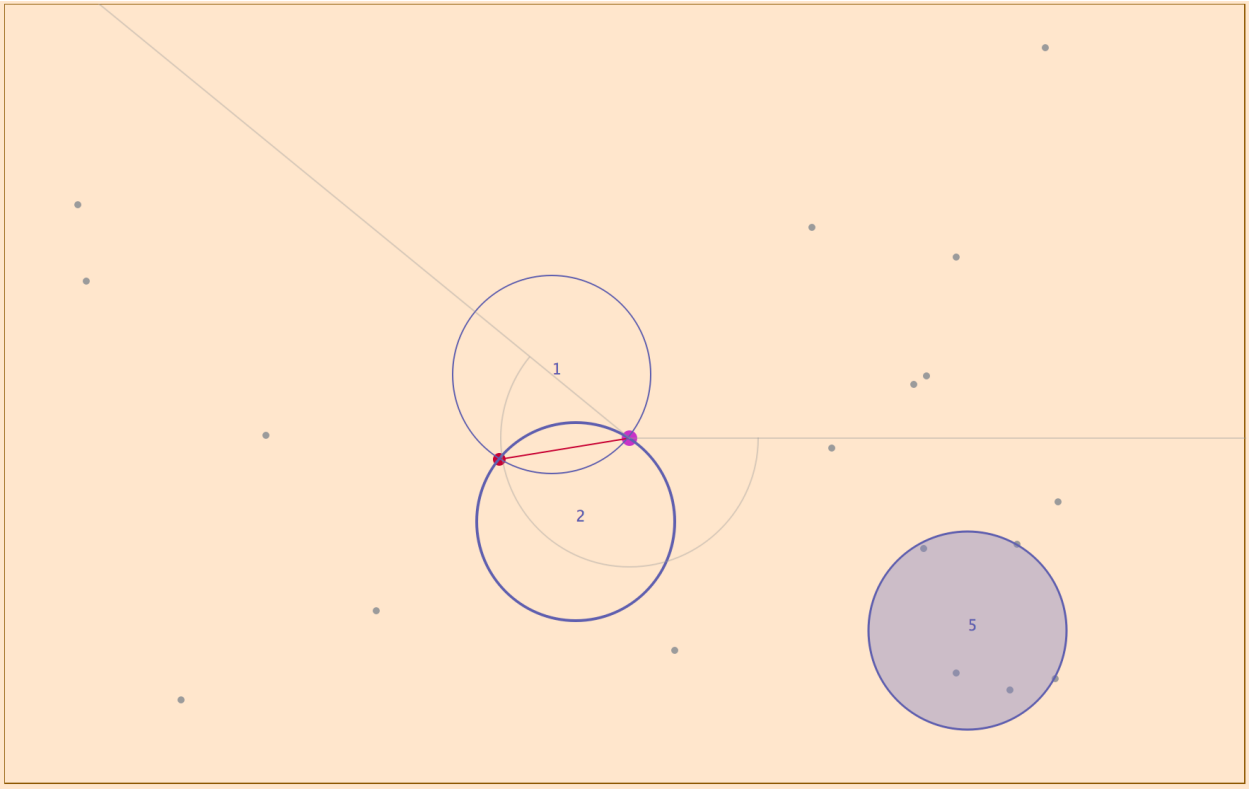
Geometrijski algoritmi @ MATF Luka Živanović 1085/2017

Algoritam za određivanje maksimalnog broja tačaka u krugu fiksnog prečnika brišućom pravom



