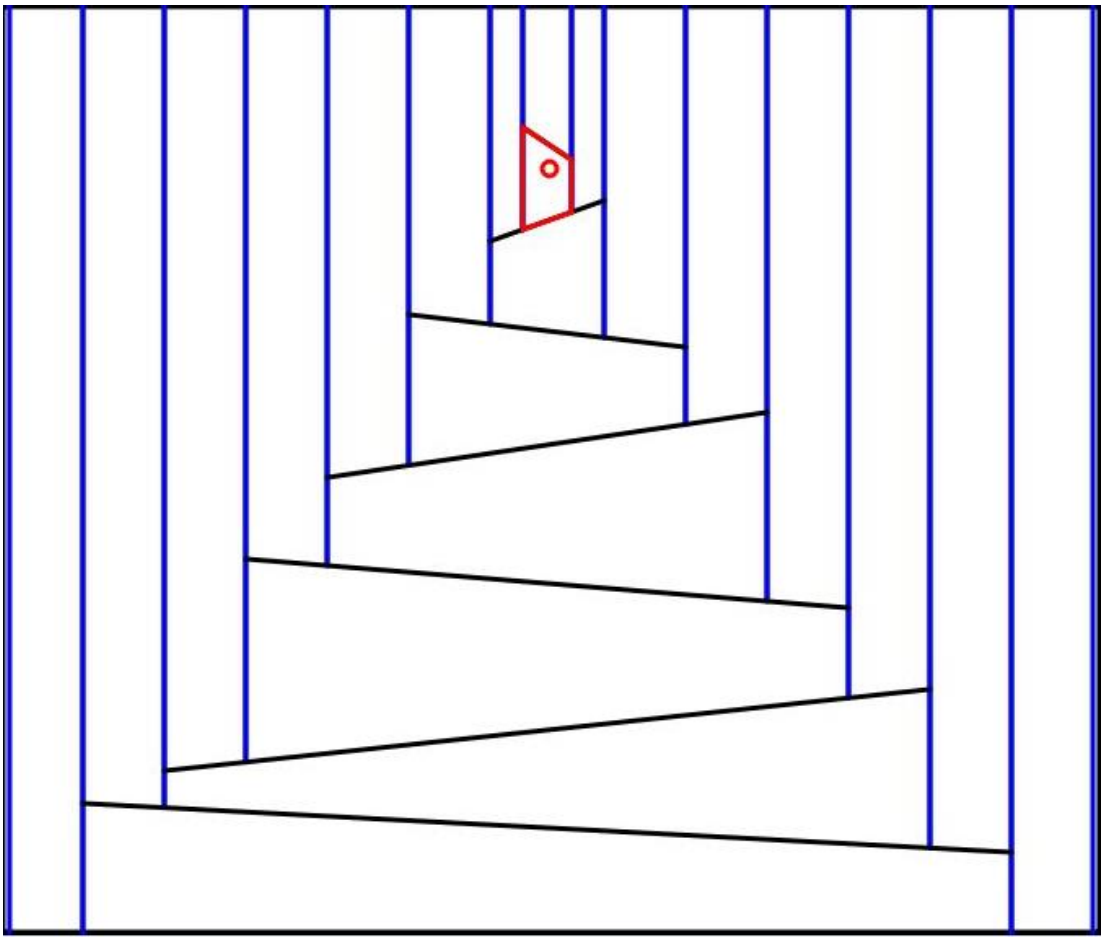
Geometrijski algoritmi @ MATF

Problem lociranja tačke

Miloš Arsić, 1052/2020



Opis problema

Aplikacije koje koriste geometrijske strukture pri svom radu često imaju potrebu za određivanjem oblasti kojoj pripada neka trenutno izabrana tačka. Ovaj problem izražen je najviše kod aplikacija koje rade sa pozicioniranjem u okviru geografskih informacionih sistema, navigacijom i određivanjem putanja kretanja robota i mašina, a i problemima računarske grafike u kojima je potrebno u svakom trenutku efikasno odgovoriti na upite o tekućoj oblasti izabrane tačke.

U ovom radu biće opisano rešenje problema lokacije tačke u ravni koji može da bude definisan sa: Za n različitih duži među kojima se nikoje dve ne seku i za zadate koordinate tačke potrebno je odrediti oblast kojoj zadata tačka pripada.

