i originalna Blok metoda, koristi prethodno generisane triangulacije od poligona sa manjim brojem temena. Osnovna razlika se ogleda u tome što Modifikvana Blok metoda koristi vrednosti iz Katalanovog trougla kako bi se izbeglo pojavljivanje duplih triangulacija, čime se postiže značajno ubrzanje u odnosu na originalnu Blok metodu.

2.2.1 Uvodne napomene i osnovne postavke

Kao što smo naglasili ranije, triangulacija predstavlja dekompoziciju poligona unutrašnjim dijagonalama koje se ne seku.

Neka P_n označava poligon sa n temena. Originalna Blok metoda predstavljena u radu [33], kao i Modifikovana Blok metoda, koristi triangulacije poligona P_b za generisanje triangulacija poligona P_n, gde je $b < n$.

Kao što je poznato, ukupan broj triangulacija $\mathcal{T}_n$ za dati n-gon se dobija pomoću Katalanovog broja C_{n-2}, ili

$$\mathcal{T}_n = C_{n-2} = \frac{1}{n-1}\binom{2n-4}{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}, \quad n \geq 3. \tag{2.13}$$

Vezano za Katalanove broje i njihovu ulogu u više kombinatornih problema može se videti u monografijama [45, 36].

Skup svih triangulacija konveksnog poligona P_n je označen sa $\mathcal{T}_n$. Dijagonala koja spaja temena i i j je označena sa $\delta_{i,j}$. Spoljašnja stranica poligona može se smatrati dijagonalom dok su ne-susedna temena povezana *unutrašnjom dijagonalom*.

U originalnoj Blok metodi [33], sledeća tvrdnja je dokazana.

Tvrđenje 2.2.1. *Svaka triangulacija iz skupa $\mathcal{T}_{n-1}$ pojavljuje se kao početni deo u tačno dve triangulacije u skupu $\mathcal{T}_n$.*

Prema tome, $2T_{n-1}$ triangulacije poligona P_n mogu se lako izvesti, dok za ostatak $T_n - 2T_{n-1}$ triangulacija mora da se odradi složeniji postupak. Glavni problem u ovom postupku je kako izbeći pojavljivanje duplih triangulacija.

Modifikovana Blok metoda, kao što je navedeno, koristi prethodno generisane triangulacije poligona P_b, gde je $b < n$. Slično kao kod metode predstavljene u radu [40], Modifikovana Blok metoda koristi vrednosti iz Katalanovog trougla kako bi se izbeglo pojavljivanje duplih triangulacija, gde je opterećenje računarskih resursa u tom postupku zanemarljivo.

Osnovno pravilo za konstrukciju Katalanovog trougla dato je formulom (2.3).

Vrednosti iz Katalanovog trougla su predstavljene u Poglavlju 1 (Tabela 1.1), koje se koriste za izbegavanje pojavljivanja duplih triangulacija.

Pre nego što izložimo osnovna teorijska zapažanja, hajde da predložimo neku notaciju.

Definicija 2.2.1. *Neka skup* $\mathcal{A} = \{A_1, \ldots, A_m\}$ *i* $\mathcal{B} = \{B_1, \ldots, B_n\}$ *budu dva skupa triangulacija. Operacija* $\mathcal{A} \otimes \mathcal{B}$ *je definisana kao*

$$\mathcal{A} \otimes \mathcal{B} = \{A_1 \cup B_1, \ldots, A_1 \cup B_n, A_2 \cup B_1, \ldots, A_2 \cup B_n, \ldots, A_m \cup B_1, \ldots, A_m \cup B_n\}.$$

Definicija 2.2.2. *Neka se notacije* $\{v_1 \mapsto u_1, \ldots, v_m \mapsto u_m\}$ *odnose na mapiranje zamene svakog temena* v_j *na odgovarajuće teme* u_j*, za* $j = 1, \ldots, m$*, i svako drugo teme za sebe.*

2.2.2 Algoritam za Modifikovanu Blok metodu

Modifikovana Blok metoda, kao i originalna Blok metoda, koristi triangulacije polgona P_b (gde je $2 < b < n$) za generisanje triangulacija poligona P_n. Očigledno, poligon P_n može nastati iz poligona P_{n-1} dodavanjem temena n. U triangulacijama poligona P_n, novo dodato teme n može biti *uvo*, ili teme unutrašnje dijagonale.

Neka $\mathcal{T}_n^E$ označava skup tirangulacija poligona P_n koje nemaju dijagonalu kroz teme n (ili u koje je teme n uvo). Sa druge strane, ako triangulacija poligona P_n ima dijagonalu kroz teme n, označenu sa $\delta_{i,n}$, u tom slučaju posmatramo dva podpoligona $P_{L,i}^n$ (odn. $P_{R,i}^n$), sa skupom triangulacija $\mathcal{T}_{L,i}^n = \mathcal{T}(P_{L,i}^n)$ (odn. $\mathcal{T}_{R,i}^n = \mathcal{T}(P_{R,i}^n)$). Imajte na umu da ni $\mathcal{T}_{L,i}^n$ ni $\mathcal{T}_{R,i}^n$ nemaju dijagonalu kroz teme n.

Prema ovom zapažanju razlikujemo dva slučaja.

- **Slučaj 1, n je uvo**

  Kada je teme n uvo (ref. [42]), generišu se odgovarajuće triangulacije poligona P_n kao $\mathcal{T}_n^E := \mathcal{T}_{n-1} \otimes \{\{\delta_{1,n-1}\}\}$.

- **Slučaj 2, n je deo unutrašnje dijagonale**

  Koristimo strategiju "**podeli pa vladaj**" (divide and conquer[1]) da dati poligon P_n podelimo na sve moguće načine kroz teme n (koristeći dijagonale $\delta_{2,n}, \ldots, \delta_{n-2,n}$).

  Deljenjem poligona P_n pomoću proizvoljne unutrašnje dijagonale $\delta_{i,n}$, $i \in \{2, \ldots, n-2\}$ generiše se *levi podpoligon* $P_{L,i}^n$ i *desni podpoligon* $P_{R,i}^n$.

  Takva particija $(P_{L,i}^n, P_{R,i}^n)$ je definisana (2.14)

$$P_{L,i}^n := \{1, \ldots, i, n\}, \quad P_{R,i}^n := \{i, \ldots, n-1, n\}, \ i = 2, \ldots, n-2. \tag{2.14}$$

[1] https://en.wikipedia.org/wiki/Divide–and–conquer_algorithm

Dakle, razmatramo sve particije $(P^n_{L,i}, P^n_{R,i})$ poligona P_n

$$\{(P^n_{L,i}, P^n_{R,i})|\ i = 2, \ldots, n-2\},$$

gde su $P^n_{L,i}$ i $P^n_{R,i}$ definisani pomoću (2.14).

Osim toga, u određenim slučajevima potrebno je izvršiti odgovarajuće mapiranje indeksa temena P_b, gde je $b < n$. Naime, indeksi temena pojavljuju se kao krajnje tačke unutrašnjih dijagonala koje opisuju triangulaciju poligona. Kada dodamo teme n, neke od dijagonala koje opisuju triangulacije P_b mogu da promene indeks jedne ili obe njegove krajnje tačke.

U Slučaju 1, takvo mapiranje nije neophodno, dok je u Slučaju 2, takvo mapiranje potrebno. Potrebno je uzeti u obzir da prethodno generisane triangulacije P_{i+1} sadrže temena iz skupa $\{1, \ldots, i, i+1\}$ kao krajnje tačke sopstvenih unutrašnjih dijagonala. Uzimajući u obzir $P^n_{L,i} := \{1, \ldots, i, n\}$, zaključuje se da se indeks $i+1$ mora transformisati u n, kad god se pojavi. Koristeći notaciju uvedenu definicijom 2.2.2, takva transformacija se može opisati kao $P^n_{L,i} := P_{i+1}\{i+1 \mapsto n\}$.

Poligon $P^n_{R,i}$ je definisan temenima $\{i, \ldots, n-1, n\}$, dok su prethodno generisane triangulacije P_{n-i+1} opisane skupom temena $\{1, \ldots, n-i, n-i+1\}$.

Prema tome, $P^n_{R,i} = P_{n-i+1}\{j \mapsto j+i-1 :\ j \in \{1, 2, \ldots, n-i+1\}\}$.

Kardinalni broj $\mathcal{T}^n_{L,i}$ (odnosno $\mathcal{T}^n_{R,i}$) se označava sa $T^n_{L,i}$ (odnosno $T^n_{R,i}$).

Sažeto,

$$\begin{aligned}
\mathcal{T}_n &= \mathcal{T}^E_n \cup \bigcup_{i=2}^{n-2} \mathcal{T}^n_{L,i} \otimes \{\{\delta_{i,n}\}\} \otimes \mathcal{T}^n_{R,i} \\
\mathcal{T}^E_n &= \mathcal{T}_{n-1} \otimes \{\{\delta_{1,n-1}\}\} \\
\mathcal{T}^n_{L,i} &= \mathcal{T}(P^n_{L,i}) = \mathcal{T}_{i+1}\{i+1 \mapsto n\} \\
\mathcal{T}^n_{R,i} &= \mathcal{T}_{n-i}\{j \mapsto j+i-1 : j \in \{1, 2, \ldots, n-i\}\} \otimes \{\{\delta_{i,n-1}\}\}
\end{aligned}$$

Broj triangulacija za $P^n_{L,i}$ i $P^n_{R,i}$ određuju se odgovarajućim vrednostima Katalanovog trougla 2.3.

Lema 2.2.1. *Particije* (2.14) *zadovoljavaju sledeća svojstva:*
(a) $T^n_{L,i} = C(i-1, i-1) = C_{i-1}$.
(b) $T^n_{R,i} = C(n-i-1, n-i-1) = C_{n-i-1}$.

Dokaz. Na osnovu formule Katalanovih brojeva (1.2) i formule za konstrukciju Katalanovog trougla (1.4) imamo da je

$$\begin{aligned}
C(i-1, i-1) &= \frac{(i-1+i-1)!(i-1-i+1+1)}{(i-1)!i!} \\
&= \frac{(2i-2)!}{(i-1)!i!} \\
&= C_{i-1} = T_{i+1}.
\end{aligned}$$

Na kraju, identiteti

$$T^n_{L,i} = T_{i+1}, \quad T^n_{R,i} = T_{n-i+1} \tag{2.15}$$

kompletiraju dokaz. □

Svi elementi $\mathcal{T}^n_{L,i}$ su neophodni za generisanje triangulacija $\mathcal{T}_n$. Ali, u generisanju $\mathcal{T}^n_{R,i}$ moramo izbegavati generisanje ponovljenih triangulacija.

Teorema 2.2.1. *Samo $P^n_{R,i}$ triangulacije koje sadrže dijagonalu $\delta_{i,n}$ mogu dovesti do generisanja duplih triangulacija u $\mathcal{T}_n$.*

Dokaz. Dijagonala $\delta_{i,n}$ se pojavljuje kao unutrašnja dijagonala koja čini triangulaciju za $P_{L,i+1}$, jer prihvatamo sve triangulacije $P^n_{L,i+1}$. Prema tome, ako prihvatimo sve triangulacije $P^n_{R,i} := \{i, \ldots, n-1, n\}$, imali bismo duple triangulacije koje sadrže trougao $(i, i+1, n)$. Da bismo ovo izbegli, razmatramo samo triangulacije $P^n_{R,i}$ u kojima se teme n pojavljuje kao uvo. U stvari, ovo su triangulacije $P^{n-1}_{R,i}$ koje se završavaju sa $\delta_{i,n-1}$. Drugim rečima, razmatramo samo prve $C(n-i-1, n-i-1)$ triangulacije od $P^n_{R,i}$ □

Teorema 2.2.2. *Modifikovana Blok metoda generiše tačan broj C_{n-2} različitih P_n triangulacija.*

Dokaz. Na osnovu Slučaja 1, doprinos ukupnom broju P_n triangulacija je $T_{n-1} = C_{n-3}$ triangulacija. Ove triangulacije $\mathcal{T}^E_n := \mathcal{T}_{n-1} \otimes \{\{\delta_{1,n-1}\}\}$ su različite kao što su triangulacije u $\mathcal{T}_{n-1}$ različite.

U Slučaju 2, kada podelimo P_n sa $\delta_{2,n}$, imamo $P^n_{L,2} := \{1, 2, n\}$ kao trougao sa jedinstvenom trivijalnom triangulacijom ($T_3 = C_1 = 1$) kombinovano sa $T_{n-2} = C_{n-4}$ triangulacijama prihvaćenim od $P^n_{R,2}$ (praveći sve triangulacije od $P^{n-1}_{R,2}$).

Podela poligona P_n sa proizvoljnim $\delta_{i,n}$, $i > 2$ daje $P^n_{L,i}$ sa $T_{i+1} = C_{i-1}$ prihvaćenim triangulacijama od $\mathcal{T}(P_{L,i})$ kombinovane sa

$$T_{n-i} = C(n-i-2, n-i-2) = C_{n-i-2}$$

triangulacije $\mathcal{T}^{n-1}_{R,i}$.

Triangulacije u $\mathcal{T}^n_{L,i} \otimes \{\{\delta_{i,n}\}\} \otimes \mathcal{T}^n_{R,i}, i \in \{2, \ldots, n-2\}$ su različite, kao što su i triangulacije u $\mathcal{T}^n_{L,i}$, a i triangulacije u $\mathcal{T}^n_{R,i}$ su različite.

Takođe, $\forall i \in \{2, \ldots, n-2\}(\mathcal{T}^E_n \cup \mathcal{T}^n_{L,i} \otimes \{\{\delta_{i,n}\}\} \otimes \mathcal{T}^n_{R,i}) = \emptyset$, zato što triangulacije u $\mathcal{T}^E_n$ ne sadrže dijagonalu $\delta_{i,n}$.

Slično, $\forall i_1, i_2 \in \{2, \ldots, n-2\}, i_1 < i_2$ sledi da

$$(\mathcal{T}^n_{L,i_1} \otimes \{\{\delta_{i+1,n}\}\} \otimes \mathcal{T}^n_{R,i_1}) \cup (\mathcal{T}^n_{L,i_2} \otimes \{\{\delta_{i_2,n}\}\} \otimes \mathcal{T}^n_{R,i_2}) = \emptyset,$$

jer nijedna triangulacija u $(\mathcal{T}^n_{L,i_1} \otimes \{\{\delta_{i+1,n}\}\} \otimes \mathcal{T}^n_{R,i_1})$ ne sadrži dijagonalu $\delta_{i_2,n}$.

Kako je $T_2 = 1$, a imajući u vidu da prihvatamo T_{i+1} triangulaciju od $P^n_{L,i}$ i T_{n-i} triangulacije iz $P^n_{R,i}$, moguće je dobiti Segnerovu formulu

$$\begin{aligned} T_{n-1} + \sum_{i=2}^{n-2} T_{i+1}T_{n-i} &= T_{n-1}T_2 + \sum_{i=3}^{n-1} T_i T_{n-i+1} \\ &= \sum_{i=2}^{n-1} T_i T_{n-i+1} \\ &= T_n. \end{aligned}$$

Korišćenjem svojstva "*izbegavanja kolizije*", sve triangulacije su različite i njihov broj je $T_n = C_{n-2}$, što potvrđuje ispravnost našeg algoritma. □

Algoritam 2.2.1 Modifikovana Blok metoda

Ulaz: Pozitivan ceo broj n i skup triangulacija $\mathcal{T}_b, b < n$ (opisan unutrašnjim dijagonalama u formi $\delta_{i,j}$).

1: **Slučaj 1: n je uvo**
   Pročitaj skup triangulacija $\mathcal{T}_{n-1}$ sa ulaza.
   $\mathcal{T}_n^E := \mathcal{T}_{n-1} \otimes \{\{\delta_{1,n-1}\}\}$.
   $\mathcal{T}_n := \mathcal{T}_n^E$.

2: **Slučaj 2: n je teme unutrašnje dijagonale**
   for $i = 2$ **to** $n-2$ **do**
   Podeli poligon P_n pomoću $\delta_{i,n}$ na dva podpoligona $P_{L,i}^n$ i $P_{R,i}^n$.
   (a) $\mathcal{T}_{L,i}^n = \mathcal{T}(P_{L,i}^n) = \mathcal{T}_{i+1}\{i+1 \mapsto n\}$
   (b) $\mathcal{T}_{R,i}^n = \mathcal{T}_{n-i}\{j \mapsto j+i-1 : j \in \{1,2,\ldots,n-i\}\} \otimes \{\{\delta_{i,n-1}\}\}$
   (c) $\mathcal{T}_n := \mathcal{T}_n \cup \mathcal{T}_{L,i}^n \otimes \{\{\delta_{i,n}\}\} \otimes \mathcal{T}_{R,i}^n$

3: Vrati $\mathcal{T}_n$

Ilustrativni primeri

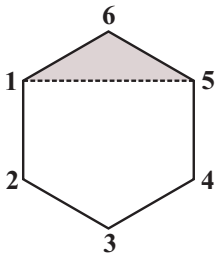
Primer 2.2.1 ilustruje operaciju $\otimes$ između dva skupa unutrašnjih dijagonala, dok primer 2.2.2 prezentuje postepenu ilustraciju Algoritma 2.2.1.

Primer 2.2.1. *Neka je skup* $\mathcal{A} = \{\{\delta_{1,3}\}, \{\delta_{2,7}\}\}$ *i skup* $\mathcal{B} = \{\{\delta_{3,7}, \delta_{3,5}\}, \{\delta_{3,7}, \delta_{4,6}\}\}$. *Onda sledi,*

$$\begin{aligned} \mathcal{A} \otimes \mathcal{B} &= \{\{\delta_{1,3}\}, \{\delta_{2,7}\}\} \otimes \{\{\delta_{3,7}, \delta_{3,5}\}, \{\delta_{3,7}, \delta_{4,6}\}\} \\ &= \{\{\delta_{1,3}\} \cup \{\delta_{3,7}, \delta_{3,5}\}, \{\delta_{1,3}\} \cup \{\delta_{3,7}, \delta_{4,6}\}, \\ &\quad \{\delta_{2,7}\} \cup \{\delta_{3,7}, \delta_{3,5}\}, \{\delta_{2,7}\} \cup \{\delta_{3,7}, \delta_{4,6}\}\} \\ &= \{\{\delta_{1,3}, \delta_{3,7}, \delta_{3,5}\}, \{\delta_{1,3}, \delta_{3,7}, \delta_{4,6}\}, \{\delta_{2,7}, \delta_{3,7}, \delta_{3,5}\}, \{\delta_{2,7}, \delta_{3,7}, \delta_{4,6}\}\}. \end{aligned}$$

Primer 2.2.2. Ilustrujmo Algoritam 2.2.1 na primeru šestougla (P_6). Mogu se uočiti dva važna slučaja.

Slučaj 1: Kada je n uvo



Slika 2.3: Šestougao i slučaj kad je n uvo

Kada je teme $n = 6$ uvo, izvodi se korak 1 Algoritma 2.2.1, gde se prihvata ceo skup $\mathcal{T}_5$ dodajući svakom njegovom elementu dijagonalu $\delta_{1,5}$. Na ovaj način dobijamo pet triangulacija od P_6, ili

$$\begin{aligned}\mathcal{T}_6^E &= \mathcal{T}_5 \otimes \{\{\delta_{1,5}\}\} \\ &= \{\{\delta_{1,3}, \delta_{1,4}, \delta_{1,5}\}, \{\delta_{2,4}, \delta_{1,4}, \delta_{1,5}\}, \{\delta_{2,5}, \delta_{2,4}, \delta_{1,5}\}, \\ &\quad \{\delta_{1,3}, \delta_{3,5}, \delta_{1,5}\}, \{\delta_{2,5}, \delta_{3,5}, \delta_{1,5}\}\}.\end{aligned}$$

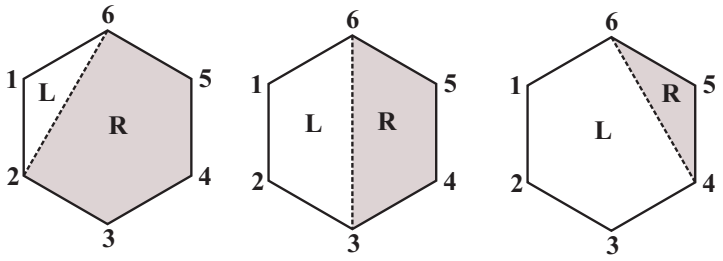
Ovaj korak je sažet u Tabeli 2.8.

