Geometrijski algoritmi @ MATF

Algoritmi za pronalaženje para najbližih tačaka

Lazar Jovanović 1099/19

Opis problema

Za dati skup tačaka u ravni, izračunati par tačaka čije je međusobno rastojanje manje ili jednako od svih drugih parova tačaka. Vremenska složenost alogritama treba da bude O($n*\log(n)$), gde je n broj tačaka.

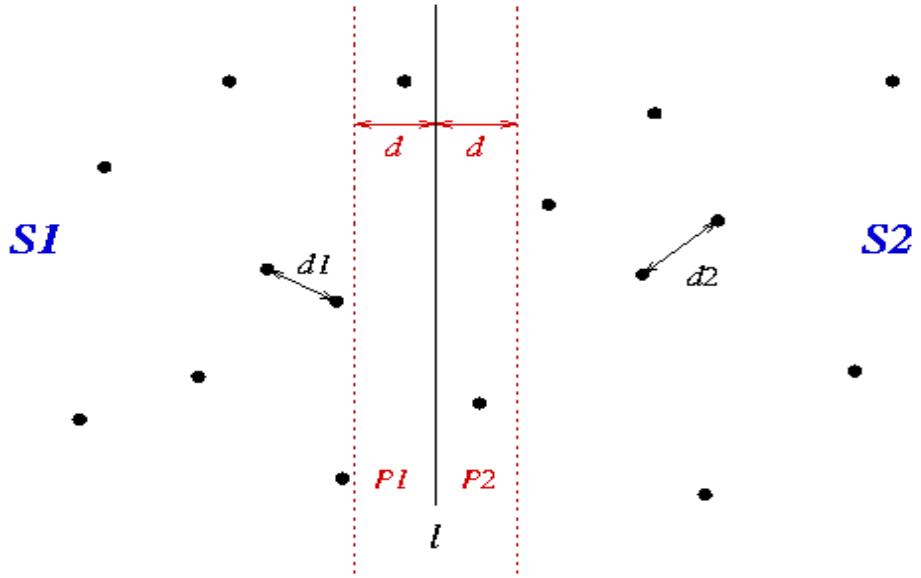
Primena: Primne ovog algoritma su brojne. Jedan primer upotrebe jeste forimranje hijerahijskog klasterovanja na osnovu najbližeg para instanci. Drugi primer upotrebe bio bi praćenje kretanja aviona u kontroli leta i detekcija mogućeg sudara ukoliko priđu preblizu.

Ulaz: Skup od n tačaka u ravni.
Izlaz: Dve najbliže tačke i rastojanje između njih.

Naivno rešenje problema

Naivno rešenje oslanja se na algoritam grube sile. Svaka tačka poredi se sa svim ostalim tačkama pri čemu se pamti minimalno rastojanje i par tačaka za koje se to rastojanje dobija. Imajući u vidu da se svaka tačka poredi sa svakom (kao i da je rastojanje simetrična relacija). Dobijamo složenost O($\frac{n*(n-1)}{2}$) tj. O(n^2).

Algoritam podeli pa vladaj



Kao pretprocesiranje sortiramo tačke neopadajuće prema x koordinati. Napravimo kopiju niza tačaka i njega sortiramo neopadajuće prema y koordinati. Pratimo sledeće korake:

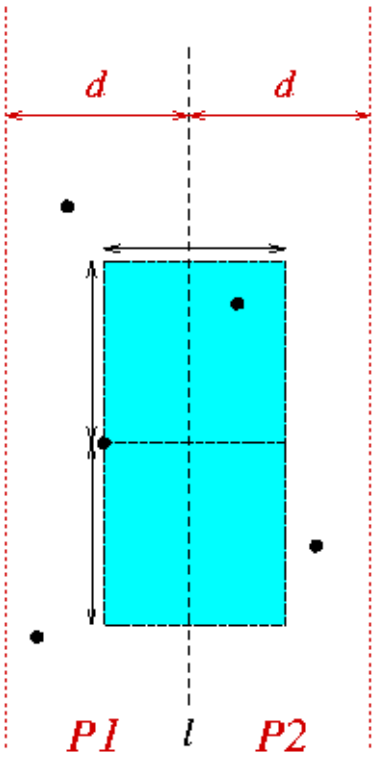
1. Nađemo srednju tačku po x koordinati u sortiranom nizu. *O(1)*
2. Podelimo tačke na dve polovine. *O(1)*
3. Rekurzivno nađemo najmanja rastojanja u levoj i desnoj polovini. Neka je minimum ova dva rastojanja *d*. *O(1)*
4. Pravimo vektor *tmp* koji sadrži tačke koje su na rastojanju najviše *d* od središnje tačke. *O(n)*
5. Nalazimo najmanje rastojanje u *tmp*. *O(1)*
6. Vratimo minimum od *d* i rastojanja koje smo izračunali u koraku 5. *O(1)*

Budući da smo u koraku pripreme podataka izvršili sortiranje koje je složenosti O($n*\log(n)$), to je i ukupna složenost algoritma.

