# Geometrijski randomizirani algoritmi

**Milačić, Luka**

**SVEUČILIŠTE U ZAGREBU**

**PRIRODOSLOVNO–MATEMATIČKI FAKULTET**

**MATEMATIČKI ODSJEK**

Luka Milačić

# GEOMETRIJSKI RANDOMIZIRANI ALGORITMI

Diplomski rad

Voditelji rada:
dr. sc. Nina Kamčev

Zagreb, siječanj, 2025.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

*Prije svega želio bih zahvaliti mentorici dr. sc. Nini Kamčev. Njeno znanje, strpljenje i mnogi savjeti uvelike su obogatili rad. Neizmjerno sam zahvalan obitelji i prijateljima za podršku tijekom studija. Dodatno, želim istaknuti da sam svoj interes prema matematici i njezinim izazovima prepoznao zahvaljujući profesorici Maji Zelčić i profesoru Zoranu Vajiću iz srednje škole.*

# Sadržaj

# Uvod

Razvoj učinkovitih algoritama predstavlja jedan od temeljnih i ključnih izazova u području računalnih znanosti. U geometriji, gdje su problemi često složeni i višedimenzionalni, randomizirani algoritmi nude značajne prednosti u odnosu na determinističke pristupe. Randomizirani algoritmi koriste slučajnost kako bi postigli jednostavniju implementaciju, ubrzali proces rješavanja problema ili osigurali bolju skalabilnost.

Ovaj rad se bavi randomiziranim algoritmima u kontekstu geometrijskih problema u dvije i tri dimenzije. Analiziramo ne samo vremenske složenosti algoritama, već i druge metrike. Recimo analiziramo broj dijelova na koji algoritam dijeli ravninu ili duljinu puta koju određeni algoritam prijeđe. Budući da su algoritmi randomizirani proučavat ćemo njihovu očekivanu složenost. Uz to jednim od važnih alata pokazat će se analiza algoritma unatrag, gdje analiziramo algoritam kao da ga izvršavamo u suprotnom smjeru.

Same strukture koje ćemo proučavati u algoritmima prikazat ćemo kao planarne grafove. Skupove elemenata planarnog grafa vrhove, bridove i strane označit ćemo s $V$,$E$ i $F$. Za sve elemente planarnog grafa podrazumijevamo da su susjedstva između njih uzajamna. Recimo svaki vrh zna u kojim stranama je sadržan te kojim bridovima je jedna krajnja točka. Također koristimo geometrijska svojstva planarnih grafova dobivenih iz poliedara.

Cilj ovog rada je istražiti prednosti randomiziranih algoritama u geometrijskim problemima te njihove koristi u stvarnom životu. Uz to želimo pokazati da jednostavniji algoritmi, unatoč svojoj naizgled manjoj sofisticiranosti, često daju jednake ili bolje rezultate u praksi. Konkretno, rad teži dokazati da randomizirani algoritmi pružaju visoku razinu praktičnosti, kako kroz pojednostavljenje implementacije tako i kroz performanse u raznim metrikama.

U prvom poglavlju bavimo se analizom konstrukcije konveksnih ljusaka skupa točaka u dvije i tri dimenzije. Uz to, uvodimo vrlo važan koncept dualnosti, s pomoću kojeg se mogu poistovjetiti određeni problemi u geometriji, kao u slučaju konveksne ljuske u tri dimenzije. Taj problem poistovjećujemo s problemom presijecajućih poluprostora, za koji dajemo randomizirani algoritam kojime dobivamo i randomizirani algoritam za konveksnu ljusku. Standardni algoritam za konveksnu ljusku u tri dimenzije poznat je po svojoj složenosti i težini implementacije. Ipak, njegov randomizirani pandan puno je jednostavniji i ima istu

očekivanu složenost.

U nastavku definiramo Voronoijeve dijagrame i njima ekvivalentne Delaunayjeve triangulacije. To su neke od najbitnijih dekompozicija ravnine u računalnoj matematici, koje ćemo koristiti za ilustraciju algoritama u kasnijim poglavljima. Osim njih, promatramo i dekompoziciju ravnine trapezima. Za sve te dekompozicije postoji randomizirani algoritam čija je očekivana vremenska složenost jednaka složenosti najbržih postojećih determinističkih algoritama za iste probleme. Također, proučavamo algoritme za particije ravnine i prostora, pri čemu je ključna metrika broj područja na koje algoritam dijeli ravninu ili prostor. Ovakvi algoritmi značajni su u računalnoj grafici, posebice za prikaz i skrivanje objekata iz različitih kutova gledišta.

Na kraju analiziramo konkretne algoritme koji se mogu primijeniti na bilo koje euklidske grafove. Definiramo izvor i odredište kao dva vrha u grafu između kojih dani algoritmi moraju pronaći put. Za svaki algoritam pokazujemo da postoje protuprimjeri gdje nikada neće biti pronađen put. Usprkos tome, pokazujemo da u Delaunayjevim triangulacijama algoritmi uvijek pronalaze put. Nadalje, analiziramo omjer između najduljeg puta koji algoritmi pronalaze i najkraćeg mogućeg puta u samom grafu. Uz to, izlažemo algoritam kod kojeg taj omjer nikada ne prelazi neku konstantnu vrijednost. Na kraju poglavlja dajemo kratku raspravu o empirijskim rezultatima svih analiziranih algoritama. Ti rezultati sugeriraju da jednostavniji algoritmi često daju bolje rezultate u praksi, iako bi ih zbog njihove jednostavnosti bilo lako podcijeniti u teorijskim analizama iz [8] i [11].

# Poglavlje 1

# Randomizirana konstrukcija konveksnih ljusaka

## 1.1 Konveksna ljuska u ravnini

Na početku iznosimo elegantan algoritam koji će prikazati glavne ideje iza brojnih randomiziranih algoritama u geometriji. Definirajmo prvo sve potrebno za zadani problem.

**Definicija 1.1.1.** ***Konveksna ljuska*** *za skup $S$ od $n$ točaka je najmanji konveksan skup koji sadrži sve točke iz $S$.*

Zamislimo da u ravnini imamo $n$ točaka i oko njega rastegnemo jako veliku gumu, konveksna ljuska je poligon koji je dobiven kada pustimo gumu da opkoli točke. Sa $\text{conv}(S)$ označit ćemo konveksnu ljusku od $S$.

Od sada pa nadalje, ovisno o broju dimenzija koje će biti unaprijed definirane, smatramo da kada za $S$ kažemo da je skup točaka, onda je on podskup od $\mathbb{R}^n$.

**Definicija 1.1.2.** ***Problem pronalaska konveksne ljuske u ravnini*** *je problem pronalaska poligona koji definira rub* $\text{conv}(S)$*, za ulaz od $n$ točaka unutar skupa $S$.*

Izlaz bi trebao vratiti listu vrhova iz $\text{conv}(S)$ po redu, u matematički pozitivnom smjeru. Radi određenosti, neka je prva točka u ovom poretku točka iz skupa $S$ s najmanjom $x$-koordinatom. Pretpostavljamo da nijedne tri točke iz $S$ ne leže na istom pravcu. Ova se pretpostavka može izostaviti u praktičnoj implementaciji uz manje modifikacije.
Za problem pronalaska konveksne ljuske postoje deterministički algoritmi koji rješavaju taj problem u složenosti $O(n \log n)$, kao na primjer "Gram scan" [1].

Slijedi primjer randomiziranog algoritma čija je očekivana složenost $O(n \log n)$. Glavna

prednost ovog algoritma je njegova jednostavnost, ostali algoritmi za dobivanje konveksne ljuske u ravnini znatno su teži za implementirati nego dani randomizirani algoritam. Štoviše ovakav algoritam se može poopćiti za pronalazak konveksnih ljusaka u više dimenzija. Prije samog opisa algoritma definirajmo neke pojmove potrebne za daljnja razmatranja.

Permutiramo točke u ulazu iz skupa $S$. Neka je $P_i$ $i$-ta točka po redu u ovoj nasumičnoj permutaciji, definirajmo tada $S_i$ kao skup $\{P_1, P_2, ..., P_i\}$. Nakon $i$-tog koraka algoritam će izračunati conv($S_i$). Tijekom tog koraka dodajemo točku $P_i$ u conv($S_{i-1}$), te stvaramo conv($S_i$). Uđimo sada u detalje.

Počinjemo od conv($S_3$), jer je trokut konveksna ljuska od 3 nekolinearne točke u ravnini. Tijekom cijelog algoritma biramo točku koja je uvijek u unutrašnjosti od conv($S$), preciznije, koristimo središte upisane kružnice conv($S_3$), koje dobivamo u konstantnom vremenu. Nazovimo tu točku $P_0$, ona je uvijek unutar conv($S_i$) radi conv($S_3$) $\subseteq$ conv($S_i$). Nakon $i$-tog koraka, držimo kružnu povezanu listu vrhova u conv($S_i$) (graf koji je jedan veliki ciklus). Za svaku točku $P \in S \setminus S_i$, držimo obostrani pokazivač na brid odnosno dužinu u rubu conv($S_i$) koju siječe polupravac $\overrightarrow{P_0P}$ koji kreće od $P_0$ i prolazi kroz $P$. Za svaki $P$ postoji jedinstvena dužina radi svojstava konveksne ljuske. Specificirajući sve strukture podataka koje su nam potrebne, prelazimo na analizu algoritma.

Odgovor na pitanje siječe li polupravac $\overrightarrow{P_0P}$ neku određenu dužinu možemo odrediti u konstantnom vremenu. Prema tome na početku, kada imamo conv($S_3$) koji ima 3 brida, svih ostalih $n-3$ točaka možemo s njima upariti u $O(n)$.

Točka $P_i$ definirana u $k$-tom koraku nalazi se unutar ili izvana conv($S_i$) (zbog naše prijašnje pretpostavke točka $P_i$ ne može ležati točno na rubu ljuske). Koristeći segment $\overline{P_0P}$ i pokazivač na $P$, to možemo odrediti u konstantnom vremenu. Ako je točka $P_i$ unutar conv($S_i$) brišemo pokazivač i nastavljamo na $(k+1)$-korak. U suprotnom, moramo modificirati našu listu koja predstavlja poligon koji je rub ljuske. Te točke su podijeljene u 3 skupa sa dodavanjem $P_i$:

1. Vrhovi conv($S_{i-1}$) koji su bili izbrisani jer nisu vrhovi conv($S_i$).
2. Dva vrha koji su postali susjedi točke $P_i$ u conv($S_i$). Nazvat ćemo ih $V_1$ i $V_2$.
3. Vrhovi od conv($S_{i-1}$) koji su u conv($S_i$) koji su sa svojim bridovima ostali nepromijenjeni.

Krajnje točke dužine koju siječe $\overline{P_0P_i}$ su tipa (1) ili (2). Hodom po rubu conv($S_{i-1}$), od danog brida, u oba smjera možemo odrediti vrhove tipa (1) ili (2). To određujemo u line-

arnom vremenu proporcionalnom broju takvih vrhova. Kako to radimo, usput detektiramo točke u $S \setminus S_i$, koje s $P_0$ sijeku bridove koje brišemo i ažuriramo im pokazivače na jedan od bridova $\overline{V_1P_0}$ ili $\overline{P_0V_2}$. Za svaku detektiranu točku nam treba konstantno vremena, jer se radi o provjeri 2 brida.

Računamo cijenu svakog koraka. Svako brisanje brida se može naplatiti cijeni kreacije tog brida, budući da brid može biti izbrisan samo ako je prvo stvoren. Za svaku novo dodanu točku dodajemo maksimalno 2 brida, pa je ukupna složenost brisanja brida proporcionalna broju točaka kojih je $n$.

Cijena ažuriranje pokazivača u $k$-tom koraku je broj točaka u $S \setminus S_i$ čiji su upareni bridovi bili izbrisani u tom koraku.

**Lema 1.1.3.** *Očekivan broj koraka za ažurirati pokazivače u algoritmu je $O(n \log n)$.*

*Dokaz.* Označimo broj ažuriranih pokazivača s $X_i$. Za ograničavanje očekivanja ove slučajne varijable $\mathbb{E}[X_i]$ koristit ćemo analizu unatrag. Zamislimo da dani algoritam izvršavamo unatrag, da iz $\text{conv}(S_i \setminus S_3)$ brišemo neku nasumičnu točku da stvorimo $\text{conv}(S_{i-1})$. Broj pokazivača ažuriranih u originalnom algoritmu jednak je broju točaka čije smo uparene bridove izbrisali (posljedicom brisanja točke).

Odredimo sada $\mathbb{E}[X_i]$. Za točku $P \in S \setminus S_i$ neka je $e_P$ brid kojeg siječe $\overline{P_0P}$. Vjerojatnost da je pokazivač od $P$ ažuriran je točno vjerojatnost da je $e_P$ izbrisan tijekom tog koraka. Brišemo $e_P$ ako je jedna od njegove 2 krajnje točke izbrisana u koraku unazad izvršenog algoritma. Budući da je taj vrh izabran uniformno iz $S_i \setminus S_3$, dakle biran između $i-3$ vrha, vjerojatnost je $O(1/i)$. Sumiranjem po svim točkama koje nisu u $S_i$ dobivamo očekivanje

$$\mathbb{E}[X_i] = O(n/k).$$

Prema tome očekivan broj koraka za ažurirati pokazivače u algoritma je zbroj očekivanja po svim koracima što nam daje

$$\sum_{k=4}^{n} \mathbb{E}[X_i] = \sum_{k=4}^{n} O(n/k) = O(n \log n)$$

Budući da su svi ostali koraci, kojih je konstantan broj, imaju linearnu složenost dobivamo da je ukupna složenost algoritma $O(n \log n)$. □

**Teorem 1.1.4.** *Postoji algoritam koji u očekivanoj složenosti $O(n \log n)$ rješava problem pronalaska konveksne ljuske.*

*Dokaz.* Koristimo prije opisani algoritam te mu računamo složenost.

Koraci permutacije i prvotnog postavljanja obostranih pokazivača zahtijevaju $O(n)$ operacija. Također cijena brisanja i stvaranja točaka kroz algoritam je, kao što smo prije komentirali $O(n)$. Sve do sada složenosti ne ovise o poretku točaka u našoj permutaciji te im je ukupna složenost $O(n)$. Iz prijašnje leme očitujemo očekivani broj koraka algoritma za ažuriranje obostranih pokazivača koji je $O(n \log n)$. Dakle ukupna očekivana složenost je

$$O(n) + O(n \log n) = O(n \log n).$$

□

## 1.2 Dualnost

***Geometrijska dualnost*** jedna je od fundamentalnih pojmova u računalnoj geometriji i igra važnu ulogu u dizajniranju algoritama.

**Definicija 1.2.1.** ***Dual*** *točke $P = (a, b)$ je pravac u ravnini s jednadžbom $ax + by + 1 = 0$. Obratno, dual pravca kojemu je jednadžba $ax + by + 1 = 0$ je točka $P = (a, b)$.*

Dualnost u ravnini je preslikavanje koje šalje točke u pravce i pravce u točke. Primijetimo da je to preslikavanje zapravo involucija, dual od duala točke je opet ta ista točka, analogno za pravac.

Ako je točka $P$ udaljena za $d \in \mathbb{R}$ (po euklidskoj udaljenosti) od središta koordinatnog sustava, njen dual (pravac $l$) je okomit na pravac koji prolazi kroz $P$ i ishodište. Brzim računom možemo dobiti tu činjenicu, isto kao i to da je udaljenost $l$ od ishodišta $1/d$. Također $l$ ne prolazi kroz kvadrant koji sadrži $P$. Od sada nadalje, ako je nije drugačije rečeno, točku $O$ definiramo kao ishodište koordinatnog sustava, ovisno o dimenziji koja će biti definirana.

Napominjemo da u definiciji dualnosti u obzir ne uzimamo pravce kroz ishodište, točke u beskonačnosti te točku ishodišta.

**Lema 1.2.2.** *Neka su $P_1$ i $P_2$ točke u ravnini te neka su redom $l_1$ i $l_2$ njihovi duali. Pravac $l$ koji prolazi kroz točke $P_1$ i $P_2$ je dual od točke presjeka pravaca $l_1$ i $l_2$.*

*Dokaz.* Neka je $P_i = (a_i, b_i)$ za $i \in \{1, 2\}$, te neka $l$ ima jednadžbu $Ax + By + 1 = 0$. Kada uvrstimo $P_1$ i $P_2$ dobivamo izraze:

$$Aa_1 + Bb_1 + 1 = 0 \text{ i } Aa_2 + Bb_2 + 1 = 0$$

Zapisano drugačije to daje

$$a_1A + b_1B + 1 = 0 \text{ i } a_2A + b_2B + 1 = 0 \,(1)$$

Neka je presjek $l_1$ i $l_2$ točka $P$, presjek tih pravaca je rješenje (jedinstveno zbog prijašnjih napomena) sustava za dane $x, y \in \mathbb{R}$

$$a_1x + b_1y + 1 = 0 \tag{1.1}$$
$$a_2x + b_2y + 1 = 0. \tag{1.2}$$

A iz (1) dobivamo da je to rješenje upravo $P = (A, B)$ što je dual od $l$. □

Primijetimo da se isti dokaz može provesti i za $d$ dimenzija, naravno imajući na umu da se $d$ $(d-1)$-dimenzionalnih ploha sijeku u jednoj točci, tako da gledamo rješenja $d \times d$ sustava.

Koristeći dualnost, poistovjetit ćemo problem konveksne ljuske s jednim drugim problemom u računalnoj geometriji. Ovo će nam služiti kao priprema za algoritam pronalaska konveksne ljuske točaka u tri dimenzije.

**Definicija 1.2.3.** ***Problem presjeka poluravnina*** *je problem u koji za ulaz uzimamo skup poluravnina $H = \{h_1, h_2, ...h_n\}$. Moramo odrediti presjek tih poluravnina.*

Izlaz bi trebao biti oblika konveksnog poligona ako je presjek neprazan. Tražimo da je izlaz u obliku odgovarajuće liste bridova i vrhova, isto kao i u problemu pronalaska konveksne ljuske. Također napominjemo da od sada pa nadalje, kada pričamo o poluravninama i poluprostorima govorimo o otvorenim skupovima.

Pokazat ćemo da su problemi pronalaska konveksne ljuske može riješiti s pomoću pronalaska presjeka poluravnina. Pretpostavimo da se u konveksnoj ljusci od $S$ iz problema pronalaska konveksne ljuske nalazi ishodište, te ono samo nije u $S$. Ako to nije tako, uzmemo središte opisane kružnice nekog trokuta u $S$ te sve točke u $S$ translatiramo tako da je to središte sada $O$. Za pravac $l$ u ravnini, koji ne prolazi kroz $O$, definirajmo $l^+$ kao poluravninu omeđenu s $l$ koja sadrži $O$. Onda za $l_i$ kao dual $P_i \in S_i$ imamo problem presjeka poluravnina dan s $h_i = l_i^+$.

**Lema 1.2.4.** *Ako neka su $P_1$ i $P_2$ točke i $l_1$ i $l_2$ njima pripadni duali. Vrijedi*

$$P_1 \in l_2^+ \iff P_2 \in l_1^+.$$

*Dokaz.* Neka je $P_i = (a_i, b_i)$ za $i \in \{1, 2\}$. Ako vrijedi $P_1 \in l_2^+$, to je ekvivalentno

$$a_1 a_2 + b_1 b_2 + 1 \geq 0.$$

Primijetimo da je i $P_2 \in l_1^+$ ekvivalentno upravo tom uvjetu što nam daje tvrdnju. □

**Teorem 1.2.5.** *Neka je $S$ takav da* conv($S$) *sadrži $O$, i $O \notin S$. Neka su $P_{k_1}$, $P_{k_2}$ i $P_{k_3}$ vrhovi u* conv($S$)*, koja se pojavljuju tim redom u izlazu. Tada $h_{k_1}$, $h_{k_2}$ i $h_{k_3}$ omeđuju presjek poluravnina $h_i$, te se pojavljuju na rubu od presjeka u tom redoslijedu.*

*Dokaz.* Budući da je presjek $h_{k_1}$, $h_{k_2}$ i $h_{k_3}$ nadskup presjeka svih $h_i$, sam presjek se nalazi unutar definiranog trokuta.

Pretpostavimo da jedan od $h_{k_1}$ nije jedan od rubova poligona stvorenog presjekom $h_i$. Definiramo pravac $l_i$ kao pravac koji inducira $h_i$. Postoji vrh u presjeku $h_{k_1}$, $h_{k_2}$ i $h_{k_3}$ koji je vrh poligona stvorenog presjekom $h_i$. Taj vrh $X$ je induciran s $h_a$ i $h_b$ tako da njihov presjek ne sadrži $l_{k_1}$. Bez smanjenja općenitosti $h_a$ ne sadrži ni dual od $P_{k_1}P_{k_2}$ dok $h_b$ ne sadrži dual od $P_{k_1}P_{k_3}$ jer su oni elementi od $l_{k_1}$. Duali pravaca $l_a$ i $l_b$ koji induciraju $h_a$ i $h_b$ su točke $a$ i $b$ za koje vrijedi $a \notin P_{k_1}P_{k_2}^+$ i $b \notin P_{k_1}P_{k_3}^+$, no to bi značilo da postoji točka dužine $\overline{ab}$ koja se nalazi u presjeku komplemenata od $P_{k_1}P_{k_2}^+$ i $P_{k_1}P_{k_3}^+$, no to znači da $P_{k_1}$ nije na konveksnoj ljusci. Dobivamo kontradikciju čime je tvrdnja dokazana.

Redoslijed dobivamo iz leme 1.2.2. □

Budući da se svaki $h_i$ može dobiti iz $P_i$ u konstantom vremenu, možemo dobiti conv($S$) iz poligona dobivenog rješavanjem problema presjeka poluravnina u linearnoj vremenskoj složenosti. Prema toma ako postoji algoritam koji računa presjek poluravnina, onda nam to daje algoritam za pronalažene konveksne ljuske.

Primijetimo da sve ideje u ovome poglavlju lako možemo proširiti na $d$ dimenzija gledajući sustave jednadžbi više dimenzije. Sljedeće poglavlje koristit će sve ove rezultate ali u 3 dimenzije.

## 1.3 Konveksna ljuska u 3 dimenzije

Cilj ovog odjeljka je razviti randomizirani inkrementalni algoritam za računanje presjeka $n$ poluravnina u 3 dimenzije. Pokazat ćemo da algoritam ima očekivano vrijeme izvođenja od $O(n \log n)$. Primjenom poopćena rezultata iz prijašnjeg odjeljka, dobit ćemo algoritam za računanje konveksne ljuske skupa od $n$ točaka u 3 dimenzije s očekivanim vremenom izvođenja od $O(n \log n)$. Precizno definirajmo dani problem.

