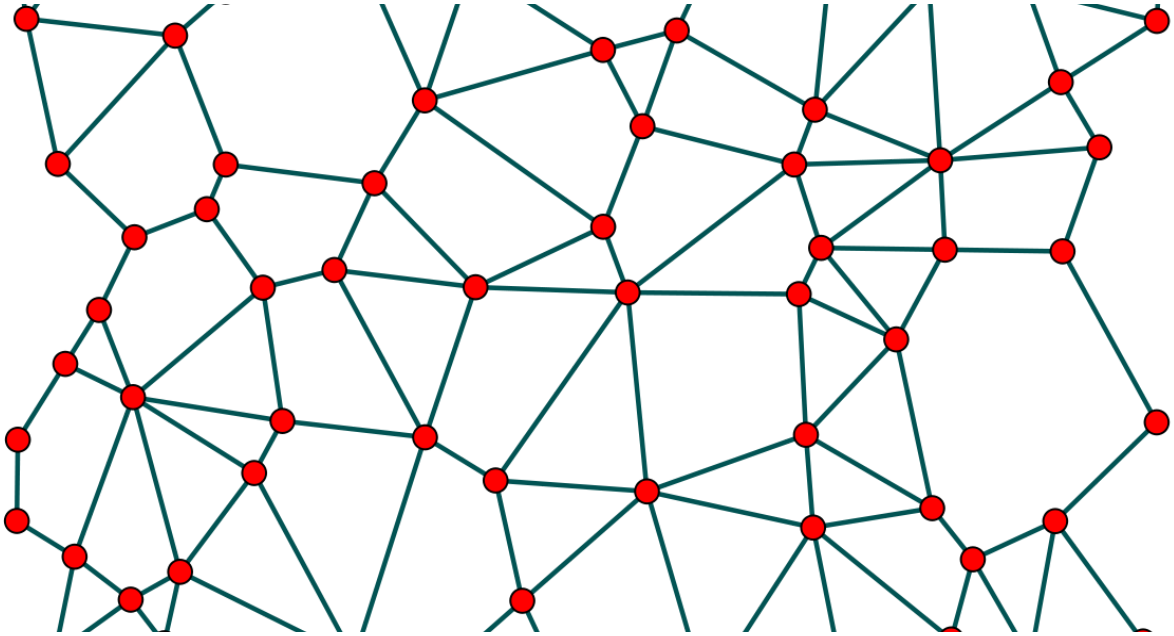
Geometrijski algoritmi @ MATF

Algoritam za lociranje tačke u planarnom razlaganju ravni

Marinela Parović



Opis problema

Neka je dato planarno razlaganje ravni indukovano skupom međusobno nepresijecajućih duži S (svake dvije različite duži u skupu S su ili disjunktne, ili imaju tačno jedno zajedničko tjeme). Zadatak je transformisati dato razlaganje u pogodnu strukturu, koja će omogućiti da se za zadatu tačku P efikasno odredi kojoj oblasti u razlaganju ona pripada. Potrebna je, dakle, struktura podataka koja omogućava brzo lociranje tačaka.

Ovaj problem ima široku primjenu u GPS sistemima, koja je očigledna ukoliko ga formulišemo neformalnije. Ako je data mapa i tačka upita P zadata svojim koordinatama, potrebno je na mapi pronaći region koji sadrži tačku P. Takođe, ovo je potproblem za mnoge druge probleme računarske geometrije kao što su planiranje kretanja robota ili izračunavanje najkraćeg puta.