Tabela 2.8: Slučaj 1: n je uvo

$\mathcal{T}_6$		
$\mathcal{T}_5$	Dodaj	$\mathcal{T}_6^E$
$\delta_{1,3}, \delta_{1,4}$	$\delta_{1,5}$	$\delta_{1,3}, \delta_{1,4}, \delta_{1,5}$
$\delta_{2,4}, \delta_{1,4}$	$\delta_{1,5}$	$\delta_{2,4}, \delta_{1,4}, \delta_{1,5}$
$\delta_{2,5}, \delta_{2,4}$	$\delta_{1,5}$	$\delta_{2,5}, \delta_{2,4}, \delta_{1,5}$
$\delta_{1,3}, \delta_{3,5}$	$\delta_{1,5}$	$\delta_{1,3}, \delta_{3,5}, \delta_{1,5}$
$\delta_{2,5}, \delta_{3,5}$	$\delta_{1,5}$	$\delta_{2,5}, \delta_{3,5}, \delta_{1,5}$

Slučaj 2: Kada je teme n jedna od krajnih tačaka unutrašnje dijagonale

Poligon P_6 se može podeliti povezivanjem temena $n = 6$ sa temenima iz skupa $\{2, ..., n-2\}$ ili u ovom konkretnom slučaju $\{2, 3, 4\}$. Dakle, možemo da podelimo poligon dijagonalama $\delta_{2,6}$, $\delta_{3,6}$ i $\delta_{4,6}$ (Slika 2.4).



Slika 2.4: Šestougao i slučaj kada je teme n deo unutrašnje dijagonale

U ovom slučaju, Korak 2 Algoritma 2.2.1 treba ponoviti $n - 3 = 3$ puta. Odgovarajuća obrada je predstavljena u posebnim delovima (A)-(C) ovog slučaja.

(A) Dijagonala podele $\delta_{2,6}$

Poligon $P^6_{L,2}$ je trougao, gde je $C(1,1) = 1$ i imamo samo jednu trivijalnu triangulaciju bez unutrašnjih dijagonala koje je opisuju. Dakle, prema Koraku 2(a) imamo $\mathcal{T}^6_{L,2} = \mathcal{T}_{2+1}\{2+1 \mapsto 6\} = \{\{\}\}\{3 \mapsto 6\} = \{\{\}\}$.

Poligon $P^6_{R,2}$ je petougao i na osnovu pravila "*izbegavanje sudara*" uzimamo samo prve $C(2,2) = 2$ triangulacija iz skupa $\mathcal{T}_5$ i pravimo mapiranje indeksa temena dobijajući $\mathcal{T}^6_{R,2}$ (Tabela 2.9).

$$\begin{aligned}
\mathcal{T}^6_{R,2} &= \mathcal{T}_{6-2}\{j \mapsto j+2-1 : j \in \{1,2,\ldots,6-2\}\} \otimes \{\{\delta_{2,6-1}\}\} \\
&= \mathcal{T}_4\{j \mapsto j+1 : j \in \{1,2,\ldots,4\}\} \otimes \{\{\delta_{2,5}\}\} \\
&= \{\{\delta_{1,3}\},\{\delta_{2,4}\}\}\{j \mapsto j+1 : j \in \{1,2,\ldots,4\}\} \otimes \{\{\delta_{2,5}\}\} \\
&= \{\{\delta_{2,4}\},\{\delta_{3,5}\}\} \otimes \{\{\delta_{2,5}\}\} \\
&= \{\{\delta_{2,4},\delta_{2,5}\},\{\delta_{3,5},\delta_{2,5}\}\}
\end{aligned}$$

Na kraju, Korak 2 (c) Algoritma 2.2.1 daje rezultate

$$\begin{aligned}
\mathcal{T}_6 &:= \mathcal{T}_6 \cup \mathcal{T}^6_{L,2} \otimes \{\{\delta_{2,6}\}\} \otimes \mathcal{T}^6_{R,2} \\
&= \{\{\delta_{1,3},\delta_{1,4},\delta_{1,5}\},\{\delta_{2,4},\delta_{1,4},\delta_{1,5}\},\{\delta_{2,5},\delta_{2,4},\delta_{1,5}\}, \\
&\quad \{\delta_{1,3},\delta_{3,5},\delta_{1,5}\},\{\delta_{2,5},\delta_{3,5},\delta_{1,5}\},\{\delta_{2,6},\delta_{2,4},\delta_{2,5}\}, \\
&\quad \{\delta_{2,6},\delta_{3,5},\delta_{2,5}\}\}.
\end{aligned}$$

Tabela 2.9: Mapiranje indeksa temena za poligon $P^6_{R,2}$

P_5	$\to \mathcal{T}^6_{R,2}$
$\{1,2,3,4,5\}$	$\to \{2,3,4,5,6\}$
$\delta_{1,3}, \delta_{1,4}$	$\to \delta_{2,4}, \delta_{2,5}$
$\delta_{2,4}, \delta_{1,4}$	$\to \delta_{3,5}, \delta_{2,5}$
$\delta_{2,5}, \delta_{2,4}$	
$\delta_{1,3}, \delta_{3,5}$	
$\delta_{2,5}, \delta_{3,5}$	

(B) Dijagonala podele $\delta_{3,6}$

Poligon $P^6_{L,3}$ je četvorougao, uzimamo sve $C(2,2) = 2$ triangulacije iz skupa $\mathcal{T}_4$. Nakon mapiranja indeksa temena dobijamo $\mathcal{T}^6_{L,3} = \mathcal{T}_{3+1}\{3+1 \mapsto 6\} = \mathcal{T}_4\{4 \mapsto 6\} = \{\{\delta_{1,3}\}, \{\delta_{2,6}\}\}$ (Tabela 2.10).

Tabela 2.10: Mapiranje indeksa temena za poligon $P^n_{L,3}$

P_4	$\to \mathcal{T}^6_{L,3}$
$\{1,2,3,4\}$	$\to \{1,2,3,6\}$
$\delta_{1,3}$	$\to \delta_{1,3}$
$\delta_{2,4}$	$\to \delta_{2,6}$

Poligon $P^6_{R,3}$ je četvorougao i na osnovu pravila "*izbegavanja sudara*" uzimamo samo prve $C(1,1) = 1$ triangulacije iz skupa $\mathcal{T}_4$, gde posle mapiranja indeksa temena, dobijamo $\mathcal{T}^6_{R,3}$ (Tabela 2.11).

Tabela 2.11: Mapiranje indeksa temena za poligon $P^6_{R,3}$

P_4	$\to \mathcal{T}^6_{R,3}$
$\{1,2,3,4\}$	$\to \{3,4,5,6\}$
$\delta_{1,3}$	$\to \delta_{3,5}$
$\delta_{2,4}$	

Na osnovu Koraka 2 (c), dobijamo
$\mathcal{T}^6_{L,3} \otimes \{\{\delta_{3,6}\}\} \otimes \mathcal{T}^6_{R,3} = \{\{\delta_{1,3}, \delta_{3,6}, \delta_{3,5}\}, \{\delta_{2,6}, \delta_{3,6}, \delta_{3,5}\}\}$.

(C) Dijagonala podele $\delta_{4,6}$

Poligon $P^6_{L,4}$ je petougao, uzimamo sve $C(3,3) = 5$ triangulacije iz skupa $\mathcal{T}_5$. Nakon mapiranja indeksa temena dobijamo:

$\mathcal{T}^6_{L,4} = \{\{\delta_{1,3}, \delta_{1,4}\}, \{\delta_{2,4}, \delta_{1,4}\}, \{\delta_{2,6}, \delta_{2,4}\}, \{\delta_{2,6}, \delta_{1,3}\}, \{\delta_{3,6}, \delta_{3,6}\}\}$ (Tabela 2.12).

Tabela 2.12: Mapiranje indeksa temena za poligon $P^6_{L,4}$

P_5	$\rightarrow \mathcal{T}^6_{L,4}$
$\{1,2,3,4,5\}$	$\rightarrow \{2,3,4,5,6\}$
$\delta_{1,3}, \delta_{1,4}$	$\rightarrow \delta_{1,3}, \delta_{1,4}$
$\delta_{2,4}, \delta_{1,4}$	$\rightarrow \delta_{2,4}, \delta_{1,4}$
$\delta_{2,5}, \delta_{2,4}$	$\rightarrow \delta_{2,6}, \delta_{2,4}$
$\delta_{1,3}, \delta_{3,5}$	$\rightarrow \delta_{1,3}, \delta_{3,6}$
$\delta_{2,5}, \delta_{3,5}$	$\rightarrow \delta_{2,6}, \delta_{3,6}$

Poligon $P^6_{R,4}$ je trougao, u ovom slučaju imamo samo jednu trivijalnu triangulaciju bez unutrašnjih dijagonala, tako da je $\mathcal{T}^6_{R,4} = \{\{\}\}$.

Prema tome, sledi, $\mathcal{T}^6_{L,4} \otimes \{\{\delta_{4,6}\}\} \otimes \mathcal{T}^6_{R,4} = \{\{\delta_{1,3}, \delta_{1,4}, \delta_{4,6}\}, \{\delta_{2,4}, \delta_{1,4}, \delta_{4,6}\}, \{\delta_{2,6}, \delta_{2,4}, \delta_{4,6}\}, \{\delta_{1,3}, \delta_{3,6}, \delta_{4,6}\}, \{\delta_{2,6}, \delta_{3,6}, \delta_{4,6}\}\}$.

Kao rezultat Modifikovane Blok metode, skup triangulacija poligona P_6 je:

$$\begin{aligned}\mathcal{T}_6 = \{&\{\delta_{1,3}, \delta_{1,4}, \delta_{1,5}\}, \{\delta_{2,4}, \delta_{1,4}, \delta_{1,5}\}, \{\delta_{2,5}, \delta_{2,4}, \delta_{1,5}\},\\ &\{\delta_{1,3}, \delta_{3,5}, \delta_{1,5}\}, \{\delta_{2,5}, \delta_{3,5}, \delta_{1,5}\}, \{\delta_{2,6}, \delta_{2,4}, \delta_{2,5}\},\\ &\{\delta_{2,6}, \delta_{3,5}, \delta_{2,5}\}, \{\delta_{1,3}, \delta_{3,6}, \delta_{3,5}\}, \{\delta_{2,6}, \delta_{3,6}, \delta_{2,5}\},\\ &\{\delta_{1,3}, \delta_{1,4}, \delta_{4,6}\}, \{\delta_{2,4}, \delta_{1,4}, \delta_{4,6}\}, \{\delta_{2,6}, \delta_{2,4}, \delta_{4,6}\},\\ &\{\delta_{1,3}, \delta_{3,6}, \delta_{4,6}\}, \{\delta_{2,6}, \delta_{3,6}, \delta_{4,6}\}\}.\end{aligned}$$

2.2.3 Komparativna analiza i eksperimentalni rezultati

Za implementaciju Originalne Blok metode i Modifikovane Blok metode korišćen je NetBeans IDE okruženje za Java programski jezik.

Vremena izvršavanja (u sekundama) za oba algortima su predstavljena u Tabeli 2.13. Kolona "Ubrzanje" u tabeli prikazuje količnik vrednosti sadržanih u prethodne dve kolone.

Testiranje je odrađeno u NetBeans testnom modulu "Profile Main Project / CPU Analyze Performance" na računaru sledećih karakteristika*: *CPU - Inter(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40GHz, RAM memorija 8GB, Grafička karta: NVIDIA GeForce 820M.*

Tabela 2.13: Eksperimentalni rezultati testiranja: Blok metoda vs Modifikovana Blok metoda.

n	**Broj triangulacija**	**Originalna Blok metoda**	**Modifikovana Blok metoda**	**Ubrzanje**
5	5	0.003	0.000	-
6	14	0.006	0.000	-
7	42	0.009	0.000	-
8	132	0.013	0.000	-
9	429	0.016	0.000	-
10	1,430	0.056	0.000	-
11	4,862	0.181	0.000	-
12	16,796	0.665	0.016	41.56
13	58,786	2.555	0.020	127.75
14	208,012	10.429	0.075	139.05
15	742,900	66.102	0.343	192.72
16	2,674,440	-	1.152	-
17	9,694,845	-	7.466	-

Efikasnost algoritama za triangulaciju konveksnih poligona se ogleda u načinu na koji tretiraju problem pojavljivanja duplih triangulacija koje se eksponencijalno povećavaju sa veličinom poligona.

Rezultati prikazani u Tabeli 2.13 potvrđuju efikasnost Modifikovane Blok metode u generisanju triangulacija i eliminisanju pojavljivanja duplikata. Tačnije, Modifikovana Blok metoda daje bolje rezultate u odnosu na originalnu Blok metodu.

Poglavlje 3

Metode za pronalaženje optimalnih triangulacija

U ovom poglavlju je tretiran problem pronalaženja optimalne (minimalne težinske) triangulacije.

U prvom delu poglavlja, predstavljena je nova metoda za pronalaženje i skladištenje optimalne triangulacije. Metoda se zasniva na funkcionalnosti kvadratne matrice (SM) u kojoj skladištimo sve rezultate triangulacija za dati n. Glavni akcenat metode je na brzini generisanja optimalne triangulacije i uštedi memorijskog prostora prilikom izračunavanja velikog broja triangulacija. Metoda skladištenja podataka je konstruisana na ideji da nije neophodno posebno skladištiti svaku težinu triangulacije, već da sve to izvodimo odjednom.

U drugom delu poglavlja, predstavljen je postupak pronalaženja optimalne triangulacije, koji se zasniva na autorskoj metodi za generisanje triangulacija (Blok metode). Analizirana su dva slučaja izračunavanja težina triangulacije (klasični slučaj i slučaj zasnovan na Blok metodi). Značaj prikazanog postupka se ogleda u činjenici da korišćenje skladištenih vrednosti može biti veoma efikasan način za pronalaženje optimalne triangulacije. Osnovni cilj predloženog postupka je brzina dobijanja optimalne triangulacije.

Za potrebe dobijanja eksperimentalnih rezultata, za oba slučaja, odrađena je implementacija u Java programskom jeziku. Nakon toga, urađena je komparativna analiza predloženih metoda sa Hurtado-Noy algoritmom, gde je na osnovu dobijenih rezultata ukazano na prednosti novih metoda. Naučni radovi na kojima se zasniva ovo poglavlje su [29, 38].

3.1 Metoda za pronalaženje i čuvanje optimalnih triangulacija na osnovu kvadratne matrice

Proces pronalaženja optimalnih triangulacija je procedura u računarskoj geometriji i računarstvu, koja zahteva određenu količinu vremena za izvršenje, kao i relativno velike zahteve za skladištenje podataka. Prema tome, važno je obezbediti efikasan način za brže generisanje optimalnih triangulacija i njihovo sladištenje tako da zauzimaju što manje memorijskog prostora. U skladu sa tim, u nastavku je predstavljen novi metod za pronalaženje optimalnih triangulacija, koji se zasniva na osnovu funkcionalnosti kvadratne matrice (SM) u kojoj skladištimo sve rezultate triangulacija za dati n.

Osnovna ideja za implementacijom ove metode je proizašla iz naprednih mogućnosti *Java* programskog jezika, koje se odnose na primenu paketa za rad sa ćelijama (poljima) tabela. Glavnu ulogu u skladištenju težina triangulacija ima *Java* paket `DefaultTableCellRenderer`[1], koji omogućava skladištenje više vrednosti u jednom polju tabele (nudi efikasnu podelu polja na više kolona i redova).

Glavni akcenat metode je na brzini generisanja optimalne triangulacije i uštedi memorijskog prostora prilikom izračunavanja velikog broja triangulacija. Metoda skladištenja podataka je konstruisana na ideji da nije neophodno posebno skladištiti svaku težinu triangulacije, već da sve to izvodimo odjednom.

Rešavanje problema određivanja minimalnih težinskih triangulacija se bazira na tehnikama dinamičkog programiranja. Pored brzog izračunavanja optimalnih triangulacija, jedan od ciljeva u implementaciji je i ušteda memorijskog prostora prilikom proračuna velikog broja triangulacija.

Metoda je konstruisana na osnovu različitih načina množenja matrica sa zagradama (optimalno planiranje množenja matrica) i zakona asocijativnosti $A \cdot (B \cdot C) = (A \cdot B) \cdot C$ za množenje matrica.

Pretpostavimo da imamo sekvencu $A_1, A_2, \ldots, A_n$ od n matrica koje se množe i neka je $P(n)$ broj različitih zagrada za ovo množenje. Ovaj proizvod od n matrica možemo podeliti između k-te i $(k+1)$ matrice za bilo koje $k = 1, 2, \ldots, n-1$ i označiti sa zagradama dve rezultujuće sekvence nezavisno kao $((A_1 A_2, \ldots, A_k)(A_{k+1} A_{k+2}, \ldots, A_n))$, ponavljanje (3.1) se dobija na ovaj način:

$$P(n) = \begin{cases} 1, & n = 1 \\ \sum_{k}^{n-1} P(k) P(n-k), & n \geq 2. \end{cases} \tag{3.1}$$

Ovo ponavljanje generiše sekvencu Katalanovih brojeva i zadovoljava $P(n) = C_{n-1}$. Optimalna triangulacija konveksnog poligona konstruisana je kao problem dinamičkog programiranja i sastoji se od sledećih koraka:

1. obeležavanje ivica sa matricom $A_i, i = 1, 2, \ldots, n$ $(A_{n\times 1})$ neparnih ivica i $A_{1\times n}$ ravnih ivica;

[1] https://docs.oracle.com/javase/7/docs/api/javax/swing/table/DefaultTableCellRenderer.html

2. podela problema na potprobleme koristeći optimalnu podelu;
3. zagraditi potprobleme optimalno;
4. kombinacija optimalnih rešenja potproblema.

Označimo težinu minimalne triangulacije konveksnog poligona kao $m[1,n]$. Težina svakog trougla u triangulaciji je dužina njegovog obima i neka $w(i,j,k)$ označava dužinu obima od $\Delta v_i v_j v_k$.

Neka konveksni poligon sadrži podpoligon sa i temena kroz k $(2 \leq k \leq i)$, i označimo težinu minimalne triangulacje takvog potpoligona kao matricu $m[i,j]$ (je minimalni broj skalarnih množenja potrebnih za izračunanje $A_{i\ldots j} = A_i \cdot A_{i+1} \cdots A_j$). Moguće vrednosti matrice $m[i,k]$ spadaju u dva slučaja:

- **Slučaj 1**: ako je $i = j$ onda je $m[i,j] = m[i,i] = 0$, $A_i = i$ i množenje matrica nije potrebno.
- **Slučaj 2**: ako je $i < j$, onda se pretpostavlja optimalna podela na k, $i \geq k < j$. U ovom slučaju $A_{i,k}$ i $A_{k+1,j}$ imaju dimenziju $v_{i-1} \times v_k$, $v_k \times v_j$ respektivno. Za svaki izbor j se izračunava.

