Geometrijski algoritmi @ MATF

**Algoritam za izračunavanje minimalnog zatvarajućeg diska**

Rastko Đorđević

horizontal line

# 

# Opis problema

Neka je dato n tačaka u ravni, potrebno je pronaći zatvoreni disk sa minimalnim poluprečnikom i to takav da pokriva sve date tačke. Krug je skup tačaka ekvidistantnih od tačke centra. Disk predstavlja skup tačaka koje se nalaze unutar kruga. Zatvoren disk sadrži i krug koji ga ograničava.

**Ulaz**: *skup od n tačaka u ravni*

**Izlaz**: *zatvoreni disk minimalnog poluprečnika koji pokriva sve tačke*

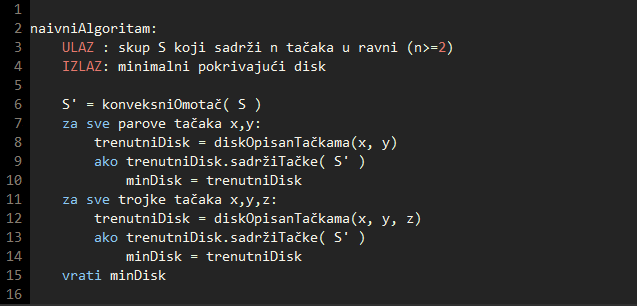
### Naivno rešenje problema

Pre diskutovanja algoritma, primetimo da je svaki krug određen sa tri tačke (kao opisan krug oko trougla koje te tačke definišu). Takođe može se dokazati da za bilo koji konačan skup tačaka najmanji pokrivajući disk sadrži bar tri tačke na njegovoj granici, ili ima dve i to takve da zajedno grade prečnik kruga koji opisuje disk.

Iz prethodnih zaključaka direktno sledi ideja za naivni algoritam vremenske složenosti O(n^4) gde je n broj tačaka datog skupa.

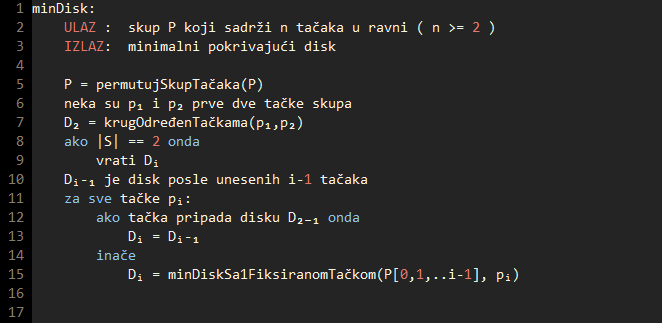
Prvo generišemo sve parove tačaka u O(n^2) vremenu, i linearno proverimo za svaki par da li disk koji generišu sadrži sve tačke. Potom generišemo u O(n^3) vremenu sve različite kombinacije trojke tačaka, i za svaku od njih linearno izvršimo istu proveru.

Ovaj algoritam se može ubrzati, ako obradjujemo samo tačke koje grade konveksni omotač skupa. Ako na ovaj način preprocesiramo skup tačaka koji nam je dat, vremenska složenost naivnog algoritma postaje O(h^4), gde je h broj tačaka konveksnog omotača.



### Randomizirani inkrementalni algoritam

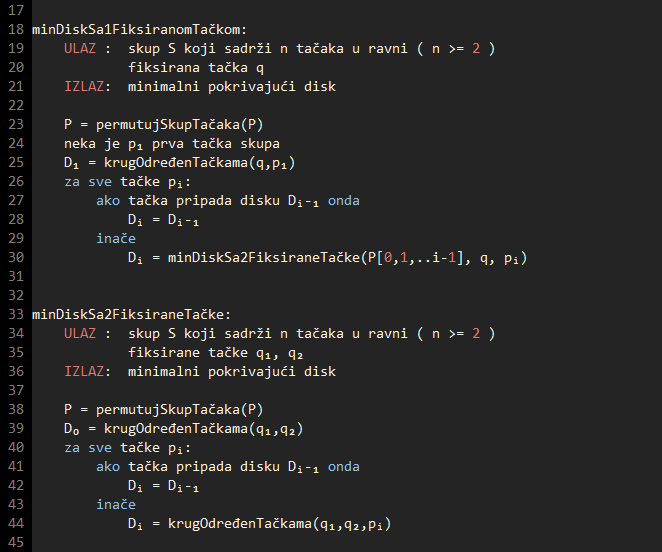
U nastavku će biti predstavljen algoritam koji rešava problem najmanjeg pokrivajućeg diska u očekivanom O(n) vremenu. Za početak ćemo permutovati dat skup tačaka. Izabraćemo bilo koje dve tačke, i izračunati jedinstveni krug čiji je prečnik određen ovim tačkama. Sa Di-1 ćemo označiti minimalni disk nakon unosa i-1 tačaka. Za tačku pi koju trenutno ubacujemo, u konstantnom vremenu možemo izračunati da li leži unutar Di-1.Ako se nalazi unutar diska, onda prelazimo na sledeću tačku a minimalni disk ostaje isti Di = Di-1. Ako ne pripada disku onda moramo da ažuriramo minimalni disk, tako da sadrži i tačku pi i to na njegovom obodu. Nakon što ubacimo poslednju tačku vratimo Dn kao minimalni pokrivajući disk.