**Definicija 1.3.1.** ***Problem presjeka poluprostora*** *je problem u koji za ulaz uzimamo skup poluprostora $H = \{h_1, h_2, ...h_n\}$. Moramo odrediti presjek tih poluprostora.*

Izlaz bi trebao biti u obliku konveksnog poliedra ako je presjek neprazan. Tražimo da je izlaz u obliku grafa koji prikazuje rezultirajući poliedar. Za dani skup $S$ od $n$ poluprostora u 3 dimenzije, definirajmo presjek inter($S$) koji čini (moguće prazan) konveksni poliedarski skup u prostoru. Primijetimo da presjek ne mora biti omeđen. Svaka ploha ovog poliedra nalazi se u ravnini koja omeđuje neku od poluravnina. Pretpostavljamo da je svaka poluravnina zadana linearnom nejednakošću čije su varijable koordinate; pripadna jednakost određuje jednadžbu koja definira ravninu koja omeđuje poluprostor. Budući da je inter($S$) poliedar (kada nije prazan), možemo ga predstaviti kao graf u kojem svaki vrh odgovara vrhu tog poliedra. Vrhovi grafa su međusobno povezani ako pripadni vrhovi na poliedru induciranom inter($S$) leže na pravcu formiranom presjekom dviju poluravnina iz $S$ (pravac je dio nekog brida poliedra).

Kada je inter($S$) neograničen pretpostavljamo, radi jednostavnosti, da postoji točka u "beskonačnosti" koja je zajednička krajnja točka svih polubeskonačnih bridova poliedra. Zadan je skup $S$, a naš cilj je izračunati graf koji predstavlja strane poliedra inter($S$). Taj graf predstavljamo tako da za sve njegove vrhove određujemo njihove položaje (u prostoru), zajedno s povezanostima između vrhova. Svaka strana sadrži sve bridove i točke koje ga definiraju. Dajemo korisnu lemu koja će nam biti važna u analizi složenosti danog algoritma.

**Lema 1.3.2.** *Za svaki konveksan poliedar s n strana vrijedi da mu je broj vrhova* $|V| = O(n)$ *i broj bridova* $|E| = O(n)$.

*Dokaz.* Poznato je da se svaki konveksni poliedar možemo prikazati kao planaran graf, prema tome vrijedi Eulerova formula za planarne grafove. Dakle naš poliedar ima $|F|$ strana, $|V|$ vrhova i $|E|$ bridova, onda vrijedi

$$|V| - |E| + |F| = 2. \quad \text{(Eulerova formula)}$$

Također primijetimo da iz svakog vrh $v \in V$ izlaze barem 3 brida (budući da se radi o poliedru), dakle $\deg(v)$ odnosno stupanj vrha $v$ veći ili jednak je 3. Sumirajući po svim vrhovima i koristeći poznati rezultat iz teorije grafova dobivamo

$$3|V| \leq \sum_{v \in V} \deg(v) = 2|E|$$

$$\Longleftrightarrow$$

$$|V| \leq 2(|E| - |V|) = 2|F| - 4.$$

Iz zadnjeg rezultata dobivamo da je $|V|$ proporcionalan broju strana u poliedru, a kako je i izraz $|E| - |V|$ također takav, dobivamo i da je $|E| = O(n)$. □

Mi ćemo pretpostaviti da svaki vrh u poliedru, osim možda točka u beskonačnosti, ima stupanj točno 3. Dakle nikoje 4 ravnine koje omeđuju poluprostore se ne sijeku u istoj točci. Uočimo da zbog proporcionalnosti broja vrhova, strana i bridova iz prijašnje leme ova pretpostavka zapravo gleda "najgori" slučaj u rješavanju ovog problema. Efektivno kada više od 3 ravnine sijeku neki vrh, smanjuje se broj maksimalnih vrhova u konstrukciji poliedra, s obzirom na broj poluprostora.

Također ćemo definirati susjedstvo strana u poliedru. Budući da poliedar sadrži strane, bridove i vrhove, mi ćemo za svaki par tih struktura držati obostrani pokazivač (par mogu biti i 2 iste strukture). Naime isto kako su vrhovi poliedra susjedni s drugim vrhovima, tako su i strane susjedne s vrhovima koji ih definiraju, u oba slučaja postavljamo obostrane pokazivače u strukturi koju koristimo za prikaz poliedra. Opišimo sada dani algoritam.

Kao i u algoritmu s konveksnom ljuskom u prostoru, prvo nasumično permutiramo skup poluprostora, neka je $h_i$ $i$-ti poluprostor u danoj permutaciji za $1 \leq i \leq n$. Definirajmo $S_i = \{h_1, h_2, ..., h_i\}$. Nakon $i$-tog koraka algoritam bi trebao odrediti inter($S_i$). Tokom $i$-tog koraka, dodaje se $h_i$ u inter($S_{i-1}$), tako stvarajući inter($S_i$). Geometrijski gledano, kada dodajemo $h_i$, zapravo odsijecamo dio inter($S_{i-1}$) koji nije sadržan u $h_i$. Tijekom koraka, neki vrhovi od inter($S_{i-1}$) su izbrisani, a neki novi vrhovi su dodani. Slijedi detaljan opis algoritma.

Prvo zbog jednostavnosti pretpostavimo da je $\{h_1, h_2, h_3, h_4\}$ omeđen, dakle inter($S$) će biti omeđeni poliedar. Zanemarit ćemo poluprostore u $S \setminus S_i$ kojima je presjek sa skupom inter($S \setminus S_{i-1}$) prazan, ti poluprostori su jednostavni za obraditi. Za sve ostale poluprostore, definirajmo $\overline{h}$ kao komplement od $h$. Za vrh $v$ u inter($S_{i-1}$) koji je u $\overline{h}$ kažemo da je u **sukobu** s $h$, također kažemo da je $h$ u sukobu s $v$. Za svaku poluravninu $h$ koja još nije u inter($S_i$) održavat ćemo obostrani pokazivač na vrh u inter($S_i$) s kojom je $h$ u sukobu ($h$ može biti u sukobu s više vrhova, ali je uparen sa samo jednim).

Na početku algoritma možemo u vremenu proporcionalnom broju poluprostora upariti svaki od njih sa jednim od vrhova iz $\{h_1, h_2, h_3, h_4\}$. Proces dodavanja poluravnine $h_i$ da se stvori inter($S_i$) počinje od vrha iz inter($S_{i-1}$) u sukobu s $h_i$. Iz tog vrha započinjemo pretraživanje grafa koji reprezentira inter($S_{i-1}$), pazeći na to da nikada ne uđemo u dio grafa inter($S_{i-1}$) $\cap$ $h_i$. Ovako pronalazimo točke koje su izbrisane dodavanjem $h_i$. Također dodajemo novonastale vrhove. To radimo tako da u trenutku kada nađemo brid čija jedna krajnja točka ulazi u inter($S_{i-1}$) $\cap$ $h_i$ odredimo dani brid te ga presijecamo s $h_i$. Sljedeća lema nam opravdava uparivanja samo jednog vrha u sukobu s pojedinom ravninom.

**Lema 1.3.3.** *Vrhovi koji su izbrisani u trenutku kada je dodan $h_i$ induciraju povezani graf u* inter($S_{i-1}$).

*Dokaz.* Uzmimo suprotnu stranu ravnine koja definira $\overline{h_i}$ i pogledajmo presjek tog poluprostora sa inter($S_{i-1}$). Graf tog novog poliedra $P$ je povezan jer je to zapravo inter($S_{i-1}$) u presjeku s poluprostorom.

Strana od $P$ definirana ravninom od $h_i$ sadrži vrhove i bridove koje ćemo nazvati crvenima. Crveni vrhovi i bridovi tvore ciklus u $P$, dok je ostatak grafa upravo onaj za koji želimo dokazati da je povezan, neka su bridovi i vrhovi tog grafa plavi.

Uzmimo neka 2 plava vrha nazovimo ih $v_1$ i $v_2$. Postoji put od $v_1$ do $v_2$. Ako je na tom putu crveni brid, budući da je on dio neke strane od $P$, možemo drugim dijelom te strane otputovati od jednog do drugog krajnjeg vrha brida koristeći plave bridove i tako maknuti crveni brid iz puta.
Ostaje nam slučaj kada iz nekog plavog vrha idemo u crveni vrh pa opet u neki plavi vrh. Ta 3 uzastopna vrha su sigurno dio jedne strane od $P$, što znači da postoji drugi put između 2 plava vrha koji sadrži samo plave bridove i točke. Dakle možemo maknuti i sve crvene točke iz puta od $v_1$ do $v_2$. Sada smo za proizvoljni izbor točaka u grafu našli plavi put između njih što dokazuje danu tvrdnju. □

Prošla lema nam govori da iz jednog vrha koji je u sukobu s poluprostorom možemo odrediti sve druge vrhove koji su također u sukobu s tim poluprostorom, bez toga da prolazimo kroz sve vrhove grafa inter($S_{i-1}$).

Cijena pretrage proporcionalna je broju vrhova uništenih i broju vrhova dodanih tokom algoritma. Kao i prije, svako brisanje točke možemo naplatiti njenom stvaranju. Pokušajmo odrediti očekivani broj vrhova dodanih u jednom koraku.

**Lema 1.3.4.** *Očekivani broj vrhova dodanih u jednom koraku algoritma je konstantan.*

*Dokaz.* Analizirajmo očekivanje broja vrhova $X_i$ dodan tijekom dodavanja $h_i$. Koristimo analizu unatrag, gdje je broj stvorenih vrhova jednak broju izbrisanih vrhova u izvršavanju algoritma unatrag, kada bi iz inter($S_i$) maknuli neku poluravninu $h$. Koristeći činjenicu da je jednako vjerojatno možemo maknuti bilo koju poluravninu $h$ iz $S_i$ dobivamo

$$\mathbb{E}[X_i] = \frac{\sum_{f \in F} \deg(f)}{i} = \frac{2|E|}{i} = \frac{O(i)}{i}.$$

S $\deg(f)$ označavamo broj vrhova koje ima strana $f$ u poliedru te s $|E|$ trenutačni broj bridova u inter($S_i$). Uočimo da je taj broj isti kao i broj bridova koji omeđuje dana strana. Drugu jednakost onda dobivamo iz toga i činjenice da u planarnog grafu svaki brid ima

točno 2 strane koje omeđuje. U zadnjoj jednakosti koristimo lemu 1.3.2. Prema danoj jednadžbi onda zaključujemo da je očekivani broj dodanih vrhova u svakom koraku konstantan. □

Ovo je bitna činjenica koja će nam trebati za kasniju analizu složenosti.

Ostaje nam potkrijepiti pretpostavku da za svaku poluravninu $h \in S \setminus S_i$ imamo obostrani pokazivač prema vrhu inter($S_{i-1}$) u sukobu s $h$. Ovdje ćemo opisati kako se te informacije mogu održavati te analizirati cijenu tog postupka. Konkretno, moramo specificirati kako se pokazivači za poluravnine u $S \setminus S_i$ ažuriraju nakon dodavanja $h_i$.

Kada izbrišemo vrh $v$ iz inter($S_{i-1}$) tijekom dodavanja $h_i$, provjeravamo postoje li pokazivači s $v$ prema poluravninama u $S \setminus S_i$. Za svaki takav pokazivač, moramo ga premjestiti na novi vrh $w \in h \cap$ inter($S_i$). Kako pronalazimo takav vrh $w$?

Postupak je sličan onome koji koristimo za ažuriranje inter($S_{i-1}$) kako bismo formirali inter($S_i$). Napominjemo da je vrh $v$ u $\overline{h} \cap \overline{h_i}$. Provodimo pretraživanje grafa koji predstavlja inter($S_{i-1}$), počinjemo od $v$ i pritom pazimo da nikada ne uđemo u $h$. Pretragu nastavljamo dok ne stignemo do prvog vrha iz inter($S_i$). Kada stignemo do takvog vrha, označimo ga kao $w$. Budući da vrijedi $w \in \overline{h}$, on je u sukobu s $h$, i premještamo pokazivač sukoba iz $h$ da pokazuje na $w$ i da pokazivač iz $w$ pokazuje na $h$.

Ostaje analizirati cijenu ovog pretraživanja. Ovdje opet koristimo lemu 1.3.3. Naime sama cijena pretraživanja proporcionalna je broju vrhova u $\overline{h} \cap \overline{h_i} \cap$ inter($S_{i-1}$) jer je naša pretpostavka da je stupanj svakog vrha u poliedru točno 3. Zbog toga bridove koje spajaju točke od kojih je jedna u $h$, a druga nije je najviše proporcionalno broju točka u prije definiranom presjeku. Dakle broj vrhova u presjeku je broj uništenih vrhova inter($S_{i-1}$) u sukobu s $h$, plus broj novostvorenih vrhova inter($S_i$) u sukobu s $h$.

Definirajmo skup novonastalih vrhova tijekom dodavanja $h_i$ u $i$-tom koraku s $N$. Broj novonastalih vrhova u sukobu s $h$, sumirano po svim $h \in S \setminus S_i$ nazovimo $X_i$. Želimo ograničiti $X_i$, a on iznosi

$$X_i = \sum_{v \in N} |\{h \in S \setminus S_i : v \text{ je u sukobu s } h\}|. \tag{1.1}$$

**Lema 1.3.5.** *Za očekivanu vrijednost sume svih $X_i$ u svim koracima jedne iteracije algoritma vrijedi*

$$\mathbb{E}\left[\sum_{i=0}^{n} X_i\right] \leq O(n \log n).$$

*Dokaz.* Za skup poluravnina $H$, neka $c(H, h)$ označava broj vrhova inter($H$) koji su u sukobu s $h$. Ponovno koristeći analizu unatrag, prvo razmatramo fiksni skup $S_i$, iz kojeg se nasumično uklanja jedna poluravnina kako bi se dobilo inter($S_{i-1}$). Budući da svaki vrh inter($S_i$) ima stupanj 3, očekivanje izraza (1.1) je stoga ograničen s

$$\frac{3}{i} \sum_{h \in S \setminus S_i} c(S_i, h). \tag{1.2}$$

Svaki vrh brojali smo 3 puta jer ima točno 3 susjedne strane. A za svaki vrh u inter($S_i$) prebrojali smo broj poluprostora s kojima je u sukobu, što je isto kao i da smo za svaki poluprostor prebrojali broj vrhova u inter($S_{i-1}$) koji su u sukobu s njime.

Budući da je $h_{i+1}$ odabran uniformno nasumično iz $S \setminus S_i$, vrijedi

$$\mathbb{E}[c(S_i, h_{i+1})] = \frac{1}{n-i} \sum_{h \in S \setminus S_i} c(S_i, h). \tag{1.3}$$

Povezujući (1.2) i (1.3), očekivanje izraza (1.1) je ograničeno s gornje strane

$$\frac{3(n-i)}{i} \mathbb{E}[c(S_i, h_{i+1})].$$

Slučajna varijabla $c(S_i, h_{i+1})$ također predstavlja očekivani broj vrhova koji se uništavaju dodavanjem $h_{i+1}$, poluravnine dodane u koraku $i + 1$. Stoga je očekivanje zbroja izraza (1.1) po svim $i$ (što mjeri ukupni rad za ažuriranje pokazivača tijekom cijelog algoritma) ograničen odozgo

$$\sum_{i=1}^{n-1} 3(n-i) \cdot \mathbb{E}[\text{Broj uništenih vrhova u trenutku } i+1]. \tag{1.4}$$

Za vrh $v$ stvoren tijekom algoritma, neka $t_c(v)$ označava vrijeme (broj koraka) kada je stvoren, a $t_d(v)$ vrijeme kada je izbrisan. Tada se izraz (1.4) može prepisati kao

$$\left( \sum_v \frac{3(n - t_d(v) - 1)}{t_d(v) - 1} \right) / n!, \tag{1.5}$$

gdje $v$ prolazi kroz sve vrhove stvorene tijekom svake moguće iteracije algoritma (dakle algoritma izvršenog za svaku permutaciju poluravnina). Budući da vrijedi $t_c(v) \leq t_d(v) - 1$, možemo izraz (1.5) ograničiti s gornje strane kao

$$\left( \sum_v 3(n - t_c(v)) \right) / n! \leq \sum_{i=1}^{n-1} \frac{3(n-1)}{i} \mathbb{E}[|\{v : t_c(v) = i\}|].$$

Desna suma, ali s $i$ zapisanim umjesto jedinice u razlomku, dobiva se iz lijeve istom logikom kao i (1.5) iz (1.4). Već smo vidjeli u lemi 1.3.4. da je $\mathbb{E}[|\{v : t_c(v) = i\}|]$ konstanta. Stoga vrijedi

$$\mathbb{E}\left[\sum_{i=0}^{n} X_i\right] \leq \sum_{i=1}^{n-1} \frac{3(n-1)}{i} \mathbb{E}[|\{v : t_c(v) = i\}|] = O(n \log n).$$

□

**Teorem 1.3.6.** *Postoji algoritam koji rješava problem presjeka poluprostora u očekivanoj složenosti $O(n \log n)$.*

*Dokaz.* Prijašnje opisani algoritam za početan zapis obostranih pokazivača između vrhova i poluprostora u sukobu ima linearnu složenost. Očekivani broj koraka u algoritmu potreban za ažuriranje obostranih pokazivača tijekom odsijecanja poliedra $\text{inter}(S_i)$ je zbog prijašnjeg razmatranja i leme 1.3.5. upravo $O(n \log n)$, što je i očekivana složenost samog algoritma. □

Prokomentirajmo još pretpostavku da presjek skupa poluprostora $\{h_1, h_2, h_3, h_4\}$ tvori zatvoreni poliedar. Budući da tražimo konveksnu ljusku od $n$ točaka u prostoru, mi možemo njih pomaknuti tako da smo sigurni da je u njoj sadržano središte (izaberemo neke 4 koje nisu na istoj ravnini i uzmemo točku i njihovoj konveksnoj ljusci). Sada ako gledamo dualni problem, ishodište mora biti sadržano u svim poluprostorima koji su duali od vrhova konveksne ljuske u prostoru opisani kao u poglavlju 1.2. Sada po lemi 1.2.4. i činjenice da je konveksna ljuska poligon koji je omeđen, pa je i presjek duala njenih vrhova omeđen. Prema tome možemo uzeti prva 3 poluprostora u našoj permutaciji, onda mora postojati $h \in S \setminus S_3$ koji s prva 3 tvori tetraedar, a kojega možemo naći u $O(n)$.

# Poglavlje 2

# Dekompozicije ravnina i prostora

U ovom poglavlju pokazujemo kako pomoću randomiziranih algoritama možemo podijeliti dvodimenzionalne i trodimenzionalne prostore tako da te podjele imaju neka korisna svojstva koja kasnije možemo koristiti za alate za rješavanje nekih drugih problema. Prednost ovakvih podjela je to što su brže i služe za unaprijed obraditi neki prostor u kojemu ćemo kasnije raditi daljnje operacije.

## 2.1 Delaunayjeve triangulacije

Vraćamo se na ravninu te gledamo jednu od najkorisnijih podjela ravnina koja se koristi u rješavanju mnogih matematičkih problema. Neka $S = \{P_1, \ldots, P_n\}$ bude skup od $n$ točaka u ravnini.

**Definicija 2.1.1.** *Za točku $P_i \in S$, neka* cell($P_i$)*, poznat kao* ***Voronoijeva ćelija*** *od $P_i$, označava skup svih točaka u ravnini koje su bliže $P_i$ nego bilo kojoj drugoj točki $P_i \in S$, za $j \neq i$.*

**Lema 2.1.2.** *Skup* cell($P_i$) *je (moguće otvoren) konveksan poligon za svaki $P_i \in S$, uz to rubovi koje dijele susjedne ćelije* cell($P_i$) *i* cell($P_j$) *su skupovi točaka jednako udaljenih od $P_i$ i $P_j$. Svi skupovi* cell($P_i$) *dijele ravninu u n konveksnih poligona.*

*Dokaz.* Skup točaka jednako udaljenih od točaka $A$ i $B$ u ravnini je simetrala dužine $\overline{AB}$. Sve točke s iste strane tog pravca kao i točka $A$ su bliže $A$, analogno za $B$. Dakle cell($P_i$) je zapravo presjek poluravnina definiranih kao simetrale dužine između $P_i$ i svake druge točke iz $S$ i usmjerenih prema $P_i$. Kada kažemo da je poluravnina usmjerena prema $P_i$, mislimo na dio ravnine omeđene pravcem koji sadrži točku $P_i$. Presjek poluravnina je (ne nužno zatvoren) poligon. Svaka od ovih $n$ točaka je definirana s takvim poligonom prema tome ravnina je podijeljena u točno $n$ dijelova koji su svi poligoni. □

**Definicija 2.1.3.** *Podjela ravnine opisana u prijašnjoj lemi poznata je kao* ***Voronoijev dijagram*** *skupa $S$, i označavat ćemo ga s* vor($S$).

**Definicija 2.1.4.** *Konveksna poligonalna regija* cell($P_i$) *koja odgovara točki $P_i$ poznata je kao* ***Voronoijeva ćelija*** *točke $P_i$. .*

Pojam Voronoijeva ćelija i dijagrama može se lako formulirati i za točke u višedimenzionalnim prostorima, no ovdje ćemo se fokusirati na točke u ravnini.

Voronoijev dijagram skupa točaka je temeljna struktura u računalnoj geometriji s mnogim primjenama. Ovdje ćemo se baviti algoritmima za konstrukciju vor($S$) i srodnih struktura za zadani skup $P$. Pretpostavljamo da nijedne četiri točke iz $S$ ne leže na istoj kružnici te da nijedne tri točke ne leže na istom pravcu. Ove pretpostavke znatno pojednostavljuju opise algoritama koji slijede, iako se one uz malo truda mogu ukloniti.

**Lema 2.1.5.** *Voronoijev dijagram skupa točaka u ravnini ima niz svojstava:*

1. *Granica između bilo koje dvije ćelije (poznata kao Voronoijev brid) je skup točaka jednako udaljenih od dvije točke iz $S$.*
2. *Promatrajući* vor($S$) *kao planarni graf, svaki vrh grafa ima stupanj* 3.
3. *Ako* cell($P_i$), cell($P_j$) *i* cell($P_k$) *dijele vrh u Voronoijevom dijagramu, tada kružnica koja prolazi kroz $P_i$, $P_j$ i $P_k$ ne sadrži nijednu drugu točku iz $S$.*
4. *Ako je $P_i$ točka iz $S$ koja se nalazi na konveksnoj ljusci skupa $S$, tada je* cell($P_k$) *otvoren.*

*Dokaz.* Iz dokaza prijašnje leme znamo da je dio simetrale dužine između točaka rub ćelije u dijagramu prema tome slijedi tvrdnja pod 1.