Opis problema

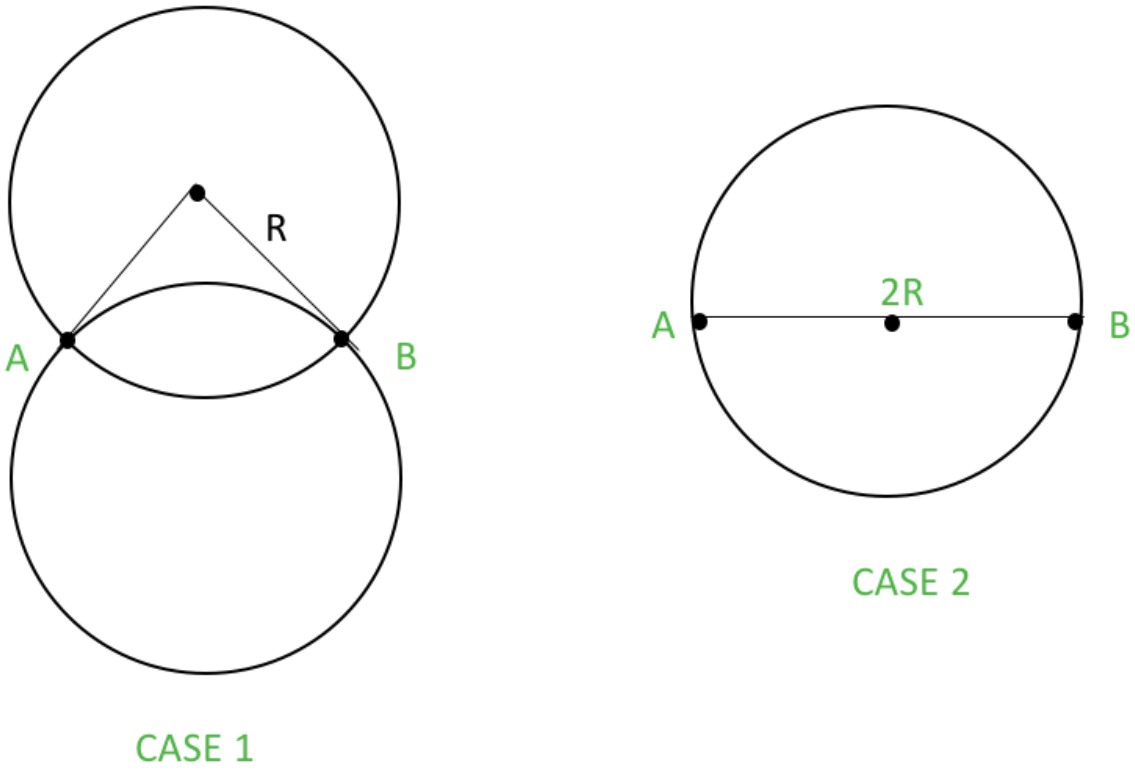
Neka je dat skup od **n** tačaka u ravni i prečnik kruga **r**. Potrebno je odrediti krug datog prečnika takav da sadrži maksimalni broj tačaka iz datog skupa. Ukoliko rešenje nije jedinstveno, vratiti bilo koji krug koji zadovoljava kriterijum.

Ulaz: Skup od **n** tačaka u ravni, poluprečnik kruga **r**

Izlaz: Najveći broj tačaka u jednom krugu

Naivno rešenje problema

Za proizvoljni par tačaka **A** i **B** iz skupa, čije je Euklidsko rastojanje manje ili jednako 2**r,** konstruišemo krugove poluprečnika **r** koji dodiruje obe tačke. Postoji maksimalno dva takva kruga.



Slika je preuzeta sa: https://www.geeksforgeeks.org

Za svaki krug brojimo koliko se tačaka nalazi u njemu koristeći Euklidsko rastojanje. Maksimalni broj tačaka u jednom krugu je rešenje.

Složenost algoritma je $O(n^3)$, što sledi iz ukupnog broja parova tačaka za čiji se izbor može uraditi u složenosti $O(n^2)$ i potom provere koliko se tačaka nalazi unutar krugova koje oni opisuju: $O(n)$ (za svaki krug).

Optimalni algoritam

Optimalni algoritam se zasniva na tehnici brišuće prave. Za razliku od algoritama koje smo obradili na kursu, brišuća prava u ovom slučaju se ne kreće pravolinijski i nije paralelna nekoj od osa, već se rotira oko tačke.

Ideja je da se za svaku tačku uradi sledeće:

1. Iteriramo kroz skup i biramo tačku
2. Konstruišemo krug poluprečnika **r**, sa središtem **C**, tako da data tačka leži na kružnici, a centar kruga se nalazi na istoj *y* koordinati na kojoj se data tačka nalazi. Postoje dva takva kruga, u ovoj implementaciji se uzima krug čiji je centar desno od date tačke.
3. Taj krug potom rotiramo oko date tačke u negativnom smeru. Svaki put kad kružnica dodirne novu tačku povećava se ukupan broj unutrašnjih tačaka, a kada postojeća prestane da pripada unutrašnjosti, smanjuje se.
4. Konačno rešenje je maksimalni broj tačaka vezan za jednu tačaka oko koje rotiramo.