 Pseudokod prvog nivoa rekurzije

Ostaje pitanje kako ažurirati minimalni disk kada unosimo novu tačku. Iako je primamljivo pretpostaviti da možemo da ga izračunamo samo koristeći tačke koje zatvaraju prethodni minimalni disk i novu tačku, postoje slučajevi koji to onemogućavaju, tako da ćemo morati da uzmemo u obzir sve do sada unete tačke. Važno tvrđenje je da ako pi nije u minimalnom disku prvih i-1 tačaka, onda se nalazi na obodu minimalnog diska prvih i tačaka.

Ažuriranje se vrši na sledeći način. Fiksiramo tačku koju ubacujemo što znamo da možemo da uradimo iz prethodnog tvrđenja, nakon čega smo efektivno smanjili dimenzionalnost problema. U ovom drugom nivou rekurzije kada dođemo do tačke koja se nalazi van trenutnog minimalnog diska, rekurzivno ćemo pozvati podproblem koji ima dve fiksirane tačke od kojih je druga trenutno ubačena tačka.

Konačno u trećem nivou rekurzije ako dođemo do tačke koja se ne nalazi u trenutnom minimalnom disku imamo podproblem koji ima tri fiksirane tačke. Ali rešenje ovog problema je jedinstveno tako je i rešenje problema jedinstveno.

 Pseudokod drugog i trećeg nivoa rekurzije

## Poredjenje efikasnosti naivnog i naprednog algoritma

Ovde treba da bude dat tabelarni i/ili grafički prikaz brzine izvršavanja oba algoritma u zavisnosti od veličine ulaza

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Alg. \ dim. ulaza** | 100 | 500 | 1.000 | 5.000 | 7.000 | 10.000 |
| **naivni** | 0,0012 | 0,098 | 0,0559 | 0,2918 | 2,6614 | 18,7632 |
| **optimalan** | 0,0002 | 0,0008 | 0,0021 | 0,0112 | 0,0189 | 0,0298 |

## Testiranje ispravnosti algoritma

Za testiranje je korišćena biblioteka google test. Pošto je algoritam koristi realne brojeve u pokretnom zarezu, svaki test je napisan imajući na umu potencijalne greške u računu. Algoritam dopušta greške od 0.0001, i očekuje se da vrednosti rezultata neće odstupati od očekivanog izlaza više od ove vrednosti koja se prožima kroz sve testove.

### SKUP TESTOVA: nalaženje jedinstvenog kruga pomoću tačaka

|  |  |  |  |
| --- | --- | --- | --- |
| Naziv testa | Opis testa | Ulaz | Očekivani izlaz |
| twoPoints | Zadavanje dve tačke kao ulaz, očekivani izlaz je krug čiji prečnik je određen tačkama sa ulaza | [{-1,0}, {1,0}] | circle.x == 0  circle.y == 0  circle.radius == 1 |
| threePoints\_twoHaveSameX | Kao ulaz dajemo tačke sa istim X koordinatama. Ako tačke grade duži paralelne sa x-osom može doći do problema pri izračunavanju. Očekivani izlaz je krug opisan oko trougla koji grade tačke sa ulaza. | [{1,2}, {1,4}, {2,3}] | circle.x == 1  circle.y == 3  circle.radius == 1 |
| threePoints\_twoHaveSameY | Kao ulaz dajemo tačke sa istim Y koordinatama. Ako tačke grade duži paralelne sa y-osom može doći do problema pri izračunavanju. Očekivani izlaz je krug opisan oko trougla koji grade tačke sa ulaza. | [{20,10}, {40,10}, {38,60}] | circle.x == 30.00  circle.y == 34.46  circle.radius == 26.59 |