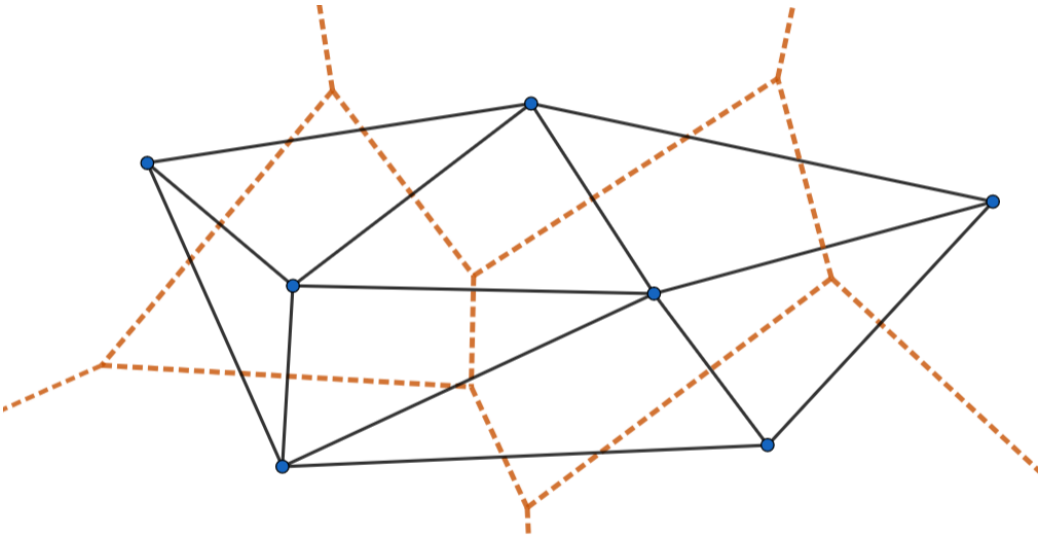
Pretpostavimo suprotno, ako bi neki vrh imao stupanj 4, to bi značilo da je to presjek 4 simetrala dužina pa postoje 4 točke na istoj kružnici što je u kontradikciji s našom pretpostavkom. Ovime dobivamo tvrdnju pod 2.

Pretpostavimo suprotno, nazovimo točku $O$ središte kružnice opisane $\triangle P_iP_jP_k$, ako je točka unutar te kružnice znači da je ona bliža $O$ od ikoje od točaka u trokutu. To bi značilo da je $O$ u Voronoijevoj ćeliji promatrane točke, što bi značilo da cell($P_i$), cell($P_j$) i cell($P_k$) ne dijele vrh u Voronoijevom dijagramu. Dakle dobivamo kontradikciju, što znači da 3. vrijedi.

Neka su $X$ i $Y$ susjedne točke od $P_i$ na konveksnoj ljusci od $S$. Pravac kroz $P_i$ paralelan s dužinom $XY$ nazovimo $l_p$. Povucimo okomiti polupravac na $l_p$ kroz $P_i$ u smjeru u kojem neće sjeći konveksnu ljusku. Svaka točka na tom polupravcu je bliža $P_i$ nego bilo kojoj drugoj točci na konveksnoj ljusci budući da se zbog svojstva ljuske sve drugo točke na njoj nalaze na suprotnom dijelu $l_P$ od $P_i$. Dakle cell($P_i$) sadrži beskonačan polupravac, što znači da je otvoren. □

Definirajmo još jednu vrstu dualnosti, ali ovaj puta između grafova.

**Definicija 2.1.6.** *Promatrajmo* vor($S$) *kao planarni graf čija svaka strana odgovara ćeliji* cell($P_i$) $\in$ vor($S$). *Planarni dual ovog grafa, u kojem se svaki vrh nalazi na točki* $P_i \in S$ *(predstavljajući stranu* cell($P_i$)*), te koji ima brid između dva vrha ako odgovarajuće ćelije dijele brid u* vor($S$)*, poznat je kao* ***Delaunayjeva triangulacija*** *skupa* $S$*, koju označavamo s* del($S$).



Narančaste crte su rubovi vor($S$), a crne su bridovi del($S$)

Iz svojstva 2. leme 2.1.5. slijedi da je del($S$) doista triangulacija (tj. svaka od njegovih strana, osim vanjske, je trokut). Naime vrhovi u vor($S$) su strane u del($S$).

Jasno je da, za zadane $S$ i vor($S$), možemo konstruirati del($S$) u vremenskoj složenosti $O(n)$, budući da kao ulaz od vor($S$) dobivamo graf sa susjedstvom njegovih strana slično kao što je opisano u potpoglavlju 1.3.

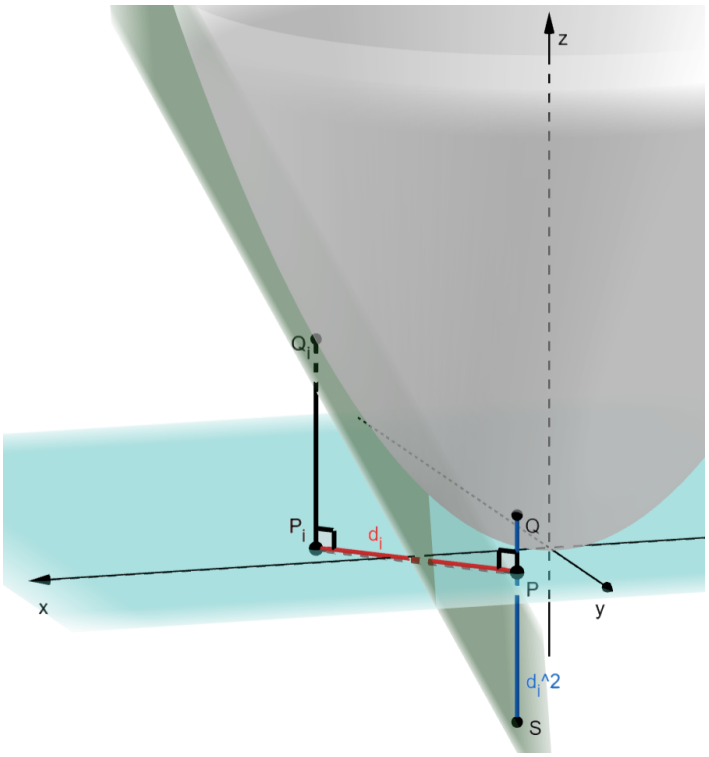
Također postoji više algoritama koji konstruiraju vor($S$) pomoću del($S$) u vremenskoj složenosti $O(n)$. Algoritmi koji izvršavaju ovu konstrukciju znatno su kompleksniji od

prijašnjeg, te ih nećemo opisivati nego samo navesti primjere. Najpoznatija 2 od kojih svaki konstruira vor($S$) pomoću del($S$) su "Hopcroft-Tarjan Planarity Algorithm" [10] i "Boyer-Myrvold Planarity Testing Algorithm" [9].

## 2.2 Konstrukcija Delaunayjeve triangulacije

U ostatku ovog odjeljka usredotočujemo se na algoritme za konstrukciju del($S$). Iz prijašnjeg razmatranja jasno je da će algoritam za izračunavanje vor($S$) implicirati del($S$). Najprije ćemo opisati paraboličnu transformaciju koja reducira problem konstrukcije vor($S$) na problem izračunavanja presjeka $n$ poluravnina u trodimenzionalnom prostoru.

Za zadane točke skupa $S$ u ravnini $xy$, razmatrajmo paraboloid $z = x^2 + y^2$. Označimo s $Q_i$ točku $(x_i, y_i, x_i^2 + y_i^2)$ na površini paraboloida koja je točno iznad ftočke $P_i = (x_i, y_i, 0)$. Neka $h_i$ označava poluprostor koji se nalazi iznad tangencijalne ravnine paraboloida u točki $Q_i$. Razmatrajmo poliedar formiran presjekom svih poluprostora $h_i$.




Primjer konstrukcije te leme 2.2.1.

Prije daljnjeg razmatranja moramo dokazati određene geometrijske rezultate.

**Lema 2.2.1.** *Neka $P$ bude točka u $xy$-ravnini na udaljenosti $d_i$ od $P_i$, i neka $Q$ bude točka na paraboloidu točno iznad $P$. Vertikalna udaljenost (po $z$ osi) između $Q$ i tangencijalne ravnine koja omeđuje $h_i$ je jednaka $d_i^2$.*

*Dokaz.* Formule za ravnine možemo dobiti pomoću svojstva derivacija višeparametarske funkcije $f(x, y) = x^2 + y^2$, presjeke tih ravnina s pravcem okomitim na $xy$-ravninu kroz $P$ je točka koja se dobije jednostavnim računom koji nećemo provoditi. Na kraju pomoću formule za udaljenost točaka dobivamo dani rezultat. □

Sada je pitanje kako doći do triangulacije koristeći ovu naizgled nepovezanu konstrukciju.

**Teorem 2.2.2.** *Neka je $S = \{P_1, \ldots, P_n\}$ i neka je $H = \{h_1, \ldots, h_n\}$ skup poluprostora kao što je opisano gore. Neka* inter($H$) *označava presjek poluprostora iznad tangencijalnih ravnina iz $H$. Primijetimo da je to zapravo poligon kao što je opisano u odjeljku 1.3. Voronoijev dijagram skupa $S$ rezultat je projiciranja bridova poligona* inter($H$) *na $xy$-ravninu.*

*Dokaz.* Neka je $X$ neka točka neke strane poliedra koji je dio ravnine koji definira $h_i$. Nazovimo projekciju $X$ na ravninu $xy$ točkom $T$. Presjek pravca $XT$ s nekom od ravnina koja definira neki $h_i$ nazovimo $k_i$. Zbog činjenice da je poliedar presjek svih poluprostora, točka $X$ je po $z$-osi bliža paraboloidu nego sve druge točke $k_i$. Iz ovoga slijedi da je njena projekcija, gledajući udaljenost u $xy$ ravnini, bliža $P_i$ nego bilo kojoj drugoj točki u $S$, zbog leme 2.2.1.

Iz ovog slijedi da su sve točke projecirane iz strane paraboloida na $xy$-ravninu najbliže upravo točci $P_i$ čija tangencijalna ravnina na paraboloid je stvorila danu stranu u poliedru. □

Primijetimo da trebamo prvo naći presjek svih ovih poluprostora, no taj presjek nije omeđen. Toga problema se rješavamo tako da dodamo ravninu koja je paralelna sa $xy$-ravninom, i udaljena od ishodišta barem duplo više od najveće udaljenosti $P \in S$ od ishodišta (ovu udaljenost možemo dobiti u $O(n)$). Onda nakon dobivanja poliedra možemo zanemariti vrhove, bridove i strane koje je stvorila ta ravnina odnosno poluprostor induciran opisanom ravninom usmjeren prema ishodištu. Zbog leme 1.3.2. projeciranje bridova poliedra na $xy$-ravninu je korak koji također zahtijeva $O(n)$ koraka.

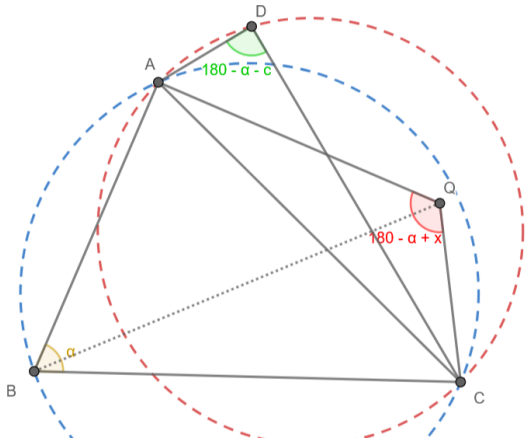
Budući da smo prije opisali algoritam koji rješava problem presjeka poluprostora u očekivanom vremenu $O(n \log n)$, slijedi da za dani skup točaka $S$ možemo dobiti vor($S$) u očekivanom vremenu $O(n \log n)$. U našem prijašnjem razmatranju komentirali smo kako

u linearnoj složenosti iz vor($S$) dobivamo del($S$). Rezultat je naime planaran graf kojemu su vrhovi, bridovi i strane iz del($S$), također za sve elemente tog grafa imamo zapisana međusobna susjedstva. Iz ovoga svega slijedi da i del($S$) možemo dobiti u očekivanoj složenosti $O(n \log n)$.

Što se događa ako za točke iz $S$ vrijedi to da tvore konveksan poligon? Ispada da postoji još brži i elegantniji algoritmi kojima možemo dobiti del($S$) po imenu Chewov algoritam. Sljedeća lema pomoći će nam u analizi složenosti i valjanosti jednog takvog algoritma.

**Lema 2.2.3.** *Neka* del($S$) *označava Delaunayjevu triangulaciju skupa $S$ točaka u ravnini koje čine konveksan poligon. Razmotrimo dodavanje nove točke $Q \notin S$ koja s točkama iz $S$ također tvori konveksan poligon. Promatramo sve trokuta iz* del($S$) *kojima je $Q$ sadržana u opisanoj kružnici. Skup svih tih trokuta čini povezanu komponentu grafa* del($S$) *(u smislu da graf koji induciraju ti trokuti bude povezan).*

*Dokaz.* Neka je $Q$ u kružnici opisanoj nekom trokutu $\delta_1$ iz del($S$). Postoji brid u trokutu $\delta_1$ koji siječe jedna od dužina koja spaja $Q$ s vrhovima od $\delta_1$, nazovimo taj brid $e$. Tvrdimo da ako je $e$ dio drugog trokuta $\delta_2$ iz del($S$), onda je $Q$ i u kružnici opisanoj $\delta_2$.




Dokaz leme 2.2.3.

Uzmimo da su trokuti $\delta_1 = \triangle ABC$ i $\delta_2 = \triangle ACD$, dakle $\overline{AC}$ je njihova zajednička stranica. Označimo $\angle CBA = \alpha$. Budući da je $Q$ u kružnici opisanoj $\triangle ABC$ i na suprotnoj strani $AC$ od $B$ (radi presjeka $\overline{QB}$ s $\overline{AC}$), vrijedi $\angle AQC \geq 180^o - \alpha$. Za točku $D$ znamo da zbog svojstava Delaunayjeve triangulacije nije u kružnici opisanoj $\triangle ABC$, te također znamo da je $D$ s iste strane $\overline{AC}$ kao i $Q$. Prema tome $\angle ADC \leq 180^o - \alpha$. Dobivamo $\angle AQC \geq \angle ADC$, iz čega slijedi da je $Q$ u kružnici opisanoj $\triangle ACD$.

Definirajmo udaljenost $Q$ od trokuta kao najmanju udaljenost $Q$ od svih točaka u trokutu ako je $Q$ u opisanoj kružnici danog trokuta. Inače definirajmo tu udaljenost kao beskonačno. Budući da je $D$ na istoj strani $AC$ kao i $Q$ te je $Q$ u kružnici opisanoj $\triangle ACD$, udaljenost $Q$ od $\triangle ACD$ manja je nego udaljenost od $\triangle ABC$.

Prijašnji postupak možemo ponoviti samo konačan broj puta, radi stalnog smanjenja udaljenosti i konačnog broja trokuta u triangulaciji. Sam postupak staje kada se dogodi da je brid $e$ brid samo jednog trokuta (isti brid opisan na početku dokaza), nazovimo taj trokut $\delta$. Brid $e$ je brid konveksnog poligona koji je "najbliži" $Q$, pa je trokut $\delta$ jedinstven zbog svojstava konveksne ljuske i prijašnje činjenice. Prema tome iz svakog trokuta koji sadrži $Q$ u opisanoj kružnici može se nizom prije opisanih koraka doći do $\delta$, čime je dokaz završen.

□

Sada ćemo pokazati da jednostavan randomizirani algoritam izračunava del($S$) u očekivanom vremenu $O(n)$ kada su točke skupa $S$ vrhovi konveksnog poligona. Zahtijevat ćemo da su točke skupa $S$ dane redoslijedom kojim se pojavljuju na tom konveksnom poligonu.

Algoritam je rekurzivan i izgleda ovako. Odaberemo slučajnu točku $P \in S$, neka $Q$ i $R$ označavaju njezine susjede na granici zadanog konveksnog poligona (ovo nisu susjedi u velikom poligonu nego susjedi u do sad obrađenim vrhovima).

Rekurzivno izračunavamo del($S \setminus \{P\}$), dokle god $|S \setminus \{P\}| > 3$. Nakon što izračunamo del($S \setminus \{P\}$), proširujemo ga kako bismo formirali del($S$) sljedećim koracima:

1. Dodajemo trokut $PQR$ u del($S \setminus \{P\}$). Neka $D$ označava rezultirajući graf.

2. Identificiramo sve trokute del($S \setminus \{P\}$) čije opisane kružnice sadrže $P$, nazovimo ih **loši** trokuti (takvi trokuti više ne mogu biti Delaunayjevi trokuti), koristeći DFS [15] u dualnom grafu del($S \setminus \{P\}$), slično pretraživanju vrhova u sukobu u algoritmu za presjek poluravnina iz odjeljka 1.3. Prema lemi 2.2.3, ti trokuti čine povezanu komponentu del($S \setminus \{P\}$). Neka $L$ označava skup koji se sastoji od ovih "loših" trokuta zajedno s trokutom $PQR$.

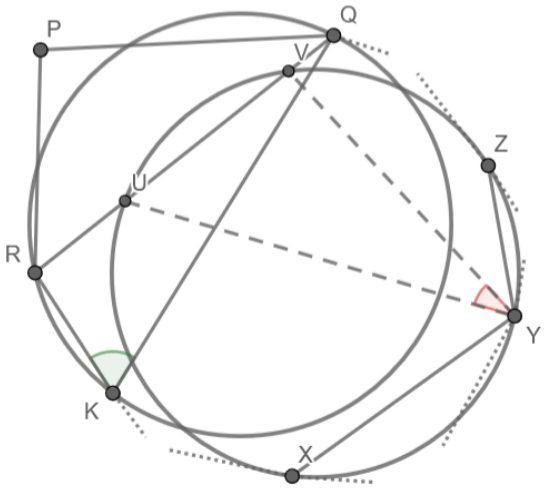
3. Uklonimo iz $D$ sve bridove koji imaju trokute iz $L$ s obje strane i ponovno trianguliramo rezultirajuće strane uvođenjem svih dijagonala koje imaju $P$ kao krajnju točku.

Analizirajmo valjanost (postupak nam uistinu daje del($S$)) i složenost ovog algoritma. Za složenost i valjanost trebati će nam još 2 bitne leme.

**Lema 2.2.4.** *Odrediti trokut od kojeg treba početi pretragu u drugom koraku može se u konstantnom vremenu.*

*Dokaz.* U ovom koraku algoritma, prvi trokut od kojega bi trebali početi pretraživanje u staroj Delaunayjevoj triangulaciji je onaj koji sadrži brid $\overline{RQ}$, nazovimo ga $\delta$. Ako $P$ nije u opisanoj kružnici od $\delta$, znamo da $P$ nije niti u jednoj kružnici opisanoj bilo kojem drugom trokutu u staroj triangulaciji. U suprotnom našli smo trokut iz kojega počinjemo pretragu.

Neke je $K$ treći vrh trokuta čiji je brid $\overline{RQ}$. Za proizvoljan trokut u triangulaciji bez $P$, dokažimo da $P$ nije u njegovoj opisanoj kružnici. Proizvoljan trokut u staroj Delaunayjevoj triangulaciji označimo s $\triangle XYZ$, kojem se vrhovi tim redom u smjeru obrnutom kazaljke na satu pojavljuju na trenutačnom konveksnom poligonu.




Skica 2.2.3.

Bez smanjena općenitosti, zbog svojstava Delaunayjeve triangulacije da se bridovi u njoj međusobno ne sijeku, vrhovi $\triangle XYZ$ pojavljuju se prije $K$, a nakon $Q$ u poretku u konveksnom poligonu (isto je ako je $K$ nakon vrhova $X$, $Y$ i $Z$ u poretku u konveksnoj ljusci, samo zrcalimo skicu). Inače bi, ako je recimo $K$ između $X$ i $Y$, dužina $\overline{RK}$ bi sjekla $\overline{XY}$.

Ako kružnica opisana $\triangle XYZ$ ne siječe pravac $RQ$ onda, $P$ sigurno nije u toj kružnici. Ako ga pak siječe onda siječe dužinu $\overline{RQ}$ u $U$ i $V$ redom, opet jer $R$ i $Q$ ne mogu biti u toj kružnici radi svojstva Delaunayjeve triangulacije.

Neka je $\angle RKQ = \alpha$ i $\angle UYV = \beta$. Vidimo da je $\angle VPU < \angle QPR < 180^o - \alpha$, zbog konveksnosti i pretpostavke da $P$ nije na kružnici. Također $\alpha < \angle RYQ < \beta$ (jer $Y$ nije na kružnici od $\triangle RKQ$ zbog svojstava Delaunayjeve triangulacije) što znači da je $\angle VPU <$

$180^o - \alpha < 180^o - \beta$, što znači da $P$ ne može biti u kružnici opisanoj $\triangle XYZ$.

□

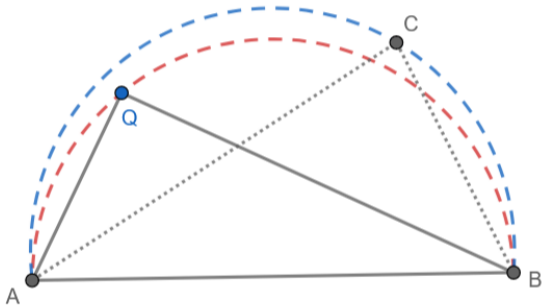
Primijetimo da bez konveksnosti poligona prethodna lema ne vrijedi. Nastavimo s analizom i dokažimo da nakon 2. i 3. koraka opet dobivamo Delauneyjevu triangulaciju.

**Lema 2.2.5.** *Nakon što iz del*$(S)$*, gdje je* $S$ *konveksan poligon, izbrišemo sve bridove čija su oba susjedna trokuta imala* $Q$ *u svojoj opisanoj kružnici (*$Q$ *je takav da je* $S \cup \{Q\}$ *konveksan poligon), te nakon toga spojimo sve vrhove novonastale strane s* $Q$ *dobivamo* del$(S \cup \{Q\})$.

*Dokaz.* Pokažimo prvo da je novi graf ponovno triangulacija. Po lemi 2.2.3. znamo da ćemo brisanjem bridova dobiti jednu novu stranu. Ta strana je također konveksan poligon jer je podskup konveksnog poligona, sada spajanjem $Q$ sa svim vrhovima te strane dobivamo triangulaciju te strane, a ostatak del$(S)$ koji su ostali trokuti ostaje nepromijenjen.