(2. korak je uprošćen radi intuitivnosti algoritma)

Implementacioni koraci su slični prethodnim (bez uzimanja degenerisanih slučajeva u obzir):

1. Inicijalizujemo globalni maksimalni broj tačaka na 1
2. Iteriramo kroz dati skup tačaka i u svakoj iteraciji biramo jednu (**P**)
3. Iz ostatka skupa izdvajamo tačke koje su udaljene od **P** najviše 2**r**
4. Svaka izabrana tačka će kreirati dva dogadjaja (kada ulazi u krug i kada izlazi iz njega)
5. Ugao koji zaklapaju x-osa i duž **PC** pri svakom događaju dodajemo u niz
   (duž **PC** je takva duž koja dodiruje tačku **P**, i tačku **C**, koja je središte kruga kada tačka iz skupa ulazi/izlazi u kružnicu)
6. Sortiramo niz po uglovima, čime obezbeđujemo da svaki put kada obrađujemo ulazni događaj povećamo broj tačaka za jedan, odnsno smanjimo za jedan pri izlaznom događaju
7. Maksimum za trenutnu tačku inicijalizujemo na 1
8. Iteriramo kroz niz (koristeći pravila iz 6.) i ažuriramo trenutni maksimum, ako je potrebno

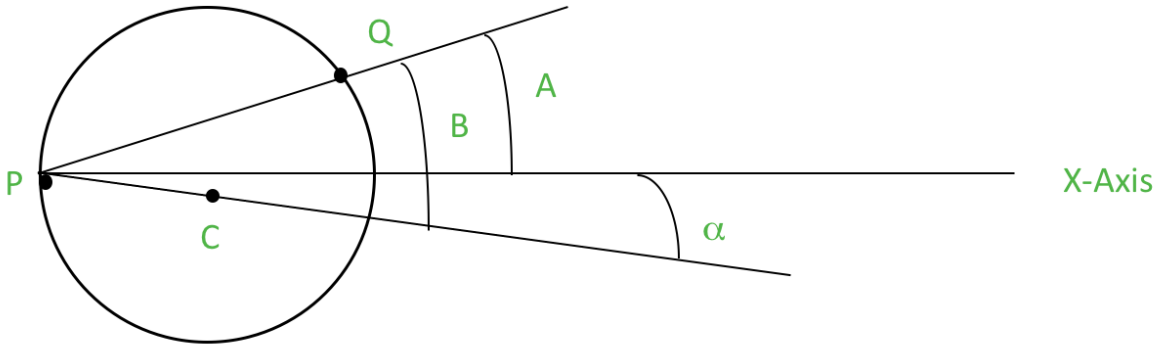
9. Ako je maksimum za trenutnu tačku veći od globalnog maksimuma, ažuriramo globalni maksimum

(korak 8. - Brišuća prava zapravo kreće od najmanjeg ugla u sortiranom nizu, ne od x ose kao što je rečeno u koraku 2 uprošćenog objašnjenja algoritma)

Ukoliko skup nema tačaka, algoritam treba da vrati 0. Taj slučaj nije ubačen u algoritam, zato što se smatra graničnim slučajem i nepotrebno bi povećalo kompleksnost algoritma.

Ulazni i izlazni ugao iz koraka 5 se računaju na sledeći način:

Neka nam je tačka oko koje se krug rotira **P**, a tačka za koju tražimo ugao **Q**. Posmatrajmo slučaj kada **Q** ulazi u krug:



Slika je preuzeta sa: https://www.geeksforgeeks.org

Tačka **C** predstavlja središte kruga kada tačka Q ulazi u njega.

Ugao **A** je ugao koji zaklapa x osa sa prvaom PQ i računa se na sledeći način:

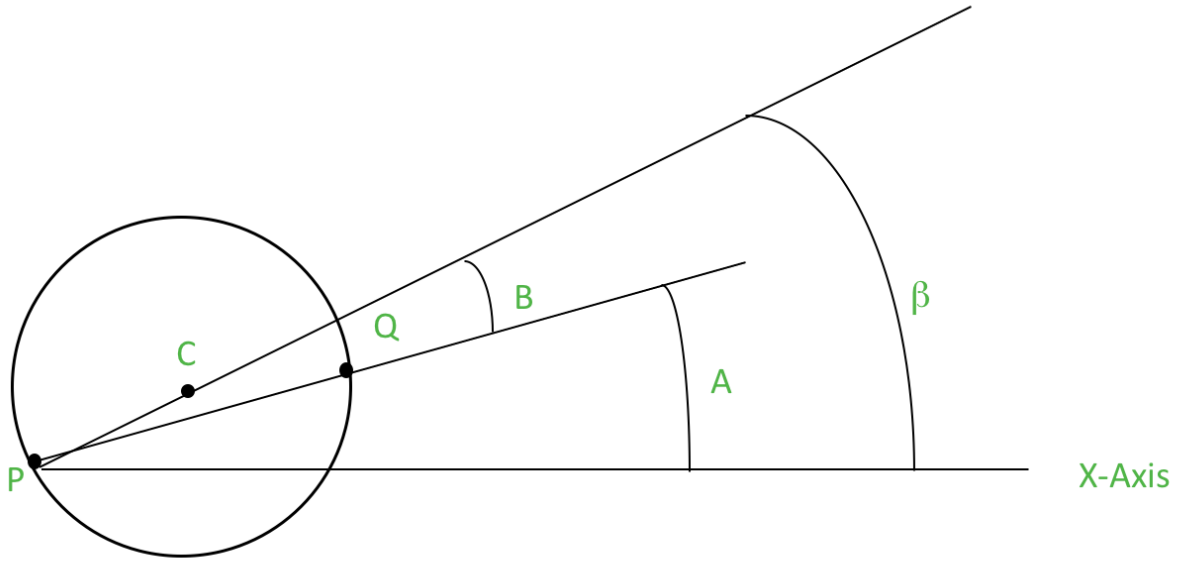
$$A = \arctan\left(\frac{P.y - Q.y}{P.x - Q.x}\right)$$

Ugao **B** je ugao koji zaklapaju prave PQ i PC i računa se na sledeći način:

$B = \arccos\left(\frac{d}{2 \cdot r}\right)$, gde je **d** Euklidsko rastojanje izmedju **P** i **Q**.

Ugao **α** računamo kao **A** – **B**. Rotacija brišuće prave počinje od x ose i kreće se u negativnom smeru, kao štoje gore navedeno, tako da **α** zaista jeste **A** – **B** (može biti negativan).

Slučaj kada **Q** izlazi iz kruga je sličan:



Slika je preuzeta sa: https://www.geeksforgeeks.org

Oznake su iste i način na koji se računaju **A** i **B**, samo što nam je u ovom slučaju **β** jednaka **A** + **B**.

Kada pravimo niz događaja, unosimo ugao alfa ili beta, i oznaku događaja (ulazni ili izlazni). Sortiranje se radi baš po tom uglu, u rastućem poretku.

Složenost optimalnog algoritma

Iteraciju kroz ceo skup tačaka i izbor trenutne tačke oko koje rotiramo krug možemo uraditi u vremenskoj složenosti $O(n)$.