### SKUP TESTOVA: naivni algoritam

Narednim skupom testova se testira ispravnost naivnog algoritma, pokriveni su granični slučajevi, kao i neki naizgled proizvoljni ulazi radi temeljne provere ispravnosti.

|  |  |  |  |
| --- | --- | --- | --- |
| Naziv testa | Opis testa | Ulaz | Očekivani izlaz |
| twoPoints | Kao ulaz imamo dve tačke.  Kao izlaz očekujemo minimalni disk nad tim tačkama, koji u ovom slučaju predstavlja krug čiji je poluprečnik određen tačkama sa ulaza. | [{10,10}, {30,10}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| threePoints | Kao ulaz imamo tri tačke.  Kao izlaz očekujemo minimalni disk nad tim tačkama. | [{10, 10}, {30, 10}, {20, 12}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| threeColinearPoints | Kao ulaz imamo tri tačke.  Kao izlaz očekujemo minimalni disk nad tim tačkama, koji u ovom graničnom slučaju predstavlja krug određen prečnikom najudaljenijih tačaka iz datog skupa. | [{10, 10}, {30, 10}, {15, 10}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| fourPointsOnDisk | Kao ulaz imamo četiri tačke koje se nalaze na obodu minimalnog diska koji određuju. Kao izlaz očekujemo taj disk. | [{10, 10}, {30, 10}, {20, 0}, {20, 20}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| fivePoints | Test sa više mogućnosti za grešku, ulaz čine 5 tačaka, a kao izlaz očekujemo minimalni disk. | [{10, 10}, {30, 10}, {20, 12}, {21, 11}, {19, 9}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |

### SKUP TESTOVA: napredni algoritam

Ovaj skup testova je izuzetno sličan prethodnom skupu, ulazi i izlazi su isti, jedina stvar koja se razlikuje je algoritam čiju ispravnost proveravamo.

|  |  |  |  |
| --- | --- | --- | --- |
| Naziv testa | Opis testa | Ulaz | Očekivani izlaz |
| twoPoints | Kao ulaz imamo dve tačke.  Kao izlaz očekujemo minimalni disk nad tim tačkama, koji u ovom slučaju predstavlja krug čiji je poluprečnik određen tačkama sa ulaza. | [{10,10}, {30,10}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| threePoints | Kao ulaz imamo tri tačke.  Kao izlaz očekujemo minimalni disk nad tim tačkama. | [{10, 10}, {30, 10}, {20, 12}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| threeColinearPoints | Kao ulaz imamo tri tačke.  Kao izlaz očekujemo minimalni disk nad tim tačkama, koji u ovom graničnom slučaju predstavlja krug određen prečnikom najudaljenijih tačaka iz datog skupa. | [{10, 10}, {30, 10}, {15, 10}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| fourPointsOnDisk | Kao ulaz imamo četiri tačke koje se nalaze na obodu minimalnog diska koji određuju. Kao izlaz očekujemo taj disk. | [{10, 10}, {30, 10}, {20, 0}, {20, 20}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |
| fivePoints | Test sa više mogućnosti za grešku, ulaz čine 5 tačaka, a kao izlaz očekujemo minimalni disk. | [{10, 10}, {30, 10}, {20, 12}, {21, 11}, {19, 9}] | minDisk.x == 20  minDisk.y == 10  minDisk.radius == 10 |

### SKUP TESTOVA: poređenje algoritama

Testovima koji slede poredimo izlaze naivnog i efikasnog algoritma na generisanim ulazima različitih veličina.

|  |  |  |  |
| --- | --- | --- | --- |
| Naziv testa | Opis testa | Ulaz | Očekivani izlaz |
| randomInput\_20points | Kao ulaz imamo proizvoljan niz tačaka, poredimo izlaze efikasnog i naivnog algoritma, pritom očekivajući da budu jednaki. | Proizvoljni niz veličine 20 | Poklapanje rezultata naivnog i efikasnog algoritma. |
| randomInput\_100points | Kao ulaz imamo proizvoljan niz tačaka, poredimo izlaze efikasnog i naivnog algoritma, pritom očekivajući da budu jednaki. | Proizvoljni niz veličine 100 | Poklapanje rezultata naivnog i efikasnog algoritma. |
| randomInput\_500points | Kao ulaz imamo proizvoljan niz tačaka, poredimo izlaze efikasnog i naivnog algoritma, pritom očekivajući da budu jednaki. | Proizvoljni niz veličine 500 | Poklapanje rezultata naivnog i efikasnog algoritma. |