Napomenimo prije nastavka da ćemo opisanu kružnicu trokuta $XYZ$ označavati s $(\triangle XYZ)$, također ako kažemo pravac $AB$, to znači da je to pravac koji prolazi kroz te točke.

Promatrajmo proizvoljan trokut u novo definiranoj triangulaciji koji sadrži točku $Q$, nazovimo ga $\triangle ABQ$. Primijetimo da je brid $\overline{AB}$ također brid u del$(S)$, jer novonastali bridovi su oni s jednim krajnjim vrhom $Q$. Neka je vrh $C$ s iste strane pravca $AB$ kao i $Q$ koji je u staroj triangulaciji tvorio trokut $ABC$. Budući da je $Q$ unutar $(\triangle ABC)$ slijedi da prostor između luka $\overset{\frown}{AB}$ od $(\triangle ABC)$ koji sadrži $C$ i pravca $AB$ obuhvaća prostor između luka $\overset{\frown}{AB}$ od $(\triangle ABQ)$ koji sadrži $Q$ i pravca $AB$. Prema tome niti jedan vrh nije s iste strane pravaca $AB$ kao i $Q$, nije unutar $(\triangle ABQ)$.

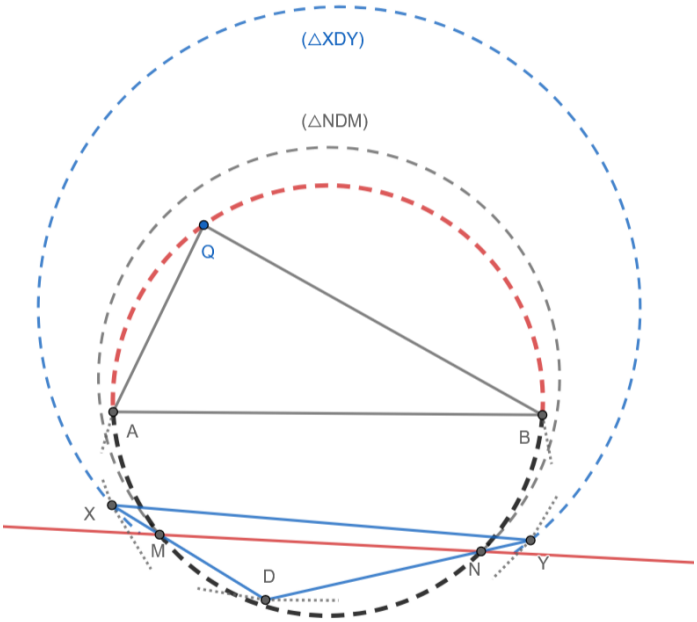



Skica 2.2.5.a.

Sada gledamo vrhove iz $S$ koji su na suprotnoj strani pravca $AB$ od točke $Q$. Nazovimo taj skup $T$. Redoslijed vrhova definiramo onim redom kojim se oni pojavljuju u poligonu

od $A$ do $B$ ispod pravca $AB$. Nazovimo vrhove iz $T$ koje su u $(\triangle ABQ)$ unutarnjima, u suprotnom su vanjski. Vrhovi $A$ i $B$ također podrazumijevamo vanjskim. Preptopstavljamo da postoji barem jedan unutarnji vrh inače smo gotovi. Pokazat ćemo da postoji trokut u novoj triangulaciji čiji us vrhovi redom vanjski, unutarnji pa opet vanjski.

Koristimo indukciju po broju vrhova u $T$. Broj tih vrhova označimo s $n$. Za $n = 3$ (jer $A$ i $B$ moraju biti na početku), vrh između $A$ i $B$ je unutarnji jer smo pretpostavili da postoji barem jedan takav. Neka ova tvrdnja vrijedi za svaki $k \leq n$, $k \in \mathbb{N}$. Neka je $n + 1$ vrhova u $T$, postoji trokut u triangulaciji vrhova iz $T$ koji sadrži brid $AB$, nazovimo zadnji vrh tog trokuta $D$. Ako je $D$ unutarnji vrh onda je dokaz završen. U suprotnom je $D$ vanjski vrh. Gledamo skupove vrhova između $A$ i $D$ te $D$ i $B$, bez smanjenja općenitosti pretpostavimo da postoji unutarnji vrh u skupu vrhova između $A$ i $D$. Pozivanjem na pretpostavku indukcije dokaz je završen.




Skica 2.2.5.b.

Promatrajmo trokut $XDY$ gdje su $X$, $D$ i $Y$ na suprotnoj strani $AB$ od $Q$, i neka je $D$ unutarnji vrh, dok su $X$ i $Y$ vanjski vrhovi. Neka su $M$ i $N$ presjeci $\overline{XD}$ i $\overline{DY}$ s $(\triangle ABQ)$. Budući da je $D$ unutar te kružnice, a $M$ i $N$ na njoj, znamo da prostor između luka $\overset{\frown}{MN}$ od $(\triangle MDN)$ koji ne sadrži $D$ i pravca $MN$ obuhvaća prostor između luka $\overset{\frown}{AB}$ od $(\triangle ABQ)$ koji sadrži $Q$ i pravca $AB$. Sada budući da su $X$ i $Y$ dalje od $D$ nego $N$ i $M$, znamo da prostor između luka $\overset{\frown}{XY}$ od $(\triangle XDY)$ koji ne sadrži $D$ i pravca $MN$ obuhvaća prostor između luka

$\overset{\frown}{MN}$ od ($\triangle MDN$) koji ne sadrži $D$ i pravca $MN$. Iz zadnje dvije tvrdnje zaključujemo da $\triangle XDY$ sadrži $Q$ što je kontradikcija s konstrukcijom nove triangulacije. Prema tome ne postoji unutarnji vrh iz $T$, dakle nova triangulacija uistinu je Delauneyjeva triangulacija. □

Iz prošle leme dobivamo valjanost algoritma s obzirom na korake 2. i 3. Analizirajmo sada složenosti algoritma. Prvi korak u algoritmu uvijek biva izvršen u konstantnom broju operacija.

U drugom koraku pomoću leme 2.2.4. možemo u konstantnom vremenu naći trokut od kojeg krećemo pretragu. Nakon toga korak se može izvršiti u vremenu $O(|L|)$. Taj broj iznosi jedan više od broja bridova uvedenih u trećem koraku (ponovna triangulacija). Za svaki trokut koji smo izbrisali stvaramo novi brid, tj. dijagonalu, a dodatna operacija je radi toga što smo dodali trokut $PQR$. Stoga je očekivani trošak ažuriranja proporcionalan očekivanom stupnju vrha $P$ u del($S$).

**Lema 2.2.6.** *Za proizvoljan konveksan poligon $S$, očekivani stupanj unifomno slučajnog vrha $u$* del($S$) *je konstantan.*

*Dokaz.* Nazovimo deg($v$) stupanj vrha $v$ u del($S$). Budući da je svaka strana osim jedne u grafu del($S$) trokut. Označimo deg($f$) broj bridova koji čine stranu $f$. Sada pomoću deg($f$) $\geq 3$ dobivamo nejednakost

$$3|F| \leq \sum_{f \in F} \deg(f) = 2|E| \quad \text{tj.} \quad |F| \leq 2(|E| - |F|), \text{ gdje}$$

smo u drugoj sumi iskoristili činjenicu da za svaki brid postoje 2 strane. Sada iz Eulerove formule za planarne grafove dobivamo

$$|V| - 2 = |E| - |F| \geq |F|/2.$$

Dakle iz ovoga lako zaključujemo da ako je $|V| = n$ onda je $|E| = O(n)$ i $|F| = O(n)$. Sada, vjerojatnost da izaberemo jedan vrh od njih $n$ je točno $1/n$. Prema tome imamo

$$\mathbb{E}[\deg(v)] = \frac{\sum_{v \in V} \deg(v)}{n} = 2|E|/n = O(n)/n.$$

Dakle očekivanje je uistinu konstantno. □

Sada smo spremni odrediti konačnu očekivanu složenost.

**Teorem 2.2.7.** *Chewov algoritam izvršava se u očekivanom broju koraka $O(n)$.*

*Dokaz.* Iz prijašnje leme dobivamo da je očekivani broj koraka za 2. i 3. korak u algoritmu konstantan. Zbrajajući ovaj očekivani trošak kroz $n-3$ rekurzivnih koraka, zajedno s ostalim konstantnim troškovima ostalih koraka, dobivamo da Chewov algoritam ima očekivanu složenost $O(n)$. □

Dalje ćemo proučavati još neke vrste dekompozicija ravnine.

## 2.3 Dekompozicija trapezima

Naš sljedeći primjer algoritma dekompozicije ravnine (ponekad poznat i kao *vertikalna dekompozicija*) dolazi iz konstrukcije trapezoidne dekompozicije za skup dužina u ravnini. Naime prije smo kao ulaz dobivali točke, sada umjesto točaka dobivamo dužine.

Trapezoidna dekompozicija je osnovna struktura za predstavljanje i manipulaciju rasporedom dužina. Neka $S$ označava skup od $n$ (moguće međusobno presijecajućih) dužina u ravnini; pretpostavljamo da su $x$-koordinate krajnjih točaka svih dužina različite. Neka $k$ označava broj točaka u kojima se presijecaju dvije ili više dužina.

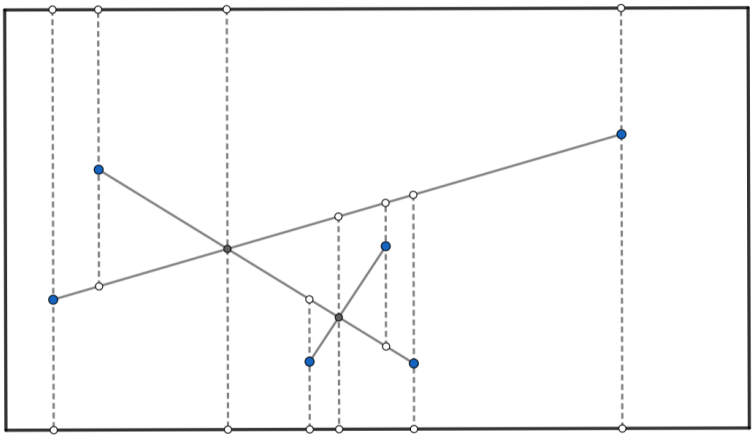
Zamislimo vertikalne linije koje prolaze kroz svaku krajnju točku svake dužine u $S$, kao i kroz svaku od $k$ točaka presjeka. Te vertikalne linije nastavljaju se dok ne naiđu na neku drugu dužinu, gdje se zaustavljaju. Neke od tih linija nastavljaju se u beskonačnost jer ne nailaze ni na jednu drugu dužinu.

Rezultirajuća dekompozicija ravnine poznata je kao **trapezoidna dekompozicija**. Svaka od regija na koje je ravnina podijeljena općenito je trapez s dvije paralelne vertikalne stranice ili trokut (degenerirani trapezi). Naravno, neke regije su beskonačne.

Zamislimo da je područje koje sadrži segmente u $S$ okruženo velikim pravokutnim "okvirom" (njega zovemo graničnom kutijom). Trapezoidnu dekompoziciju skupa $S$ možemo promatrati kao planarni graf, čiji su vrhovi

1. krajnja točka nekog segmenta u $S$
2. točka u kojoj se dva ili više segmenata iz $S$ sijeku
3. točka u kojoj vertikalna linija kroz vrh tipa 1. ili 2. presijeca segment ili okvir.

Važno je napomenuti da strana ovog planarnog grafa može imati proizvoljan broj vrhova, iako je geometrijski oblik trapez.



Primjer dekompozicije, plave točke su vrhovi dužina, presjeci sivi i sjecišta s produljenima bijeli.

Planarni graf stvoren na ovako opisan način im barem $\Omega(n+k)$ bridova. Sljedeća lema pokazuje da ovaj problem ima najmanju očekivanu složenost.

**Lema 2.3.1.** *Računanje danog grafa dekompozicije trapezima zahtjeva barem* $\Omega(n \log n)$ *usporedbi.*

*Dokaz.* Primijetimo da ako imamo algoritam za ovaj problem, onda se s njim može riješiti problemom pronalaska jedinstvenog element u skupu od $n$ brojeva. Naime ako uzmemo $n$ vertikalnih pravaca i neka dva se sijeku, onda skup nema sve jedinstvene elemente, u protivnom ima.

Za problem jedinstvenosti $n$ elemenata u skup [2] znamo da je u najgorem slučaju složenost $\Omega(n \log n)$. Prema tome i složenost danog problema je najmanje $\Omega(n \log n)$. □

Jedan od poznatijih algoritam za računanje ovog problema je "Bentley–Ottmann" algoritam [13], odvija se u vremenskoj složenosti $O((n+k)\log n)$, postoje i algoritmi koji rješavaju problem u složenosti $O(n \log n + k)$, no oni su dosta kompleksni, pa ih nećemo izlagati ovdje. Mi ćemo dati algoritam kojemu je očekivana složenost $O(n \log n + k)$, dakle po našoj prijašnjoj lemi, ako je $k$ dovoljno malen, ovaj algoritam ima najbolju moguću složenost.

Opišimo sada algoritam. Neka trap($S$) označava trapezoidnu dekompoziciju skupa $S$, predstavljenu kao planaran graf u ravnini. U nastavku opisujemo jednostavan algoritam s

nasumičnim inkrementalnim dodavanjem za izračunavanje trap($S$), s očekivanim vremenom izvođenja $O(n \log n + k)$.

Pretpostavimo da niti jedna dužina u ulaznim podacima nije vertikalna. Algoritam prvo nasumično permutira segmente pravaca u $S$. Neka $s_i$ bude $i$-ti segment u ovom nasumičnom redoslijedu. Neka $S_i$ označava skup $\{s_1, s_2, \ldots, s_i\}$. Algoritam se odvija u $n$ koraka, nakon $i$-tog koraka izračunat ćemo trap($S_i$). Tijekom $i$-tog koraka ($i > 1$), dodaje se $s_i$ u trap($S_{i-1}$), čime nastaje trap($S_i$).

Prvo navodimo detalje ovog koraka ažuriranja, a zatim analiziramo vremensku složenost. Za $i > 1$, neka $S \setminus S_i$ označava skup segmenata u $S$ koji se dodaju u inkrementalnoj konstrukciji nakon $i$-tog koraka (tj. skup $\{s_{i+1}, s_{i+2}, \ldots, s_n\}$). Za svaki segment u $S \setminus S_i$, održavamo obostrani pokazivač na stranu trap($S_i$) koja sadrži njegovu lijevu krajnju točku (s obzirom na $x$-os). Tako, za zadana stranu trap($S_i$), možemo u vremenu linearnom u broju takvih krajnjih točaka odrediti sve segmente u $S \setminus S_i$ sadržane u toj strani.

Zatim opisujemo kako se trap($S_{i-1}$) ažurira u trap($S_i$) dodavanjem segmenta $s_i$. Počinjemo identificiranjem strane trap($S_{i-1}$) koja sadrži lijevu krajnju točku $s_i$. Potom se "krećemo" duž $s_i$ prema njegovoj desnoj krajnjoj točki, ažurirajući strukture podataka tijekom prolaza. Pogledajmo kakve sve vrste ažuriranja ćemo trebati provesti.

Povlačimo vertikalnu liniju kroz lijevu krajnju točku $s_i$, određujući obje krajnje točke te vertikalnog pravca (točke gdje ona najprije dodiruje segment u $S_{i-1}$ ili horizontalni rub granične kutije), nazovimo rezultat vertikalnim priključkom za lijevu krajnju točku $s_i$.

Kako nastavljamo duž $s_i$, moramo podijeliti svaku stranu $S_{i-1}$ koje ona presijeca na dvije strane. Konkretno, kad god segment u $S_{i-1}$ bude presječen s $s_i$, izračunavaju se vertikalni priključci za točku presjeka. Kada stignemo do desne krajnje točke $s_i$, ponovno se izračunavaju vertikalni priključci za ovu točku. Zapravo idemo oko svih bridova strane gdje je lijeva točka od $s_i$ sadržana, kada nađemo gdje $s_i$ siječe stranu, podijelimo trenutačnu stranu u 2 nove, prepoznamo stranu u koju je "ušao" $s_i$, te ponovimo isti postupak dok de dođemo do strane koja sadrži najdesniju točku od $s_i$.

Nakon što su izračunati novi vertikalni priključci koji proizlaze iz dodavanja $s_i$, provodi se drugi prolaz kroz rezultirajući planarni graf (označimo ga s $G_i$). Kad god $s_i$ presiječe vertikalni brid trap($S_{i-1}$), jedan dio tog vertikalnog brida se briše, čime se dvije strane $G_i$ spajaju. Ovo je jednostavno spojiti jer u izbrisanom bridu imamo zapisane njegove susjedne strane.

Završni korak ažuriranja uključuje ažuriranje dvosmjernih pokazivača za segmente u $S \setminus S_i$. Potrebno je ažurirati samo pokazivače za segmente čije su lijeve krajnje točke bile sadržane u stranama $\text{trap}(S_{i-1})$ koje $s_i$ presijeca.

Za stranu $f$ iz $\text{trap}(S_{i-1})$, neka $n(f)$ označava broj vrhova $\text{trap}(S_{i-1})$ koji omeđuju $f$, a $l(f)$ označava broj segmenata $S \setminus S_i$ čije lijeve krajnje točke leže u $f$. Odredimo sada očekivani trošak za dodavanje $s_i$.

**Lema 2.3.2.** *Očekivani trošak za dodavanje $s_i$ u i-tom koraku proporcionalan je*

$$\frac{1}{i} \sum_{s \in S_i} \sum_{f \in F(s)} [n(f) + l(f)].$$

*Gdje je $F(s)$ skup svih strana u $\text{trap}(S_i)$ koje sadrže barem jednu točku dužine $s$.*

*Dokaz.* Koristimo analizu unatrag za određivanje očekivanog troška ažuriranja $\text{trap}(S_{i-1})$ kako bismo dobili $\text{trap}(S_i)$. Zamislite da se u $i$-tom koraku segment nasumično odabran iz $\text{trap}(S_i)$ briše. Kao i prije, ovo je valjano jer je bilo koji od $i$ segmenata u $S_i$ jednako vjerojatno označen kao $s_i$ u početnoj nasumičnoj permutaciji.

Micanjem $s$ iz $S_i$ ne rekonstruiramo trenutačni planaran graf, već samo pomoću njega pokušavamo brojati korake koji su napravljeni izvođenjem algoritma u normalnom smjeru.

1. Iz prijašnje rasprave micanje $s$ iz grafa ekvivalentno je njegovom dodavanju, samo sve korake radimo unatrag. Budući da tijekom njegovog dodavanja obilazimo bridove svih strana koje $s$ siječe, taj broj koraka je proporcionalan sa $\sum_{f \in F(s)} n(f)$.

2. Kada bi rekonstruirali prijašnji graf te ga presjekli sa $s$, nakon toga bi za svaku stranu koju $s$ siječe morali ažurirati pokazivače. Budući da $s$ siječe svaku stranu na 2 dijela, te tako prvo stvara 2 nove strane, za svaki pokazivač u staroj strani nam treba konstantno vremena za odrediti u kojoj se novoj strani nalazi. Nakon toga neke strane moramo spojiti, jer nam nestaju neke dužine na kojima su priključci. To spajanje za svaki pokazivač zahtijeva konstantna broj operacija radi toga što znamo sve strane koje se spajaju u jednu, onda samo svaki pokazivač pokazuje na tu novo stvorenu stranu. Iz ovoga slijedi da je posao za ažurirati pokazivače u ovom koraku proporcionalan $\sum_{f \in F(s)} l(f)$.

Iz prijašnje rasprave slijedi tvrdnja.

□

Ostaje ograničiti izraz u terminima $n$ i $k$.

**Lema 2.3.3.** *Očekivana vrijednost dodavanja $s_i$ je proporcionalna $(n + \mathbb{E}[k_i])/k$, gdje je $k_i$ broj presjeka dužina u $S_i$.*

*Dokaz.* Jasno je da je izraz $\sum_{s \in S_i} \sum_{f \in F(s)} l(f)$ proporcionalan ukupnom broju pokazivača za segmente u $S \setminus S_i$, što je $n - k$ (nijedna dva segmenta nemaju krajnje točke s istom $x$-koordinatom, pa se strana $f$ pojavljuje u $F(s)$ za najviše četiri segmenta $s$).

Dalje, primjećujemo da je izraz $\sum_{s \in S_i} \sum_{f \in F(s)} n(f)$ proporcionalan $i + k_i$, gdje je $k_i$ broj točaka u kojima se dva ili više segmenata skupa $S_i$ sijeku. Ovaj rezultat opet slijedi iz toga da se strana $f$ pojavljuje u $F(s)$ za najviše 4 segmenta i leme 1.3.2. (budući da svaki vrh u grafu ima stupanj veći ili jednak 3). Stoga je očekivani trošak ažuriranja prilikom dodavanja $s_i$ proporcionalan $(n + \mathbb{E}[k_i])/i$. □

Ostaje izračunati očekivanje $\mathbb{E}[k_i]$, pod uvjetom da je $S_i$ slučajan podskup od $i$ segmenata iz $S$.

**Lema 2.3.4.** *Očekivanje $\mathbb{E}[k_i]$, gdje je $k_i$ broj presjeka dužina u $S_i$ iznosi $O(ki^2/n^2)$*

*Dokaz.* Neka je $x$ jedna od $k$ točaka u kojima se dva segmenta (recimo $r$ i $q$) skupa $S$ sijeku. Sad, $x$ se pojavljuje u trap($S_i$) ako i samo ako su oba segmenta $r$ i $q$ u $S_i$. Vjerojatnost za to proporcionalna je $i^2/n^2$. Korištenjem linearnosti očekivane vrijednosti $k$ mogućih izbora za $x$ slijedi da je $\mathbb{E}[k_i] = O(ki^2/n^2)$. □

Ovdje eksplicitno koristimo činjenicu da je $S_i$ slučajan podskup $S$, to znači da je svaki podskup od $i$ ima jednaku vjerojatnost da bude izabran u nekoj iteraciji algoritma. To je činjenica koju nismo eksplicitno koristili u prethodnim analizama unazad jer nam nije bila potrebna (koristili smo samo činjenicu da se iz skupa $S_i$ slučajno briše element za analizu unazad).