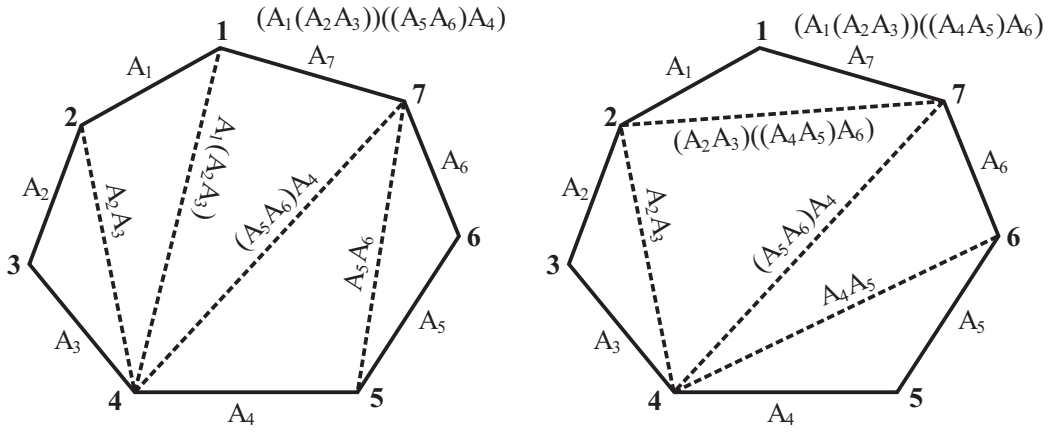
$$m[i,j] = m[i,k] + m[k+1,j] + w(i,j,k) \quad for \quad k = i, i+1, i+2, \ldots, j-1. \tag{3.2}$$

Ova dva slučaja daju sledeću rekurzivnu formulu (3.3):

$$m\,[i,\ j] = \begin{cases} 0 & if\ i = j \\ \min_{i \leq k < j} m\,[i,\ k] + \ m\,[k+1,\ j] + \ w(i,j,k) & if\ i < j \end{cases} \tag{3.3}$$

Konveksni poligon $v_0 v_1 \ldots v_{n-1}$ sa $(n-3)$ nepresećajućih dijagonala je podeljen na $(n-2)$-trougla. U većini slučajeva, moguće je pronaći kriterijum za izračunavanje težine triangulacije, tj. vrednosti obima, sumu visina ili dužine najdužeg medijana prema broju trouglova [47]. Optimalna triangulacija se može izračunati pomoću rekurentne formule (3.3). U nastavku ćemo opisati algoritam za izračunavanje težine (w) trougla $w(i,j,k)$.

U cilju ilustracije, posmatrajte dve triangulacije konveksnog sedmougla, kao što je prikazano na Slici 3.1.



Slika 3.1: Dve triangulacije konveksnog sedmougla sa pripadajućim težinama

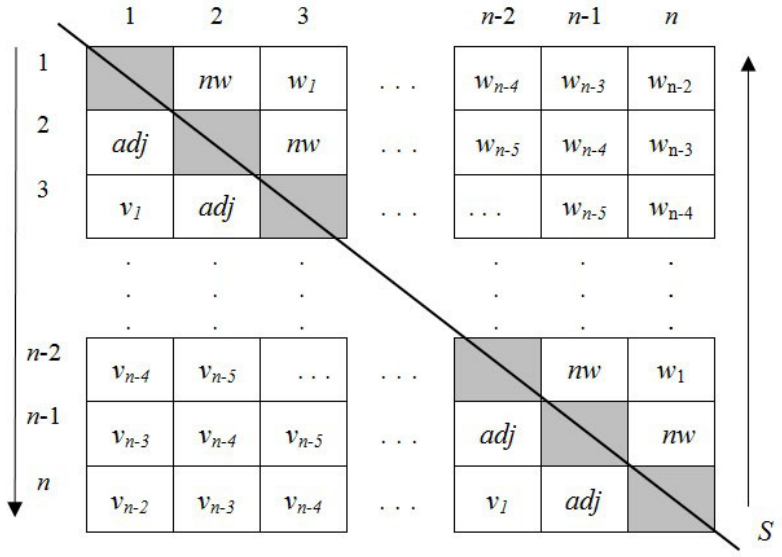
3.1.1 Skladištenje težina na osnovu kvadratne matrice (SM model)

Metoda za skladištenje težina triangulacija se zasniva na kvadratnoj matrici ($i = j = n$) sa dimenzijama ($n \times n$), gde i označava broj redova, j predstavlja kolone, a n broj temena poligona. Matrica $\mathcal{A}_{ij}$ se koristi za organizaciju k-temena trouglova (i, j, k) i prezentaciju njihovih težina (w)..

U našem slučaju, matrica A se može trajno konvertovati u tabelu pomoću `Java JDBC API`. Matrica je dijagonalno podeljena na dva dela. Dijagonala $i = j$ deli matricu na levi i desni deo. Pozicije $(1, 1), (2, 2), (3, 3), \ldots, (n, n)$ su popunjene nulama (videti Sliku 3.2).

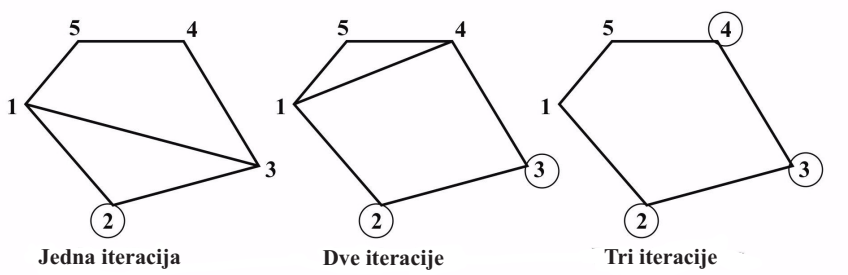
Temena trouglova se skladište na levoj strani matrice dok se njihove izračunate težine (3.3) skladište na desnoj strani:

$$k[i, j] = w[j, i]$$



Slika 3.2: Generalna šema popunjavanja matrice

Primer 3.1.1. Ovaj primer prikazuje koliko k vrednosti postoji u skupu v_k za petougao (Slika 3.3).



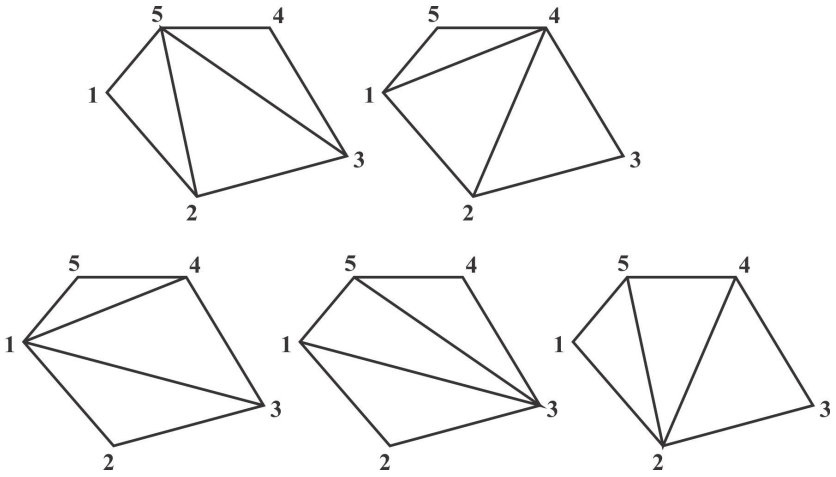
Slika 3.3: Primer povećanja broja k-temena u kvadratnoj matrici

Ne postoji teme k između temena 1 i 2, zato što su susedni (označeni sa adj), tako da nema druge težine na drugoj strani dijagonale (označena sa nw, Slika 3.2). Tačnije, ne postoji trougao (i, j, k) koji ima definisanu težinu.

Broj temena između (i, j) se povećava sa povećanjem broja i -redova. Na primer, između temena 1 i 3 postoji samo jedno teme $k = 2$, zatim, postoje dva temena $k = 2, 3$ između temena 1 i 4, postoje tri temena $k = 2, 3, 4$ između temena 1 i 5, itd.

Kako se rastojanje između temena (i, j) povećava, tako se povećava i broj mogućih k-temena unutar skupa v i broj težina (w) na drugoj strani dijagonale (j, i).

Na osnovu $P(n) = C_{n-1}$ za $n = 5$ (petougao) dobijamo pet različitih triangulacija. Postoji samo jedna univerzalna matrica (5×5) za svih pet triangulacija kao i sve težine trouglova neophodne za određivanje minimalne težine za svaku triangulaciju odvojeno.



Slika 3.4: Triangulacije petougla

U nastavku je predstavljena matrica petougla 5×5, na osnovu generalne šeme za popunjavanje matrice.

$$opt5 = \begin{bmatrix} 0 & \{nw\} & \{25\} & \{25, 30\} & \{16, 80, 35, 60\} \\ \{adj\} & 0 & \{nw\} & \{10, 30\} & \{35, 11, 15\} \\ \{2\} & \{adj\} & 0 & \{nw\} & \{30\} \\ \{2, 3\} & \{3\} & \{adj\} & 0 & \{nw\} \\ \{2, 3, 4\} & \{3, 4\} & \{4\} & \{adj\} & 0 \end{bmatrix}$$

Metoda za skladištenje težina je kombinovana sa našom metodom za generisanje triangulacija [34].

U nastavku predstavljamo Algoritam 3.1.1 za skladištenje težina triangulacija i pronalaženje optimalne triangulacije. Na početku, algoritam očekuje broj temena poligona n, gde svako teme dobija sopstvene koordinate. Tabela se proširuje novom kolonom koja skladišti težine triangulacija trenutnog procesa.

Algoritam 3.1.1 sadrži tri glavna koraka:

1. Formira se kvadratna matrica $i \times j$ i popunjavaju se polja duž dijagonale sa vrednostima 0 (gde je $i = j$). Zatim se upisuje vrednost adj na poziciji (i, j) i vrednost nw na poziciji (j, i) u odnosu na prvu dijagonalnu liniju (videti Sliku 3.2). To su u pitanju susedna temena gde ne postoje k-vrednosti za njih, koje se mogu koristiti za formiranje trougla (i, j, k). Ta polja imaju vrednost 0 i nakon toga popunjavaju se sve moguće vrednosti za k između (i, j).
2. Izračunavanje svih težinskih triangulacija (w) za redove i kolone matrice gde je $(j-i) > 1$ na osnovu formule (3.1) i nastavlja se na sledeći korak. Dodaju se odgovarajuće težine za dodeljene k vrednosti na pozicijama (j, i).
3. Dobijene težine se dodeljuju odgovarajućoj triangulaciji poligona.

Algoritam 3.1.1 Algoritam za skladištenje težina i pronalaženje optimalne triangulacije.

Ulaz: n, tabela T_n

1: Kreiranje nove matrice $(i \times j)$
for $(i = 1; i \leq n; i++)$
$v[i, j] = 0$, gde je $i = j$
$v[i, j] = adj$, gde je $j - i = 1$

2: Popunjavanje ostalih polja
for$(i = 1; i \leq n; i++)$
for$(j = 2; j \leq n; j++)$
$j = i + 1$
for$(k = i + 1; i \leq j - 1; k++)$
$m[i, j] = min\{m[i, k] + m[k + 1, j] + w(i, j, k)\}$
$W[T_i] = sumAll[m[i, j]]$
if $W[T_i] < m[i, j]$ then
$W[T_i] = m[i, j]$
return $m[i, j]$

3: Selektovanje $(n - 2)$ trouglova (i, j, k) iz matrice i identifikacija u poligonu T_n
Evaluacija $Opt[w] = minW[T_i]$

Primer 3.1.2. Dobijanje optimalne triangulacije za nepravilni konveksni petougao je pojašnjen u naredna tri koraka.

Korak 1: Formira se matrica 5×5, dijagonalno se popunjavaju polja nulama i vrednostima za susedna temena. Posle toga, popunjavaju se vrednosti za sve trouglove (i, j, k). Nakon toga matrica poprima sledeći oblik:

$$opt5 = \begin{bmatrix} 0 & \{nw\} & \{35\} & \{35\} & \{16\} \\ \{adj\} & 0 & \{nw\} & \{10\} & \{11\} \\ \{2\} & \{adj\} & 0 & \{nw\} & \{30\} \\ \{\langle 2\rangle, 3\} & \{\langle 3\rangle\} & \{adj\} & 0 & \{nw\} \\ \{2, 3, \langle 4\rangle\} & \{3, 4\} & \{4\} & \{adj\} & 0 \end{bmatrix}$$

Korak 2: Izračunava se suma težina (w) za C_{n-2} redova za datu tabelu T_5. Tabela je kreirana uz pomoć algoritma za generisanje triangulacija koji je opisan detaljnije u radu [34].

Tabela za skladištenje triangulacija se proširuje novom kolonom W (Tabela 3.1).

Tabela 3.1: Proširena tabela $\mathcal{T}_5$

i	D_1	D_2	$\boldsymbol{W}$
1	$\delta_{1,3}$	$\delta_{1,4}$	76
2	$\delta_{1,3}$	$\delta_{3,5}$	71
3	$\delta_{2,4}$	$\delta_{1,4}$	61
4	$\delta_{2,4}$	$\delta_{2,5}$	37
5	$\delta_{2,5}$	$\delta_{3,5}$	51

Sume svih težina trouglova se čuvaju u novoj koloni W za $[T_i]$, gde je i trenutni red.

Dijagonale u redovima formiraju $(n-2)$-trouglova i za svaki trougao, uzimaju se težine iz matrice A:

$$W[T_1] = 25 + 35 + 16 = 76$$
$$W[T_2] = 16 + 25 + 30 = 71$$
$$W[T_3] = 10 + 35 + 16 = 61$$
$$\cdots$$

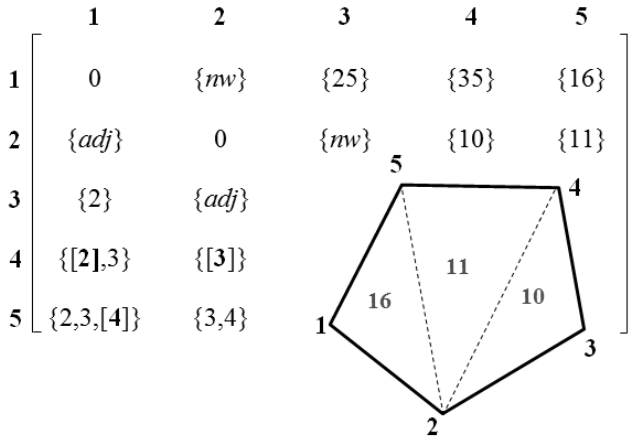
Korak 3: U ovom koraku se izračunava najniža suma u redu (optimalna triangulacija):

$$OptT_w = min\{W[T_1], \ldots, W[T_i]\} \tag{3.4}$$

gde je $1 \leq i \leq T_n$.

Na osnovu formule (3.4) dobija se optimalna triangulacija $OptT_w = 37$.

Nakon pronalaženja optimalne triangulacije iz prethodne tabele, sledi crtanje iste na osnovu unutrašnjih dijagonala $\delta_{i,j}$ koji odgovaraju vrednostima iz reda i kolone u matrici, tj. paru (i, j): $optT = \{\delta_{2,4}, \delta_{2,5}\}$ (Slika 3.5).



Slika 3.5: Odgovarajuće vrednosti u matrici i optimalna triangulacija

3.1.2 Komparativna analiza izračunavanja težina

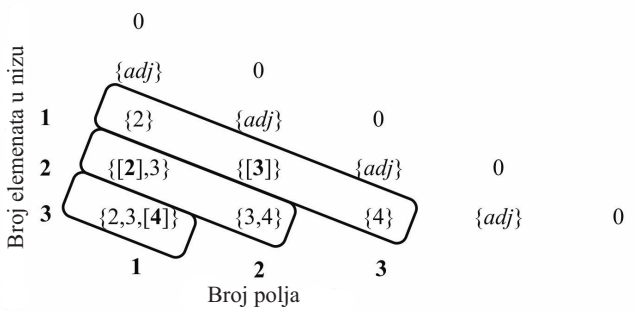
Sada pogledajmo uštede, u smislu opterećivanja računarskih resursa prilikom pronalaženja težine svake triangulacije, posebno kada se radi o Hurtado-Noy metodi.

U najgorem slučaju, ukupan broj izračunavanja težina (sa ponavljanjem) je rezultat ukupnog broja triangulacija n-tougla i broja njegovih dijagonala:

$$TM_w = T_n \times (n-2) \tag{3.5}$$

Ako želimo da izračunamo težine triangulacije pentagona, na osnovu formule (3.5), imamo ukupno 15 kalkulacija (5 različitih triangulacija sa po tri trougla).

Sada je potrebno da nađemo koje su uštede u smislu izračunavanja težine trougla (i, j, k). Na početku određujemo koliko k-vrednosti ima u matrici. Na osnovu generalne šeme punjenja matrice (Slika 3.2), broj k-temena se povećava kako idemo dalje od dijagonale. Prvi red ne sadrži elemente (jer oni predstavljaju tzv. susedne dijagonale), drugi ima jedan element, treći ima dva i tako dalje. Poslednji dijagonalni red sadrži $(n-2)$ k-temena jer je razlika između prvog i poslednjeg n-temena uvek dva, jer se prvo i poslednje teme uvek oduzimaju od ukupnog broja temena (Slika 3.6).



Slika 3.6: Primer izračunavanja ukupnog broja k-temena

Ako označimo ukupan broj uskladištenih k-temena za poligon sa n-temena sa V_n onda se broj trouglova (i, j, k) ili k-temena koje su unete u tabelu, dobija iz sledeće jednačine:

$$\begin{aligned} V_n &= \sum_{i=1}^{n-2} i((n-2)-(i-1)) \\ V_n &= \sum_{i=1}^{n-2} i(n-i-1) \end{aligned} \tag{3.6}$$

Dekompozicijom formule 3.6 možemo predstaviti vrednost V_n kao sumu broja polja u redovima i broj sekvencioniranih elemenata tog polja.

Primer 3.1.3. U ovom primeru se izračunava broj k-temena za $n = \{5, 6, 7\}$, koji su označeni sa V_5, V_6 i V_7:

$$\begin{aligned} V_5 &= (1 \times 3) + (2 \times 2) + (3 \times 1) = 10 \ \text{ (popunjeno ukupno od 10 k-vrednosti)} \\ V_6 &= (1 \times 4) + (2 \times 3) + (3 \times 2) + (4 \times 1) = 20 \\ V_7 &= (1 \times 5) + (2 \times 4) + (3 \times 3) + (4 \times 2) + (5 \times 1) = 35 \end{aligned}$$

Za triangulacije petougla (Slika 3.4), nalazi se 10 dijagonala uključujući ponavljanje (5 različitih triangulacija koje sadrže po 2 dijagonale), prikazane u obliku kako sledi: $\delta_{2,5}\delta_{3,5}, \delta_{2,4}\delta_{1,4}, \delta_{1,3}\delta_{1,4}, \delta_{1,3}\delta_{3,5}$ i $\delta_{2,4}, \delta_{2,5}$. Na osnovu formule (3.6) sledi da je broj izračunatih težina jednak broju trouglova:

$$SM_w = \sum_{i=1}^{n-2} i(n-i-1) \tag{3.7}$$

Prema tome, imamo deset različitih trouglova u triangulaciji za petougao, predstavljen u Primeru 3.1.3. Koristeći tradicionalno izračunavanje za sve triangulacije petougla zasebno, tj. pomoću $T_n \times (n-2)$ dobijamo 15 trouglova (uključujući ponavljanje) za sve triangulacije pentagona. Na osnovu $TM_w - SM_w$ sačuvali smo 5 kalkulacija.

Za drugo poređenje, sa našom SM metodom, uzet ćemo dobro poznati algoritam dat u radu [11]. Hurtardo i Noy su predložili algoritam za generisanje triangulacije poligona P_n na osnovu triangulacije P_{n-1}. Štaviše, definisali su stablo triangulacija gde su sve triangulacije P_n, tj. triangulacije iz $\mathcal{T}_n$, raspoređene na n-tom nivou ovog stabla (na osnovu ovog stabla, može se izračunati broj težina u skladištu).

Svaka triangulacija na novou n ima "oca" u skupu $\mathcal{T}_{n-1}$ i dva ili više "sinova" u skupu $\mathcal{T}_{n+1}$. Sinovi od istg oca su "braća". Postoji poređenje među decom triangulacije, a samim tim i među svim triangulacijama. U slučaju Hurtado-Noy hijerarhije (algoritma), imamo sledeće. Za svaku triangulaciju na nivou $n-1$ potrebno je da se izvrši $2n-5$ provera da bi pronašli dijagonale koje su povezane sa temenom $n-1$. Ukupan broj ovih provera je jednak $(2n-5)C_{n-3}$.