Ulaz: *planarno razlaganje ravni indukovano skupom S od n nepresijecajućih duži i tačka P*

Izlaz: *oblast u razlaganju koja sadrži tačku P (na primjer, naziv tražene oblasti)*

Naivno rešenje problema

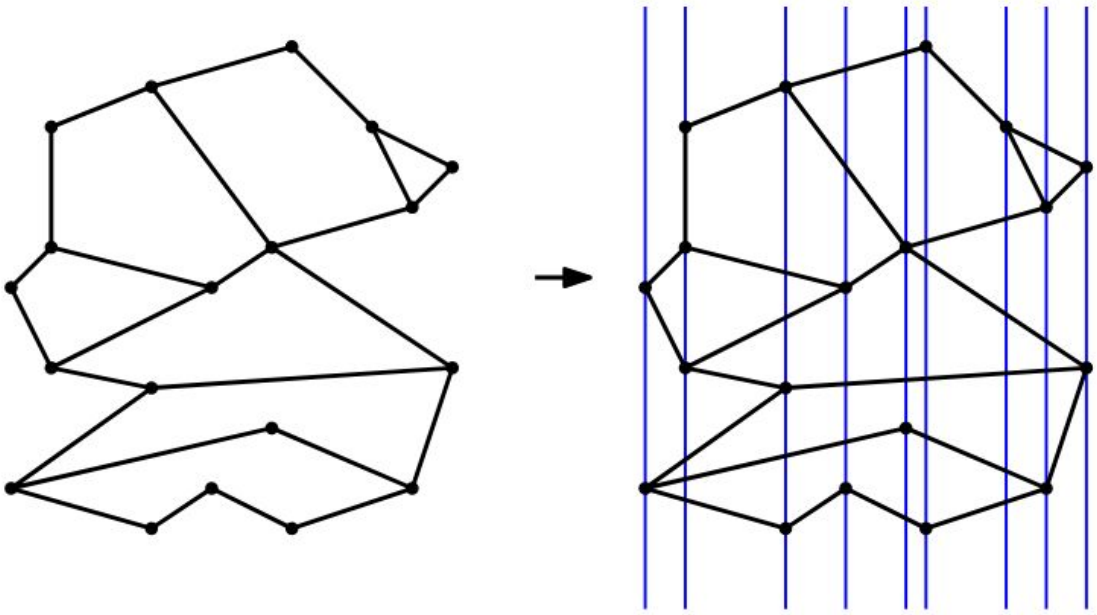
Naivni algoritam daje jednostavnu strukturu podataka za izvršavanje upita lociranja tačke P. Zamislimo da smo povukli vertikalne prave kroz sva tjemena svih duži u razlaganju. Na taj način, ravan je podijeljena na vertikalne trake. U soritanom nizu čuvaju se x-koordinate tjemena, što omogućava da se u vremenu od O(logn) odredi kojoj vertikalnoj traci pripada tačka P. Takođe, ni u jednoj vertikalnoj traci nema tjemena duži iz S, što znači da dio razlaganja unutar svake trake ima veoma specifičnu formu. Naime, sve duži koje sijeku traku je u potpunosti presijecaju tako da im nijedno tjeme nije unutar nje, i nijedna od duži ne siječe nijednu drugu duž. To znači da unutar svake trake imamo definisano uređenje duži, pa duži unutar jedne trake takođe možemo čuvati u sortiranom nizu. Dodatno, uz svaku duž možemo čuvati informaciju o tome koja se oblast nalazi iznad nje.

U nastavku je dat pseudo kod naivnog algoritma.

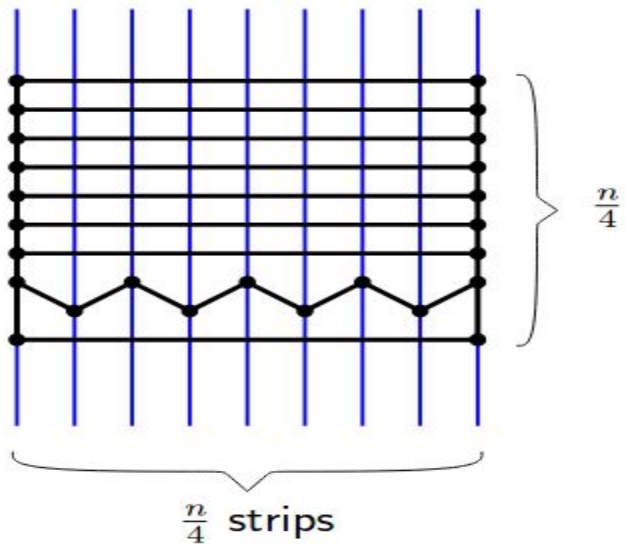
Ulaz: *planarno razlaganje ravni indukovano skupom S od n nepresijecajućih duži i tačka P*