**Teorem 2.3.5.** *Očekivani trošak konstrukcije trapezoidne dekompozicije od n segmenata u ravnini je $O(n \log n + k)$, gdje je k broj točaka u kojima se dva ili više segmenata sijeku.*

*Dokaz.* Sumirajući troškove ažuriranja preko svih koraka, dobivamo

$$\sum_{i=1}^{n} (n + \mathbb{E}[k_i])/i = \sum_{i=1}^{n} (n/i + O(ki^2/n^2)/i) = O(n \log n) + k \sum_{i=1}^{n} O(i/n^2) = O(n \log n + k),$$

gdje koristimo $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ da iz predzadnje jednakosti dobivamo posljednju. Zbog prethodnog sumiranja, dobivamo rezultat. □

## 2.4 Binarne particije ravnine i prostora

U ovom odjeljku nećemo gledati očekivanu složenost svih operacija, već očekivani broj dijelova u koje će naš algoritam podijeliti ravninu odnosno prostor. Za ulaz od $n$ segmenata htjet ćemo podijeliti ravninu ili prostor tako da svaki od segmenata bude u zasebnom dijelu definirane particije. Uz to bit će nam dani određeni uvjeti, naime ravninu dijelimo segmentima pravaca, a prostor sa segmentima ravnina, ali te ravnine ni pravci nisu proizvoljni. Zašto gledati veličine particije?

Pri prikazu scene na grafičkom terminalu, često se suočavamo sa situacijom u kojoj scena ostaje nepromijenjena, ali se promatra iz više smjerova (na primjer, u simulatoru letenja, gdje simulirano kretanje aviona uzrokuje promjenu točke gledanja). Problem eliminacije skrivenih linija je sljedeći: nakon što saznamo točku gledanja i smjer promatranja, želimo prikazati samo dio scene koji je vidljiv, eliminirajući one objekte koji su "skriveni iza" drugih objekata "ispred" njih u odnosu na točku gledanja. U takvoj situaciji, možemo biti spremni uložiti određeni napor u prethodnu obradu scene kako bismo omogućili brz prikaz s eliminiranim skrivenim linijama za zadani smjer gledanja.

Jedan takav pristup je **binarna particija prostora** u tri dimenzije. Sada, s obzirom na smjer gledanja, koristimo ideju poznatu kao algoritam slikara (painter's algorithm) [7] za prikaz same scene. Ne izlažemo detalje algoritma samo navodimo da njegova složenost ovisi o veličini binarne particije prostora za koju u ovome poglavlju izlažemo randomizirane algoritme koji je stvaraju. Budući da je priprema odvija samo jednom, sama vremenska složenost te pripreme nije toliko bitna. Svako malo mijenjat će se točka gledišta u npr. nekoj igrici koju korisnik igra, te će se svako malo koristiti i slikarov algoritam koji ovisi o veličini binarne particije.

Prikažimo prvo slučaj u dvije dimenzije, te iz njega možemo lako poopćiti slučaj u tri dimenzije.

**Definicija 2.4.1.** ***Binarna particija ravnine*** *sastoji se od binarnog stabla s određenim informacijama koje sada opisujemo. Svaki unutarnji čvor stabla ima dva djeteta. Uz svaki čvor $v$ stabla povezana je regija $r(v)$ u ravnini. Uz svaki unutarnji čvor $v$ od stabla povezan je i pravac $p(v)$ čiji segment presijeca regiju $r(v)$ (onaj dio pravca unutar dane regije). Regija koja odgovara korijenu stabla je cijela ravnina. Regija $r(v)$ dijeli se segmentom pravca $p(v)$ na dvije regije $r_1(v)$ i $r_2(v)$, koje odgovaraju dvama potomcima čvora $v$. Dakle $r(v)$ još uvijek postoji, samo su dodani $r_1(v)$ i $r_2(v)$. Dakle, svaka regija $r$ particije ograničena je dijelovima particijskim pravcima na putu od korijena do čvora koji odgovara regiji $r$ u stablu.*

Naglasimo da regija može sadržavati druge regije, tako da postoje elementi koji mogu biti u više regija odjednom, no mi ćemo te elemente upariti samo s jednom regijom u budućem razmatranju. Reći ćemo da je neki element u svojoj regiji ako je s njome uparen. Naš sljedeći primjer je konstrukcija binarne particije ravnine skupa $n$ ne presijecajućih dužina, problem s primjenama u računalnoj grafici. Definirajmo sada i sami problem.

**Definicija 2.4.2.** ***Problem binarne particije ravnine*** *za ulaz prima skup $S = \{s_1, s_2, \ldots, s_n\}$ od $n$ dužina u ravnini koje se međusobno ne sijeku. Kao izlaz želimo pronaći binarnu particiju ravnine takvu da je najviše jedna dužina (ili dio jedne dužine) uparen s točno jednom regijom iz binarne particije.*

Primijetite da nam definicija omogućuje podijeliti ulaznu dužinu $s_i$ na nekoliko segmenata $s_{i_1}, s_{i_2}, \ldots$, od kojih svaki leži u različitoj regiji. Naglasimo da ako podijelimo ravninu s dijelom pravca $l$, i on siječe neku dužinu $u$ u 2 dijela, onda oba dijela moraju biti u svojoj regiji.

Rješenju ovog problema pristupit ćemo s pomoću posebnog načina particije ravnine.

**Definicija 2.4.3.** *Za skup dužina $S = \{s_1, s_2, \ldots, s_n\}$,* ***autoparticija ravnine*** *je particija koja koristi samo segmente iz skupa pravaca $\{p(s_1), p(s_2), \ldots, p(s_n)\}$.*

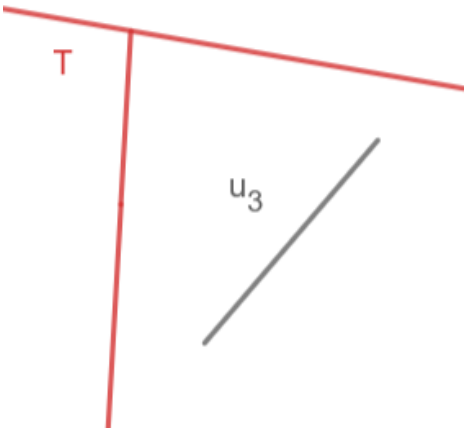
Pomoću autoparticije ravnine rješavamo sam problem, no prvo definirajmo još jednu bitnu definiciju koja će nam biti potreba za daljnje razmatranje.

**Definicija 2.4.4.** *Za dani skup $T$* ***produljenje dužine*** *$s$ je najveća dužina (moguće beskonačna) koja sadrži $s$ i ne sadrži ni jednu točku iz $T \setminus s$. Označavamo je sa $l(s)$.*

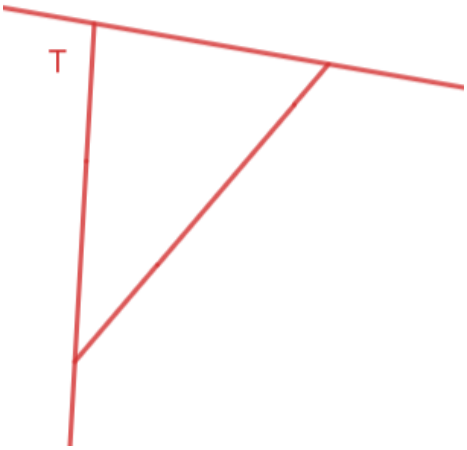
Produljenje zapravo intuitivno konstruiramo tako da iz svake od rubnih točaka $s$ krenemo crtati polupravce (u smjeru suprotnom od druge rubne točke od $s$) dok ne naiđemo na neku točku iz $T$, gdje prestajemo crtati taj polupravac.

Prvo što radimo je permutiramo dužine u $S$, tako da odredimo kojim redoslijedom ćemo ih produživati. Također imamo skup $T$ koji je na početku prazan, on će predstavljati našu trenutačnu particiju ravnine, u obliku svih rubova regija koje tvori dana particija. Nazovimo tu permutaciju $\pi$ i particiju induciranu njome $P_\pi$. U $k$-tom koraku uzimamo dužinu $u$ koja je $k$-ta u permutaciji, te je produljujemo s obzirom na trenutačni $T$ (dok u oba smjera ne siječe rub neke regije koja je trenutačno u podjeli ravnine). Tada u $T$ dodamo $l(u)$, tj. novi $T = T \cup l(u)$ (primjer se može vidjeti na slikama 2.1. i 2.2.). Iz regija koje sadrže $u$ dobili smo nove regija, no napominjemo da nismo izbrisali staru (zbog toga kako funkcionira binarna particija). Budući da je svaki $u$ na rubu 2 nove regije, možemo tu dužinu upariti s jednom od njih po nekom proizvoljnom pravilu. Ovo uparivanje ne remeti kasnije uparivanje jer ne uparujemo nikad dužinu ili dijelove dužine sa starim regijama.

Kada presiječemo neku dužinu koja još nije u $T$, o tome možemo razmišljati kao da smo izbrisali tu dužinu i dodali 2 nove, pa sada za svaku od njih moramo naći para u particiji.

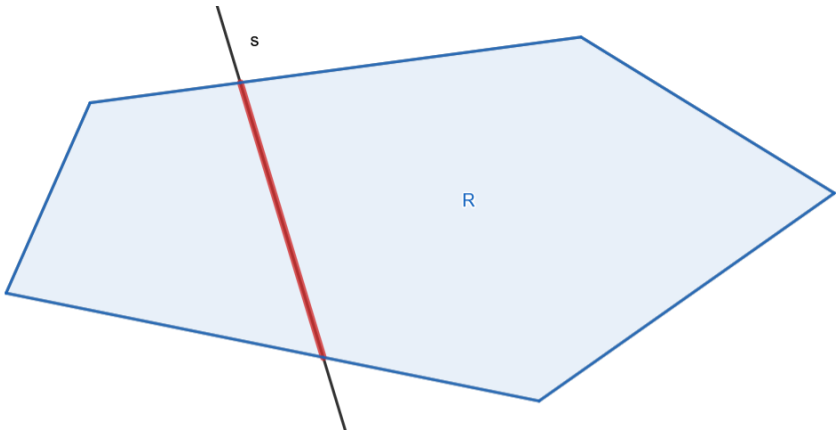



Slika 2.1: Prije produljenja $u_3$




Slika 2.2: Nakon produljenja $u_3$

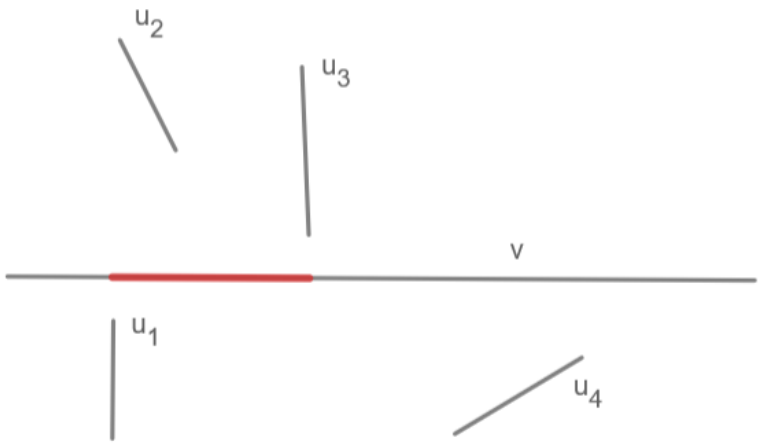
Pretpostavimo da u nekom trenutku algoritma imamo dio ravnine $R$ i dužinu $s$ koja prolazi kroz $R$. Jasno je da je korisno odmah podijeliti $R$ duž $s$ (nismo produžili $s$, samo podijelili $R$ odmah), jer se time sprječava da dio $s \cap R$ bude podijeljen u nekoj kasnijoj fazi. Čim $s$ bude presječen nekim produljenjem trebamo dodatan dio $R$ da smjestimo taj novi dio. Nadalje, ovakva podjela ne povećava ukupan broj podijeljenih dužina, budući da $s$ već dijeli regiju $R$. Takva podjela naziva se slobodni rez. Dopuštamo i slobodan rez u rješavanju našeg problema.




Primjer slobodnog reza

Ovaj algoritam je, kao i svi drugi u ovom radu, izrazito jednostavan. Trik leži u analizi očekivanog broja regija kojega ćemo dobiti. Navedimo prvo pojmove koji nam trebaju.

Za dužine $u$ i $v$, definirajmo **index**$(u, v)$ kao $i$ ako $l(u)$ siječe $i - 1$ drugih dužina prije nego što dosegne $v$, te index$(u, v) = \infty$ ako $l(u)$ ne presijeca $v$. Označimo još i događaj da $l(u)$ presijeca $v$ s $u \dashv v$. Budući da se dužina $u$ može produžiti u dva smjera, moguće je da vrijedi index$(u, v) =$ index$(u, w)$ za dvije različite dužine $v$ i $w$. Uočimo da je broj regija jednak broju presjeka produljenja dužina s drugim dužinama tj. $u \dashv v$ (onda moramo smjestiti dodatnu dužinu u neku regiju). Promotrimo sada što se događa sa slobodnim rezovima.




Slika 2.3

Razmotrimo sliku 2.3. Pretpostavimo da redoslijed zadan nasumično odabranom permutacijom $\pi$ glasi $u_1, u_3, u_4, u_2, v$. Tada je $v$ presječen od strane $u_1$, $u_3$ i $u_4$, ali ne i od $u_2$. Kada je $v$ presječen od $u_1$ i $u_3$, dio $v$ između tih presjeka dijeli regiju i time stvara slobodan rez te regije. Korisno je zamisliti ulaznu dužinu u problemu (poput $v$) kao da je zid između nekih krajnjih točka. Kada se na $v$ naprave dva presjeka, dio između tih presjeka biva "srušen" i uklanja se iz ostatka problema, više nikada neće biti presječen. Preostaju dva dijela zida $v$, svaki izgrađen do jedne krajnje točke. Tijekom daljnje obrade, svaki dio može izgubiti još dijelova zbog presjeka. To se nastavlja sve dok $v$ ne bude izabran u $\pi$, u kojem trenutku $l(v)$ postaje linija koja dijeli regiju (ili regije) koje sadrže preostali dio (ili dijelove) $v$, i $v$ više ne biva presječen.

Za ulazni segment $s$, razmatramo one segmente $u$ takve da $l(u)$ presijeca $s$, i označavamo ih kao $u_1, u_2, \ldots, u_k$ na temelju redoslijeda sjecišta pravaca $l(u_i)$ s $s$, s lijeva na desno. Na-

zovimo taj skup $\text{cuts}(s) = \{u_1, u_2, \ldots, u_k\}$. Proučavamo koliko je vjerojatno da će neki od tih segmenata presjeći $s$ u $P_\pi$.

**Lema 2.4.5.** *Za vjerojatnost događaja $u_i \dashv v$, gdje je $u_i \in \text{cuts}(v)$, vrijedi*

$$P(u_i \dashv v) < \frac{1}{i+1} + \frac{1}{k-i+2} - \frac{1}{k+1}.$$

*Dokaz.* Uočimo $l(u_i)$ presijeca $v$ samo ako $u_i$ prethodi svim $v$, $u_1, u_2, \ldots, u_{i-1}$ ili ako $u_i$ prethodi svim $v$, $u_{i+1}, \ldots, u_k$ u $\pi$. Vjerojatnost prvog događaja je $1/(i+1)$, a vjerojatnost drugog događaja je $1/(k-i+2)$. Oba događaja uključuju događaj da $u_i$ predhodi svima $v$, $u_1, u_2, \ldots, u_k$ u redoslijedu induciranom permutacijom $\pi$, što ima vjerojatnost $1/(k+1)$. Slijedi dana vjerojatnost. □

Sada ćemo dokazati naš glavni rezultat.

**Teorem 2.4.6.** *Očekivani broj regija u prije opisanom algoritmu je $O(n \log n)$.*

*Dokaz.* Neka $C_{u,v}$ bude indikator varijabla koja je 1 ako $u \dashv v$ i 0 inače. Iz prijašnje leme znamo $\mathbb{E}[C_{u_i,v}] = \Pr[u_i \dashv v] < P(u_i \dashv v) < \frac{1}{i+1} + \frac{1}{k-i+2} - \frac{1}{k+1}$. Ako $u_i$ nikad ne siječe $v$ ova vjerojatnost je 0. Veličina $P_\pi$ jednaka je $n$ plus broj presijecanja zbog slobodnih rezova. Prema tome, njezino očekivanje je $n + \mathbb{E}[\sum_u \sum_v C_{u,v}]$, a prema linearnosti očekivanja to je jednako

$$n + \sum_u \sum_v \Pr[u \dashv v] \leq n + \sum_v \sum_{u_i \in \text{cuts}(v)} \frac{1}{i+1} + \frac{1}{k_v - i + 2} - \frac{1}{k_v + 1}.$$

Gdje smo s $k_v$ označili broj elemenata u $\text{cuts}(v)$. Budući da indeks $i$ od $u_i$ u zadnjoj desnoj sumi ne prelazi $k_v$ znamo da vrijedi

$$\sum_{u_i \in \text{cuts}(v)} \frac{1}{i+1} + \frac{1}{k_v - i + 2} - \frac{1}{k_v + 1} \leq \log(k_v) \leq log(n).$$

Pa onda dobivamo da je

$$\sum_v \log(n) \leq n \log n.$$

Budući da je ukupno $n$ dužina u samom skupu. Iz prijašnjeg dobivamo tvrdnju. □

U ovom algoritmu smo mogli zanemariti slobodan rez i dobiti isti rezultat. Dokaz i analiza se iznose vrlo slično kao sa slobodnim rezom, pa to nećemo obraditi. Pokazujemo isti problem poopćen na 3 dimenzije, u tome algoritmu znatno smanjujemo složenost ako imamo slobodan rez. Definirajmo sada spomenute pojmove.

**Definicija 2.4.7.** ***Autoparticija prostora*** *je particija ravnine gdje možemo samo koristiti dijelove od produženih trokuta (u 3 dimenzije) kao ravnine. Ravninu dobivenu produljenjem trokuta $\delta$ nazivamo $h(\delta)$.*

Na analogan način particiji ravnine, možemo govoriti o particiji u trodimenzionalnom prostoru.

**Definicija 2.4.8.** ***Problem binarne particije prostora*** *je problem u kojem kao ulaz primamo skup $T = \{u_1, u_2, \ldots, u_n\}$ od n međusobno ne podudarajućih trokuta u prostoru. Za podjelu ravnine možemo koristit samo dijelove ravnina iz skupa $\{h(u_1), h(u_2), \ldots, h(u_n)\}$. Želimo dobiti particiju prostora u kojoj je svaki trokut (ili dio svakog trokuta, ako ga podijelimo s ravninom) nalazi u svojoj regiji particije.*

Kao i slučaju particije ravnine, isto permutiramo ulaz te danu permutaciji nazivamo $\pi$ i rezultirajuću particiju s $P_\pi$. U tri dimenzije, kada ravnina $h(u)$ siječe trokut $v$, ona može presjeći nekoliko podstrana trokuta $v$ koje leže u različitim regijama već stvorene particije. To su one strane trokuta stvorena kada ravnina presijeca trokut, presjek trokuta i ravnine je pravac koji siječe trokut gledamo u 2 dimenzije. Neka je $Y_k$ ukupan broj dodatnih presjeka koje stvara $u_{\pi(k)}$, dakle $k$-ti trokut u permutaciji, a $Y_{k,u}$ broj tih presjeka na ulaznom trokutu $u \in \{u_1, u_2, \ldots, u_n\} \setminus \{u_{\pi(k)}\}$. Stoga ukupna broj svih "fragmentacija" trokuta (svih podstrana na svim trokutima na kraju algoritma) iznosi

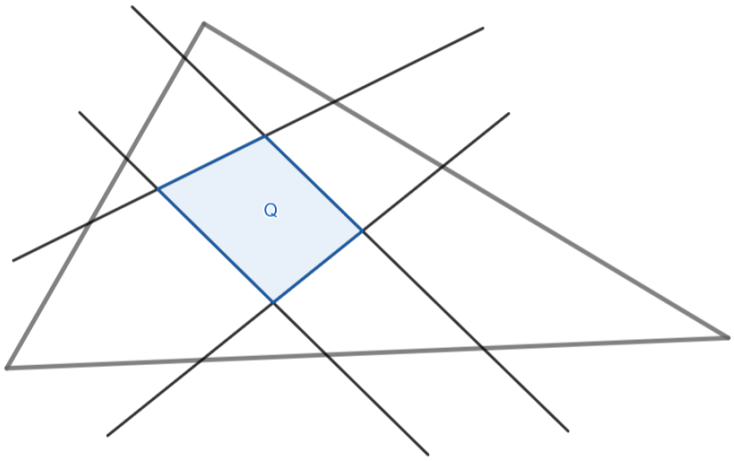
$$\sum_k Y_k = \sum_k \sum_u Y_{k,u}.$$

Primijetimo da je ovo ekvivalentno broju regija u koje je podijeljen prostor u danoj particiji, jer svaka podstrana trokuta mora biti u jednoj regiji (slična logika kao i u dvodimenzionalnom slučaju). Cilj je pokazati da je $\mathbb{E}[Y_{k,u}] = O(1)$, nakon čega rezultat slijedi iz linearnosti očekivanja.

**Lema 2.4.9.** *Vrijedi $\mathbb{E}[Y_{k,u}] = O(1)$.*

*Dokaz.* Za izračunati $Y_{k,u}$ razmatramo sve podstrane trokuta $u$ koje presijeca $h(u_{\pi(k)})$. Promatramo raspored $L_{\pi,k}$ dužina $\{l_{\pi(1)}, l_{\pi(2)}, \ldots, l_{\pi(k)}\}$ na trokutu $u$, gdje je dužina $l_{\pi(i)}$ presjek $h(u_{\pi(i)})$ s trokutom $u$, za $1 \leq i \leq k$ (vidi sliku 2.4.). Bez slobodnih presjeka, podstrane bi bile upravo one regije iz $L_{\pi,k-1}$ koje presijeca $l_{\pi(k)}$. Međutim, zbog slobodnih presjeka u trokutu $u$, bilo koje unutarnje podstrane $L_{\pi,k-1}$ već bi bile "uklonjene". Stoga, $Y_{k,u}$ je broj vanjskih regija koje presijeca $l_{\pi(k)}$.

Za raspored $L$ od $k$ dužina $l_1, l_2, \ldots, l_k$ na trokutu $u$, i za $1 \leq i \leq k$, neka je $x(L, i)$ broj vanjskih regija u rasporedu $L \setminus \{l_i\}$ koje siječe $l_i$. Primijetimo da $\sum_{i=1}^{k} x(L, i)$ odgovara ukupnom broju bridova koji omeđuju vanjske podstrane $L$. Na primjer, na slici 2.4.2,




Slika 2.4: Prikaz gdje 4 pravca sijeku trokut s unutarnjom podstranom Q.