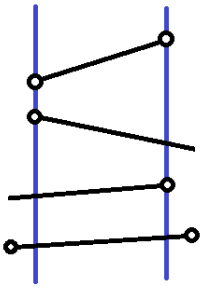
Kako bismo ograničili oblast koju posmatramo konstruišemo obuhvatajući pravougaonik (eng. bounding box). Rešenje će biti realizovano kroz nekoliko koraka od kojih će prvi biti trapezno

razlaganje ravni. Trapez je konveksna četvorougaona geometrijska figura u ravni čije su dve naspramne stranice (osnovice) paralelne. Pored toga biće detaljno definisana i ažurirana pretraživačka struktura za efikasno pretraživanje i određivanje odgovarajuće oblasti prilikom upita.

Ulaz: *n različitih duži među kojima se nikoje dve ne seku i koordinate tačke u ravni*

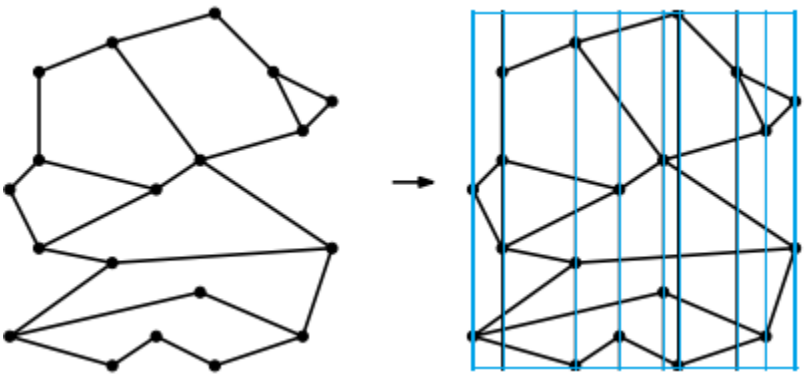
Izlaz: *pokazivač na trapez kojoj data tačka pripada*

Naivni algoritam



Slika 1. Položaj duži unutar sektora

Prvi korak naivnog algoritma predstavlja konstrukcija pravih paralelnih sa ordinatom, a koje sadrže krajeve duži. Tako ravan možemo da podelimo na vertikalne pravougaone sektore u kojima se nalaze duži (cele duži ili odsečci) koje nemaju međusobnih preseka i čiji krajevi pripadaju vertikalnim stranicama sektora. Odsečke duži možemo da sortiramo prema y-koordinati vrhova unutar sektora. Svakom odsečku dodelićemo oznaku oblasti koja je iznad njega (čija je on donja ivica). Na taj način se može dobiti trapezna mapa koja se sastoji od trouglova i trapeza. Primer takve mape prikazan je na sledećoj slici.



Slika 2. Trapezna mapa

Upit kojim se određuje kojoj oblasti pripada neka tačka može se realizovati u tri koraka:

1) Binarnom pretragom odredimo sektor kome pripada x-koordinata tačke.

2) Binarnom pretragom u okviru sektora odredimo oblast kojoj pripada tačka (duži između kojih se tačka nalazi).

3) Oznaka oblasti donje duži (odsečka) je oznaka tražene oblasti. Ukoliko se tačka nalazi iznad prvog ili ispod poslednjeg odsečka u sektoru kažemo da ona pripada spoljašnjosti (koju smo ograničili pravougaonikom).

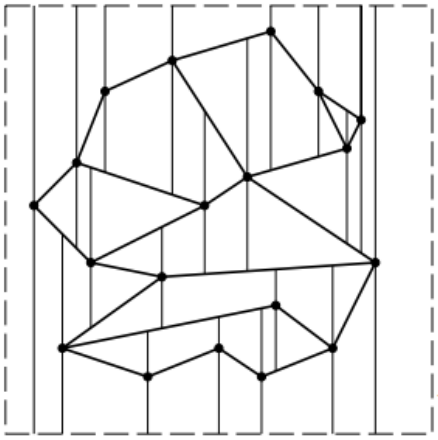
Vremenska složenost algoritma je dobra. Prva binarna pretraga vrši se u nizu sektora čija je dužina najviše 2n, a druga u nizu odsečaka čija je dužina najviše n. Tako zaključujemo da je složenost pretrage prilikom upita $O(\log n)$.

Prostorna složenost, sa druge strane, u najgorem slučaju je $O(n^2)$.