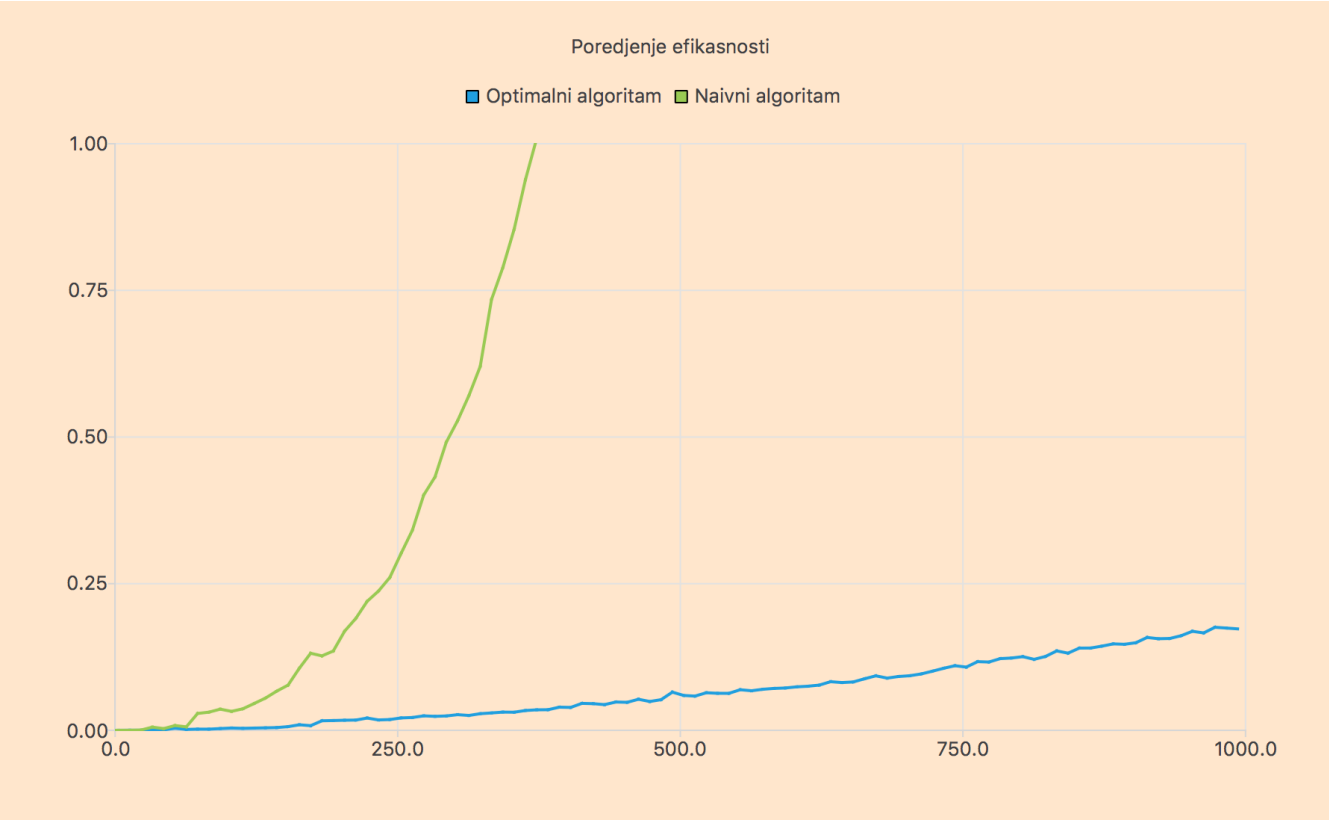
Za izabranu tačku treba obraditi sledeće:

1. Proveru da li je tačka u blizini, i dodavanje u skup možemo uraditi u vremenskoj složenosti $O(n)$.
2. Sortiranje bliskih tačaka možemo odraditi u vremenskoj složenosti $O(n\log(n))$.
3. Za ažuriranje maksimuma treba da obradimo svaki dogadjaj, što je 2 * broj bliskih tačaka, što se može uraditi u vremenskoj složenosti $O(n)$.

   Najveću vremensku složenost u ovom delu ima 2. korak : $O(n\log(n))$

Ukupno, optimalni algoritam ima složenost $O(n)\cdot O(n\log(n))$ što je $O(n^2\log(n))$

Poređenje naivnog i optimalnog algoritma



Ulaz (broj tačaka)	Naivni algoritam (s)	Optimalni algoritam (s)
10	3.5e-05	0.000428
50	0.00625	0.002351
100	0.03592	0.010321
200	0.137309	0.017531
300	0.467008	0.025055
400	1.15257	0.037379
500	2.07504	0.05422

Testiranje ispravnosti algoritma

Naziv Testa	Opis testa	Ulaz	Očekivani izlaz
Circle_efficient. noPoints	Ulaz koji nema ni jednu tačku	[]	0
Circle_efficient. lonelyPoint	Ulaz ima samo jednu tačku	{1,1}	1
Circle_efficient. twoSamePoints	Ulaz ima dva tačke koje imaju iste koordinate	{{1,1},{1,1}}	2
Circle_naive. noPoints	Ulaz koji nema ni jednu tačku	[]	0
Circle_naive. lonelyPoint	Ulaz ima samo jednu tačku	{1,1}	1
Circle_compare. twoSamePoints	Ulaz ima dva tačke koje imaju iste koordinate	{{1,1},{1,1}}	2
Circle_compare. randomInput_1	Poredjenje naivnog i optimalnog algoritma	Nasumično izabrano 30 tačaka	naivnoRešenje = optimalnoRešenje
Circle_compare. randomInput_2	Poredjenje naivnog i optimalnog algoritma	Nasumično izabrano 300 tačaka	naivnoRešenje = optimalnoRešenje
Circle_compare. circleRingInput	2, 3 ili 4 tačke će se nalaziti na samoj kružnici	{{0, 70}, {70, 0}, {140, 70}, {70, 140}}	4