$\sum_{i=1}^{k} x(L,i) = 12$. Sada se pozivamo na standardni rezultat iz kombinatorne geometrije, ukupan suma bridova koji neki pravac siječe u planarnog grafu stvorenom presjekom $n$ pravaca je $O(n)$ [3]. Primijetimo da je naš "veliki" trokut $u$ sačinjen od 3 takva pravca. Ukupan broj bridova koji omeđuju vanjska lica do $L$ jest $3O(k)$ uz još maksimalno $3k$ novostvorenih bridova (to su bridovi koji su sadržani u stranicama trokuta $u$). Prema tome $\sum_{i=1}^{k} x(L,i) = O(k)$ za bilo koji raspored $L$ na trokutu $u$ (ovaj dokaz može se naći i u [12]).

Budući da je $\pi$ slučajna permutacija, $l_{\pi(k)}$ je jednako vjerojatno bilo koja od dužina u rasporedu $L$. Stoga vrijedi

$$\mathbb{E}[Y_{k,u}] = \frac{1}{k}\sum_{i=1}^{k} x(L,i) = O(1).$$

□

Iz prijašnjeg razmatranja i u kombinaciji s lemom 2.4.9. Dobivamo da je očekivana složenost prije opisanog algoritma $O(n^2)$.

# Poglavlje 3

# Algoritmi nad Delauneyjevim triangulacijama

Pronalaženje puta ili usmjeravanje ključno je za niz područja, uključujući geografske informacijske sustave, urbano planiranje, robotiku i komunikacijske mreže. U mnogim slučajevima, informacije o okruženju u kojem se usmjeravanje odvija nisu unaprijed poznate, pa vozilo/robot/paket mora učiti ove informacije putem istraživanja. Algoritmi za usmjeravanje u takvim okruženjima nazivaju se ***online*** algoritmi usmjeravanja.

U ovom poglavlju razmatramo *online* usmjeravanje u sljedećem apstraktnom okruženju: Okruženje je planarni graf s ravnim bridovima, $T$, s $n$ vrhova, čije težine bridova su definirane Euklidskom udaljenošću između krajnjih točaka, izvor $v_{\text{src}}$ i odredište $v_{\text{dst}}$ su vrhovi $T$, a paket se može kretati samo po bridovima $T$. U početku, paket zna samo $v_{\text{src}}$, $v_{\text{dst}}$ i $N(v_{\text{src}})$, gdje $N(v)$ označava skup vrhova susjednih $v$.

Klasificiramo *online* algoritme usmjeravanja na temelju njihove uporabe memorije i/ili nasumičnosti. Definirajmo $v_{\text{cur}}$ kao trenutačni vrh na kojem se paket nalazi. Algoritam usmjeravanja naziva se **algoritmom bez memorije** ako sljedeći korak koji paket poduzima ovisi samo o $v_{\text{cur}}$, $v_{\text{dst}}$ i $N(v_{\text{cur}})$. U svakom koraku algoritma poznate su koordinate $v_{\text{cur}}$ i $v_{\text{dst}}$, te se iz njih mogu dobiti određene dužine i pravci koji će nam pomoći u izvođenju određenog algoritma. Sam algoritam je nasumičan ako je sljedeći korak koji paket poduzima odabran nasumično iz $N(v_{\text{cur}})$. Nasumični algoritam bez memorije koristi distribuciju koja ovisi samo o $v_{\text{cur}}$, $v_{\text{dst}}$ i $N(v_{\text{cur}})$.

Opravdanje za proučavanje zahtjeva za memorijom algoritama usmjeravanja proizlazi iz komunikacijskih mreža, u kojima memorija koju koristi algoritam rezultira dodatnim informacijama u zaglavlju koje putuje s paketom. Budući da se te informacije koriste samo

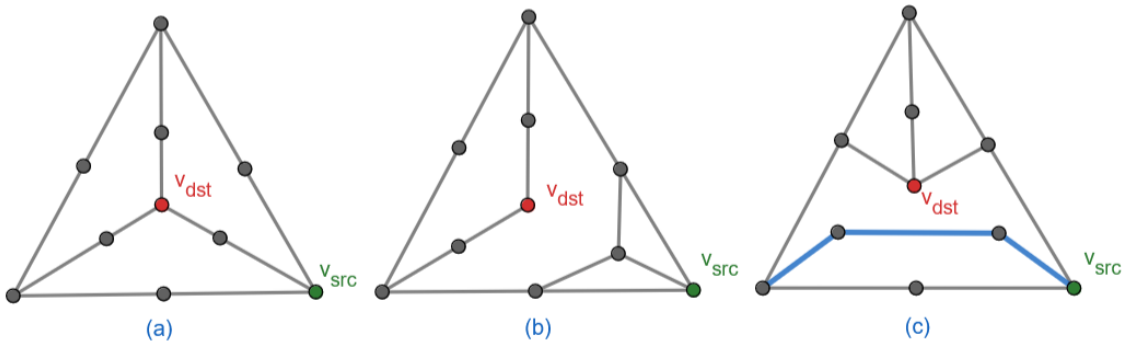
za potrebe usmjeravanja i nemaju koristi za pošiljatelja ili primatelja, one efektivno smanjuju propusnost komunikacije.

Za algoritam $\mathcal{A}$ kažemo da ga graf **savladava** ako postoji par izvor/odredište za koji paket nikada ne stigne na odredište kada krene s izvora. Ako $\mathcal{A}$ pronađe put $P$ od $v_{\text{src}}$ do $v_{\text{dst}}$, onda $P$ nazivamo $\mathcal{A}$-putem od $v_{\text{src}}$ do $v_{\text{dst}}$. Ovdje koristimo pojam puta u intuitivnom smislu, a ne u strogom smislu teorije grafova, jer $P$ može više puta posjetiti isti vrh.

U ovom poglavlju također gledamo posebne slučajeve dekompozicije ravnine i triangulacija koje smo obradili u prijašnjim poglavljima. To su naime Voronoijevi dijagrami i Delaunayjeve triangulacije.

## 3.1 Nedostaci deterministčkih algoritama bez memorije

Treba napomenuti da deterministički algoritmi bez memorije imaju određena bitna ograničenja. Razmotrimo što se događa kada takav algoritam pokušava usmjeravati s jednog od vrhova vanjske strane do $v_{\text{dst}}$ na grafovima prikazanim na Slici 3.1 U svakom od ovih grafova, susjedstva kutnih vrhova izgledaju isto. Stoga, svaki deterministički algoritam bez memorije mora donijeti iste odluke na kutovima u svakom od grafova. Postoje četiri slučaja za razmatranje:




Slika 3.1: Nijedan deterministički algoritam bez memorije ne radi za sve 2-povezane planarne grafove.

1. Na sva tri kuta algoritam odabire koristiti brid konveksne ljuske. U ovom slučaju, algoritam će biti savladan na grafu na Slici 3.1.a jer nikada neće ući u unutrašnjost konveksne ljuske i stoga nikada neće stići do $v_{\text{dst}}$.

2. Na dva od kutova algoritam odabire koristiti brid konveksne ljuske, a na trećem kutu ne koristi. Bez smanjenja općenitosti, možemo pretpostaviti da je treći kut donji desni kut. U ovom slučaju, algoritam će biti savladan na grafu prikazanom na Slici 3.1.b jer je jedini način da stigne do $v_{\text{dst}}$ izvan konveksne ljuske putem jednog od dva puta u druga dva kuta.

3. Na jednom od kutova algoritam odabire koristiti brid konveksne ljuske, a na ostala dva kuta ne koristi bridove od konvesksne ljuske. Bez smanjenja općenitosti, možemo pretpostaviti da je kut koji koristi unutarnji brid gornji kut. U ovom slučaju, algoritam će biti savladan na grafu na Slici 3.1.c jer će završiti zarobljen ciklički se krećući po bridovima prikazanim podebljano.

4. Na svim kutovima algoritam odabire bridove koji nisu bridovi konveksne ljuske. U ovom slučaju algoritam će također biti savladan na grafu na Slici 3.1.c iz istog razloga kao u slučaju 3.

Budući da su grafovi na Slici 3.1 svi 2-povezani, imamo sljedeći negativan rezultat.

**Lema 3.1.1.** *Nijedan deterministički algoritam bez memorije ne radi za sve* 2*-povezane planarne grafove.*

Ipak postoje deterministički algoritmi koji nisu savladani ni od jednog planarnog grafa, ako taj graf ima određena svojstva. Naravno, česti primjeri će biti Delaunayjeve triangulacije.

## 3.2 Pohlepno usmjeravanje

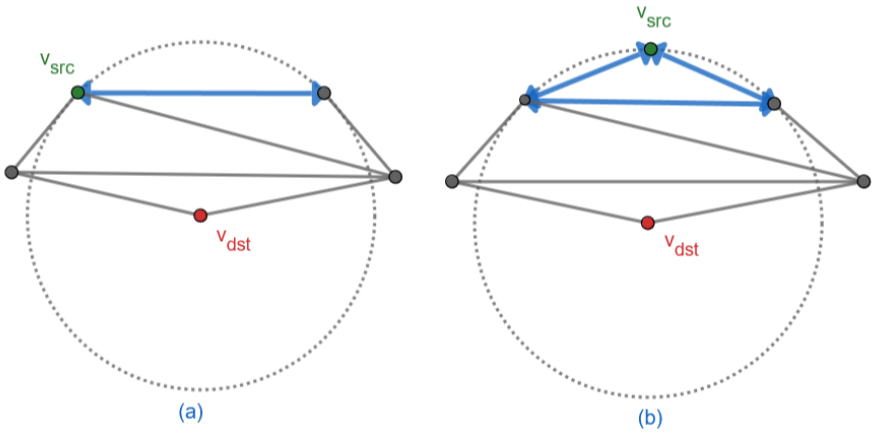
**Algoritam pohlepnog usmjeravanja** uvijek premješta paket na susjeda od vrha $v_{\text{cur}}$, kojeg zovemo $\text{gdy}(v_{\text{cur}})$, koji minimizira $\text{dist}(\text{gdy}(v_{\text{cur}}), v_{\text{dst}})$, gdje $\text{dist}(p, q)$ označava Euklidsku udaljenost između $p$ i $q$. U slučaju izjednačenja, jedan od vrhova odabire se proizvoljno.

Algoritam pohlepnog usmjeravanja može biti svladan triangulacijom $T$ na dva načina (prvi način je važan posebni slučaj drugog):

1. Paket je zarobljen micanjem naprijed/natrag po bridu triangulacije (Slika 3.2.a).

2. Paket može biti zarobljen u ciklusu od tri ili više vrhova (Slika 3.2.b).

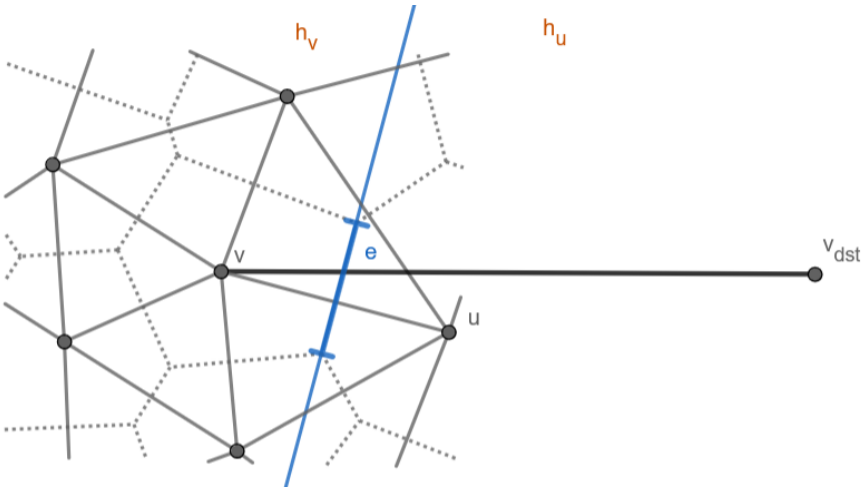
Međutim, kao što pokazuje sljedeći teorem, niti jedna od ovih situacija ne može nastati ako je $T$ Delaunayjeva triangulacija.




Slika 3.2: Triangulacije koje svladavaju pohlepno usmjeravanje

**Teorem 3.2.1.** *Ne postoji skup točaka čija Delaunayjeva triangulacija savladava algoritam pohlepnog usmjeravanja.*

*Dokaz.* Pokazat ćemo da svaki vrh $v$ iz $T$ ima susjeda koji je strogo bliži $v_{\text{dst}}$ nego što je $v$. Stoga, na svakom koraku usmjeravanja, paket se približava $v_{\text{dst}}$ i nakon najviše $n$ koraka stiže do $v_{\text{dst}}$. Pratimo sliku 3.3.




Slika 3.3: Dokaz teorema 3.2.1.

Razmotrimo Voronoijev dijagram VD($T$) vrhova $T$ i neka je $e$ brid najbliži $v$ u VD($T$) koji presijeca dužina čije su krajnje točke $v$ i $v_{\text{dst}}$. Imajte na umu da je $e$ na granici dviju

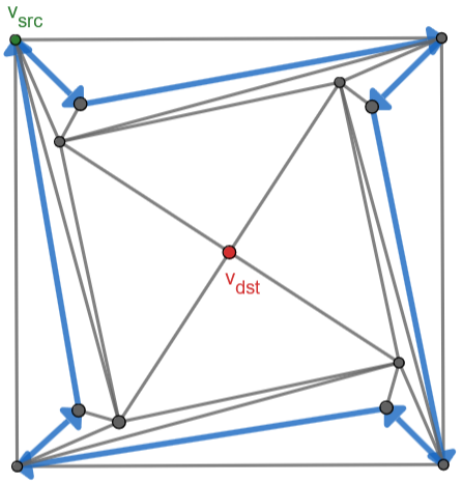
Voronoijevih ćelija, jedne za $v$ i druge za neki drugi vrh $u$, a pravac koji je produljenje od $e$ dijeli ravninu na dvije otvorene poluravnine $h_v = \{p : \text{dist}(p, v) < \text{dist}(p, u)\}$ i $h_u = \{p : \text{dist}(p, u) < \text{dist}(p, v)\}$. Budući da je Voronoijev dijagram dual planarnog grafa Delaunayjeve triangulacije, brid $(u, v) \in T$. Također, prema odabiru $e$, $v_{\text{dst}} \in h_u$, tj. $\text{dist}(u, v_{\text{dst}}) < \text{dist}(v, v_{\text{dst}})$. □

Sljedeći algoritam je isto deterministički, no služit će nam kao baza za jedan sličan randomizirani algoritam sličan njemu.

## 3.3 Usmjeravanje kompasom

**Algoritam usmjeravanja kompasom** uvijek premješta paket na vrh $\text{cmp}(v_{\text{cur}})$ koji minimizira kut $\angle v_{\text{dst}} v_{\text{cur}} \text{cmp}(v_{\text{cur}})$ među svim vrhovima susjednim vrhu $v_{\text{cur}}$ (cmp je susjed od $v_{cur}$). Ovdje se kut uzima kao manji od dva kuta mjerenih u smjeru kazaljke na satu i suprotnom smjeru. U slučaju jednakosti, jedan od (najviše 2) vrhova odabire se pomoću proizvoljnog determinističkog pravila.

Prvim pogledom na algoritam čini se moguće da usmjeravanje kompasom uvijek može biti korišteno za pronalaženje puta između bilo koja dva vrha u triangulaciji. Međutim, triangulacija na slici 3.4. savladava usmjeravanje kompasom. Kada počinjemo od jednog od vrhova na vanjskoj strani od $T$, i usmjeravamo prema $v_{\text{dst}}$, algoritam usmjeravanja kompasom zarobi paket u ciklusu prikazanom podebljanim plavim strelicama.

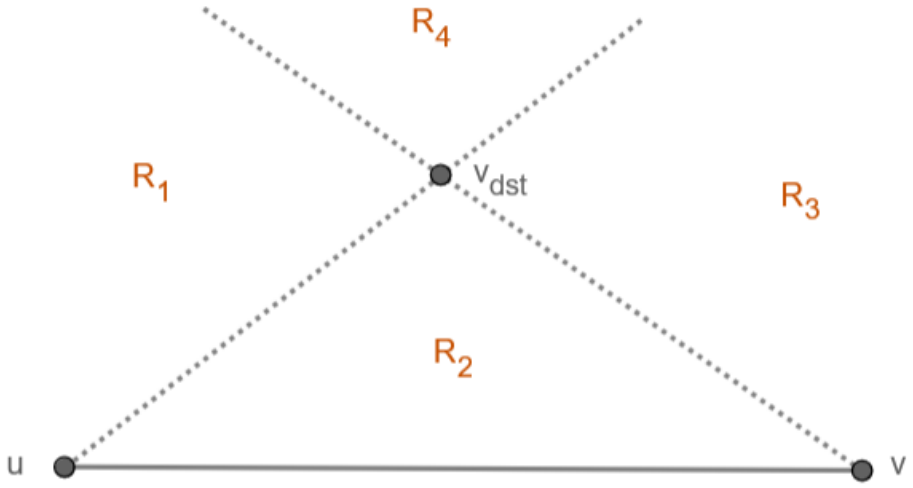



Slika 3.4: Primjer triangulacije koja savladava algoritam usmjeravanja kompasom

Sljedeća lema pokazuje da svaka triangulacija koja savladava usmjeravanje kompasom uzrokuje da paket bude zarobljen u ciklusu.

**Lema 3.3.1.** *Neka je $T$ triangulacija koja savladava usmjeravanje kompasom. Neka $v_{\text{dst}}$ bude vrh za koji usmjeravanje kompasom ne uspijeva usmjeriti paket prema $v_{\text{dst}}$ kada se kao izvor koristi neki drugi vrh. Tada postoji ciklus $C = v_0, \ldots, v_{k-1}$ ($k \geq 3$) u $T$ takav da je $\text{cmp}(v_i) = v_{i+1}$ za sve $0 \leq i < k$.*

*Dokaz.* Budući da $T$ savladava usmjeravanje kompasom i budući da algoritam kompasnog usmjeravanja donosi istu odluku svaki put kada posjeti neki vrh, tada ili postoji brid $(u, v)$ takav da je $\text{cmp}(u) = v$ i $\text{cmp}(v) = u$, ili postoji situacija opisana u lemi. Dokazujemo da ne može postojati takav brid $(u, v)$. Pretpostavimo da takav brid $(u, v)$ postoji.




Slika 3.5: Dokaz leme 3.3.1.

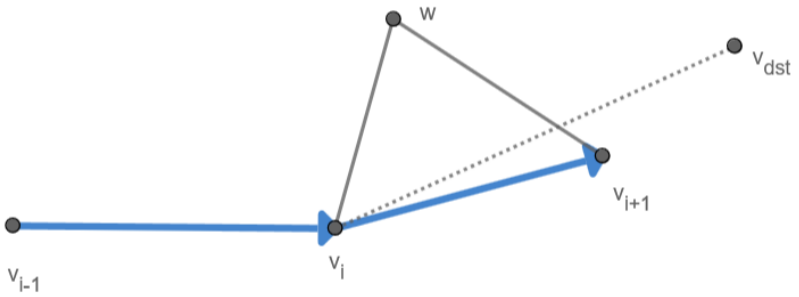
Tada postoji trokut $(u, v, w)$ u $T$ takav da se $w$ nalazi u istoj poluravnini omeđenoj pravcem kroz $u$ i $v$ kao i $v_{\text{dst}}$. Pozivajući se na sliku 3.5, vrh $w$ mora biti u jednoj od regija 1, 2, 3 ili 4. No, to je kontradikcija, jer $w$ ne može biti u regiji 4 jer je onda $v_{dst}$ u trokutu, što krši pravila triangulacije. Ako je pak $w$ u regiji 1, tada je $\text{cmp}(v) = w$. Ako je $w$ u regiji 2, tada je $\text{cmp}(u) = w$ (i $\text{cmp}(v) = w$), a ako je $w$ u regiji 3, tada je $\text{cmp}(u) = w$. □

Takav ciklus $C$ nazivamo **blokirajućim ciklusom** u $T$ za $v_{\text{dst}}$. Sljedeće karakteriziramo blokirajuće cikluse u smislu svojstva vidljivosti triangulacija. Neka su $t_1$ i $t_2$ dva trokuta u $T$. Tada kažemo da $t_1$ **zaklanja** $t_2$ s obzirom na točku gledišta $v_{\text{dst}}$ ako postoji polupravac koji kreće iz $v_{\text{dst}}$, a koji najprije pogađa $t_1$, a zatim $t_2$. Neka su $u$ i $v$ bilo koja dva vrha iz $T$ takva da je $\text{cmp}(u) = v$. Tada definiramo $\triangle uv$ kao trokut iz $T$ koji se nalazi u zatvorenoj poluravnini omeđenoj pravcem $uv$ u kojoj je $v_{\text{dst}}$, koji sadrži brid $\overline{uv}$ (tom trokutu treći vrh

nije nužno točka $v_{\text{dst}}$, već neka točka u poluravnini gdje se nalazi $v_{\text{dst}}$). Dobivamo sljedeću korisnu karakterizaciju blokirajućih ciklusa.

**Lema 3.3.2.** *Neka je $T$ triangulacija koja savladava usmjeravanje kompasom. Neka $C = v_0, \ldots, v_{k-1}$ bude zarobljavajući ciklus u $T$ za vrh $v_{\text{dst}}$. Tada je $\triangle v_i v_{i+1}$ ili identičan $\triangle v_{i-1} v_i$ ili zaklanja $\triangle v_{i-1} v_i$ za sve $0 \leq i < k$. (pod identičan se misli da su trokuti $\triangle v_{i-1} v_i$ i $\triangle v_{i-1} v_i$ isti.)*

*Dokaz.* Pretpostavimo da $\triangle v_i v_{i+1}$ i $\triangle v_{i-1} v_i$ nisu identični; inače je lema trivijalno točna. Neka je $w$ treći vrh $\triangle v_i v_{i+1}$ dakle imamo trokut $w v_i v_{i+1}$ (pogledajte sliku 3.6.).




Slika 3.6: Dokaz leme 3.3.2.