Dalje, moramo proći kroz dijagonale i kopirati neke bez transformacije, a neke od njih potrebno je transformisati i umetnuti dve nove dijagonala, za svaku incidentnu dijagonalu koja se pronađe. Na taj način pravimo $2n-3$ parova koji predstavljaju jednu novu triangulaciju. Ukupan broj incidentnih dijagonala je jednak C_{n-2}. Sve zajedno, u slučaju Hurtado-Noy algoritma, potreban je

$$HN_w = (2n-5)C_{n-3} + (2n-3)C_{n-2}$$

broj izračunavanja težina.

Kako se broj temena povećava, tako se i broj ponavljanja izračunavanja drastično povećava (takođe, ubrzanje i uštede drastično se povećavaju, vidi Tabelu 3.2).

Tabela 3.2: Komparativna analiza izračunavanja težina

$\boldsymbol{n}$	TM_w	HN_w	SM_w	D	R
7	210	45	35	10	**1.29**
8	792	161	56	105	**2.88**
9	3003	588	84	504	**7.00**
10	11440	2178	120	2058	**18.15**
11	43758	8151	165	7986	**49.40**
12	167960	30745	220	30525	**139.75**
13	646646	116688	286	116402	**408.00**
14	2496144	445094	364	444730	**1222.79**

TM_w – Ukupan broj izračunavanja težina (sa ponavljanjem) ($T_n \times (n-2)$)
HN_w – Broj izračunavanja na osnovu Hurtado-Noy hijerarhije
SM_w – Broj izračunavanja na osnovu SM metode (3.7))
D – Razlika između HN_w i SM_w (**ušteda**, SM-HN)
R – Odnos između HN_w i SM_w (**ubrzanje**, SM/HN)

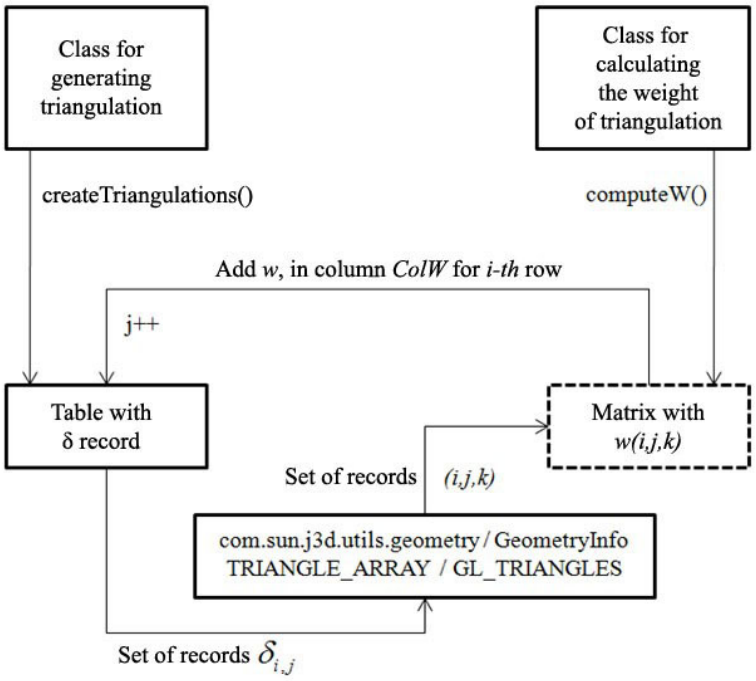
Metoda razvijena u ovom istraživanju izbegava iste proračune i daje pravilo za izračunavanje istih trouglova (i, j, k). Kod tradicionalnih metoda dolazi do značajnog porasta istog izračunavanja koji posebno dolazi do izražaja kada broj temena poraste iznad $n > 7$ nasuprot predloženoj `SM metodi`.

3.1.3 Implementacija i eksperimentalni rezultati

Java NetBeans okruženje [9] ima paket `OptimalTriangulation` koja radi kao dopuna aplikaciji koja generiše triangulacije konveksnog poligona. Glavna klasa deluje preko metode (`compute()`), koja vrši izračunava svih težina unutar tabele T_n.

Da bi radili sa ćelijama tabele, koristi se `DefaultTableCellRendererclass` iz `Swing` paketa. Ova klasa nasleđuje klasu `Table` i omogućava manipulaciju ćelija tabele (u ovom

slučaju, to nam omogućava da dodeljujemo broj nezavisnih vrednosti jednoj ćeliji tabele). Primena paketa `Geometry` i `GeometryInfo` klase [25, 1] u procesu kombinovanja metoda za pronalaženje optimalne triangulacije i generisanja triangulacije prikazana je na Slici 3.2. `GeometryInfo` klasa i njena metoda `TRIANGLE_ARRAY` obuhvata skup od tri temena koji formiraju trougao (i, j, k) gde se upotrebom metode `GL_TRIANGLES` formiraju trouglovi.



Slika 3.7: Postupak skladištenja triangulacija

U cilju dobijanja eksperimentalnih rezultata `SM` metode, u nastavku su predstavljeni ključni elementi implementacije u Java okruženju. Kao rezultat primene, dobija se tabela uskladištenih težina koja odgovara svim mogućim triangulacijama određenog poligona.

Aplikacija daje opciju grafičkog prikaza optimalne triangulacije i izvodi se kao što je opisano u Algoritmu 3.1.1:

1. Težine se izračunavaju pritiskom na dugme "*CALCULATE WEIGHTS*".
2. Nakon uspešnog izračunavanja sledi provera JDBC konekcije.
3. Kada se prikaže poruka za uspešnu konekciju, poziva se metoda za generisanje svih triangulacija i formira se odgovarajuća tabela T_n. Pronalaženje tabele T_n je jednostavno (broj n dobija istu vrednost kao i broj vertikala nacrtanih na *JPanelapplication*).
4. Po otkrivanju adekvatne tabele T_n (ako postoji), nove kolone za težinu (w) će se popunjavati jedna po jedna za svaku triangulaciju (tj. red po red). Popunjavanje nove kolone znači otkrivanje (i, j, k) vrednosti matrice na osnovu dijagonala $(\delta_{i,j})$ koje su već u tabeli T_n.

5. Kada se izračunaju sve težine za triangulacije, korisnik će dobiti izlaznu poruku "status-OK" za svaku triangulaciju.
6. Klikom na "*Show Optimal Triangulation*" JPanel prikazuje optimalnu triangulaciju.

Važnost skladištenja na predloženi način odražava se u činjenici da rezultati korišćenja ove metode mogu na efikasan način prikazati optimalnu triangulaciju pod uslovom da postoji metoda generisanja svih triangulacija.

Tabela 3.3 prezentuje rezultate za skup $n \in \{5, 6, \ldots, 14\}$ (Vremena su izražena u sekundama). Tabela prikazuje vreme potrebno za izračunavanje težina (*korak 1 i 2, bez grafičkog generisanja optimalne triangulacije*) i ukupno vreme za pronalaženje optimalne triangulacije (*3 i 4, sa grafičkom prezentacijom*).

Tabela 3.3: Eksperimentali rezultati testiranja metode za pronalaženje i čuvanje optimalnih triangulacija

n	***Broj triangulacija***	***Popunjavanje matrice***	***Ukupno vreme izvršavanja***
5	5	0.01	1.7
6	14	0.03	2.4
7	42	0.09	2.8
8	132	0.12	3.7
9	429	0.19	5.2
10	1430	0.27	14.8
11	4,862	0.41	27.4
12	16,796	0.62	48.7
13	58,786	1.04	64.6
14	208,012	1.59	85.5

Testiranje je izvršeno na računaru sa sledećim tehničkim specifikacijama: *CPU Intel Core2 Duo, 2.40GHz,Cache 4MB, RAM: 2Gb, Graphic: NVIDIA GeForce 8600M GS.*

Značaj predložene metode ogleda se u tome što je korišćenje skladištenih vrednosti veoma efikasan način za pronalaženje optimalne triangulacije, što se može zaključiti na osnovu dobijenih rezultata koji su predstavljeni u Tabeli 3.3.

3.2 Pronalaženje optimalnih triangulacija na osnovu Blok metode

U ovom delu poglavlja je predstavljen novi predlog u pronalaženju optimalne triangulacije koja se zasniva na našoj autorskoj metodi za generisanje triangulacija (Blok metoda [33]). Predstavljena su dva slučaja izračunavanja težina triangulacije (klasični slučaj i slučaj zasnovan na Blok metodi). Takođe, predstavili smo njihovu jednakost i uspostavljenu vezu u izračunavanju težina za oba modela, s naglaskom na jednostavnost proračuna koja se javlja u drugom slučaju. Osnovni cilj predložene metode je brzina dobijanja optimalne triangulacije.

Značaj prikazane metode ogleda se u činjenici da korišćenje skladištenih vrednosti može biti veoma efikasan način za pronalaženje optimalne triangulacije. To znači da se u pronalaženju dozvoljene triangulacije (kombinacija unutrašnjih dijagonala koje se ne seku), direktno se skladište vrednosti za temena i njihove težine. Ovo je pogodno za brzo iscrtavanje optimalne triangulacije nepravilnog konveksnog poligona.

3.2.1 Algoritam za Blok metodu

Blok metoda se zasniva na prepoznavanju triangulacije poligona sa manjim brojem temena (blokova) u skupu temena koji odgovara poligonu sa većim brojem temena. Tačnije, ovaj metod se zasniva na korišćenju već generisanih triangulacija koje su sačuvane u formi unutrašnjih dijagonala ($\delta_{i,j}$).

Jedna od prednosti Blok metode se ogleda u korišćenju već dobijenih triangulacija poligona P_{n-1} za dobijanje triangulacija poligona P_n, dok je za pronalaženje preostalih unutrašnjih dijagonala razvijen čitav set pravilnosti. Termin blok predstavlja skup svih triangulacija za dati poligon P_{n-1}, koje se pojavljuju dva puta u celini u skupu triangulacija za poligon P_n. Na ovaj način izbegava se ponovno generisanje iste triangulacije (tehnika rekurzije sa memoizacijom).

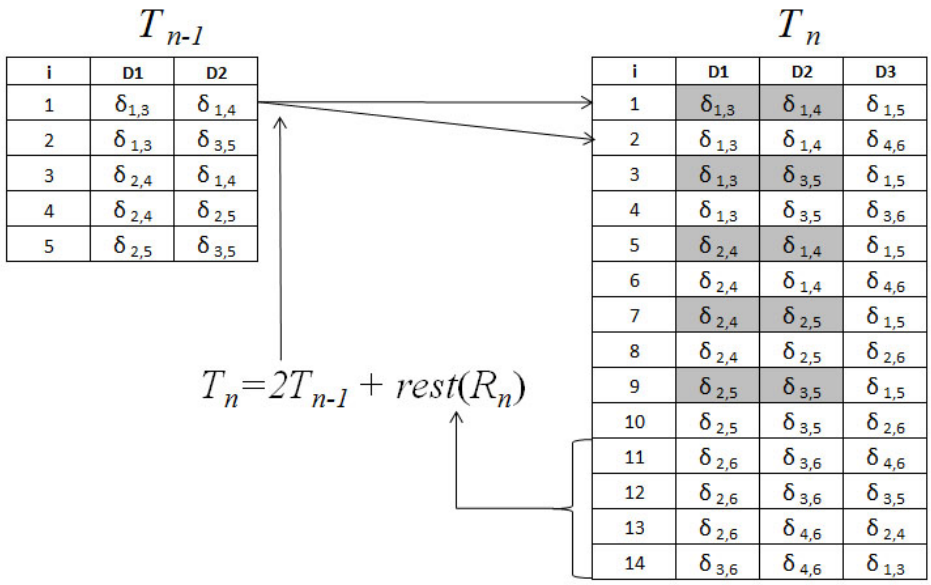
Algoritam koji realizuje Blok metodu može se podeliti u četiri faze:

1. U prvoj fazi ove metode transponuju se svi zapisi triangulacije manjeg poligona (korak 1 u Algoritmu 4 [33]).
2. U drugoj fazi, transponovani zapisi poligona P_{n-1} se snimaju u privremenu bazu podataka gde je planirano skladištenje zapisa triangulacije poligona P_n (korak 2).
3. U sledećoj fazi, na sve zapise se dodaje jedan dodatni zapis za dodatnu unutrašnju dijagonalu (korak 3). U ovom koraku pozivamo Algoritam 3 [33], koji je odgovoran za pronalaženje preostalih dozvoljenih temena, koji takođe poziva dodatni Algoritam 2 [33] za eliminaciju svih zabranjenih (zatvorenih) temena (definicija zabranjenog temena data je u radu [33]).
4. U poslednjoj fazi Blok metode nalazimo sve preostale triangulacije koje imaju najmanje dve unutrašnje dijagonale koje sadrže poslednje teme n (korak 4.1). U

ovom segmentu još jednom pozivamo dodatni Algoritam 3 [33], da bismo pronašli preostala dozvoljena temena (korak 4.2).

Obzirom da Blok metoda radi sa bazama podataka, ovde se podsećamo nekih detalja o skladištenju: (1) Svi rezultati za blokove (bazu) se snimaju u `.db` formatu; (2) Svaki blok se snima u obliku `[`T_n`_`*base*`].db`; (3) Baza podataka sadrži skup unutrašnjih dijagonala za svaku liniju pojedinačno, što definiše triangulaciju poligona.

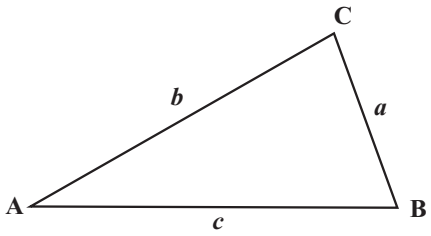
Originalna šema Blok metode je data na sledećoj Slici.



Slika 3.8: Postupak generisanja triangulacija na osnovu Blok metode

3.2.2 Izračunavanje optimalne triangulacije i primena Blok metode

Osnovu svih triangulacija čini trougao, tako da ovde razmatramo kako pronaći obim trougla (P), kao jednu od mera u pronalaženju optimalne triangulacije.



Slika 3.9: Trougao $P_{\triangle ABC} = a + b + c$

Pre izračunavanja obima za sve trouglove prvo treba da dobijemo rastojanje između dva temena (dužine stranica) koristeći sledeću jednačinu:

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \tag{3.8}$$

Napomena 3.2.1. *Svaka tačka u ravni je definisana koordinatama x, y*

U postupku pronalaženja optimalne triangulacije koristi se ukupna suma obima trouglova, izražena u težinama (W), koje definišu jednu triangulaciju.

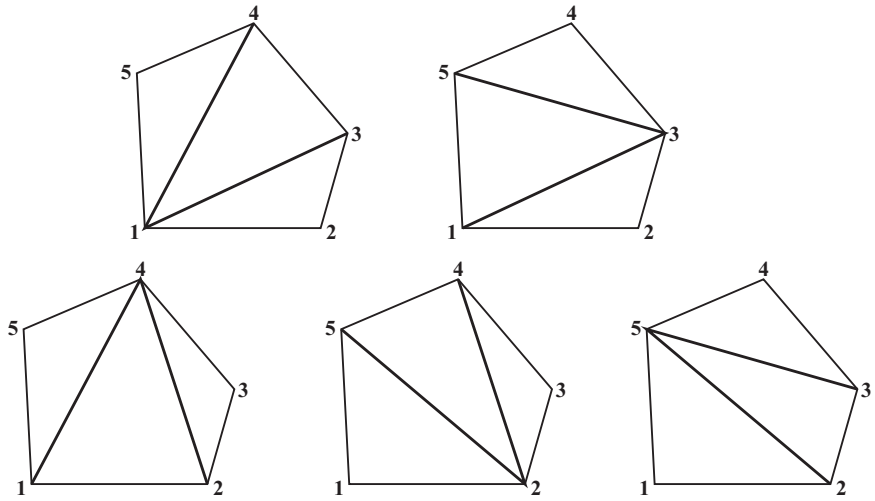
$$W_n = \sum P(T_n) \tag{3.9}$$

Slučaj 1: Klasično izračunavanje težina triangulacije

Za klasični proračun optimalne triangulacije, uzet ćemo primer konveksnog pentagona koji ima 5 različitih triangulacija 3.10.

$$T_n = \frac{1}{n-1}\begin{pmatrix} 2n-4 \\ n-2 \end{pmatrix} = \frac{(2n-4)!}{(n-1)!(n-2)!} \tag{3.10}$$

Primer 3.2.1. Pronalaženje optimalne triangulacije pomoću klasičnog proračuna obima:



Slika 3.10: Sve kombinacije triangulacije petougla

Tabela 3.4: Dimenzije petougla

Stranica	Dužina	Dijagonala	Dužina
$\delta_{1,2}$	9cm	$\delta_{1,3}$	11.51cm
$\delta_{2,3}$	5cm	$\delta_{1,4}$	11.96cm
$\delta_{3,4}$	7.5cm	$\delta_{3,5}$	11.32cm
$\delta_{4,5}$	6.5cm	$\delta_{2,4}$	11.10cm
$\delta_{5,1}$	8cm	$\delta_{2,5}$	12.41cm

$$\begin{aligned}
W_1(T_5) &= P_{\triangle 123} + P_{\triangle 134} + P_{\triangle 145} \\
&= (9 + 5 + 11.51) + (11.51 + 7.5 + 11.96) + (11.96 + 6.5 + 8) \\
&= 25.51 + 30.97 + 26.46 \\
&= 82.94
\end{aligned}$$

$$\begin{aligned}
W_2(T_5) &= P_{\triangle 123} + P_{\triangle 135} + P_{\triangle 345} \\
&= (9 + 5 + 11.51) + (11.51 + 11.32 + 8) + (11.32 + 7.5 + 6.5) \\
&= 25.51 + 30.83 + 25.32 \\
&= 81.66
\end{aligned}$$

$$\begin{aligned}
W_3(T_5) &= P_{\triangle 145} + P_{\triangle 124} + P_{\triangle 234} \\
&= (11.96 + 6.5 + 8) + (9 + 11.10 + 11.96) + (5 + 7.5 + 11.10) \\
&= 26.46 + 32.06 + 23.6 \\
&= 82.12
\end{aligned}$$

$$\begin{aligned}
W_4(T_5) &= P_{\triangle 125} + P_{\triangle 245} + P_{\triangle 234} \\
&= (8 + 9 + 12.41) + (12.41 + 11.10 + 6.5) + (11.10 + 5 + 7.5) \\
&= 29.41 + 30.01 + 23.6 \\
&= 83.02
\end{aligned}$$

$$\begin{aligned}
W_5(T_5) &= P_{\triangle 125} + P_{\triangle 235} + P_{\triangle 345} \\
&= (8 + 9 + 12.41) + (5 + 11.32 + 12.41) + (11.32 + 6.5 + 7.5) \\
&= 29.41 + 28.73 + 25.32 \\
&= 83.46
\end{aligned}$$

Ako primenimo formulu 3.9 na triangulacije petougla sa slike 3.10, dobijamo sledeći skup:

$$W(T_5) = (82.94; 81.66; 82.12; 83.02; 83.46)$$

Nakon izračunavanja težina svih triangulacija nalazimo minimalnu (optimalnu) težinu, kao što sledi:

$$\begin{aligned}
&\min W(T_5) = \min(82.94; 81.66; 82.12; 83.02; 83.46) \\
&\min W(T_5) = 81.66 \Rightarrow W_2(T_5)
\end{aligned}$$

U ovom slučaju optimalna triangulacija definisana je sledećim trouglovima: $\triangle(1,2,3); \triangle(1,3,5); \triangle(3,4,5)$.