Optimalni algoritam

Zbog velike prostorne složenosti prethodna struktura podataka nije zanimljiva u praksi jer iako se najgori slučaj ne dešava često on je moguć. Trebalo bi odrediti finiju podelu koja će poput već prikazane dekompozicije olakšati upite o lokaciji tačaka, ali i smanjiti prostornu složenost već pomenutog rešenja. Unapređeno rešenje imaće i jedan novi preduslov - ne postoje dva kraja duži sa istom x-koordinatom. Uz ovaj preduslov obezbediće se da dobijena mapa nema više krajeva duži na istoj vertikalnoj liniji. Skup duži koji ispunjava preduslov zvaćemo skup duži u opštem položaju.

Može se primetiti da je u prethodom razlaganju određeni broj trouglova i trapeza bio suvišan jer su delili iste gornje i donje duži i bili susedni pa će se poboljšanje rešenja odnositi i na spajanje pomenutih trouglova i trapeza u jedan čime se smanjuje broj objekata, a samim tim i informacija koje je potrebno pamtiti. Konstrukcija bočnih stranica trapeza u ovom slučaju vršiće se konstrukcijom vertikalnih duži od krajeva duži ka gore i dole sve do prve duži na koju se pri konstrukciji naiđe. Tako se dobija trapezna mapa prikazana na slici 3.



Slika 3. Nova trapezna mapa

Pored smanjenja broja oblasti ova podela daje nam i jedinstven način za opisivanje svakog trapeza pomoću četiri parametra: gornja i donja duž, tačka levog i tačka desnog kraja (tačka koja pripada levoj i desnoj stranici trapeza). Uz to, zbog preduslova, svaki trapez može imati najviše dva leva i dva desna suseda što olakšava izgradnju pretraživačke strukture.

Pretraživačka struktura koja će biti korišćena jeste usmereni aciklični graf koji se sastoji od tri vrste čvorova – čvor tačke (TCvor), čvor duži (Dcvor) i čvor trapeza (List). Svaki od čvorova nasleđuje klasu Cvor koja u sebi sadrži tri virtuelna metoda

```
virtual std::string procitajImeCvora();
virtual Trapez* trapezCvora();
virtual Cvor* sledeciCvor(QPointF);
```

Pomoću prvog možemo da proverimo da li smo stigli do lista prilikom pretrage, pomoću drugog dobijamo pokazivač na Trapez koji se nalazi u Listu, a pomoću trećeg određuje se koji je čvor u strukturi sledeći prilikom pretrage. Kod čvora tačke poslednji metod proverava da li se zadata tačka nalazi levo ili desno u odnosu na tačku čvora, a kod čvora duži da li se zadata tačka nalazi iznad ili ispod u odnosu na duž čvora.

Na početku se u strukturi nalazi list koji predstavlja obuhvatajući pravougaonik. U klasi TrapeznaMapa postoje sledeći metodi:

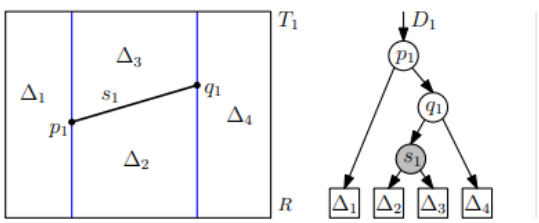
```
Trapez* pronadjiTrapez(QPointF t);
void dodavanjeduzi(Duz *novaduz);
```

Metod `pronadjiTrapez` služi za pretragu i poziva metod `sledeciCvor` sve dok se u pretrazi ne stigne do lista.

Metod `dodavanjeduzi` dodaje duž u strukturu uz ažuriranje. Prvo se metodom `pronadjiTrapez` dobijaju pokazivači ka dva trapeza, početnom i krajnjem trapezu, kome pripadaju početak i kraj duži. Potom razlikujemo dva slučaja:

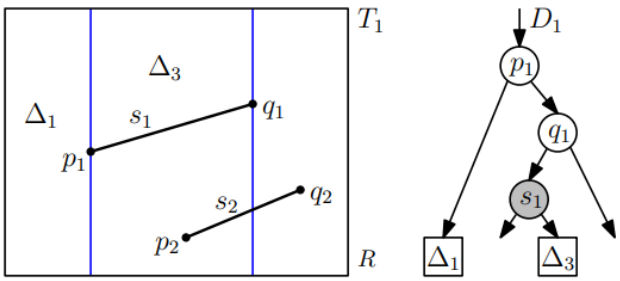
1) Početni i krajnji trapez se poklapaju (duž pripada jednom trapezu);
2) Početni i krajnji trapez se razlikuju.