Imajući u vidu da je jedino 4. korak netrivijalan i predstavlja srž optimizacije njemu posvećujemo dodtanu pažnju.
Naime, u pojasu oko srenje tačke tražimo najmanje rastojanje. Porđenje svake tačke sa svakom u ovom koraku vodi kvadratnoj složenosti. Međutim, to nije slučaj zahvaljujući sledećem zapažanju.
Za bilo koju tačku *p* u nekom pojasu, dovoljno je proveriti tačke koje zadovoljavaju sledeće uslove:

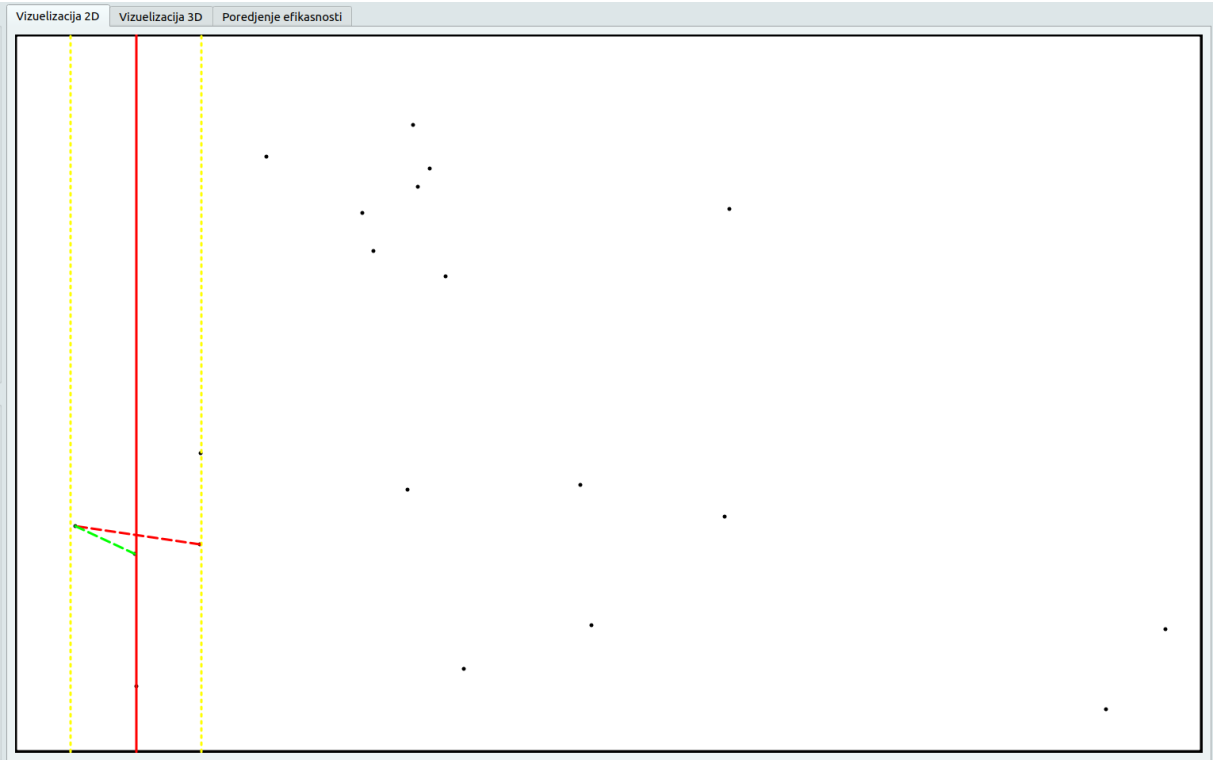
- da su na rastojanju *d* od tačke *p* u x smeru,
- da su na rastojanju *d* od tačke *p* u y smeru.

Ovo svojstvo proizilazi iz činjenice da tačke izvan kvadrada određenog dužinom *d* ne mogu biti na rastojanju manjem od *d* od tačke *p.* Zahvaljujući tome što smo u koraku pripreme sortirali tačke prema njihovoj y koordinati ovu proveru možemo izvesti u linearnom vremenu.