Izlaz: *oblast u razlaganju koja sadrži tačku P (na primjer, naziv tražene oblasti)*

1. U nizu sa sortiranim x-koordinatama razlaganja, binarnom pretragom pronaći vertikalnu traku T u kojoj se nalazi tačka P.
2. U nizu koji odgovara traci T binarnom pretragom pronaći duž s takvu da je tačka P direkno iznad nje ili ustanoviti da takva duž ne postoji (to znači da je P u neograničenoj oblasti razlaganja).
3. Oblast koja je sačuvana uz duž s je oblast koja sadrži tačku P.



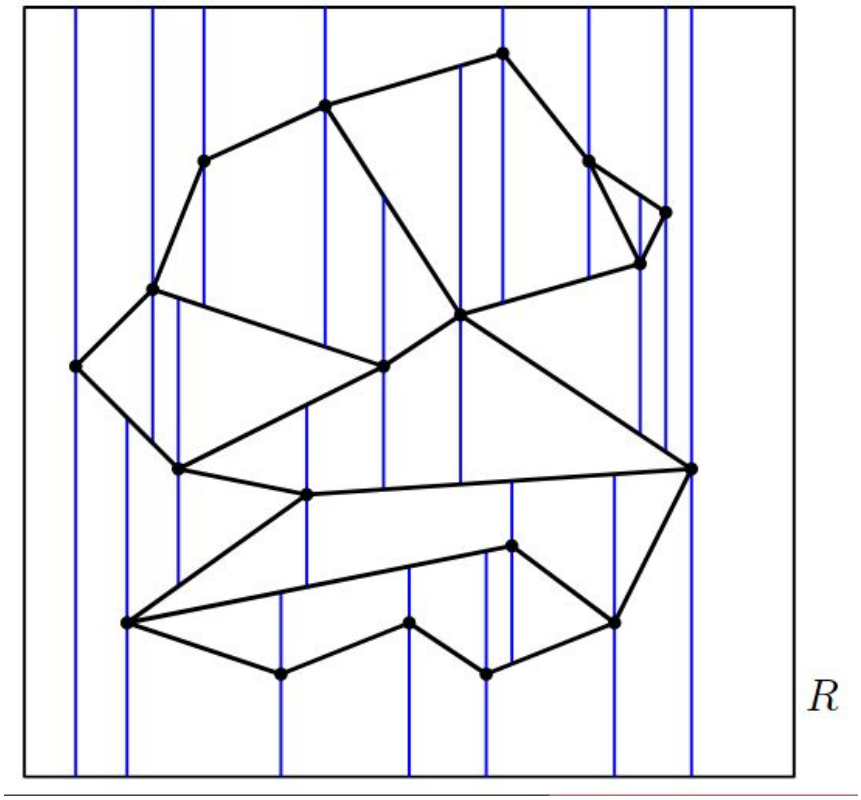
Složenost lociranja tačke naivnim algoritmom kada je odgovarajuća struktura već formirana je O(logn), jer se vrše dvije binarne pretrage u nizovima od kojih je prvi dužine najviše 2n, a drugi dužine najviše n. Međutim, prostorna složenost ovakvog profinjenja polaznog razlaganja u najgorem slučaju ima složenost O(n^2), što je neprihvatljivo u gotovo svim praktičnim primjenama, čak i za pristojno velike vrijednosti broja n.