Slučaj 2: Izračunavanje težna triangulacija na bazi Blok metode

Pre izračunavanja predstavimo neka svojstva našeg predloga:

- Pošto su spoljne stranice iste za sve kombinacije triangulacije datog poligona, isključujemo njihov obračun jer ne utiče na konačni rezultat;
- Imajući u vidu da su samo unutrašnje dijagonale različite i da one određuju jedinstvenost konačnog rezultata, predlažemo da samo njih koristimo u pronalaženju optimalne triangulacije.

Primer 3.2.2. Predloženi scenario zasnovan na Blok metodi za dati poligon.

Iščitavanje blokova triangulacije petouga (T_5) iz baze podataka

i	D_1	D_2
1	$\delta_{1,3}$	$\delta_{1,4}$
2	$\delta_{1,3}$	$\delta_{2,5}$
3	$\delta_{2,4}$	$\delta_{1,4}$
4	$\delta_{2,4}$	$\delta_{2,5}$
5	$\delta_{2,5}$	$\delta_{3,5}$

Ažuriranje tabele sa novom kolonom za čuvanje zbirne dužine internih dijagonala (L_{int}) za svaku liniju (označenu sa S).

$$S = \sum L_i nt$$

i	D_1	D_2	S
1	$\delta_{1,3}$	$\delta_{1,4}$	
2	$\delta_{1,3}$	$\delta_{2,5}$	
3	$\delta_{2,4}$	$\delta_{1,4}$	
4	$\delta_{2,4}$	$\delta_{2,5}$	
5	$\delta_{2,5}$	$\delta_{3,5}$	

Izračunavanje udaljenosti između temena i njihove dužine za svaku liniju pojedinačno, što definiše triangulaciju poligona:

- Izračunavanje udaljenosti;
- Izračunavanje dužine unutrašnjih dijagonala i njihove sume.

Ažuriranje tabele sa svim podacima.

i	D_1	D_2	S
1	$\delta_{1,3}$	$\delta_{1,4}$	23.47
2	$\delta_{1,3}$	$\delta_{2,5}$	**22.83**
3	$\delta_{2,4}$	$\delta_{1,4}$	23.06
4	$\delta_{2,4}$	$\delta_{2,5}$	23.51
5	$\delta_{2,5}$	$\delta_{3,5}$	23.73

Određivanje optimalne triangulacije na osnovu svih podataka iz tabele:

- Pronalaženje minimalne vrednosti poslednje kolone $\min(S)$.

Nakon dobijanja težina za sve triangulacije, nalazimo minimalno jedno (optimalno), kao što sledi:

$$\min S(T_5) = \min(23.47; 22.83; 23.06; 23.51; 23.73)$$
$$\min S(T_5) = 22.83$$

U ovom slučaju optimalna triangulacija je određena sa sledećim unutrašnjim dijagonalama: $D1(1,3); D2(2,5)$.

Tačnije, ako izrazimo istu triangulaciju preko trouglova $\triangle(1,2,3); \triangle(1,3,5); \triangle(3,4,5)$, dobijamo istu optimalnu triangulaciju kao u Primeru 3.2.1.

Zaključak je da dobijemo isti rezultat u oba slučaja, samo što u drugom slučaju vršimo računanje sa manje podataka, što se ogleda u jednostavnosti predložene metode.

Jednakost slučaja 1 i slučaja 2

Ako se posmatra metod izračunavanja u Primeru 3.2.1, može se primetiti da u računskom skupu obima trouglova (težina) u jednoj triangulaciji, svaka unutrašnja dijagonala se uvek koristi dva puta.

Ako se eliminiše dužina spoljašnjih dijagonala (L_{out}) datog poligona, ne narušava se jedinstvo izračunavanja težine triangulacije. Dalje, eliminisanjem, u sumi dužina unutrašnjih dijagonala koje se pojavljuju dva puta, dobijamo istu vrednost kao u Primeru 3.2.1 (tabela u koraku 4).

Postupak eliminacije je dat u sledećoj jednačini:

$$\frac{W(T_n) - \sum L_{out}}{2} = S \tag{3.11}$$

Primena jednačine 3.11 na konkretnom primeru za slučaj 1 i slučaj 2, dat je u nastavku:

1. $\frac{(82.94-36)}{2} = 23.47$
2. $\frac{(81.66-36)}{2} = 22.83$
3. $\frac{(82.12-36)}{2} = 23.06$
4. $\frac{(83.02-36)}{2} = 23.51$
5. $\frac{(83.46-36)}{2} = 23.73$

3.2.3 Komparativna analiza i eksperimentali rezultati

U ovoj sekciji su navedeni rezultati testiranja u pronalaženju optimalne triangulacije kroz dva algoritma. Uzimajući u obzir da se predloženi metod oslanja na pronalaženje optimalne triangulacije iz skupa svih generisanih triangulacija, za potrebe komparativne analize izabran je Hurtado-Noy algoritam [11], kao i detalji njegove implementacije u radu [27].

Tabela 3.5: Eksperimentalni rezultati testiranja: Blok metoda i vreme za pronalaženje optimalne triangulacije

n	***Broj triangulacija***	***Blok metoda***	***Optimalna triangulacija***
5	5	0.16	0
6	14	0.26	0.01
7	42	0.34	0.02
8	132	0.41	0.03
9	429	0.46	0.05
10	1,430	0.54	0.08
11	4,862	0.85	0.14
12	16,796	1.32	0.21
13	58,786	4.40	0.34
14	208,012	24.13	0.59
15	742,900	85.30	1.07

Tabela 3.6: Eksperimentalni rezultati testiranja: Hurtado-Noy metoda i vreme za pronalaženje optimalne triangulacije

n	***Broj triangulacija***	***Hurtado-Noy metoda***	***Optimalna triangulacija***
5	5	0.19	0
6	14	0.34	0.01
7	42	0.42	0.03
8	132	0.49	0.05
9	429	0.67	0.07
10	1,430	1.18	0.11
11	4,862	3.81	0.21
12	16,796	12.46	0.34
13	58,786	43.51	0.58
14	208,012	119.05	0.79
15	742,900	318.63	2.75

Tabela 3.7: Ukupno vreme izvršenja za dva algoritma i ubrzanje

n	***BM - Optimalna triangulacija***	***HN - Optimalna triangulcija***	***Ubrzanje***
5	**0.16**	**0.19**	**1.19**
6	**0.27**	**0.35**	**1.3**
7	**0.36**	**0.45**	**1.25**
8	**0.44**	**0.54**	**1.23**
9	**0.51**	**0.74**	**1.45**
10	**0.62**	**1.29**	**2.08**
11	**0.99**	**4.02**	**4.06**
12	**1.53**	**12.8**	**8.37**
13	**4.74**	**44.09**	**9.3**
14	**24.72**	**119.84**	**4.85**
15	**86.37**	**321.38**	**3.72**

Testiranje se izvršeno u NetBeans testnom modulu "Profile Main Project / CPU Analyze Performanse" na konfiguraciji računara*: CPU - Intel(R) Core(TM)2Duo CPU, T7700, 2.40 GHz, L2 Cache 4 MB (On-Die,ATC,Full-Speed), RAM Memory - 2 Gb, Graphic card - NVIDIA GeForce 8600M GS.

I u ovom slučaju dobijeni rezultati (Tabela 3.3) ukazuju na prednost korišćenje skladištenih vrednosti, kao veoma efikasan način za pronalaženje optimalne triangulacije, dok autorska Blok metoda utiče na brzini dobijanja težina triangulacija.

Poglavlje 4

PHP/MySQL okruženje i algoritmi računarske geometrije

U ovom poglavlju su predstavljene implementacije generisanja triangulacija konveksnog poligona u veb okruženju, optimizovanih po pitanju brzine i mogućnosti skladištenja. PHP i MySQL su popularne tehnologije otvorenog koda, savršene za brz i efikasan razvoj veb aplikacija koje su prilagođene za rad sa bazama podataka.

Veb aplikacije često koriste kombinaciju skripti koje se izvršavaju na serveru (ASP,PHP) i one koje se izvršavaju na klijentu (HTML, JavaScript). Klijentska strana je zadužena za prezentaciju, dok je serverska strana za primanje i skladištenje podataka.

Funkcionisanje klijent-server arhitekture se zasniva na vršenju transakcija ili razmeni podataka između klijenta i servera preko HTTP (Hyper Text Transfer Protocol) protokola. Klijent započinje razmenu tako što šalje zahtev serveru, a server koji je u stalnom stanju spremnosti, pruža uslugu klijentu tako što šalje podatke koje klijent zahteva.

Spoj popularnih veb tehnologija i kompleksnih algoritama iz računarske grafike predstavlja izazov, koji smo pokušali približiti u ovom poglavlju. U nastavku je prvo predstavljena klasična implementacija klijent-server arhitekture dok je u drugom predstavljena implementacija klijent-server arhitekture gde su korišćene napredne tehnike u cilju dobijanja što boljih rezultata pri generisanju triangulacija.

4.1 Osnove veb tehnologija

4.1.1 PHP

PHP je server skriptni jezik i moćan alat za izradu dinamičnih i interaktivnih veb sadržaja. Milioni veb stranica na kojima se koristi PHP, dokaz su njegove popularnosti i kapaciteta. Koriste ga programeri, koji cene njegovu fleksibilnost i brzinu, kao i veb dizajneri, kojima odgovaraju njegove mogućnosti i lakoća upotrebe. Pogodan je za izradu svih vrsta veb aplikacija, od onih malih, koje predstavljaju samo deo jedne internet stranice, pa do velikih kompleksnih sajtova. Više o PHP programskom jeziku možete naći u [14, 20, 21, 26].

Prema indeksu Zajednice programa TIOBE[1], PHP programski jezik je jedan od 10 najpopularnijih programskih jezika (2004. godine je proglašen jezikom godine). 80% od prvih 10 miliona sajtova koristi PHP na neki način, uključujući Facebook i Vikipediju.

4.1.2 MySQL baza podataka

MySQL je sistem otvorenog koda za upravljanje relacionim bazama podataka koji je zasnovan na SQL programskom jeziku. Ovaj sistem je moguće koristiti na mnogim platformama, poput Linux, Unix ili Windows. Može se lako integrisati sa širokim spektrom programskih jezika i smatra se veoma pouzdanom opcijom. Jedna od najistaknutijih karakteristika jeste njegova laka skalabilnost.

Na osnovu portala DB-Engines[2], koji rangira sisteme za upravljanje bazama podataka prema njihovoj popularnosti, MySQL zauzima drugo mesto, odmah posle Oracle Database.

Kao što je gore navedeno, MySQL je relaciona baza podataka. To znači da su podaci smešteni unutar strukture sposobni da prepoznaju odnose između sačuvanih informacija. Svaka baza podataka sadrži tabele. Svaka tabela (koja se takođe naziva relacija), sadrži jednu ili više kategorija podataka sačuvanih u kolonama (koji se takođe nazivaju atributima). Svaki red (koji se takođe naziva zapis ili zbirka), sadrži jedinstven podatak (koji se inače naziva ključ) za kategorije definisane u kolonama. Više o funkcionalnosti MySQL bazama podataka možete pročitati u [5, 43, 48].

4.1.3 PHP i MySQL

PHP je skriptni jezik koji se izvršava na serveru, a njegova glavna svrha je dinamičko kreiranje web stranica. MySQL je sistem za upravljanje relacionim bazama podataka, i zajedno sa programskim jezikom PHP, predstavlja jedno od najpopularnijih rešenja za kreiranje dinamičkih veb stranica i veb aplikacija zasnovanih na bazama podataka.

Troslojna arhitektura predstavlja tip klijent-server arhitekture, u kojoj su korisnički interfejs, procesi poslovne logike i pristup podacima, projektovani i upravljani kao nezavisni moduli.

Osnovne komponente (slojevi) ove arhitekture su:

- Klijentski (prezentacioni) sloj;
- Sloj poslovne (aplikativne) logike;
- Sloj podataka.

Kombinacija PHP-a i MySQL-a može se koristiti za izgradnju jednostavnih ili složenih i visoko prometnih veb stranica (o temi pogledajte više u [4, 15, 22, 35]).

PHP je izuzetno moćan i izuzetno brz - može da radi čak i na najosnovnijem hardveru i jedva da ostavlja tragove u sistemskim resursima.

[1] https://www.tiobe.com/tiobe-index/
[2] https://db-engines.com/en/ranking

MySQL je brz, moćan i jednostavan za upotrebu, sistem baze podataka, koji nudi skoro sve što bi veb stranici trebalo da se od podataka predstavi u pretraživačima.

4.2 Blok metoda za izračunavanje triangulacija konveksnog poligona u PHP/MySQL okruženju

U ovoj sekciji predstavljena je implementacija Blok metode [33] za triangulaciju konveksnog poligona u veb okruženju. Za implementaciju je korišćeno popularno razvojno okruženje PHP/MySQL. Polazna tačka jedne ovakve implementacije jeste da prikažemo prednosti korišćenja veb tehnologija u izvođenju složenog algoritma iz računarske grafike. Osnovna pretpostavka je da se jednom dobijeni rezultati skladište u bazu podataka (MySQL) i koriste za druga izračunavanja. Baze podataka su prikladne i strukturirane metode za deljenje i preuzimanje podataka. Odrađena je i komparativna analiza razvijenog programa u odnosu na dva kriterijuma: CPU vreme u generisanju triangulacija; i CPU vremena u čitanju rezultata iz baze podataka.

Naučni rad na kome se bazira ova sekcija je [41].

4.2.1 Blok metoda za izračunavanje triangulacija konveksnog poligona

U nastavku je prikazana Blok metoda za triangulaciju konveksnog poligona [33] koja je predmet naše implementacije.

Generalna strategija metode je da se problem razloži na manje zavisne potprobleme. Svaki potproblem se rešava samo jednom i koristi se mnogo puta izbegavajući nepotrebna ponovna izračunavanja.

Metoda se zasniva na upotrebi prethodno generisanih triangulacija poligona sa manjim brojem temena. Tačnije, algoritam generiše skup $\mathcal{T}_n$ koristeći sve prethodno generisane triangulacije iz skupa $\mathcal{T}_b$, gde je $b < n$. Skup $\mathcal{T}_b$ se koristi onoliko puta koliko je potrebno kao blok, tj. ponavlja se nekoliko puta u skupu $\mathcal{T}_n$.

U Blok metodi [33], sledeća tvrdnja je dokazana.

Tvrđenje 4.2.1. *Svaka triangulacija iz skupa $\mathcal{T}_{n-1}$ pojavljuje se kao početni deo u tačno dve triangulacije u skupu $\mathcal{T}_n$.*

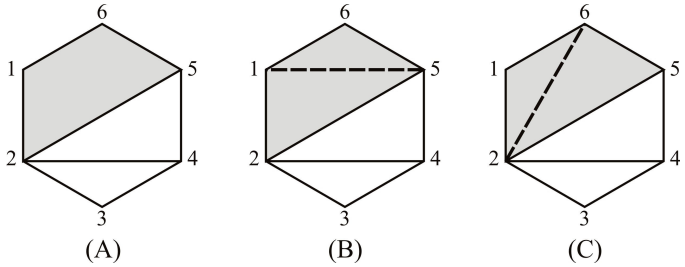
Formalna postavka Blok metode data je sledećom jednačinom:

$$T_n = 2T_{n-1} + \text{rest}(R_n). \tag{4.1}$$

Ova opšta ideja Blok metode u korišćenju skupa triangulacija T_{n-1} u generisanju triangulacija T_n je ilustrovana na Slici 4.1, gde je predstavljen jedan slučaj procesa transformacije triangulacija P_5 u dve odgovarajuće triangulacije P_6.

Na Slici 4.1, deo (A), može se videti da dijagonale $\delta_{2,4}$ i $\delta_{2,5}$ dele poligon P_6 tako da je većina temena zatvoreno osim temena 1, 2, 5, i 6 koji formiraju četvorougao. Na delovima (B) i (C) je prikazano kako se trianguliše četvorougao i kako se formiraju dve triangulacije poligona P_6 koje imaju triangulacije poligona P_5 kao početni blok.

Transformacija prikazana na Slici 4.1 se može predstaviti izrazom unutrašnjih dijagonala $P_5 = \{(2,4),(2,5)\} \rightarrow P_6 = \{\{(2,4),(2,5),(1,5)\}\&\{(2,4),(2,5),(2,6)\}\}$.



Slika 4.1: Transformacija triangulacija poligona P_5 u triangulacije poligona P_6.

Polazeći od pretpostavke da triangulacija može da ima najmanje dva uva, i u najgorem slučaju, jedno uvo može biti teme n, onda uvek imamo barem jedno uvo među ostalim temenima.

Za ispravnost algoritma, u postupku koji se koristi za pronalaženje i eliminisanje zatvorenih temena, autori su uveli listu uređenih parova forme.

$$L = \{(1,1),(2,2),\ldots,(n,n)\}. \tag{4.2}$$

Nakon eliminacije $n-l$ parova, lista L postaje

$$L = \{(s,i_s),\ s = 1,\ldots,l\},\ 4 \leq l \leq n,\ i_l = n. \tag{4.3}$$

Vrednosti $i_s,\ s = 1,\ldots,l$ su oznake temena, dok vrednosti $1,\ldots,l$ predstavljaju relativnu poziciju temena u listi L.

U nastavku ćemo ponoviti dva dodatna algoritma, Algoritam 4.2.1 - Eliminacija parova i Algoritam 4.2.2 - Formiranje četvorougla, koji su sastavni deo Algoritma 4.2.3 - Blok metode.

Algoritam 4.2.1 Eliminacija parova

Ulaz: Lista L, u formi (4.3) i temena i_p; i_q, gde je $d(p,q) = 2$

1: Uklanja se iz liste L zabranjeno teme smešteno između (p,i_p) i (q,i_q), kružno,
2: Vrši se pomeranje svih narednih temena od pozicije izbačenog zabranjenog temena i smanjuje se broj parova u listi L za 1.

Algoritam 4.2.2 Formiranje četvorougla

Ulaz: Lista L u formi (4.2), ceo broj n i niz od $n-4$ dijagonala (red u tabeli za $\mathcal{T}_n$))

1: Pronalazi dijagonalu δ_{i_p,i_q} gde je $d(p,q)=2$ u listi L

2: Poziva Algoritam 4.2.1 za parametre i_p i i_q

3: Ponavlja korake 1 i 2, $n-4$ puta

Osnovni algoritam za Blok metodu je prikazan u nastavku [33].

Algoritam 4.2.3 Algoritam za Blok metod

Ulaz: Ceo broj n i $\mathcal{T}_b$ sa brojem redova $row_b = C_{n-3}$ i brojem kolona $col_b = n-4$

1: Kreira praznu tabelu za $\mathcal{T}_n$ sa brojem redova $row_n = C_{n-2}$ i kolona $col_n = n-3$

2: Popunjava tabelu za $\mathcal{T}_n$ sa triangulacijama iz bloka $\mathcal{T}_b$ duplirajući svaki red iz $\mathcal{T}_b$

3: Popunjavanje ostatka za unete blokove (zadnja kolona tabele za prvih $2row_b$):

for $(i = 1; i \leq 2row_b; i+=2)$
{
Kreira listu L u formi (4.2)
Poziva Algoritam 4.2.2 za red i iz tabele za $\mathcal{T}_n$
Za preostala četiri temena u listi L kreira dijagonalu δ_{i_1,i_3} i smešta u poslednjoj koloni za dati red i, zatim kreira dijagonalu δ_{i_2,i_4}i smešta u poslednjoj koloni za red $i+1$.
}

4: Popunjavanje ostataka za tabelu koja $\mathcal{T}_n$ sadrži $T_n - 2T_b$ redova

4.1: Popunjavanje prvih $n-4$ kolona u poslednjih $row_n - 2row_b$

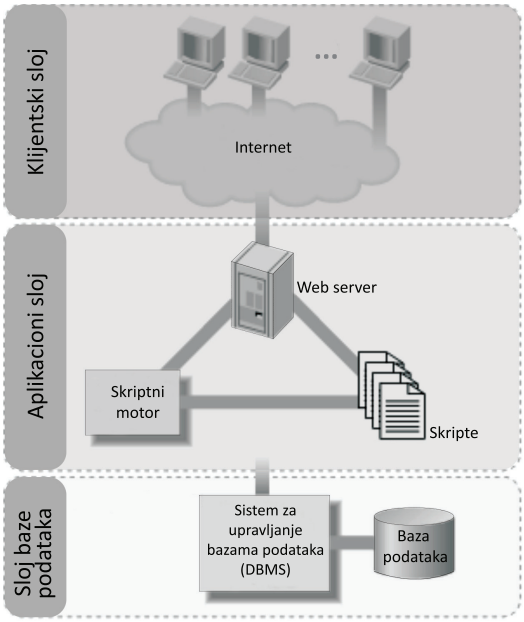
$i = 2 * row_b + 1$
Kreira listu L, u formi (4.2)
Eliminiše susedna temena do n i poziva Algoritam 4.2.1 za parametre 1 i $n-1$
Popunjava tekući red tabele i dijagonalama $\delta_{2,n}, \delta_{3,n}, ..., \delta_{n-2,n}$.
Prvih $n-4$ kolona u ostatku $row_n - 2row_b - 1$ se popunjava sa zadnjim temenom sve dok su ostala temena kombinacije $(n-4)$-nivoa u skupu $\{2, 3, ..., n-2\}$.
Broj ovakvih kombinacija je $\begin{pmatrix} n-3 \\ n-4 \end{pmatrix} = n-3$.