U prvom slučaju potrebno je napraviti samo četiri nova trapeza i ažurirati susede i pretraživačku strukturu, a potom i susede novih i starih susednih trapeza. Umesto lista kome pripadaju početna i krajnja tačka postavljamo prvo čvor tačke čiji je levi sused čvor levog trapeza Δ_1, a desni sused čvor desnog kraja duži. Levi sused čvora desnog kraja duži je čvor duži, a desni sused je čvor trapeza Δ_4. Čvor duži $s_1$ima kao levi sused čvor gornjeg trapeza Δ_2, a kao desni čvor donjeg trapeza Δ_3. Ažuriranje strukture prikazano je na slici 4.



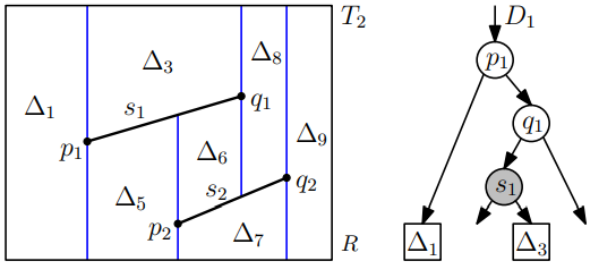
Slika 4. Ažuriranje pretraživačke strukture prilikom dodavanja duži koja pripada jednom trapezu

U drugom slučaju kada krajevi duži ne pripadaju jednom trapezu prolaskom kroz sve susedne trapeze pravimo listu presečenih trapeza čije listove na kraju uklanjamo iz pretraživačke strukture što je prikazano na slici 5.



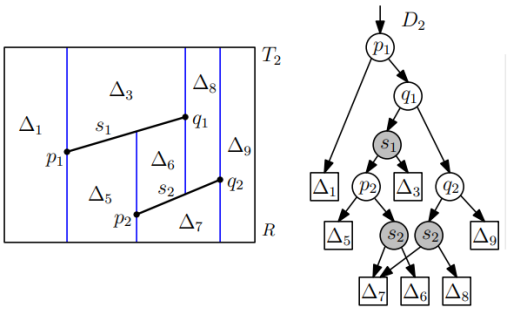
Slika 5. Uklanjanje čvorova listova

Potom pravimo nove trapeze i ažuriramo trapeznu mapu. Pored toga prolaskom kroz sve susedne trapeze iznad duži i sve susedne trapeze ispod duži spajamo susedne trapeze u jedan ukoliko imaju istu gornju i donju duž i ažuriramo susede trapeza što je prikazano na slici 6.



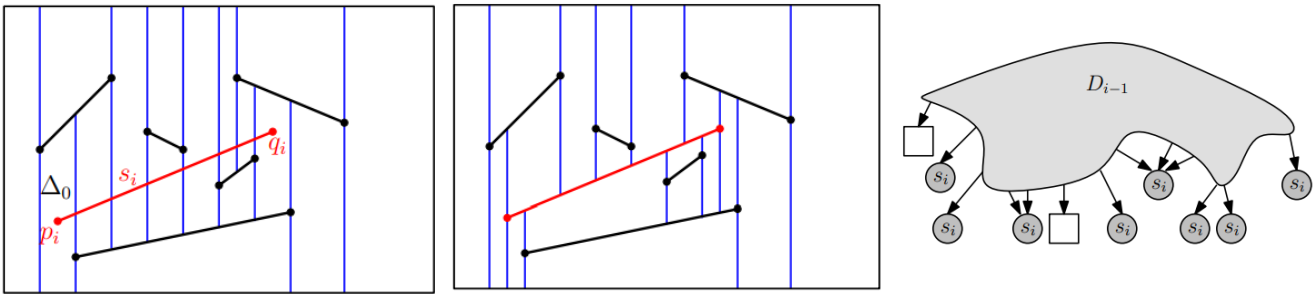
Slika 6. Generisanje novih trapeza i ažuriranje suseda

Na mesto čvorova starih presečenih trapeza u pretraživačkoj strukturi postavljamo odgovarajuće čvorove tačaka i duži. Umesto lista početnog trapeza postavljamo čvor levog kraja duži, a umesto lista krajnjeg trapeza postavljamo čvor desnog kraja duži. Za svaki od njih (levi i desni čvor redom) unutrašnji susedi su čvorovi duži pomoću koje se može razlikovati novi gornji od novog donjeg trapeza.