Optimalni algoritam - Trapezno razlaganje ravni

Trapezno razlaganje ravni takođe predstavlja profinjenje polaznog razlaganja uz garanciju da će složenost tog profinjenja biti linearna u odnosu na broj duži. Uz pretpostavke da nema vertikalnih duži, kao ni tačaka sa jednakim x-koordinatama povucimo 2 vertikalne ekstenzije iz svake tačke - jednu na gore i jednu na dolje, ali samo da sljedeće duži. Pljosni ovakvog razlaganja su trapezi (trougao se može shvatiti kao specijalan slučaj trapeza čija je jedna osnovica dužine 0). Svaki trapez određen je sa 2 tjemena duži (zvaćemo ih lijevo i desno tjeme trapeza) i dvije duži iz skupa S (gornja i donja duž trapeza).

Dva trapeza su susjedi ako dijele vertikalnu ivicu. Uz navedne pretpostavke svaki trapez ima najviše 4 susjeda. Svaki trapez je objekat koji ima lijevo i desno tjeme, gornju i donju duž, kao i pokazivače na najviše 4 trapeza koji su mu susjedni.



Teorema: Trapezna dekompozicija planarnog razlaganja ravni indukovanog sa n duži ima najviše 6n+4 tjemena i najviše 3n+1 trapeza.

Za kreiranje trapeznog razlaganja koristi se randomizovani inkrementalni algoritam, kojim se, osim trapezne mape gradi i struktura pretrage čiji su listovi trapezi u trapeznom razlaganju. Unutrašnji čvorovi su ili x čvorovi ili y čvorovi. U x čvorovima se čuva tačka i u njima se pretraga nastavlja lijevo ako je tačka koja se traži lijevo od tačke u tom čvoru, a desno inače. U y čvorovima se čuvaju duži i u njima se pretraga nastavlja lijevo ako je tačka koja se traži ispod duži u tom čvoru, a desno inače. Uz uslov da svakom trapezu odgovara tačno jedan list stuktura pretrage ne mora biti stablo, već je usmjereni aciklični graf.

U nastavku je dat pseudo kod optimalnog algoritma.

Ulaz: *planarno razlaganje ravni indukovano skupom S od n nepresijecajućih duži i tačka P*

Izlaz: *oblast u razlaganju koja sadrži tačku P (na primjer, naziv tražene oblasti)*

1. Odrediti dovoljno veliki pravougaonik R koji sadrži sve duži iz S i inicijalizovati trapeznu mapu i strukturu pretrage na R.
2. Izračunati slučajnu permutaciju $s_1, s_2, ..., s_n$ duži iz skupa S.
3. Za sve $i = 1, ..., n$ uraditi sljedeći korak.
4. Pronaći trapeze $D_0, ..., D_k$ koje siječe duž s_i. Ukloniti njih iz T i zamijeniti ih novim trapezima koji nastaju zbog umetanja duži s_i. Ukloniti listove za $D_0, ..., D_k$ iz strukutre pretrage i kreirati nove listove za nove trapeze. Povezati nove listove sa postojećim unutrašnjim čvorovima dodavanjem novih unutrašnjih čvorova.

Pseudokod algoritma za pronalaženje niza $D_0, ..., D_k$ trapeza koje siječe nova duž s_i dat je u nastavku.