4.2: Popunjavanje zadnje kolone u poslednjih $row_n - 2row_b$ redova

for $(i = 2row_b + 2; i \leq row_n; i++)$
{
Kreira listu L u formi (4.2)
Poziva Algoritam 4.2.2 za red i iz tabele za $\mathcal{T}_n$
Za preostala četiri temena u listi L kreira dijagonalu δ_{i_1,i_3} i smešta u poslednjoj koloni za dati red i
}

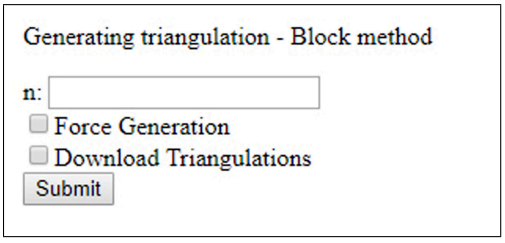
4.2.2 Implementacija Blok metode u PHP/MySQL okruženju

Najčešće korišćena arhitektura za razvoj veb aplikacija je troslojna arhitektura (Slika 4.2). Troslojna veb arhitektura je jedinstveni sistem za razvoj veb aplikacije sa bazom podataka, koja radi oko troslojnog modela, koji se sastoji od sloja baze podataka na dnu, sloja aplikacije u sredini i sloja klijenta na vrhu.



Slika 4.2: Model troslojne arhitekture veb aplikacija

Veb interfejs naše aplikacije dat je u nastavku, Slika 4.3



Slika 4.3: Veb interfejs aplikacije

Prema modelu troslojne arhitekture naša aplikacija je organizovana na sledeći način:

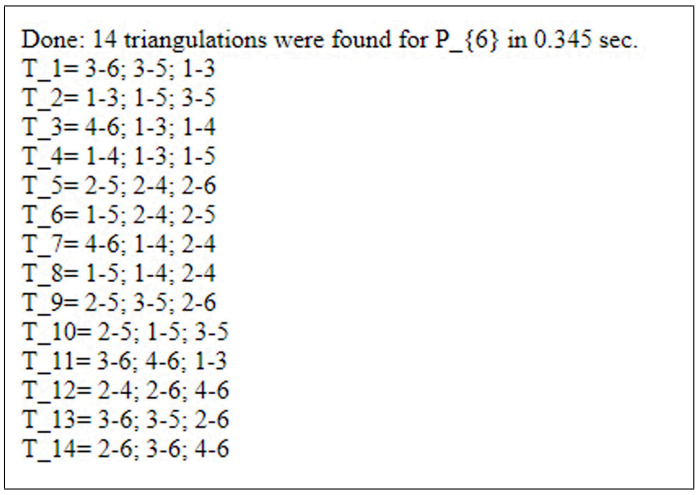
- Na *klijentskom sloju* imamo veb interfejs;
- Algoritam Blok metode se izvršava na *sloju aplikacije*;
- Generisane triangulacije se skladište na *sloju baze podataka*;

U nastavku je opisan detaljan prikaz scenarija aplikacije:

Korak 1.	Prvo je potrebno uneti vrednost za promenljivu n koja određuje za koji konveksni poligon se vrši izračunavanje triangulacija. *Preduslovi*: $n \geq 4$
Korak 2.	Kada se pritisne dugme submit, aplikacija vrši pretraživanje baze:
	Slučaj 1: Force Generation = Nije označeno Da li u bazi podataka postoje rezultati generisanja trianulacija za n; - Ako postoje, aplikacija iščitava rezultate za T_n u pretraživaču; - Ako ne postoje rezultati za T_n, aplikacija proverava da li u bazi postoje rezultati za T_{n-1}: * Ako postoje rezultati, aplikacija poziva Algoritam 4.2.3 * Ako ne postoje rezultati, preduslovi za pozivanje Algoritma 4.2.3 nisu ispunjeni
	Slučaj 2: Force Generation = Označeno Da li u bazi podataka postoje rezultati generisanja trianulacije za $n-1$; - Ako postoje, aplikacija poziva Algoritam 4.2.3 i iščitava rezultate za T_n u pretraživaču; - Ako ne postoje rezultati, preduslovi za pozivanje Algoritma 4.2.3 nisu ispunjeni
Korak 3.	Ako označimo "Download Triangulation", rezultati se mogu preuzeti u `CSV` formatu.

Primer 4.2.1. *U nastavku je prikazana ilustracija kako aplikacija radi na primeru generisanja triangulacija šestougla koristeći već poznate triangulacije petougla.*

Korak 1.	$n = 6$; Preduslovi su ispunjeni: $6 \geq 4$;
Korak 2.	Dugme submit je pritisnuto **Slučaj 1: Force Generation = Nije označeno** - Aplikacija vrši proveru da li u bazi podataka postoje rezultati tiangulacije za T_6; - Ako ne postoje rezultati za T_6, aplikacija proverava da li u bazi postoje rezultati za T_5; $\rightarrow$ Ispunjen uslov, u bazi postoje rezultati za T_5 * Ako postoje rezultati, aplikacija poziva Algoritam 4.2.3 $\rightarrow$ Aplikacija generiše triangulacije za T_6 i prikazuje rezultate u pretraživaču (Figure 4.4)



Slika 4.4: Generisanje rezultata za T_6

4.2.3 Komparativna analiza i eksperimentalni rezultati

Osnovna ideja naše implementacije je da obezbedimo odgovarajuću klijent-server veb aplikaciju, u besplatnom open source PHP/MySQL razvojnom okruženju, koristeći minimalne resurse: internet pretraživač i operativni sistem.

Za komparativnu analizu u predstavljanju prednosti veb tehnologija, implementiran je i dodatni algoritam iz oblasti računarske grafike (OTM metoda [40]). Oba algoritma se zasnivaju na upotrebi prethodno generisanih triangulacija manjeg poligona.

Vreme izvršenja, u odnosu na dva kriterijuma, predstavljeno je u Tabeli 4.3. Kolona u tabeli "Ubrzanje", prikazuje količnik vrednosti sadržanih u prethodne dve kolone.

Testiranje je odrađeno na računaru sa tehničkom specifikacijom*: *CPU - Inter(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40GHz, RAM memory 8GB, Graphic card: NVIDIA GeForce 820M.*

Tabela 4.3: Eksperimentalni rezultati testiranja: Blok metoda vs OTM metoda

n	Broj triangulacija	BM generisanje	BM - čitanje iz DB	Ubrzanje	OTM generisanje	OTM - čitanje iz DB	Ubrzanje
5	5	0.256	0.003	**85.33**	0.067	0.001	**67.00**
6	14	0.345	0.003	**115.00**	0.088	0.001	**88.00**
7	42	0.391	0.003	**130.33**	0.123	0.001	**123.00**
8	132	0.457	0.004	**114.25**	0.185	0.002	**92.50**
9	429	0.756	0.008	**94.50**	0.927	0.002	**463.50**
10	1,430	1.606	0.019	**84.53**	1.524	0.003	**508.00**
11	4,862	3.915	0.063	**62.14**	3.182	0.008	**397.75**
12	16,796	26.657	0.461	**57.82**	10.081	0.024	**420.04**
13	58,786	185.566	2.482	**74.76**	29.713	0.075	**396.17**
14	208,012	883.726	6.802	**129.92**	121.749	0.248	**490.92**
15	742,900	4,498.768	25.697	**175.07**	536.326	0.975	**550.08**

Na osnovu dobijenih eksperimentalnih rezultata, možemo zaključiti da su prednosti korišćenja baze podataka u izvođenju složenih algoritma opravdane.

4.3 Napredne tehnike PHP/MySQL-a i triangulacija poligona

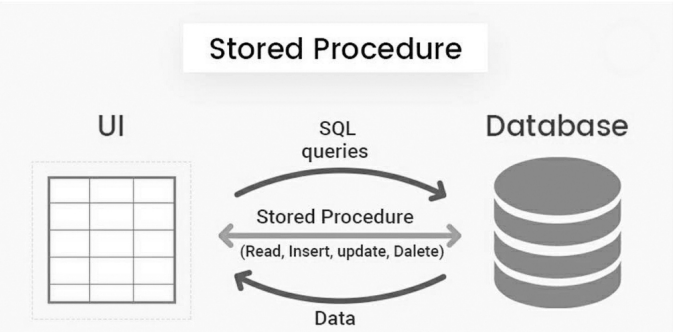
Razvojno okruženje PHP/MySQL je pogodno i za implementaciju nekih naprednih tehnika (procedura i funkcija). Obzirom da je triangulacija poligona kompleksan problem, gde algoritmi koji se koriste za generisanje triangulacija značajno opterećuju hardverske resurse, implementaciom uskladištenih procedura, pokušali smo da proces generisanja spustimo na nivo baze podataka kako bi dobili na brzini.

Implementacijom usladištenih procedura u našu veb aplikaciju, postigli smo bolje rezultate pri generisanju triangulacija u odnosu na klasičnu primenu troslojne arhitekture koja je prikazana u prethodnoj sekciji. Više o naprednim tehnikama PHP/MySQL okruženja možete videti u [14, 15, 20].

4.3.1 Osnove uskladištenih procedura

Po definiciji, uskladištene procedure predstavljaju skup prevedenih SQL funkcija (instrukcija), koje su smeštene (uskladištene) u samu bazu podataka. Obzirom da su smeštene u samu bazu podataka, a ne u korisničke aplikacije (na front-end kraju), ovakav skup instrukcija se brže izvršava jer se proces kompajliranja instrukcija vrši na SQL serveru. Svrha uskladištenih procedura je automatizacija koda i brzina rukovanja podacima [12, 13].

Uskladištene procedure pomažu u smanjenju mrežnog saobraćaja između aplikacija i MySQL servera, zato što, umesto slanja više dugačkih SQL upita, aplikacije pozivaju samo ime i parametre uskladištenih procedura.



Slika 4.5: Šematski prikaz uskladištenih procedura

Jedna od prednosti uskladištih procedura je da iste procedure mogu da koriste više aplikacija. Uskladištene procedure pomažu u smanjenju napora dupliranja iste logike u mnogim aplikacijama i čine vašu bazu podataka konzistentnijom.

Korišćenjem uskladištenih procedura može se povećati bezbednost baza podataka, tako što se dodeljuje odgovarajuća privilegija aplikacijama za pristup određenim uskladištenim procedurama, bez davanja ikakvih privilegija na osnovne tabele.

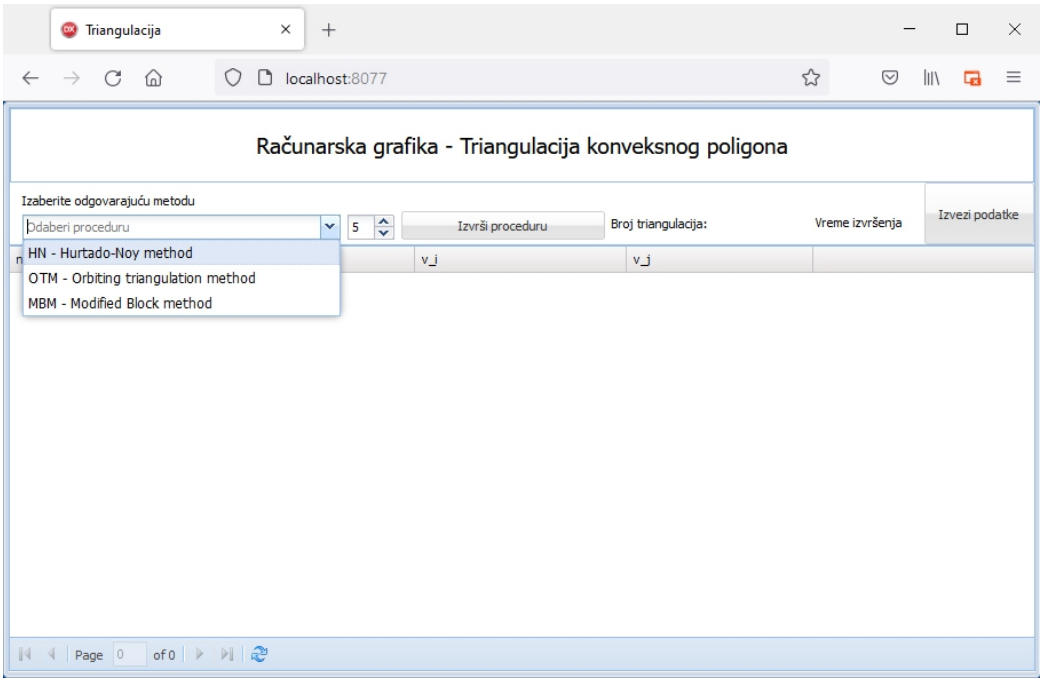
Još jedna prednost uskladištenih procedura je da su prenosive. Kada napišete svoju uskladištenu proceduru u SQL-u, znate da će ona raditi na svakoj platformi na kojoj MySQL radi, bez obaveze da instalirate dodatni paket okruženja za izvršavanje, ili da postavite dozvole za izvršavanje programa u operativnom sistemu, ili da primenite različite pakete ako imate različite tipove računara. To je prednost pisanja u SQL-u umesto na spoljnom jeziku kao što su Java ili C ili PHP. Uskladištene procedure su uvek dostupne kao *izvorni kod* u samoj bazi podataka.

4.3.2 Detalji implementacije

Troslojna veb arhitektura obezbeđuje konceptualni okvir za aplikacije koje rade sa bazama podataka. Primenom uskladištenih procedura, promenili smo logiku generisanja triangulacija u odnosu na prethodnu implementaciju. U ovom slučaju smo, proces generisanja triangulacija sa aplikacinog sloja, prebacili na sloj baze podataka.

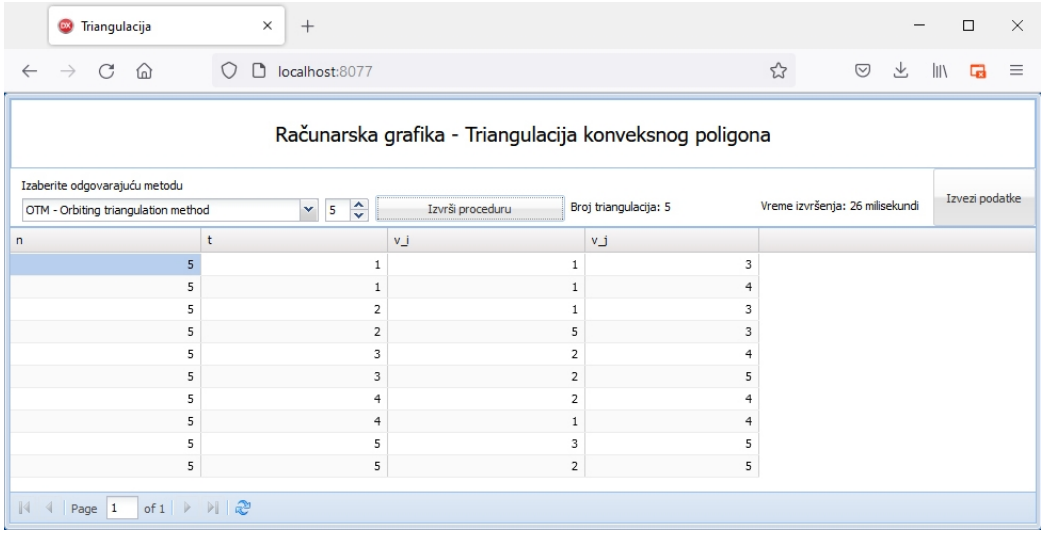
Na aplikaciji su implementirana tri algoritma: Hurtado-Noy metod [11]; OTM metoda [40]; i Modifikovana Blok methoda; gde je svaki algoritam predstavljen kao zasebna uskladištena procedura.

Na formi aplikacije korisnik može da izabere kojom procedurom želi da generiše triangulacije za traženi poligon, kao što je prikazano na Slici 4.6.



Slika 4.6: Veb forma aplikacije

Rezultati generisanja triangulacija se prikazuju direktno na radnoj površini aplikacije i isti se mogu preuzeti u *CSV* formatu klikom na dugme "Izvezi rezultate".



Slika 4.7: Veb forma aplikacije sa prikazanim reezultatima

4.3.3 Komparativna analiza i eksperimentalni rezultati

U nastavku su data vemena izvršavanja veb aplikacije za sve tri metode (Hurtado-Noy metoda, OTM metoda i Modifikovana Blok metoda). Vremena izvršavanja (u sekundama) su predstavljena u Tabeli 4.4.

Testiranje je odrađeno na računaru sledećih karakteristika*: *CPU - Inter(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40GHz, RAM memorija 8GB, Grafička karta: NVIDIA GeForce 820M.*

Tabela 4.4: Eksperimentalni rezultati testiranja: Hurtado-Noy algoritma, OTM metode i Modifikovane Blok metode

n	**Broj triangulacija**	**HN generisanje**	**HN - čitanje iz DB**	**OTM generisanje**	**OTM - čitanje iz DB**	**MBM generisanje**	**MBM - čitanje iz DB**
5	5	0.010	0.000	0.010	0.000	0.000	0.000
6	14	0.016	0.000	0.014	0.000	0.000	0.000
7	42	0.031	0.000	0.025	0.000	0.016	0.000
8	132	0.068	0.000	0.057	0.000	0.020	0.000
9	429	0.198	0.000	0.135	0.000	0.075	0.000
10	1,430	0.594	0.010	0.442	0.000	0.343	0.000
11	4,862	2.396	0.021	2.010	0.018	1.152	0.010
12	16,796	9.979	0.073	8.732	0.065	7.431	0.051
13	58,786	50.162	0.235	42.406	0.198	40.352	0.125
14	208,012	172.649	0.880	156.273	0.773	128.038	0.682
15	742,900	868.516	2.943	765.579	2.130	582.311	1.586

I, u ovom slučaju, rezultati komparativne analize pokazuju prednosti korišćenja baze podataka u izvođenju složenih algoritma. Primenom naprednih funkcionalnosti baza podataka, uskladištenih procedura, uspeli smo da smanjimo vreme generisanja triangulacija.

Iz svega navedenog, možemo zaključiti da su prednosti korišćenja baze podataka u izvođenju složenih algoritma opravdane. Ovakav način implementacije pruža dobru osnovu za dalju primenu veb tehnologija u računanju drugih algoritama.

Poglavlje 5

Zaključna razmatranja

U disertaciji su predstavljeni novi algoritmi za generisanje triangulacija konevksnog poligona i njihova primena u veb okruženju. Rezultati koje donosi ova disertacija se mogu svrstati u tri kategorije. Prvu kategoriju čine rezultati, koji se odnose na nove algoritme, koji se primenjuju u generisanju triangulacija konveksnog poligona. Drugu kategoriju čine rezultati, koji se odnose na pronalaženju optimalnih triangulacija. Treću kategoriju čine rezultati, koji se odnose na primeni PHP/MySQL okruženja za generisanje triangulacija konveksnih poligona.