Tada $w$ ne može ležati u dijelu ravnine definiranom s polupravcima $\overrightarrow{v_i v_{\text{dst}}}$ i $\overrightarrow{v_i v_{i+1}}$. Inače bi vrijedilo $\text{cmp}(v_i) = w$. Ali tada dužina koja spaja $w$ i $v_{i+1}$ zaklanja $v_i$, a time $\triangle v_i v_{i+1}$ zaklanja $\triangle v_{i-1} v_i$. Iako na skici nema trećeg vrha $\triangle v_{i-1} v_i$, on nam nije ni bitan jer smo našli polupravac koji prvo pogađa $\triangle w v_i v_{i+1}$ ( to je zapravo $\triangle v_i v_{i+1}$) prije nego $\triangle v_{i-1} v_i$ i to točno u njegovom vrhu $v_i$. □

Postoje triangulacije gdje ovaj algoritam ne biva savladan, te triangulacije imaju "slabija" svojstva od Delauneyjeve triangulacije. Pokažimo primjer triangulacije koja ne savladava opisani algoritam. Regularna triangulacija [16] je triangulacija dobivena ortogonalnom projekcijom strane donjeg dijela poliedra na ravninu. Napominjemo da je Delaunayjeva triangulacija poseban slučaj regularne triangulacije u kojem svi vrhovi poliedra leže na paraboloidu. Edelsbrunner je u [6] pokazao da, ako je $T$ regularna triangulacija, tada $T$ nema skup trokuta koji međusobno zaklanjaju jedan drugog ciklički iz bilo koje točke gledišta. Ovaj rezultat (koji ne dokazujemo), u kombinaciji s lemom 3.3.2, daje glavni rezultat o kompasnom usmjeravanju.

**Teorem 3.3.3.** *Ne postoji regularna triangulacija koja savladava algoritam usmjeravanja kompasom.*

Sljedeće pokazujemo obećani randomizirani algoritam sličan prijašnjem.

## 3.4 Randomizirano usmjeravanje kompasom

U ovom odjeljku razmatramo randomizirani algoritam usmjeravanja koji nije savladan niti jednom triangulacijom. Neka $\mathrm{cw}(v)$ bude vrh u $\mathrm{N}(v)$ koji minimizira kut $\angle v_{\mathrm{dst}}, v, \mathrm{cw}(v)$ u smjeru kazaljke na satu, a neka $\mathrm{ccw}(v)$ bude vrh u $\mathrm{N}(v)$ koji minimizira kut $\angle v_{\mathrm{dst}}, v, \mathrm{ccw}(v)$ u suprotnom smjeru kazaljke na satu. Tada **algoritam randomiziranog usmjeravanja kompasom** (RCR) pomiče paket prema jednom od $\{\mathrm{cw}(v_{\mathrm{cur}}), \mathrm{ccw}(v_{\mathrm{cur}})\}$ s jednakom vjerojatnošću. Prednost ovog algoritma je što je broj mogućih puteva koje bi mogao obići puno veći od prošlog algoritma. To se može gledati na način da ako nam jedno skrentanje ne paše rađe ćemo uzeti drugo. Na takav način može se opovrgnuti dosta kontraprimjera.

Prije nego što možemo iznijeti tvrdnje o tome koje triangulacije savladavaju randomizirano usmjeravanje kompasom, moramo definirati što znači da triangulacija savladava randomizirani algoritam. Kažemo da triangulacija $T$ savladava (randomizirani) algoritam usmjeravanja ako postoji par vrhova $v_{\mathrm{src}}$ i $v_{\mathrm{dst}}$ u $T$ takav da paket koji započinje u $v_{\mathrm{src}}$ s odredištem $v_{\mathrm{dst}}$ ima vjerojatnost 0 da dosegne $v_{\mathrm{dst}}$ u konačnom broju koraka. Napomena: dokazivanje da triangulacija $T$ ne savladava algoritam usmjeravanja bez memorije implicira da paket stiže na svoje odredište s vjerojatnošću 1.

Dobro je poznato da će slučajna šetnja (random walk) naposljetku posjetiti svaki vrh povezanog grafa [14]. Dakle, slučajni hod je randomizirani algoritam usmjeravanja koji nije savladan niti jednim povezanim grafom. Međutim, ovaj rezultat nije zadovoljavajuć iz dva razloga: (1) Budući da slučajni hod ne uzima u obzir odredište, put koji slijedi nije ni približno izravan, i (2) broj slučajnih bitova potrebnih pri svakom koraku slučajnog hoda iznosi $\log d_{\mathrm{cur}}$, gdje je $d_{\mathrm{cur}}$ stupanj trenutačnog vrha. U suštini rute su jako dugačke i koristi se previše memorije. Nasuprot tome, randomizirano usmjeravanje kompasom zahtijeva samo 1 bit pri svakom koraku i vjerojatnije je da će slijediti izravniji put prema odredištu.

Sljedeći teorem pokazuje svestranost randomiziranog usmjeravanja kompasom.

**Teorem 3.4.1.** *Ne postoji triangulacija koja savladava algoritam randomiziranog usmjeravanja kompasom.*

*Dokaz.* Pretpostavimo, suprotno tvrdnji, da postoji triangulacija $T$ koja savladava algoritam randomiziranog usmjeravanja kompasom. Tada postoji vrh $v_{\mathrm{dst}}$ u $T$ i minimalan skup $S$ vrhova takav da:

1. $v_{\mathrm{dst}} \notin S$
2. Podgraf $H$ induciran skupom $S$ je povezan
3. Za svaki $v \in S$ vrijedi $\mathrm{cw}(v) \in S$ i $\mathrm{ccw}(v) \in S$.

Vrh $v_{\text{dst}}$ leži u nekoj strani $F$ podgrafa $H$. Neka $v$ bude vrh na rubu $F$ takav da je dužina $(v, v_{\text{dst}})$ sadržan u $F$. Postojanje takvog vrha $v$ je garantirano, jer ako trianguliramo dani poligon u kojem je vrh $v$, zatim uzmemo trokut u kojem je $v$, taj trokut sigurno postoji, te onda spojimo tu točku s bilo kojim od vrhova trokuta. Budući da je trokut konveksan skup onda je i ta dužina sadržana u trokutu prema tome i u poligonu. Jača tvrdnja može se naći u [4].

Dakle $\overline{v_{dst}v}$ je u $F$. Dva susjeda vrha $v$ na rubu $F$ moraju biti $\text{cw}(v)$ i $\text{ccw}(v)$, i to ne mogu biti isti vrhovi (budući da $F$ sadrži $(v, v_{\text{dst}})$ u svojoj unutrašnjosti). Prema definicijama $\text{cw}(v)$ i $\text{ccw}(v)$, te činjenici da je $T$ triangulacija, trokut $(\text{cw}(v), v, \text{ccw}(v))$ je u $T$, no tada je $F$ taj trokut. Ali to je kontradikcija, jer tada $v_{dst}$ nije i u jednom trokutu triangulacije $T$.

□

Do sada smo razmatrali samo pitanje mogu li algoritmi usmjeravanja pronaći put između bilo koja dva vrha u $T$. Prirodan smjer za daljnje istraživanje je razmatranje duljine puta koji pronađe algoritam usmjeravanja. Kažemo da je algoritam $\mathcal{A}$ **c-kompetitivan** ako za svaki $T$ i za bilo koji par $(v_{\text{src}}, v_{\text{dst}})$ u $T$ duljina (zbroj duljina bridova) puta između $v_{\text{src}}$ i $v_{\text{dst}}$ koji pronađe $\mathcal{A}$ nije veća od $c$ puta duljine najkraćeg puta između $v_{\text{src}}$ i $v_{\text{dst}}$ u $T$. U slučaju randomiziranih algoritama, koristi se očekivana duljina puta. Kažemo da $\mathcal{A}$ ima **omjer kompetitivnosti** $c$ ako je $c$-kompetitivan.

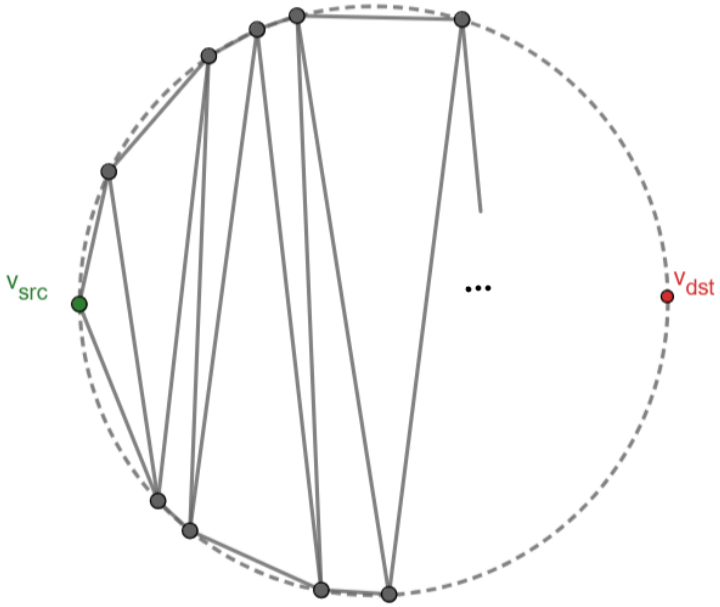
Sljedeći odjeljak bavi se pitanjima o omjeru kompetitivnosti algoritama opisanih do sada, kao i novog algoritma posebno osmišljenog za Delaunayjeve triangulacije.

## 3.5 Negativni rezultati

Nije teško osmisliti triangulacije za koje nijedan od naših algoritama nije $c$-kompetitivan za bilo koju konstantu $c$. Stoga je prirodno ograničiti našu pažnju na dobro definiranu klasu triangulacija. Nažalost, čak ni za Delaunayjeve triangulacije nijedan od do sada opisanih algoritama nije $c$-kompetitivan.

**Teorem 3.5.1.** *Postoje Delaunayjeve triangulacije za koje nijedan od algoritama pohlepnog usmjeravanja, usmjeravanja kompasom ili randomiziranog usmjeravanja kompasom nije c-kompetitivan za bilo koju konstantu c.*

*Dokaz.* Počinjemo s pohlepnim usmjeravanjem. Razmotrimo skup točaka postavljenih na kružnicu koje su zatim triangulirane kako bi se dobila cik-cak triangulacija $T$, prikazana na slici 3.7. Budući da su točke na istoj kružnici, ovo je valjana Delaunayjeva triangulacija (u 2. poglavlju smo radi jednostavnosti zanemarili uvjet da više od 3 točke ne mogu biti na istoj kruržnici). Točke su postavljene tako da svaki vrh $v$ ima susjeda na suprotnoj strani pravca kroz $v_{\text{src}}$ i $v_{\text{dst}}$ koji je bliži $v_{\text{dst}}$ od dva susjeda na istoj strani pravca.



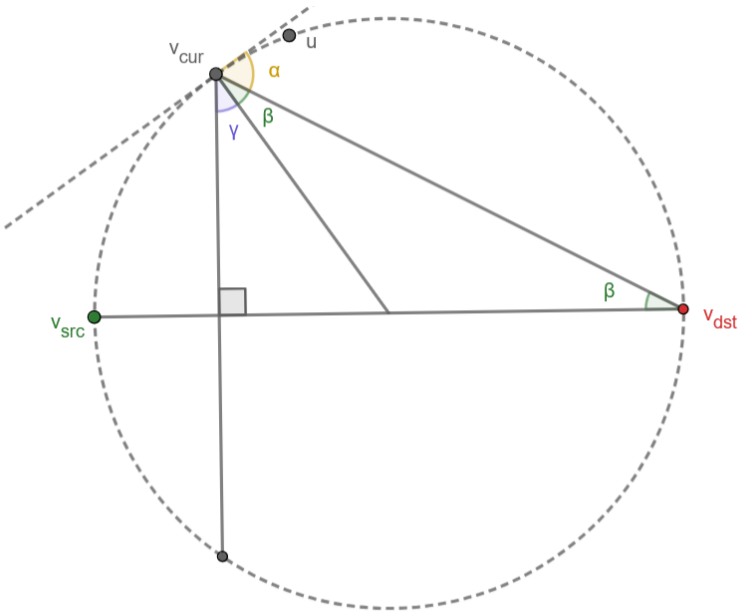

Slika 3.7: Dokaz teorema 3.5.1.

Napomenimo da postoji put između $v_{\text{src}}$ i $v_{\text{dst}}$ duljine približno $(\pi/2) \cdot \text{dist}(v_{\text{src}}, v_{\text{dst}})$, što je gornja granica duljine najkraćeg puta između $v_{\text{src}}$ i $v_{\text{dst}}$. Duljina "cik-cak" puta koji koristi dijagonale $T$ između $v_{\text{src}}$ i $v_{\text{dst}}$ iznosi $\Theta(n) \cdot \text{dist}(v_{\text{src}}, v_{\text{dst}})$, a ovo je put koji prati pohlepni algoritam usmjeravanja. Stoga pohlepno usmjeravanje nije $c$-kompetitivno za ovu triangulaciju.

Kako bismo pokazali da usmjeravanje kompasom nije $c$-kompetitivno, ponovno razmatramo skup točaka na jednoj kružnici i konstruiramo cik-cak triangulaciju. Neka je $v_{\text{cur}}$ bilo koja točka na kružnici s promjerom $\overline{v_{\text{src}} v_{\text{dst}}}$. Razmotrimo kut $\alpha$ između tangente koja prolazi kroz $v_{\text{cur}}$ i pravca kroz $v_{\text{src}}$ i $v_{\text{dst}}$. Usporedimo to s kutom između pravca okomitog na $v_{\text{src}}$ i $v_{\text{dst}}$ koji prolazi kroz $v_{\text{cur}}$ i pravca kroz $v_{\text{src}}$ i $v_{\text{dst}}$. Pozivajući se na sliku 3.8, imamo

$$\alpha = \pi/2 - \beta,$$
$$\gamma = \pi/2 - 2\beta,$$

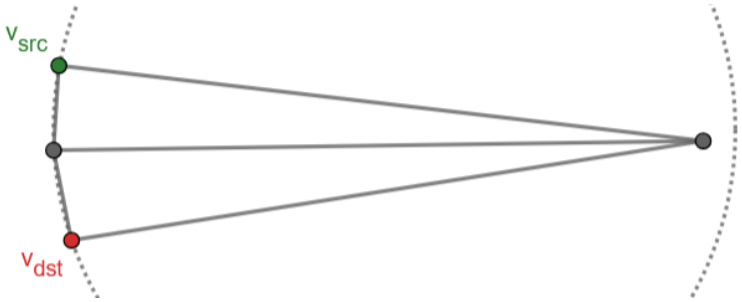
pa vrijedi $\gamma + \beta = \pi/2 - \beta = \alpha$, tj. dva kuta su jednaka. Prijašnji rezultat dobivamo pomoću poznatog teorema o kutu između tangente i tetive te talesovog poučka. Napomenimo da skica ne prikazuje graf već služi kao ideja za njegovu konstrukciju. Sada, postavljanjem točke $u$ na kružnicu blizu $v_{\text{cur}}$, možemo postići da $\angle u, v_{\text{cur}}, v_{\text{dst}} = \alpha - \epsilon$ za proizvoljno mali $\epsilon > 0$. Slično, postavljanjem točke $v_{\text{nxt}}$ na suprotnu stranu kružnice možemo postići da $\angle v_{\text{nxt}}, v_{\text{cur}}, v_{\text{dst}} = \alpha - \epsilon - \delta$ za proizvoljno mali $\delta > 0$, tako da je $\text{cmp}(v_{\text{cur}}) = v_{\text{nxt}}$. Budući da




Slika 3.8: Dokaz teorema 3.5.1.

$\epsilon$ i $\delta$ mogu biti proizvoljno mali, ovu konstrukciju možemo ponavljati koliko god želimo, čime put usmjeravanja kompasom može postati proizvoljno dug.

Kako bismo pokazali da randomizirano usmjeravanje kompasom nije $c$-kompetitivno, razmotrimo konfiguraciju točaka poput one na slici 3.9. Postavljanjem $v_{\text{src}}$ i $v_{\text{dst}}$ gotovo kolinearnim s trećom točkom moguće je proizvesti proizvoljno dugačke tanke trokute čiji omjer najkraće stranice s najduljom ili drugom najduljom stranicom biva proizvoljno velik.




Slika 3.9: Dokaz teorema 3.5.1.

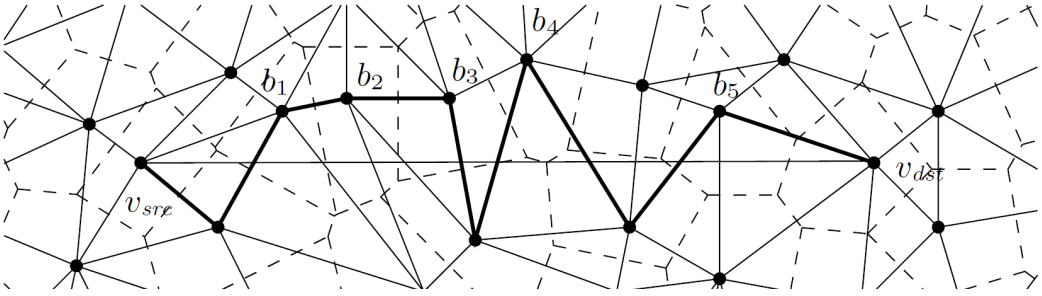
Nadalje, u ovoj konfiguraciji vjerojatnost da je put randomiziranog usmjeravanja kompasom izabrao dulju od 2 stranice od izvora iznosi $1/2$, pa očekivana duljina puta randomiziranog usmjeravanja kompasom također može postati proizvoljno velika. □

Budući da nijedan od algoritama opisanih u odjeljku 2 nije kompetitivan, čak ni za Delaunayjeva triangulacije, očito se postavlja pitanje postoji li algoritam koji je kompetitivan za Delaunayjeve triangulacije. U sljedećem odjeljku odgovaramo na to pitanje potvrdno. Zapravo, dokazujemo još jači rezultat dajući algoritam koji pronalazi put čiji je trošak najviše konstanta puta dist($v_{\text{src}}, v_{\text{dst}}$).

## 3.6 C-kompetitivan algoritam za Delaunayjeve triangulacije

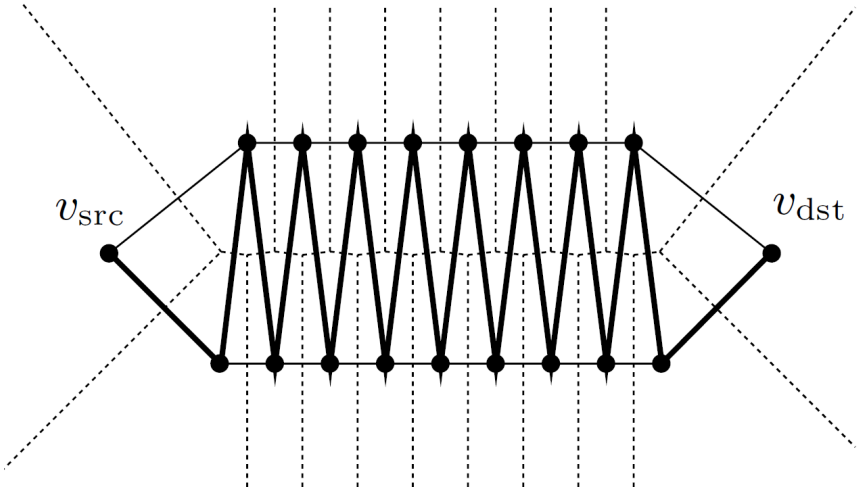
Algoritam koji ćemo sada opisati temelji se na izvanrednom dokazu Dobkina, Friedmana i Supowita [5] da Delaunayjeva triangulacija aproksimira potpuni euklidski graf unutar konstantnog faktora u smislu duljine najkraćeg puta. Detalje se mogu vidjeti i u [8]. U nastavku ćemo koristiti oznaku $x(p)$ (odnosno, $y(p)$) za označavanje $x$-koordinate (odnosno, $y$-koordinate) točke $p$ te oznaku $|X|$ za označavanje euklidske duljine puta $X$.

Razmotrimo dužinu od $v_{\text{src}}$ do $v_{\text{dst}}$. Ta dužina presijeca regije Voronoijeva dijagrama nekim redoslijedom, recimo $R_0, \ldots, R_{m-1}$, gdje je $R_0$ Voronoijeva ćelija za $v_{\text{src}}$ i $R_{m-1}$ Voronoijeva ćelija za $v_{\text{dst}}$. Algoritam **Voronoijeva usmjeravanja** za Delaunayjeva triangulacije pomiče paket od $v_{\text{src}}$ do $v_{\text{dst}}$ duž puta $v_0, \ldots, v_{m-1}$, gdje je $v_i$ točka koja definira $R_i$. Ovaj put nazivamo **Vornojievom putanjom**. Primjer puta dobivenog Voronoijevim algoritmom usmjeravanja prikazan je na slici 3.10. Budući da se Voronoijeva ćelija vrha $v$ može izračunati samo s obzirom na susjede $v$ u Delaunayjevoj triangulaciji, slijedi da je Voronoijev algoritam usmjeravanja algoritam s memorijom $O(1)$. Naime to znači da se u jednom koraku algoritma pamte koordinate $v_{src}$ i $v_{dst}$ i dužina koju te točke tvore.




Slika 3.10: Primjer puta u algoritmu Voronoijeva usmjeravanja. (Vernojieva putanja)

Međutim, sam Voronoijev algoritam usmjeravanja nije $c$-kompetitivan za sve Delaunayjeve triangulacije, kao što je vidljivo iz slike 3.11




Slika 3.11: Algoritam Voronoijeva usmjeravanja nije c-kompetitivan za Delauneyjeve triangulacije.

Ipak, ima neka svojstva koja nam omogućuju izvedbu $c$-kompetitivnog algoritma. Neka je $T'$ graf dobiven iz $T$ uklanjanjem svih bridova $T$ koji presijecaju dužinu $(v_{\text{src}}, v_{\text{dst}})$, a neka je $F$ strana grafa $T'$ koja sadrži $v_{\text{src}}$ i $v_{\text{dst}}$ kao i dužinu koja ih spaja. Pretpostavimo, bez smanjenja općenitosti, da $v_{\text{src}}$ i $v_{\text{dst}}$ leže na $x$-osi i da vrijedi $x(v_{\text{src}}) < x(v_{\text{dst}})$. Sljedeće dvije leme slijede iz rada Dobkina, Friedmana i Supowita [5].