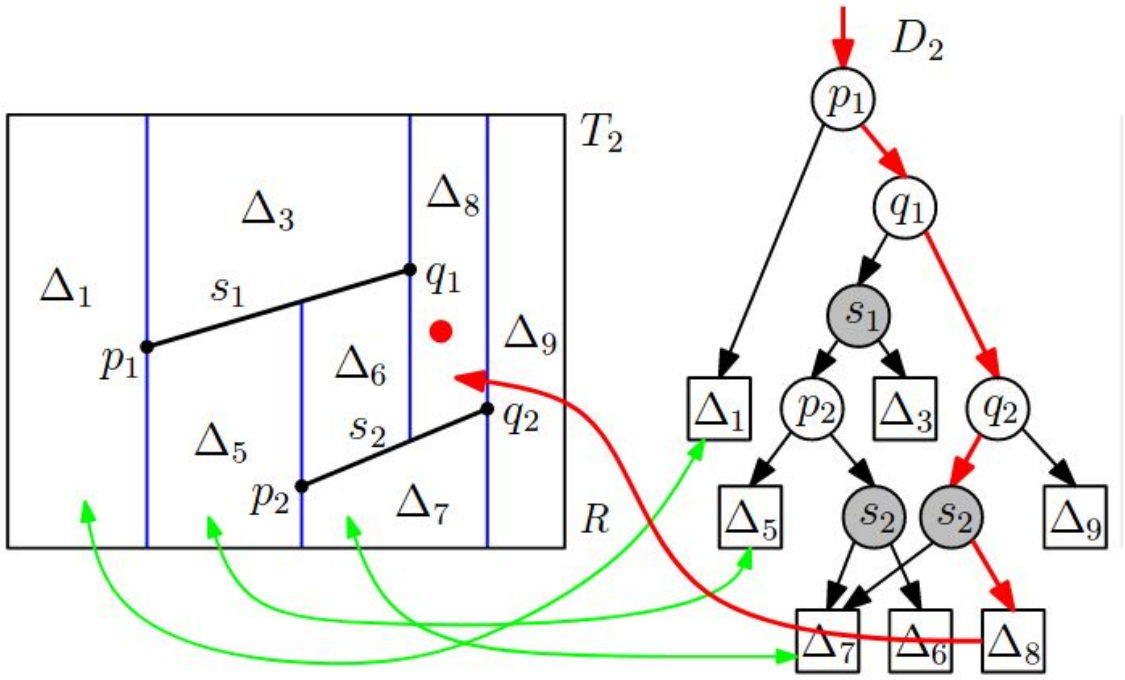
Ulaz: *trapezna mapa T, struktura pretrage D za mapu T i nova duž* s_i.

Izlaz: *niz* $D_0, ..., D_k$ trapeza koje siječe duž s_i .

1. Neka su p i q lijevo i desno tjeme duži s_i.
2. Locirati tačku p u strukturi D radi pronalaženja trapeza D_0.
3. Inicijalizovati brojač j na 0.
4. Sve dok je q sa desne strane desnog tjemena trapeza D_j ponavljati sljedeći korak.

5. Ako je desno tjeme od D_j iznad s_i onda je D_{j+1} donji desni susjed od D_j, a inače je D_{j+1} gornji desni susjed od D_j. Uvećati j za 1.
6. Vratiti niz $D_0, ..., D_k$.