Sledi kratak pregled rezultata izloženih u poglavljima disertacije:

1. Predstavljena su dva nova algoritma za generisanje triangulacija konveksnog poligona.

   A) Prva metoda za generisanje triangulacija naziva se *Orbiting triangle metoda - OTM metoda*. Metoda se zasniva na korišćenju skupa triangulacija poligona P_{n-1} za generisanje triangulacija poligona P_n. Osnovno svojstvo metode proizilazi iz tehnike obilaska poligona P_n, gde se generiše uređena lista (`OT-lista`), koja se koristi za generisanje triangulacija poligona P_n. Glavna karakteristika OTM metode je postupak mapiranja triangulacija kroz dve operacije, komplementiranja i rotiranja, kao i upotrebu Katalanovog trougla za identifikaciju važećih triangulacija.

   Na osnovu komparativne analize, predložene *OTM metode* sa *Hurtado-Noy* algoritmom i *Blok metodom*, dobijeni rezultati potvrđuju efikasnost novog algoritma u generisanju triangulacija i prevenciji pojavljivanja duplikata. Na osnovu eksperimentalnih rezultata, koji su dobijeni uz pomoć *Java* implementacije, može se utvrditi da ukupno vreme koje je izraženo u sekundama za skup $n \in \{5, 6, ..., 18\}$, odnosno skup od ukupno 14 testiranja kod *Hurtado-Noy* algoritma je 164.680, kod Blok metode je 117.603, dok je kod *OTM metode* 58.203. Znači, sa novim algoritmom smo postigli prosečno ubrzanje za 2.021 u odnosu na Hurtado-Noy algoritam i 2.829 u odnosu na Blok metodu, što procentualno iznosi 64.66% odnosno 50.51%.

B) Druga metoda za generisanje triangulacija naziva se *Modifikovana Blok metoda.* Ova metoda predstavlja modifikaciju originalne Blok metode, koja takođe koristi triangulacije poligona P_{n-1} za generisanje triangulacija poligona P_n. Obzirom da poligon P_n može nastati iz poligona P_{n-1} dodavanjem temena n, Modifikovana Blok metoda, teme n posmatra kroz dva slučaja: slučaj kada je n uvo poligona; i slučaj kada je n teme unutrašnje dijagonale. Glavno svojstvo ove metode se ogleda u korišćenju vrednosti iz Katalanovog trougla kako bi se izbeglo pojavljivanje duplih triangulacija.

U komparativnoj analizi za testiranje prvih 11 poligona (za $n \in \{5, 6, ..., 15\}$), ukupno potrošeno vreme kod originalne Blok metode je 80.035 sekundi, dok je kod Modifikovane Blok metode utrošeno samo 0.454 sekundi. Zanimljivo je da, na istom računaru gde je vršeno testiranje za $n \in \{16, 17\}$, kod originalne Blok metode generisanje triangulacija je bilo bezuspešno, dok je Modifikovana Blok metoda bez problema generisala rezultate. Na osnovu svega navedenog možemo zaključiti da predložena metoda sa svojim novim pristupom daje jako dobre rezultate.

2. Kada su u pitanju optimalne triangulacije, disertacija predstavlja novu metodu za pronalaženje i skladištenje optimalnih triangulacija. Metoda se zasniva na funkcionalnosti kvadratne matrice (SM), u kojoj skladištimo sve rezultate triangulacija za dati poligon n. Glavni akcenat metode je na brzini generisanja optimalne triangulacije i uštedi memorijskog prostora prilikom izračunavanja velikog broja triangulacija. Metoda skladištenja podataka je konstruisana na ideji da nije neophodno posebno skladištiti svaku težinu triangulacije, već da sve to izvodimo odjednom. Na osnovu komparativne analize izračunavanja težina, utvrđeno je da, predložena metoda izbegava izračunavanje istih proračuna i daje pravilno izračunavanje istih trouglova (i, j, k). Kod tradicionalnih metoda, dolazi do značajnog porasta istog izračunavanja, koji posebno dolazi do izražzaja kada broj temena poraste iznad $n > 7$, nasuprot predloženoj SM metodi.

   U drugom delu poglavlja, predstavljen je postupak pronalaženja optimalne triangulacije, koji se zasniva na autorskoj metodi za generisanje triangulacija (Blok metode). Analizirana su dva slučaja izračunavanja težina triangulacije (klasični slučaj i služaj zasnovan na Blok metodi). Prednost korišćenja Blok metode je značajna, jer se ogleda u činjenici da, korišćenjem skladištenih vrednosti, može biti veoma efikasan način za pronalaženje optimalne triangulacije. Osnovni cilj predloženog postupka je brzina dobijanja optimalne triangulacije.

3. Naveden je novi pristup rešavanja problema triangulacija u veb okruženju.

   Osnovna ideja naše implementacije je da obezbedimo odgovarajuću klijent-server veb aplikaciju, u besplatnom open source PHP/MySQL razvojnom okruženju, koristeći minimalne resurse: internet pretraživač i operativni sistem.

Na osnovu komparativne analize, implementiranih algoritama (OTM metode i Blok metode) na klijent-server veb aplikaciji, dobijeni su sledeći rezultati: ukupno potrošeno vreme za skup $n \in \{5, 6, ..., 15\}$, odnosno skup od ukupno 11 testiranja, kod direktnog generisanja rezultata Blok metode je 5602.443, a kod iščitavanja iz baze podataka 35.545; kod OTM metode taj odnos iznosi 703.965 prema 1.358 (izvršavanje algoritma se vrši na klijentskom sloju). Iz ovoga se može zaključiti da, korišćenjem baza podataka, dolazimo do znatno boljih rezultata, ubrzanje kod Blok metode iznosi 157.615 a kod OTM metode 518.383.

U drugom delu poglavlja, implementacijom uskladištenih procedura, proces generisanja triangulacija smo spustili sa nivoa aplikacije na nivo baze podataka, gde smo eksperimentalnom analizom dokazali opravdanost takve postavke. Komparativnom analizom implementiranih algoritama, *Hurtado-Noy* algoritma, OTM metode i Modifikovane Blok metode, za skup $n \in \{5, 6, ..., 15\}$, dobili smo sledeće rezultate: ukupno potrošeno vreme u generisanju triangulacija za 11 testiranja iznosi 1104.619, odnosno 975.683, odnosno 759.738; a, ukupno potrošeno vreme u čitanju istih iz baze podataka iznosi 4.162, odnosno 3.184, odnosno 2.454; dok ukupno ubrzanje iznosi 256.406, odnosno 306.433, odnosno 309.592.

Dobijeni rezultati ukazuju na značajna unapređenja, zato što smo kompleksna izračunavanja spustili na nivo baze podataka, gde smo dokazali polaznu ideju, da se jednom obrađeni rezultati koriste iščitavanjem iz baze podataka, kako bi se vreme obrade svelo na najmanje moguće.

Otvaraju se nova pitanja za dalja istraživanja u vezi oblasti triangulacije, a odnose se na mogućnosti primene predstavljenih algoritama. U praktičnom delu disertacije smo implementacijom složenih algoritama u veb okruženju (PHP/MySQL), ostavili prostor za neka dalja ispitivanja u korišćenju drugih popularnih programskih jezika tipa *Python* i *Oracle* baze podataka. Ovakav način implementacije pruža i dobru osnovu za dalju primenu veb tehnologija u izračunavanju drugih algoritama.

Prilozi

I) Detalji implementacije u PHP/MySQL okruženju

A) Klasična implementacija

U nastavku su prikazani segmenti izvornog kôda klijent-server veb aplikacije koja je predstavljena u Poglavlju 4:

Izvorni kôd za kreiranje MySQL baze podataka:

```
CREATE DATABASE IF NOT EXISTS triangulation;';
        if ($conn->query($sql)) {
            $conn->select_db('triangulation');
        } else {
            die('Could not create database: ' . $conn->error . '<br/>');
        }
```

Izvorni kôd za konekciju sa bazom podataka:

```
// Connection
$conn = new mysqli('localhost', 'root', '');
if ($conn->connect_errno) {
    die('Could not connect: (' . $conn->connect_errno . ')
        ' . $conn->connect_error . '<br/>');
    }
```

U našoj implementaciji koristimo samo jednu tabelu za čuvanje generisanih triangulacija.

```
CREATE TABLE IF NOT EXISTS Triangulation
        (
            n int,
            T int,
            i int,
            j int,
            INDEX Triangulation_n_idx (n),
            INDEX Triangulation_T_idx (T),
            INDEX Triangulation_i_idx (i),
            INDEX Triangulation_j_idx (j)
        );
```

Izvorni kôd implementacije Algoritma 4.2.3 - korak 3:

```
        // Step 3
        // diagonal \delta_{i_1 ,i_3}
        $sql .= '
        INSERT INTO Triangulation
        SELECT DISTINCT a.n,
            a.T,
            1 AS i,
            ' . ($n-1) . ' AS j
        FROM Triangulation a
        WHERE a.n=' . $n . '
            AND a.T%2=0;
        ';
        // diagonal \delta_{i_2 ,i_4}
        $sql .= '
        INSERT INTO Triangulation
        SELECT ' . $n . ' AS n,
            a.T,
            a.v AS i,
            ' . $n . ' AS j
        FROM
            (SELECT a.T,
                a.j AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1
                AND a.i=1
            UNION
            SELECT DISTINCT a.T,
                2 AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1) a
            INNER JOIN
            (SELECT a.T,
                a.i AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1
                AND a.j=' . ($n-1) . '
            UNION
            SELECT DISTINCT a.T,
                ' . ($n-2) . ' AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1) b
            ON a.T=b.T
                AND a.v=b.v; ';
```

Izvorni kôd implementacije Algoritma 4.2.3 - korak 4:

```
// Step 4
for ($k = 1; $k <= $n-4; $k++) {
    $sql .= '
    INSERT INTO Triangulation
    SELECT ' . $n . ' AS n,
        (CASE a.T
            WHEN @curTn_1 THEN @curTn
            ELSE @curTn := @curTn + SIGN(@curTn_1 := a.T) * SIGN(@curK := 1)
                            * SIGN(@lastV := ' . ($n-2) . ')
        END) + ' . $N . ' AS T,
        (CASE
            WHEN a.i=' . $n . ' THEN @lastV
            ELSE a.i
        END) AS i,
        SIGN(
            CASE WHEN a.j=' . ($n-1) . ' AND @curK = ' . ($k+1) . '
                THEN @lastV:= a.i
                ELSE 1
            END) *
            (CASE
                WHEN a.j=' . ($n-1) . ' AND @curK <= ' . $k . '
                THEN ' . $n . ' * SIGN(@curK := @curK+1)
                ELSE a.j
            END) AS j
    FROM
    (SELECT a.T,
        a.i,
        a.j
    FROM Triangulation a
    WHERE a.n=' . ($n-1) . '
        AND a.T IN
    (SELECT a.T
    FROM Triangulation a
    WHERE a.n=' . ($n-1) . '
        AND a.j=' . ($n-1) . '
    GROUP BY a.T
    HAVING count(a.i)>=' . $k . ')
    UNION SELECT a.T,
        ' . $n . ' AS i,
        ' . $n . ' AS j
    FROM Triangulation a
    WHERE a.n=' . ($n-1) . '
        AND a.j=' . ($n-1) . '
    GROUP BY a.T
    HAVING count(a.i)>=' . $k . ' ) a ,
    (SELECT @curTn := 0, @curTn_1 := 0, @curK := 0, @lastV := ' . ($n-2) . ') b
    ORDER BY a.T,
        a.i,
```

```
        a.j;
    ';
    if ($result = $conn->query('
        SELECT a.T
        FROM Triangulation a
        WHERE a.n=' . ($n-1) . '
            AND a.j=' . ($n-1) . '
        GROUP BY a.T
        HAVING count(a.i)>=' . $k . ';
        '
        )) {
        $N += $result->num_rows;
        $result->close();
        }
        else {
            die('Could not access Triangulation table: ' . $conn->error . '<br/>');
        }
    }
```

B) Implementacija sa uskladištenim procedurama

U nastavku su prikazani segmenti izvornog kôda klijent-server veb aplikacije koja je predstavljena u Poglavlju 4:

U nastavku se nalazi izvorni kôd uskladištene procedure Hurtado-Noy metode [11]:

```
CREATE PROCEDURE hn(IN required_n INT)
BEGIN
    DECLARE current_n   INT;
    DECLARE next_n      INT;
    SELECT MAX(n) INTO current_n FROM polygon;
    WHILE required_n > current_n DO
        SET next_n      = current_n + 1;
        START TRANSACTION;
        INSERT INTO polygon(n) VALUES (next_n);
        INSERT INTO diagonal(n,t,v_i,v_j)
            SELECT n,t,v_i,v_j
            FROM ((
                SELECT
                    next_n AS n,
                    dense_rank() over (ORDER BY d1.t,d2.v_i) AS t,
                    d1.v_i,
                        (CASE WHEN d1.v_j = d1.n AND d1.v_i < d2.v_i
                         THEN next_n ELSE d1.v_j END) AS v_j
                    FROM diagonal d1
                    INNER JOIN diagonal d2
                        ON d1.n = d2.n AND d1.t = d2.t
                    WHERE d1.n = current_n AND d2.v_j = d2.n
```

```
            )
            UNION ALL
            (
                SELECT
                    next_n AS n,
                    dense_rank() over (ORDER BY d1.t,d2.v_i) AS t,
                    d1.n AS v_i,
                    next_n AS v_j
                FROM diagonal d1
                INNER JOIN diagonal d2
                    ON d1.n = d2.n AND d1.t = d2.t
                WHERE d1.n = current_n AND d2.v_j = d2.n
                                       AND d1.v_i = 1
                                       AND d1.v_j = 2
            )
            UNION ALL
            (
                SELECT
                    next_n AS n,
                    dense_rank() over (ORDER BY d1.t,d2.v_i) AS t,
                    d2.v_i AS v_i,
                    next_n AS v_j
                FROM diagonal d1
                INNER JOIN diagonal d2
                    ON d1.n = d2.n AND d1.t = d2.t
                WHERE d1.n = current_n AND d2.v_j = d2.n
                                       AND d1.v_i = 1
                                       AND d1.v_j = 2
            )
        ) AS T
        ORDER BY t, v_i,v_j;
    INSERT INTO triangulation(n,t)
        SELECT DISTINCT n,t
        FROM diagonal
        WHERE n = next_n
        ORDER BY t;
    COMMIT;
    SET current_n = next_n;
END WHILE;
```

U nastavku se nalazi izvorni kôd uskladištene procedure OTM metode [40]:

```
CREATE PROCEDURE otm(IN required_n INT)
BEGIN
    DECLARE current_n   INT;
    DECLARE current_n_2 INT;
    DECLARE next_n      INT;
    DECLARE j           INT;
    SELECT MAX(n) INTO current_n FROM polygon;
    WHILE required_n > current_n DO
```

```
SET next_n      = current_n + 1;
SET current_n_2 = current_n - 2;
START TRANSACTION;
INSERT INTO catalan(n,j,c)
    SELECT
        current_n_2 AS n,
        c.j,
        @cc := @cc + c.c AS c
    FROM
        catalan c
    JOIN
        (SELECT @cc:=0) x
    WHERE c.n = current_n_2 - 1
    ORDER BY c.j;
INSERT INTO catalan(n,j,c)
    SELECT
        current_n_2,
        current_n_2,
        MAX(c.c)
    FROM
        catalan c
    WHERE n = current_n_2;

INSERT INTO polygon(n) VALUES (next_n);
INSERT INTO diagonal(n,t,v_i,v_j)
    SELECT dd.n, dd.t, dd.v_i, dd.v_j FROM
    (
        WITH tt AS
        (
            SELECT
                next_n AS n,
                row_number() OVER (ORDER BY c1.j DESC, t.t ASC) AS t,
                t.t AS src_t,
                next_n - 2 - c1.j AS v_i,
                (next_n - 2 - c1.j + next_n + 1)
            FROM
                catalan c1
            INNER JOIN
                catalan c2
            ON c1.n = c2.n AND c1.n = c2.j
            INNER JOIN
                triangulation t
            ON c1.n + 2 = t.n and c2.c - c1.c < t.t
            WHERE c1.n = next_n - 3
        )
        ( (
            SELECT
                tt.n,tt.t,tt.v_i,tt.v_j, 0 AS src_id
            FROM
```

```
                    tt
                )
                UNION ALL
                (
                SELECT
                    tt.n, tt.t,
                    (next_n - d.v_i + tt.v_i + next_n) % next_n + 1 AS v_i,
                    (next_n - d.v_j + tt.v_i + next_n) % next_n + 1 AS v_j,
                    d.id AS src_id
                FROM
                    diagonal d
                INNER JOIN
                    tt
                    ON d.n = tt.n - 1 AND d.t = tt.src_t
                )
            )
        ) AS dd
        ORDER BY dd.n,dd.t, src_id;
    INSERT INTO triangulation(n,t)
        SELECT DISTINCT n,t
        FROM diagonal
        WHERE n = next_n
        ORDER BY t;
    COMMIT;
    SET current_n = next_n;
END WHILE;
```

U nastavku se nalazi izvorni kôd uskladištene procedure Modifikovane Blok metode:

```
CREATE PROCEDURE mbm(IN required_n INT)
BEGIN
    DECLARE current_n   INT;
    DECLARE current_n_2 INT;
    DECLARE next_n      INT;
    DECLARE current_t   INT;
    DECLARE j           INT;
    SELECT MAX(n) INTO current_n FROM polygon;
    WHILE required_n > current_n DO
        SET next_n      = current_n + 1;
        SET current_n_2 = current_n - 2;
        START TRANSACTION;
        INSERT INTO catalan(n,j,c)
            SELECT
                current_n_2 AS n,
                c.j,
                @cc := @cc + c.c AS c
            FROM
                catalan c
            JOIN
```

```
    (SELECT @cc:=0) x
WHERE c.n = current_n_2 - 1
ORDER BY c.j;
SELECT MAX(c) INTO current_t FROM catalan WHERE n = current_n_2;
INSERT INTO catalan(n,j,c) VALUES (current_n_2, current_n_2, current_t);

INSERT INTO polygon(n) VALUES (next_n);

INSERT INTO diagonal(n,t,v_i,v_j)

SELECT dd.n, dd.t, dd.v_i, dd.v_j FROM
(
    WITH tt AS
    (
    SELECT
        next_n AS n,
        current_t + row_number() OVER (ORDER BY c1.j, tl.t, tr.t) AS t,
        tl.n AS nl,
        tl.t AS tl,
        tr.n AS nr,
        tr.t AS tr
    FROM
        catalan c1
    INNER JOIN
        catalan c2
        ON next_n - 4 - c1.j = c2.n AND c2.n = c2.j
    INNER JOIN
        triangulation tl
        ON tl.n = c1.j + 3
    INNER JOIN
        triangulation tr
        ON tr.n = next_n - c1.j - 1 AND tr.t <= c2.c
        WHERE c1.n = next_n - 4
    )
    ( (
    SELECT next_n AS n, t.t, 1 AS v_i, current_n AS v_j
        FROM
            triangulation t
            WHERE t.n = current_n
        )
        UNION ALL
        (
        SELECT next_n AS n, d.t, d.v_i, d.v_j
        FROM
            diagonal d
        WHERE d.n = current_n
        )
        UNION ALL
        (
```

```
                SELECT
                    tt.n, tt.t, tt.nl - 1 AS v_i, tt.n AS v_j
                FROM
                    tt
                )
                UNION ALL
                (
                SELECT
                    tt.n, tt.t, d.v_i, d.v_j
                FROM
                    tt
                INNER JOIN
                    diagonal d
                    ON tt.nl = d.n AND tt.tl = d.t
                    WHERE d.v_j < d.n
                )
                UNION ALL
                (
                SELECT
                    tt.n, tt.t, d.v_i, tt.n AS v_j
                FROM
                    tt
                INNER JOIN
                    diagonal d
                    ON tt.nl = d.n AND tt.tl = d.t
                    WHERE d.v_j = d.n
                )
                UNION ALL
                (
                SELECT
                    tt.n, tt.t, d.v_i + tt.nl - 2 AS v_i, d.v_j + tt.nl - 2 AS v_j
                FROM
                    tt
                INNER JOIN
                    diagonal d
                    ON tt.nr = d.n AND tt.tr = d.t
                )
            )
        ) AS dd
        ORDER BY dd.n,dd.t, dd.v_i, dd.v_j;
    INSERT INTO triangulation(n,t)
        SELECT DISTINCT n,t
        FROM diagonal
        WHERE n = next_n
        ORDER BY t;
    COMMIT;
    SET current_n = next_n;
END WHILE;
```

Spisak algoritama

Spisak tabela

Spisak slika

Literatura

[1] B. Chen, H. Cheng, *Interpretive OpenGL for computer graphics*, Computers and Graphics, **29** (2005), 331–339.