Vizuelizacija

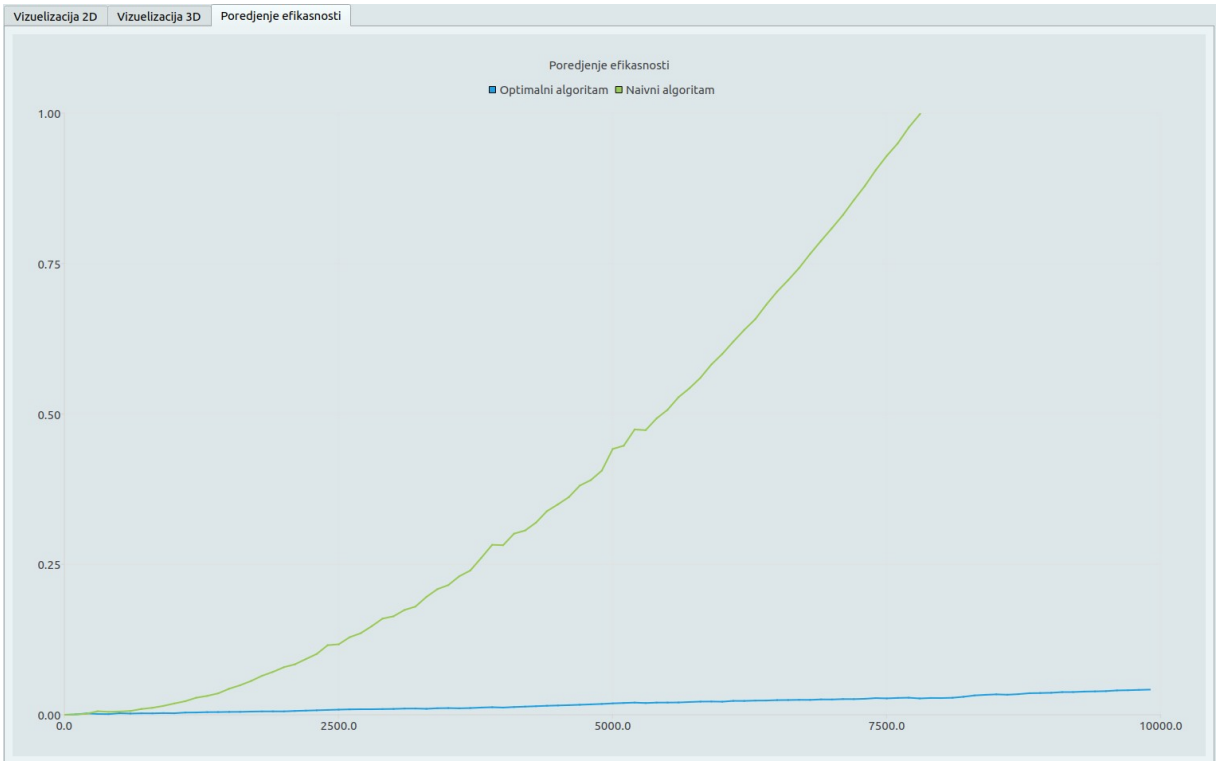
Vizuealizacija je izvršena u skladu sa navedenim načinom rada algoritma. Pronalazi se srednja tačka kroz koju se povlači crvna linija, levo i desno od nje se na rastojanju *d* nalaze isprekidane žute linije unutar kojih se vrši traženje manjeg rastojanja. Isprekidana zelena linija je trenutno minimalno rastojanje, koje je na kraju izvršavanja globalno minimalno rastojanje.



Vizuelizacija algortitma pronalaženja najbližih tačaka

Poređenje efikasnosti naivnog i naprednog algoritma

Na grafikonu je prikazano poređenje naivnog i optimalnog algortima. Možemo primetiti da je naivni algoritam za manje ulaze dovoljno dobar, dok je za veće ulaze gotovo neupotrebljiv.



Testiranje ispravnosti algoritma

Optimalni i naivni algoritam su testirani korišćenjem *Google Test* biblioteke. Karakteristike testova mogu se videti u tabili ispod.

Naziv testa	Opis testa	Ulaz	Očekivani izlaz
test_two_points	Testiranje dve tačke optimizovanog algoritma	Datoteka sa dve tačke	141.42135
test_two_points_naive	Testiranje dve tačke naivnog algoritma	Datoteka sa dve tačke	141.42135

optimized_equals_file_input	Testiranje optimizovanog algoritma u odnosu na fiskni ulaz iz fajlaTestiranje optimizovanog algoritma u odnosu na fiskni ulaz iz fajla	Datoteka sa 20 tačaka	Dužina: 39.6232 Tačke: (850, 339), (889, 323)
naive_equals_file_input	Testiranje naivnog algoritma u odnosu na fiskni ulaz iz fajla	Datoteka sa 20 tačaka	Dužina: 39.6232 Tačke: (850, 339), (889, 323)
naive_optimized_equals_big_input	Testiranje jednakosti između naivnog i optimizovanog algoritma	Nasumično generisanih 250 tačaka	Poklapanje rezulzata naivnog I optimizovanog algoritma