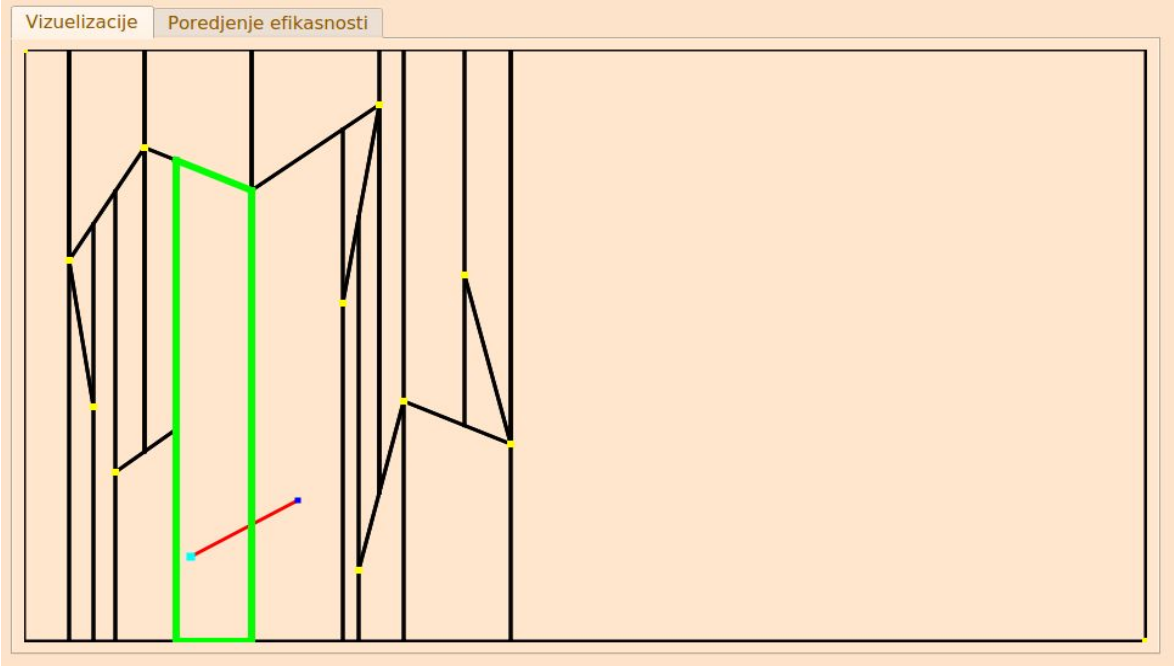
Teorema: Navedeni algoritam izračunava trapeznu mapu T i strukturu pretrage D za skup S od n nepresijecajućih duži u ravni u očekivanom vremenu O(nlogn). Očekivana prostorna složenost strukture pretrage D je O(n), a očekivano vrijeme lociranja tačke u strukturi D je O(logn).