[2] D. Du, F. Hwang, *Computing in Euclidean Geometry*, World Scientific, (1992).

[3] D.E. Knuth, *The Art of Computer Programming (vol4: Generating all trees: history of combinatorial generation*), Addison-Wesley Professional, (2006).

[4] D. Lane, H. Williams, *Web Database Applications with PHP and MySQL, 2nd Edition*, O'Reilly Media, (2009).

[5] D. Nichter, *Efficient MySQL Performance: Best Practices and Techniques*, O'Reilly Media, (2022).

[6] D. Singmaster, *An elementary evaluation of the Catalan numbers*, American Math. Monthly **85**, (1978), 366–368.

[7] D.F. Bailey, *Counting Arrangements of 1's and -1's*. Mathematics Magazine, **69** (1996), 128–131.

[8] D.M. Campbell, *The computation of Catalan numbers*, Mathematics Magazine **57 (4)**, (1984), 195–208.

[9] D.R. Heffelfinger, *Java EE 6 Development with NetBeans 7*, Birmingham, UK: Pack publishing, 2011.

[10] F. Hurtado, M. Noy, *Ears of triangulations and Catalan numbers*, Discrete Mathematics, **149**(1996), 319–324.

[11] F. Hurtado, M. Noy, *Graph of Triangulations of a Convex Polygon and Tree of Triangulations*, Computational Geometry **13** (1999), 179–188.

[12] G. Harrison, S. Feuerstein, *MySQL Stored Procedure Programming*, O'Reilly Media, (2006).

[13] G. Harrison, S. Feuerstein, *MySQL Stored Procedure Programming: Building High-Performance Web Applications in MySQL*, O'Reilly Media, (2006).

[14] G. Schlossnagle, *Advanced PHP Programming, 1st Edition*, (2004).

[15] J. Duckett, *PHP & MySQL: Server-side Web Development, 1st Edition* Wiley, (2022).

[16] J. Luo, *Ming Antu and his power series expansions, in Seki, founder of modern mathematics in Japan*, Springer, Tokyo, (2013), 299—310.

[17] J. O'Rourke, *Art gallery theorems and algorithms*, Oxford Univ. Press, Oxford (1987).

[18] J. R. Loera, J.Rambaou, F. Santos, *Triangulations Of Point Sets: Applications, Structures, Algorithms*, Universidad de Cantabria, (July 21, 2003), preuzeto sa: http://personales.unican.es/ santosf/MSRI03/chapter1.pdf

[19] J. R. Loera, J.Rambaou, F. Santos, *Triangulations: Structures for Algorithms and Applications*, Springer Verlag, (2010).

[20] L. Ullman, *PHP Advanced and Object-Oriented Programming: Visual QuickPro Guide*, Pearson Education India, (2013).

[21] L. Ullman, *PHP for the Web: Visual QuickStart Guide, 5th edition*, Peachpit Press, (2016).

[22] L. Welling, *PHP and MySQL Web Development, 5th edition*, Addison-Wesley, (2016).

[23] L.W. Shapiro, *A Catalan triangle.* Discrete Mathematics, **14**(1976), 83-–90.

[24] M. Berg, M. Kreveld, M. Overmars, O. Schwarzkopf, *Polygon Triangulation, Computational Geometry (2nd ed.)*, Springer-Verlag, (2000), 45—61.

[25] M. Berg, O. Cheong, M. Kreveld, H. Overmars, *Computational Geometry: Algorithms and Applications. 3rd edition*, New York, USA, Springer Verlag, 2008.

[26] M. Rahman, *PHP 7 Data Structures and Algorithms: Implement Linked Lists, Stack, and Queues Using PHP*, Packt Publishing, (2017).

[27] M. Saračević, P.S. Stanimirović, S. Mašović, E. Biševac, *Implementation of the convex polygon triangulation algorithm*, Facta Universitatis, series: Mathematics and Informatics **27(2)**, (2012), 213–228.

[28] M. Saračević, P. Stanimirović, P.V. Krtolica, S. Mašović, *Construction and Notation of Convex Polygon Triangulation based on Ballot problem*, ROMJIST- Journal of Information Science and Technology, **17 (3)**,(2014), 237–251.

[29] M. Saračević, S. Mašović, P. Stanimirović , P. Krtolica, *Method for finding and storing optimal triangulations based on Square Matrix*, Applied Sciences–Geometry Balkan Press, **20**, (2018), 167–180.

[30] M.R. Garey, D.S. Johnson, F.P. Preparata, R.E. Tarjan, *Triangulating a simple polygon*, Inform. Process. Lett. **7**, (1978), 175–180.

[31] O. Hjelle, M. Daehen, *Triangulations and Applications, Mathematics And Visualization*, Springer, (2006).

[32] P. Hilton, J. Pedersen, *Catalan Numbers, Their Generalization, and Their Uses*, The mathematical intelligencer **13 (2)**, (1991), 64–75.

[33] P. Stanimirović, P. Krtolica, M. Saračević, S. Mašović, *Block Method for Triangulation Convex Polygon*, ROMJIST - Journal of Information Science and Technology **15(4)**, (2012), 344–354.

[34] P. Stanimirović, P. Krtolica, M. Saračević, S. Mašović, *Decomposition of Catalan numbers and Convex Polygon Triangulations*, International Journal of Computer Mathematics, **91**, (2013), 1315–1328.

[35] R. Nixon, *Learning PHP, MySQL & JavaScript, 5th Edition*, O'Reilly Media, (2018).

[36] R. Stanley, *Enumerative Combinatorics – vol. 2*, Cambridge University Press, Cambridge, (1999).

[37] R. Stanley, *Catalan Numbers*, Cambridge. University Press, (2015).

[38] S. Mašović, M. Saračević, *Finding Optimal Triangulation based on Block Method*, Southeast European Journal of Soft Computing, ISSN: **3 (2)**, (2014), 14–18.

[39] S. Mašović, M. Saračević, P. S. Stanimirović, *Alpha-Numeric notation for one Data Structure in Software Engineering*, Acta P.Hungarica: Journal of Applied Sciences, **11 (1)**, (2014), 193–204.

[40] S. Mašović, I.A. Elshaarawz, P.S. Stanimirović, P.V. Krtolica, *Orbiting triangle method for convex polygon triangulation*, Applicable Analysis and Discrete Mathematics, **12** (2018), 439–454.

[41] S. Mašović, M. Saračević, P. Stanimirović, *Computing triangulations of the convex polygon in PHP/MYSQL Environment*, Facta Universitatis, series: Mathematics and Informatics, **34 (1)**, (2019), 137–147.

[42] S.S. Gupta, K. Mukhopadhyaya, B.B. Bhattacharya, B.P. Sinha, *Geometric Classification of Triangulations and Their Enumeration in a Convex Polygon*, Computers Math.Applic. **27 (7)** Vol. **27**, (1994), 99–115.

[43] S. Tahaghoghi, H. Williams, *Learning MySQL: Get a Handle on Your Data*, O'Reilly Media, (2006).

[44] T. Davis, *Catalan Numbers*, Mathematical Circles Topics, November 24, (2010), preuzeto sa: http://www.geometer.org/mathcircles/catalan.pdf

[45] T. Koshy, *Catalan Numbers with Applications*, Oxford University Press, New York, (2009).

[46] T. Koshy, G. Zhenguang, *Some divisibility properties of Catalan numbers* Mathematical Gazette **95**,(2011), 96-–102.

[47] T. Mirzoev, S. Vassilev, *New Results on Optimal Area Triangulations of Convex Polygons*, In: Proceedings of the XII Encuentros de Geometria Computacional, Valladolid, Spain, 2007.

[48] V. Vaswani, *MySQL Database Usage & Administration*, McGraw Hill, (2009).

[49] W. Feller, *An Introduction to Probability Theory and its Applications, Volume I (3rd ed.)*, Wiley publisher, (1968), 69.

Biografija autora

Sead H. Mašović je rođen 29.11.1981. godine u Novom Pazaru. Osnovnu i srednju tehničku školu je završio u rodnom gradu. 2002. godine, na Fakultetu za informatiku i informacione tehnologije Internacionalnog Univerziteta u Novom Pazaru, upisuje osnovne akademske studije. Iste, završava u roku, 2006. godine, prosečnom ocenom 9,67 i time stiče zvanje diplomirani inženjer informacionih tehnologija. Postdiplomske studije je upisao 2007. godine na Tehničkom fakultetu "Mihajlo Pupin" Univerziteta u Novom Sadu. Odbranom master teze, 25.02.2009. godine, pod nazivom "Projektovanje informacionog sistema Matične službe lokalne samouprave", završava postdiplomske studije sa prosečnom ocenom 9,57, gde dobija zvanje diplomirani inženjer informatike – master. Godine 2019. upisuje doktorske akademske studije na Prirodno-matematičkom fakultetu Univerziteta u Nišu, na Departmanu za računarske nauke. Sve ispite i studijsko-istraživačke radove položio je u roku sa prosečnom ocenom 9,67.

U periodu od 2006 – 2009. je bio angažovan kao saradnik u nastavi na više stručnih predmeta na Fakultetu za Informatiku i informacione tehnologije Internacionalnog Univerziteta u Novom Pazaru. Od 2007. godine je uposlenik Gradske uprave za izvorne i poverene poslove grada Novog Pazara u Odeljenju za informacione tehnologije.

Pohađao je stručno usavršavanje, u trajanju od 3 meseca, preko Kancelarije za evropske integracije u Indiji na kursu "Advanced Course in Computer Networks Engineering and Management – ITEC/SCAAP courses", u Centre for Development of Advanced Computing, Mohali. Završio je i stručno usavršavanje Oracle akademije na kursevima: Database Design and Programming with SQL; Database Programming with PL/SQL; i JAVA Fundamentals (Oracle Certied instructor).

Autor je i menadžer više projekata: "Uvođenje GIS-a za Grad Novi Pazar" – broj granta UNOPS-EP-2015-Grant-108; "Uspostavljanje elektronske uprave Novog Pazara", projekat finansiran od strane Ministarstva državne uprave i lokalne samouprave; "Geografski informacioni sistem kao sredstvo za konkurentniji razvoj Novog Pazara i Tutina" – broj granta UNOPS-EUPRO-2019-Grant-120; i "eUprava – Uprava po meri građana" – broj granta UNOPS-SWISSPRO-2019-Grant-031.

Autor je 24 naučnih i stručnih radova publikovanih u međunarodnim i domaćim časopisima, među kojima je 7 njih u časopisima sa SCI/SCIE liste.

Bibliografija

1. **S. Mašović**, M. Saračević, P. Stanimirović, P. Krtolica, *Computing triangulations of the convex polygon in PHP/MYSQL Environment*, Facta Universitatis, series: Mathematics and Informatics, (2019), **34 (1)**, 137–147.

2. **S. Mašović**, I. Elshaarawy, P. Stanimirović, P. Krtolica, *Orbiting Triangle Method for Convex Polygon Triangulation*, Applicable Analysis and Discrete Mathematics, (2018) **12 (2)**, 439–454. (M22)

3. M. Saračević, **S. Mašović**, P. Stanimirović, P. Krtolica, *Method for finding and storing optimal triangulations based on square matrix*, Applied Sciences, Balkan Society of Geometers, (2018), **20**, 167–180.

4. P. Stanimirović, P. Krtolica, M. Saračević, **S. Mašović**, *Decomposition of Catalan Numbers and Convex Polygon Triangulations*, International Journal of Computer Mathematics, (2014), **91 (6)**, 1315–1328. (M22)

5. M. Saračević, P. Stanimirović, P. Krtolica, **S. Mašović**, *Construction and Notation of Convex Polygon Triangulation Based on Ballot Problem*, Romanian Journal of Information Science and Technology, (2014), **17 (3)**, 237–251. (M23)

6. **S. Mašović**, M. Saračević, P. Stanimirović, *Alpha Numeric notation for one Data Structure in Software Engineering*, Acta Polytechnica Hungarica: Journal of Applied Sciences, (2014), **11 (1)**, 193–204. (M23)

7. P. Stanimirović, P. Krtolica, M. Saračević, **S. Mašović**, *Block method for Triangulation Convex Polygon*, Romanian Journal of Information Science and Technology, (2012), **15 (4)**, 344–354. (M23)

8. P. Stanimirović, M. Tasic, M. Saračević, **S. Mašović**, *UML Based Modeling for the Moore-Penrose inverse computation*, Revista Metal. International, (2012), **17 (12)**, 99–106. (M23)

9. **S. Mašović**, M. Saračević, H. Kamberović, M. Kudumović, *Java technology in the design and implementation of web applications*, Technics Technologies Education Management, (2012), **7 (2)**, 504–512. (M23)

10. **S. Mašović**, M. Saračević, *Finding Optimal Triangulation based on Block Method*, Souteast European Journal of Soft Computing, (2014), **3 (2)**, 14–18.

11. M. Saračević, P. Stanimirović, **S. Mašović**, *Implementation of some algorithms in computer graphics in Java*, Technics Technologies Education Management, (2013), **8 (1)**, 293–300.

12. M. Saračević, P. Stanimirović, **S. Mašović**, *Object-oriented analysis and design for one algorithm of computational geometry: Forward, reverse and round-trip engineering*, Journal of Information Technology and Applications, (2013), **3 (2)**, 96–106.

13. M. Saračević, **S. Mašović**, D. Milošević, M. Kudumović, *Proposal for applying the optimal triangulation method in 3D medical image processing*, Balkan Journal of Health Science (BJHS), (2013), **1 (1)**, 27–34.

14. M. Saračević, **S. Mašović**, *Advantages of ACID compliance in application development in FIREBIRD databases*, International Journal of Strategic Management and Decision Support Systems, (2013), **18 (1)**, 53–61.

15. M. Saračević, **S. Mašović**, *Model implementacije sistema za e-plaćanje, baziran na poslovnoj inteligenciji*, FBIM Transactions: Journal for Finance, Business, Information, Industrial technologies and Management, (2013), **1 (2)**, 136–144.

16. M. Saračević, E. Elfić,Š. Plojović, **S. Mašović**, *Implementacija transportnog problema primenom metode meta-heurističkog pristupa*, Ekonomski izazovi, (2013), **3**, 39–48.

17. M. Saračević, **S. Mašović**, D. Milošević, *Java implementation for triangulation of convex polygon based on Lukasiewicz's algorithm and binary trees*, Souteast European Journal of Soft Computing, (2013), **2 (2)**, 40–45.

18. M. Saračević, **S. Mašović**,Š. Plojović, *UML modeling for traveling salesman problem based on genetic algorithms*, Souteast European Journal of Soft Computing, (2012), **1 (2)**, 72–79.

19. M. Saračević, P. Stanimirović, **S. Mašović**, E. Biševac, *Implementation of the convex polygon triangulation algorithm*, Facta Universitatis, series: Mathematics and Informatics, (2012), **27 (2)**, 213–228.

20. M. Saračević, **S. Mašović**, M. Šemsović, *Inovacije u visokom obrazovanju sa osvrtom na konkretan razvoj kursa prema ADDIE modelu za potrebe realizacije e-učenja na univerzitetu*, Socioeconomica – The Scientific Journal for Theory and Practice of Socioeconomic Development, (2012), **1 (2)**, 267–280.

21. M. Saračević, D. Milošević, **S. Mašović**, *Inovacije i unapređenje nastave matematike primenom JAVA apleta u sistemima za e-učenje*, Nastava i vaspitanje/Journal of Education, (2012), **61 (4)**, 723–740.

22. M. Saračević, D. Milošević, **S. Mašović**, *Uporedna analiza uspešnosti savladavanja gradiva na tradicionalan način i putem internet*, Inovacije u nastavi - časopis za savremenu nastavu, (2012), **25 (4)**, 67–77.

23. **S. Mašović**, M. Saračević, *Zastupljenost e-servisa u javnim upravama Srbije*, Info M - Journal of Information Technology and Multimedia Systems, (2012), **11 (41)**, 21–25.

24. M. Saračević, **S. Mašović**, H. Kamberović, *Application of JAVA and UML tools to better quality of some matrices computations*, CDQM Journal: Communications in Dependability and Quality Management, (2012), **15 (3)**, 21–31.

Izjave autora

IZJAVA O AUTORSTVU

Izjavljujem da je doktorska disertacija, pod naslovom

ALGORITMI ZA TRIANGULACIJU POLIGONA I NJIHOVA IMPLEMENTACIJA U VEB OKRUŽENJU

koja je odbranjena na Prirodno-matematičkom fakultetu Univerziteta u Nišu:

- rezultat sopstvenog istraživačkog rada;
- da ovu disertaciju, ni u celini, niti u delovima, nisam prijavljivao na drugim fakultetima, niti univerzitetima;
- da nisam povredio autorska prava, niti zloupotrebio intelektualnu svojinu drugih lica.

Dozvoljavam da se objave moji lični podaci, koji su u vezi sa autorstvom i dobijanjem akademskog zvanja doktora nauka, kao što su ime i prezime, godina i mesto rođenja i datum odbrane rada, i to u katalogu Biblioteke, Digitalnom repozitorijumu Univerziteta u Nišu, kao i u publikacijama Univerziteta u Nišu.

U Nišu, 26.09.2022. god.

Potpis autora disertacije:

Sead H. Mašović

IZJAVA O ISTOVETNOSTI ŠTAMPANOG I ELEKTRONSKOG OBLIKA DOKTORSKE DISERTACIJE

Naslov disertacije:

ALGORITMI ZA TRIANGULACIJU POLIGONA I NJIHOVA IMPLEMENTACIJA U VEB OKRUŽENJU

Izjavljujem da je elektronski oblik moje doktorske disertacije, koju sam predao za unošenje u **Digitalni repozitorijum Univerziteta u Nišu,** istovetan štampanom obliku.

U Nišu, 26.09.2022. god.

Potpis autora disertacije:

Sead H. Mašović

IZJAVA O KORIŠĆENjU

Ovlašćujem Univerzitetsku biblioteku „Nikola Tesla“ da u Digitalni repozitorijum Univerziteta u Nišu unese moju doktorsku disertaciju, pod naslovom:

ALGORITMI ZA TRIANGULACIJU POLIGONA I NJIHOVA IMPLEMENTACIJA U VEB OKRUŽENJU

Disertaciju sa svim prilozima predao sam u elektronskom obliku, pogodnom za trajno arhiviranje.

Moju doktorsku disertaciju, unetu u Digitalni repozitorijum Univerziteta u Nišu, mogu koristiti svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (Creative Commons), za koju sam se odlučio.

1. Autorstvo **(CC BY)**
2. Autorstvo – nekomercijalno **(CC BY-NC)**
3. **Autorstvo – nekomercijalno – bez prerade (CC BY-NC-ND)**
4. Autorstvo – nekomercijalno – deliti pod istim uslovima **(CC BY-NC-SA)**
5. Autorstvo – bez prerade **(CC BY-ND)**
6. Autorstvo – deliti pod istim uslovima **(CC BY-SA)**

U Nišu, 26.09.2022. god.

Potpis autora disertacije:

Sead H. Mašović