**Lema 3.6.1.** *Voronoijev put je x-monoton, tj. vrijedi $x(v_i) < x(v_j)$ za sve $i < j$.*

**Lema 3.6.2.** *Neka je $P'$ kolekcija svih maksimalnih podputeva iz $v_0, \ldots, v_{m-1}$ koji ostaju iznad x-osi, tj.*

$$P' = \{v_i, \ldots, v_j : y(v_{i-1}) < 0 \;\; i \;\; y(v_{j+1}) < 0 \;\; i \;\; y(v_k) \geq 0 \;\; \textit{za sve } i \leq k \leq j\}.$$

*Tada vrijedi*

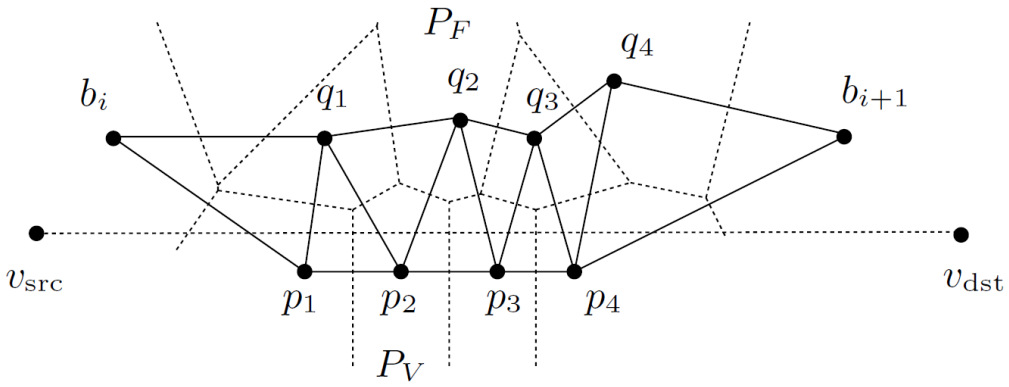
$$\sum_{X \in P'} |X| \leq \left(\frac{\pi}{2}\right) \cdot \text{dist}(v_{\text{src}}, v_{\text{dst}}).$$

Dokazi obije leme definiraju točke $p_i$ na $x$-osi koje se nalaze na granici između $\text{vor}(v_{i-1})$ i $\text{vor}(v_i)$, za $i = 1, \ldots, m-1$. Uz to definiraju se i kružnice $C_i$ čije je središte $p_i$ te sadrže točke $v_{i-1}$ te $v_i$.

Ideja za dokaz prve leme je gledati nožišta visina iz $v_i$ te ih usporediti s $p_i$ te iskoristiti definicije Voronojievih ćelija.

Za drugu tvrdnju dokazuje se da kružnica radijusa $\overline{v_{src}v_{dst}}$ sadrži sve vrhove Voronojieve putanje koristeći $C_i$. Ako gledamo sve puteve koji su iznad $x$-osi, najviše prolazimo pola opsega te kružnice.

Neka je $b_0, \ldots, b_{l-1}$ podniz vrhova $v_0, \ldots, v_{m-1}$ koji su iznad ili na segmentu $(v_{\text{src}}, v_{\text{dst}})$. (Pogledajte sliku 3.3.) Razmotrimo dva vrha $b_i = v_j$ i $b_{i+1} = v_k$, gdje $k \neq j + 1$, tj. Voronoijev put (koji smo na početku algoritma dobili) između $b_i$ i $b_{i+1}$ nije direktan brid. Neka je $P_V = (b_i = p_0, \ldots, p_x = b_{i+1})$ dio Voronoijeva puta između $b_i$ i $b_{i+1}$, a neka je $P_F = (b_i = q_0, \ldots, q_y = b_{i+1})$ gornja granica $F$ između $b_i$ i $b_{i+1}$ (vidi sliku 3.12.)




Slika 3.12: Definicije $P_V$ i $P_F$.

Tada vrijedi sljedeće.

**Lema 3.6.3.** *Neka je* $c_{dfs} = \left(1 + \sqrt{5}\right)\frac{\pi}{2}$. *Tada vrijedi*

$$|P_V| \leq c_{dfs} \cdot (x(b_{i+1}) - x(b_i)) \quad ili \quad |P_F| \leq c_{dfs} \cdot (x(b_{i+1}) - x(b_i))$$

.

*Dokaz.* Neka su $c_0, \ldots, c_z$ donji dio konveksne ljuske od $P_F$, a $P_j$ put dobiven Voronoijevim usmjeravanjem od $c_j$ do $c_{j+1}$. Iz [5] vrijedi

$$|P_V| \leq c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i)) \text{ ili } \sum_{j=0}^{z-1} |P_j| \leq c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i)) \,. \tag{1}$$

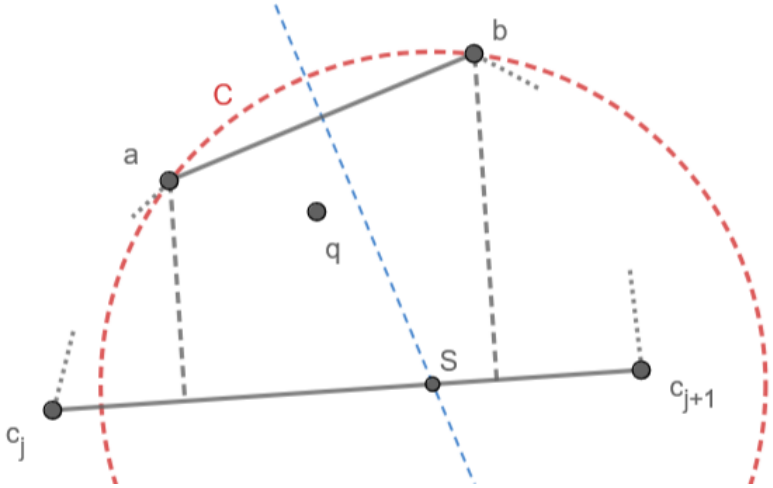
Ideja dokaza ove tvrdnje svodi se na to da se gleda unija $C_i$, te rub te unije kao put kojim se krećemo umjesto samih bridova. Pomoću tih puteva i korištenjem leme 3.6.2 s određenim ograničavanjem dobiva se rezultat.

Tvrdimo da (1) implicira lemu 3.6.3. i dokazujemo to pokazujući da $P_j$ prolazi kroz sve vrhove $P_F$ između $c_j$ i $c_{j+1}$. Po nejednakosti trokuta vrijedi

$$|P_F| \leq \sum_{j=0}^{z-1} |P_j|.$$

Pretpostavimo suprotno, neka postoji vrh $q$ na $P_F$ između $c_j$ i $c_{j+1}$ koji nije na $P_j$. Kao dio svog dokaza, Dobkin i dr. pokazuju da $P_j$ ostaje u potpunosti iznad segmenta $(c_j, c_{j+1})$ (ideja je da se pretpostavi suprotno, te se dokaže da takva točka ne može postojati kao vrh u triangulaciji, koristeći pritom uniju svih $C_i$ te svojstva donje konveksne ljuske). Stoga neka je $Q$ poligon omeđen $P_j$ i segmentom $(c_j, c_{j+1})$. Budući da je $q$ na $P_F$ između $c_j$ i $c_{j+1}$, mora ležati unutar $Q$.

Budući da je $Q$ monoton u smjeru od $c_j$ prema $c_{j+1}$, može se podijeliti na trapeze čije su gornje stranice bridovi $P_j$, donje stranice na segmentu $(c_j, c_{j+1})$, a lijeve i desne stranice okomite na $(c_j, c_{j+1})$. Vidi sliku 3.13.




Slika 3.13: Dokaz leme 3.6.3.

Neka su $a$ i $b$ dva vrha $P_j$ koja definiraju trapez koji sadrži $q$. Tvrdimo da $a$ i $b$ ne mogu biti uzastopni na $P_j$ jer njihove Voronojeve ćelije ne dijele brid koji presijeca $(c_j, c_{j+1})$. Ovo dokazujemo pokazivanjem da dio simetrala stranice $a$ i $b$ u Voronojevom dijagramu $q$, $a$, i $b$ ne presijeca segment $(c_j, c_{j+1})$. Ovo je dovoljno, jer ta simetrala sadrži dio od $a$ i $b$ u cijelom Voronojevom dijagramu.

Neka je $C$ kružnica sa središtem na $(c_j, c_{j+1})$ i sadrži $a$ i $b$. Ako dio simetrale stranice dužine točaka $a$ i $b$ u Voronojevom dijagramu $q$, $a$, i $b$ presijeca segment $(c_j, c_{j+1})$ u točki

$S$, tada $q$ ne smije biti unutar kružnice $C$. Inače bi $S$ bila bliže $q$ nego $a$ ili $b$, to znači da sigurno nije u ćeliji ni od $a$ ni od $b$ što je u kontradikciji s našom pretpostavkom. Međutim, unutar kružnice $C$ nalaze se gornje, lijeve, desne i donje stranice trapeza u kojem je $q$. Ovo vrijedi jer je $\overline{ab}$ tetiva od $C$. No nije moguće da su sve stranice trapeza unutar $C$, jer tada se cijeli trapez nalazi unutar $C$, a u njemu je $q$. Zaključujemo da nema točke $q$ na granici $F$ između $c_j$ i $c_{j+1}$ koja nije na $P_j$. □

Naš algoritam za $c$-kompetitivno usmjeravanje posjetit će sve vrhove $b_0, \ldots, b_{l-1}$ redom. Ako su $b_i$ i $b_{i+1}$ uzastopni na Voronojevoj putanji (tj. $b_i = v_j$ i $b_{i+1} = v_{j+1}$ za neki $j$), tada naš algoritam koristi Voronojevu putanju (tj. izravni brid) od $b_i$ do $b_{i+1}$. S druge strane, ako $b_i$ i $b_{i+1}$ nisu uzastopni na Voronojevoj putanji, tada prema lemi 3.6.3. postoji put od $b_i$ do $b_{i+1}$ duljine najviše

$$c_{\mathrm{dfs}} \cdot (x(b_{i+1}) - x(b_i)) .$$

Poteškoća se javlja jer algoritam unaprijed ne zna koji put odabrati. Rješenje je simulirati istraživanje oba puta "paralelno" i zaustaviti se kada prvi put dosegne $b_{i+1}$. Formalnije, neka su $P_V$ i $P_F$ definirani kao u lemi 3.6.3. Algoritam za pronalaženje puta od $b_i$ do $b_{i+1}$ opisan je sljedećim pseudokodom radi jednostavnijeg razmatranja kasnije.

Algoritam paralelnog pretraživanja

$j \leftarrow 0$, $l_0 \leftarrow \min\{\mathrm{dist}(p_0, p_1), \mathrm{dist}(q_0, q_1)\}$
**ponavljaj**
    Istraži $P_F$ do $b_{i+1}$ ili do $q_x$ takve da $|q_0, \ldots, q_{x+1}| > 2l_j$
    **ako** je $b_{i+1}$ "dosegnut" **onda**
        završi
    **inače**
        vrati se na $b_i$
    $j \leftarrow j + 1$, $l_j \leftarrow |q_0, \ldots, q_{x+1}|$
    Istraži $P_V$ do $b_{i+1}$ ili do $p_y$ takav da $|p_0, \ldots, p_{y+1}| > 2l_j$
    **ako** $b_{i+1}$ je dosegnut **onda**
        završi
    **inače**
        vrati se na $b_i$
    $j \leftarrow j + 1$, $l_j \leftarrow |p_0, \ldots, p_{y+1}|$
**dok ne** $b_{i+1}$ je "dosegnut"

**Lema 3.6.4.** *Koristeći algoritam paralelnog pretraživanja opisan gore, paket dostiže $b_{i+1}$ nakon što prijeđe udaljenost od najviše*

$$9 \cdot c_{dfs} \cdot (x(b_{i+1}) - x(b_i)) \sim 45.75 \cdot (x(b_{i+1}) - x(b_i)).$$

*Dokaz.* Jasno je da algoritam dođe do $b_{i+1}$ u konačnom broju koraka jer linije 8 i 14 osiguravaju da oba puta napreduju barem za jedan brid pri svakoj iteraciji. Neka je $k$ maksimalna vrijednost $j$, i neka je $d_j$ udaljenost prijeđena tijekom $j$-tog koraka istraživanja algoritma. Ukupna udaljenost $d$ koju paket prijeđe dana je izrazom

$$d = \sum_{j=0}^{k} d_j.$$

Budući da algoritam nije završio za $j = k - 1$, prema lemi 3.6.3. imamo

$$d_k < 2 \cdot c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i)).$$

Slično, budući da algoritam nije završio za $j = k - 1$ ili $j = k - 2$, imamo

$$l_{k-1} < 2 \cdot c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i)).$$

Budući da $l_j \geq 2l_{j-1}$ za svaki $j > 0$, imamo

$$d \leq \sum_{j=0}^{k-1} 2l_j + d_k,$$

što daje

$$d \leq \sum_{j=0}^{k-1} \frac{2l_{k-1}}{2^j} + d_k, \tag{3.1}$$

što odmah daje ogradu

$$d \leq 10 \cdot c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i)).$$

Za dobivanje jače ograde, primjećujemo da $d_k > c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i))$ implicira $l_{k-1} < c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i))$. Pod ovim uvjetom, izraz (3.1) je maksimalan kada je $l_{k-1} = 2 \cdot c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i))$, što daje

$$d \leq \sum_{j=0}^{k-1} \frac{4 \cdot c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i))}{2^j} + c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i)).$$

Iz ovoga slijedi

$$d < 9 \cdot c_{\text{dfs}} \cdot (x(b_{i+1}) - x(b_i)).$$

□

S obzirom na položaje $v_{\text{dst}}$ i $v_{\text{src}}$, paralelni algoritam pretrage opisan gore lako je implementirati kao dio algoritma usmjeravanja s $O(1)$ memorijom. Kombinaciju Voronoijeva algoritma usmjeravanja s ovim paralelnim algoritmom pretrage nazivamo **paralelnim Voronoijevim algoritmom usmjeravanja**.

**Teorem 3.6.5.** *Paralelni Voronoijev algoritam usmjeravanja proizvodi put čija je duljina najviše* $(9 \cdot c_{dfs} + \pi/2) \cdot dist(v_{src}, v_{dst})$.

*Dokaz.* Algoritam ima dva troška vezana za duljinu put koji je prijeđen:

1. Trošak kretanja po podputevima Voronoijeve putanje koji ostaju iznad $y$-osi.

2. Trošak primjena paralelnog algoritma pretrage.

Prema lemi 3.6.2., prvi trošak je najviše $(\pi/2) \cdot \text{dist}(v_{\text{src}}, v_{\text{dst}})$. Prema lemi 3.6.3. i činjenici da su vrhovi $b_0, \ldots, b_{l-1}$ $x$-monotoni (lema 3.6.1.), drugi trošak je pak najviše $9 \cdot c_{\text{dfs}} \cdot \text{dist}(v_{\text{src}}, v_{\text{dst}})$. □

Iako je prirodno nalaziti protuprimjere triangulacija za koje predstavljeni algoritmi nisu kompetitivni, u praksi to često nije tako. Razumnije je testirati algoritme po nekom tipičnom ulazu ili po nasumičnom ulazu, jer su takvi ulazi bolji prikazi podataka u stvarnim primjenama. Dani algoritmi su bili testirani u radu Bosa i Morina [8] na skupovima točaka uniformno distribuiranim u jediničnom kvadratu. Sami rezultati pokazali su da randomizirano usmjeravanje kompasom i paralelni Voronoijev algoritam usmjeravanja imaju znatno veće kompetitivne omjere od drugih algoritama proučavanih u ovom poglavlju. Dakle iako su protuprimjeri lagani za naći, vjerojatnost da će nastati u nasumičnom skupu točaka je vrlo mala, te je njihova očekivana kompetitivnost u praksi puno bolja.

Primijetimo da su znatno jednostavniji algoritmi oni koji su ipak bolji u praksi. I za algoritme koji nisu nužno geometrijski ovo je česta pojava. Za algoritme u drugim poglavljima nisu napravljeni testovi, no oni također za prednost imaju svoju jednostavnost. U praksi bi i oni mogli biti bolji od svojih determinističkih analogona.

# Bibliografija


[1] Aarti_Rathi, *Convex Hull using Graham Scan*, https://www.geeksforgeeks.org/convex-hull-using-graham-scan/.

[2] Michael Ben-Or, *Lower bounds for algebraic computation trees*, Proceedings of the fifteenth Annual ACM Symposium on Theory of Computing, 1983, str. 80–86.

[3] Bernard Chazelle, Leo J Guibas i Der Tsai Lee, *The power of geometric duality*, BIT Numerical Mathematics **25** (1985), br. 1, 76–90.

[4] Vasek Chvátal, *A combinatorial theorem in plane geometry*, Journal of Combinatorial Theory, Series B **18** (1975), br. 1, 39–41.

[5] S. J. Friedman i K. J. Supowit D. Dobkin, *Delaunay graphs are almost as good as complete graphs*, Discrete Comput. Geom. **5** (1990), 399–407, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=048ada24425179bc9862d216353df3a0d15313cf.

[6] Herbert Edelsbrunner, *An acyclicity theorem for cell complexes in d dimensions*, Proceedings of the fifth annual symposium on Computational geometry, 1989, str. 145–151.

[7] Henry Fuchs, Zvi M Kedem i Bruce F Naylor, *On visible surface generation by a priori tree structures*, Proceedings of the 7th annual conference on Computer graphics and interactive techniques, 1980, str. 124–133.

[8] Prosenji Bose i Pat Morin, *ONLINE ROUTING IN TRIANGULATIONS*, Siam J. Comput. **33** (2004), br. 4, 937–951.

[9] John M. Boyer i Wendy J. Myrvold, *Journal of Graph Algorithms and Applications*, **8** (2004), br. 3, 241–273, https://www.emis.de/journals/JGAA/accepted/2004/BoyerMyrvold2004.8.3.pdf.

[10] William Kocay, *The Hopcroft-Tarjan Planarity Algorithm*, (1993), http://acm.math.spbu.ru/~sk1/courses/1819s_au3/conspect/papers/tarjan1974-planarity.pdf.

[11] Rajeev Motwani, *Randomized Algor ithms*, Cambridge University Press, 1995.

[12] Michael S Paterson i F Frances Yao, *Efficient binary space partitions for hidden-surface removal and solid modeling*, Discrete & Computational Geometry **5** (1990), 485–503.

[13] Michiel Smid, *Computing intersections in a set of line segments: the Bentley-Ottmann algorithm*, (2003).

[14] Daniel A Spielman, *Spectral graph theory and its applications*, 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), IEEE, 2007, str. 29–38.

[15] Robert Tarjan, *Depth-first search and linear graph algorithms*, SIAM journal on computing **1** (1972), br. 2, 146–160.

[16] Günter M Ziegler, *Recent progress on polytopes*, Technische Universität Berlin. Fachbereich 3-Mathematik, 1996.

# Sažetak

Ovaj rad istražuje ključne aspekte geometrijskih algoritama i struktura, s posebnim naglaskom na randomizaciju, dekompoziciju prostora i navigacijske strategije te koristi algoritama u stvarnom životu.

U uvodnim dijelovima obrađuje se probleme konstrukcija konveksnih ljuski u ravnini i prostoru. Uz to definira se i koncept dualnosti s pomoću kojeg dobivamo "duale" prije opisanih problema. Za dane algoritme analiziramo njihovu očekivanu vremensku složenost.

Nadalje, analiziraju se različite tehnike dekompozicije prostora uz korist randomizacije. Prvo definiramo Voronojieve dijagrame i Delaunayjeve triangulacije te dajemo randomizirani algoritam za njihovu konstrukciju koji je usko vezan s problemom konstrukcije konveksne ljuske u tri dimenzije. Uz to dajemo još jedan algoritam za slučaj triangulacije konveksnog poligona. Nakon toga dajemo randomizirane algoritme za konstrukcije trapezoidne dekompozicije i binarnih particija ravnine. Sve dekompozicije imaju široku primjenu u računalnoj geometriji s naglaskom na računalnoj grafici.

U zadnjem poglavlju poseban fokus stavljen je na algoritme vezane uz Delaunayjeve triangulacije, gdje se razmatraju različite strategije usmjeravanja, uključujući razne jednostavne deterministčke algoritme s naglaskom na pohlepnim algoritmima, te njihova proširenja kroz randomizaciju. Za sve algoritme mjerimo koliko su dobri definirajući im $c$-kompetitivnost. Ispituje se ograničenja determinističkih algoritama bez memorije i analiziraju njihovi negativni rezultati u specifičnim scenarijima. Na kraju poglavlja dajemo algoritam koji je $c$-kompetitivan. Završni dio sadrži kratku raspravu o algoritmima.

Rad sintetizira teorijska saznanja i praktične primjene u području računalne geometrije, te pokazuje da su u nekim slučajevima jednostavniji algoritmi bolji za rješavanje određenih problema.

# Summary

This paper explores key aspects of geometric algorithms and structures, with a special focus on randomization, space decomposition, and navigation strategies, as well as the practical applications of these algorithms in real life.

In the introductory sections, the paper addresses the problems of constructing convex hulls in the plane and in space. The concept of duality is also defined, allowing us to obtain the ”duals” of the previously described problems. The expected time complexity of these algorithms is analyzed.

Further, various space decomposition techniques utilizing randomization are explored. First, Voronoi diagrams and Delaunay triangulations are defined, and a randomized algorithm for their construction is presented, which is closely related to the problem of constructing a convex hull in three dimensions. Additionally, another algorithm for the triangulation of convex polygons is provided. The paper then presents randomized algorithms for the construction of trapezoidal decompositions and binary partitions of the plane. All these decompositions have broad applications in computational geometry, with an emphasis on computer graphics.

The final chapter focuses on algorithms related to Delaunay triangulations, where different routing strategies are examined, including various simple deterministic algorithms with an emphasis on greedy algorithms, and their extensions through randomization. For all algorithms, their performance is evaluated by defining their $c$-competitiveness. The limitations of memoryless deterministic algorithms are examined, and their poor results in specific scenarios are analyzed. The chapter concludes with the presentation of the only $c$-competitive algorithm. The concluding section includes a brief discussion on the algorithms.

This paper synthesizes theoretical insights and practical applications in the field of computational geometry, showing that in some cases, simpler algorithms are more effective for solving certain problems.

# Životopis

Rođen sam 18. studenog 2000. godine u Zagrebu. Osnovnoškolsko obrazovanje sam završio u Osnovnoj školi Dragutina Domjanića u Zagrebu. Nakon toga sam upisao prirodoslovno-matematički smjer u Gimnaziji Lucijana Vranjanina. Tijekom srednje škole sam se aktivno bavio natjecateljskom matematikom i programiranjem, gdje sam sudjelovao na nekolicini nacionalnih i internacionalnih natjecanja.

Akademske godine 2019./2020. sam upisao Preddiplomski sveučilišni studij Matematika na Prirodoslovno-matematičkom fakultetu u Zagrebu. Nakon završetka preddiplomskog studija, akademske godine 2022./2023. sam upisao Diplomski sveučilišni studij Računarstvo i matematika. Tijekom studija sam se nastavio sam biti aktivan u vodama natjecateljske matematike, natucajući se na IMC-u te održavanjem predavanja za pripreme za mnoga matematička natjecanja.