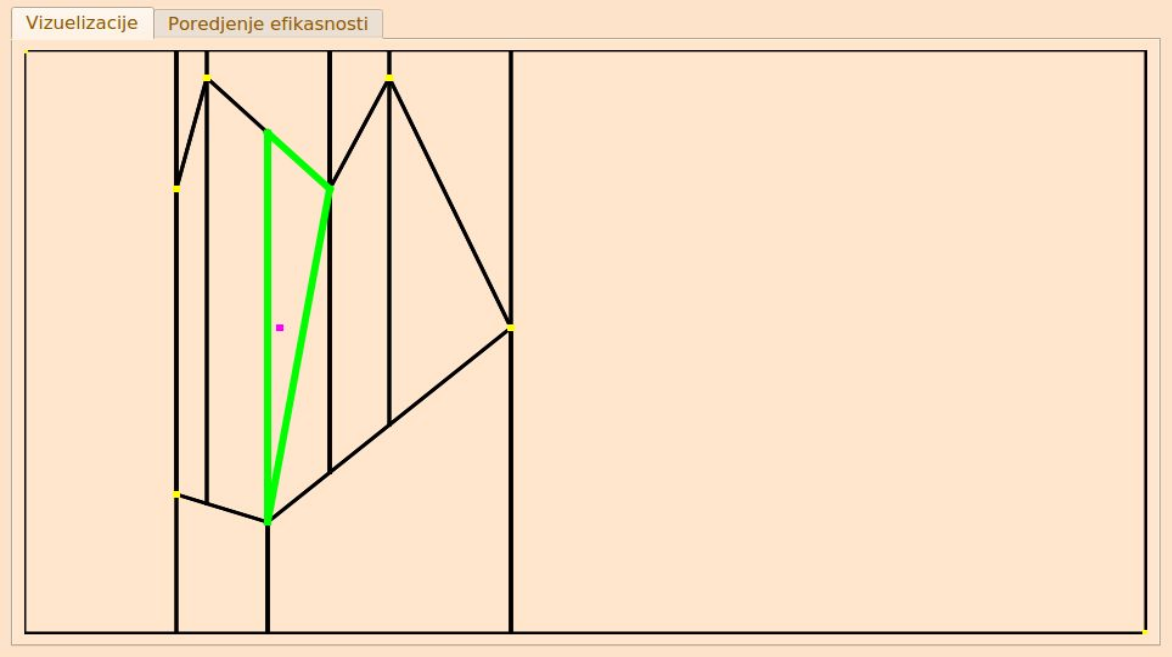


Vizuelizacija algoritma

U vizuelizaciji implementiranog algoritma ivice trapeznog razlaganja ravni su crne boje, a tjemena su žute boje. Duž koja se dodaje u trenutnom koraku algoritma je crvene boje, dok su njena tjemena tamno plave boje. Tjeme te duži koje se trenutno locira u strukturi pretrage trapeznog razlaganje je svijetlo plave boje. Prilikom lociranja tačke u strukturi pretrage provjeravaju se neke tačke i duži, sve dok se ne dođe do trapeza koji je list i koji

sadrži traženu tačku. Čvorovi strukture pretrage koji se provjeravaju prilikom lociranja u vizuelizaciji algoritma predstavljeni su zelenom bojom. Tačka čije koodinate korisnik zadaje preko grafičkog korisničkog interfejsa i koja se nakon izgradnje trapeznog razlaganja locira predstavljena je rozom bojom. Naredne slike prikazuju vizuelizaciju.





Poredjenje efikasnosti naivnog i naprednog algoritma

U tabeli koja slijedi prikazano je vrijeme izvršavanja optimanog i naivnog algoritma u sekundama, pri čemu se dimenzija ulaza zadaje kao broj tačaka.

Algoritam/ Dimenzija ulaza	100	500	1000	5000	10000
Optimalni	0.013444	0.072455	0.148142	0.883808	1.89405
Naivni	0.026447	0.470072	1.65069	37.70590	142.908

Testiranje ispravnosti algoritma

U tabeli koja slijedi prikazano je na kojim testovima je izvedeno testiranje implementiranog algoritma. Testiranje je izvedeno unit testovima pri čemu je korišćen Google Test. Testovi obuhvataju sposobnost programa da detektuje neispravne ulazne podatke, i time spriječi grešku i fazi izvršavanja, poređenje jednakosti izlaza naivnog i optimalnog algoritma, kao i provjeru ispravnosti izvršavanja i naivnog i optimalnog algorima.

Naziv testa	Opis testa	Ulaz	Očekivani izlaz
InvalidInput1	Niz duži je prazan. Algoritam neće biti izvšen.	[] Traži se tačka (0,0).	AlgorithmStatus ::INVALID_INPUT
InvalidInput2	U nizu postoje duži čije se unutrašnjosti	[(100,100),(200,200)], [(100,200),(200,100)]	AlgorithmStatus ::INVALID_INPUT

	sijeku. Algoritam neće biti izvršen.	Traži se tačka (0,0).	
compareOutputs1	Zadavanje nasumičnog niza tačaka dimenzije 30.	Niz od 30 tačaka. Traži se tačka (100,100).	Poklapanje rezultata naivnog i optimalnog algoritma.
compareOutputs2	Zadavanje nasumičnog niza tačaka dimenzije 300.	Niz od 300 tačaka. Traži se tačka (100,100).	Poklapanje rezultata naivnog i optimalnog algoritma.
smallExample	Razlaganje ravni na 2 oblasti.	Zadate 2 oblasti. Traži se tačka (100, 100).	1F
biggerExample	Razlaganje ravni na 3 oblasti.	Zadate 3 oblasti. Traži se tačka (190, 170).	1F
pointOnTheLine	Razlaganje ravni na 3 oblasti, a tačka koja se traži je na jednoj od duži razlaganja.	Zadate 3 oblasti. Traži se tačka (300, 190).	2F
randomTest	Zadavanje nasumičnog niza tačaka dimenzije 20.	Niz tačaka dimenzije 20. Traži se tačka (100,100).	0F