Slika 7. Ažuriranje suseda u pretraživačkoj strukturi

U posebnom slučaju, kada nova duž ne seče dva susedna trapeza već više njih, pored prethodno pomenutih postupaka u kojima se umesto početnog i kranjeg trapeza postavlja čvor leve i čvor desne tačke potrebno je umesto listova svih presečenih unutrašnjih trapeza dodati novi čvor duži koji će odvojiti novi gornji od novog donjeg trapeza. Takav slučaj prikazan je na slici 8.

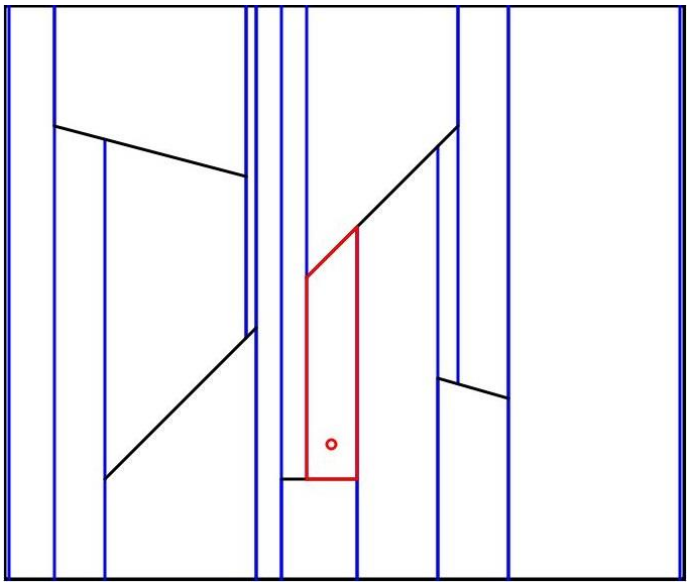


Slika 8. Slučaj kada nova duž seče više od jednog trapeza

Vremenska složenost pretprocesiranja datog skupa duži optimalnog rešenja poklapa se sa složenošću naivnog i iznosi $O(nlog\ n)$. Slično je i sa vremenskom složenošću izvršavanja upita koja je $O(log\ n)$. Optimalno rešenje ima ipak bolju prostornu složenost koja je $O(n)$.

Vizuelizacija algoritma

Vizuelizacija algoritma izvršena je kroz nekoliko koraka. Prilikom instanciranja objekta klase PointLocation napraviće se nova mapa i nova pretraživačka struktura. Nakon toga biće nacrtana trapezna mapa i kod naivnog i kod optimalnog algoritma. Ukoliko je vrednost promenljive koja predstavlja traženi trapez NULL znaće se da pretraga nije još uvek izvršena i neće biti daljih promena. Klikom na dugme Započni izvršava se pretraga i u promenljivu tTrapez postavlja se pokazivač na traženi trapez. Tada se i ažurira crtež i trapez se obeležava crvenom bojom što je prikazano na slici 9.



Slika 9. Vizuelizacija algoritma – prikaz rezultata

Testiranje ispravnosti algoritama

Kako je algortam Lokacija tačke implementiran u jednoj klasi za testiranje će biti dovoljno instanciranje objekta te klase i pozivanje odgovarajućih metoda za određivanje naivnog i optimalnog rešenja. Testovi su realizovani pomoću radnog okvira GoogleTest, a njihov opis naveden je u sledećoj tabeli.

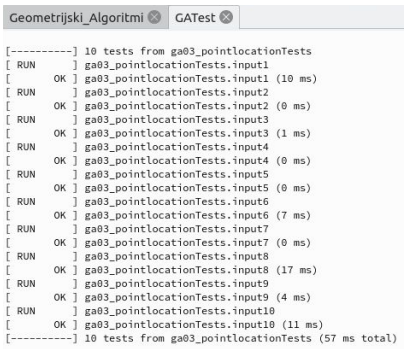
Naziv testa	Opis testa	Ulaz	Očekivani izlaz
input1	Zadavanje jednostavnog ulaza koji sadrži jednu duž i tačku ispod nje.	Niz sa jednom duži i tačkom ispod nje učitan iz datoteke input1.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input2	Zadavanje jednostavnog ulaza koji sadrži dve duži čije projekcije na x--osu nemaju zajedničkih tačaka. Zadata tačka je iznad druge duži.	Niz sa dve duži i tačkom iznad druge duži učitan iz datoteke input2.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input3	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza, ali samo sa donje strane. Zadata tačka pripada baš takvom trapezu.	Niz sa pet duži i tačkom ispod poslednje od njih učitan iz datoteke input3.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input4	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza, ali samo sa donje strane. Zadata tačka pripada prvom desnom susedu spojenog trapeza.	Niz sa pet duži i tačkom ispod druge duži učitan iz datoteke input4.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input5	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza, ali samo sa donje strane. Zadata tačka pripada trapezu koji je iznad jedne od duži, ali nije u spojenom trapezu.	Niz sa tri duži i tačkom iznad druge duži učitan iz datoteke input5.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input6	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza i sa gornje i sa donje strane. Zadata tačka pripada jednom od dobijenih spojenih trapeza koji se nalaze iznad duži.	Niz sa pet duži i tačkom iznad prve duži učitan iz datoteke input6.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input7	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza i sa gornje i sa donje strane. Zadata tačka pripada trapezu koji se nalazi ispod i iznad dva spojena trapeza.	Niz sa pet duži i tačkom ispod treće duži učitan iz datoteke input7.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input8	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza sa donje strane. Zadata tačka pripada trapezu koji je pravougaonik.	Niz sa dve duži i tačkom između njih učitan iz datoteke input8.txt	Poklapanje rezultata naivnog i optimalnog algoritma

input9	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza sa donje strane. Zadata tačka pripada jednom od trapeza koji nisu dobijeni spajanjem.	Niz sa sedam duži i tačkom između dve duži učitan iz datoteke input9.txt	Poklapanje rezultata naivnog i optimalnog algoritma
input10	Zadavanje ulaza koji sadrži duži čije projekcije na x-osu imaju zajedničkih tačaka, a samim tim postoji i spajanje trapeza i sa gornje i sa donje strane. Zadata tačka pripada trapezu koji sa sve tri strane (levo, gore i desno) ima kao suseda trapez koji je nastao spajanjem.	Niz sa tri duži i tačkom ispod prve duži učitan iz datoteke input10.txt	Poklapanje rezultata naivnog i optimalnog algoritma

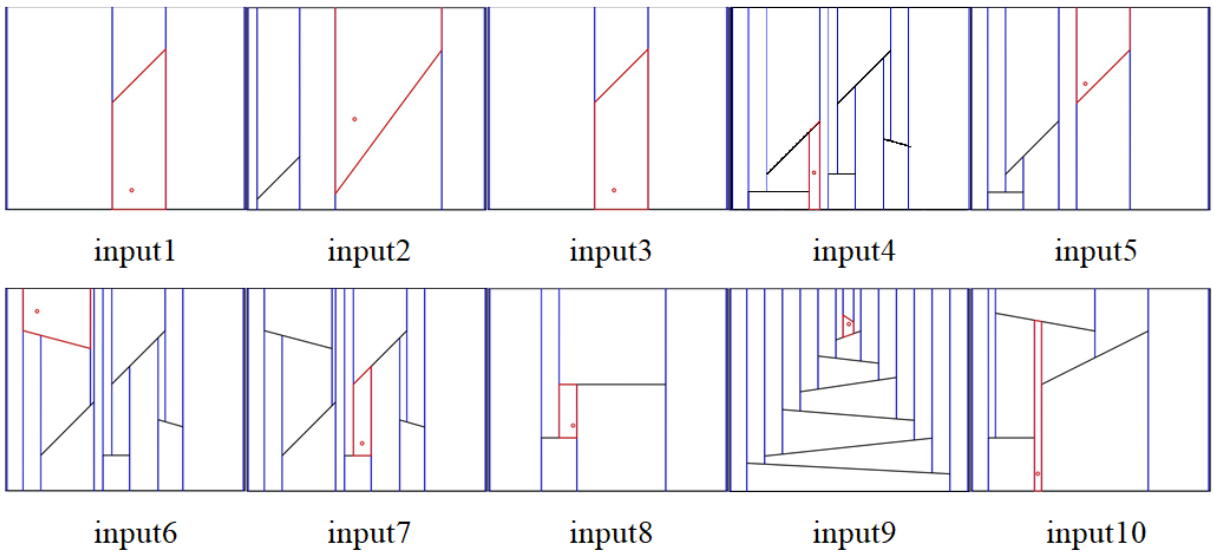
Tabela 1. Opis testova

Svi testovi su uspešno izvršeni i u svakom testu dobijeno je očekivano poklapanje rezultata naivnog i optimalnog algoritma što se može videti na slici 10.

Na slici 11. nalazi se vizuelni prikaz svih testova i dobijenog rezultata.



Slika 10. Izvršavanje testova



Slika 11. Vizuelni prikaz rezultata testova