Matematički fakultet

Projekat iz predmeta Konstrukcija i analiza algoritama 2

školska 2018/2019

Prave oslonca konveksnog mnogougla

Student:
Marija Radović
1065/2018

Mentori:
dr Vesna Marinković
Mirko Spasić

Februar, 2019

1. Uvod

Problem koji će biti predstavljen u ovom radu jeste iz oblasti geometrije, a rešenje izloženo u ovom radu predstavlja algoritam iz oblasti geometrijskih algoritama. Nakon predstavljenog zadatka, u nastavku rada će biti date definicije i objašnjenja pojmova koje će pomoći čitaocu da bolje razume izneto rešenje. Nakon toga biće izloženo teorijsko rešenje kao i implementacija sa rezultatima testiranja i vremenima izvršavanja.

2. Zadatak

Dat je konveksan mnogougao $P_0P_1...P_{n-1}$ i tačka van njega. Kako se mogu konstruisati dve prave oslonca mnogougla iz date tačke algoritmom složenosti $O(logn)$?

3. Definisanje pojmova

Tačka je u kodu predstavljena strukturom *coordinates* koja sadrzi *x* i *y* koordiante, kao i pridružen ugao u odnosu na pravu AP_0.

Konveksni mnogougao je mnogougao kod koga su svi unutrašnji uglovi manji od π radijana.

Prava oslonca konveksnog mnogougla je prava koja sadrži barem jednu tačku mnogougla i sve ostale tačke mnogougla se nalaze sa iste strane prave.

4. Teorijsko rešenje

Indeksu $i \in \{0,1,...,n-1\}$ pridružujemo veličinu ugla $\alpha_i = \angle P_0AP_i$. Jasno je da je $-\pi < \alpha_i < \pi$. Neka je $\alpha_n = 0$. Ako su svi uglovi α_i istog znaka (odnosno pozitivni, što se može pretpostaviti bez smanjenja opštosti), onda je AP_0 prava oslonca, a druga prava AP_m određena je temenom P_m za koje ugao α_m ima najveću apsolutnu vrednost. Teme P_m nalazi se binarnom pretragom intervala 1,2,...,n−1, jer niz $|\alpha_i|$ najpre raste, pa opada. U dve "srednje" tačke i, $i+1$ intervala indeksa posmatra se razlika uglova $\alpha_{i+1}-\alpha_i$, pa:

- ako je pozitivna, onda je $m \geq i+1$,
- ako je negativna, onda je $m \leq i$,
- ako je $\alpha_{i+1} = \alpha_i$, onda je, na primer, $m = i$, ali prava oslonca P_0P_m sadrži celu stranicu P_mP_{m+1}.

Ako pak nisu svi uglovi α_i istog znaka (tj. kada odredujemo P_0), onda postoje dve simetrične mogućnosti: uglovi rastu, opadaju, pa rastu, ili opadaju, rastu, pa opadaju. U oba slučaja se binarnom pretragom nalaze dva susedna temena sa uglovima suprotnog znaka. Ta dva temena razdvajaju interval indeksa na dva podintervala, u kojima se na opisani način (tražeći maksimalni ugao po apsolutnoj vrednosti) pronalazi binarnom pretragom po jedna prava oslonca. S obzirom da osnovna komponenta vremenske složenosti ovog algoritma potiče od binarne pretrage, onda je ukupna složenost algoritma $O(\log n)$.

5. Implementacija

Implementacija prethodno objašnjenog rešenja je data u programskom jeziku C. Pseudo kod implemetiranog rešenja je dat na sledećoj slici:

Algoritam: Pronalazak pravih oslonaca
Ulaz: tačke mnogougla, tačka iz koje se traže prave
Izlaz: prave oslonca

```
begin
        uglovi_istog_znaka = provera_uglovi_istog_znaka()

    if (uglovi istog znaka)
        AP0 je jedna prava oslonca
        APn = binarna_pretraga_maksimalnog_ugla()
        APn je druga prava oslonca
    else
        // ako su uglovi suprotnog znaka

        // binarna pretraga dva susedna temena
        // sa uglovima suprotnog znaka
        binarna_pretraga_intervala()
        APn = binarna_pretraga_maksimalnog_ugla_interval1()
        APm = binarna_pretraga_maksimalnog_ugla_interval2()
        APn i APm -prave oslonca

end
```

Algoritam počinje proverom da li su svi uglovi koje tačke mnogougla zaklapaju sa pravom AP_0 istog znaka. Ukoliko su uglovi istog znaka, AP_0 je jedna prava oslonca i druga prava oslonca se nalazi binarnom pretragom maksimalnog ugla u odnosu na pravu AP_0. Ukoliko, pak, mnogougao sadrži uglove suprotnog znaka, onda se binarnom pretragom pronalaze dve susedne tačke koje imaju uglove suprotnih znakova. Ove tačke dele mnogougao na dva intervala u kojima se binarnim pretragama maksimalnog ugla u odnosu na pravu AP_0 nalaze dve prave oslonca. Implementacija celokupnog algoritma je data na *Slici 1*.

```
/* Funkcija koja vrsi pronalazak pravih oslonaca za zadati mnogougao i tacku iz koje se trazi prava oslonca*/
static int processPolygon(struct coordinates* coord, struct coordinates A, int num)
{
    int start = 0;
    int end = num - 1;

    int start_case = check_sign(coord[0], A, coord[1], coord[num-1]);

    switch(start_case)
    {
        case 1:
        {
            /* Svi uglovi su istog znaka. AP0 je jedna prava oslonca, binarna pretraga druge po maks uglu*/
            coord[0].oslonac = 1;
            /* Provera da li i susedne tacke pripadaju mozda pravoj oslonca */
            if(get_angle(coord, A, 0) == get_angle(coord, A, 1))
            {
                coord[1].oslonac = 1;
            }
            else if(get_angle(coord, A, 0) == get_angle(coord, A, num-1))
            {
                coord[num-1].oslonac = 1;
            }
            binary_search_max(coord, A, 1, num-1);
        }
        break;
        default:
        {
            /* nisu svi uglovi istog znaka. */
            int i, j;
            /* Pretraga intervala, pronalazak dve uzastopne tacke suprotnog znaka*/
            binary_search_interval(coord, A, &i, &j, num);
            /* Binarna pretraga max ugla u oba podintervala */
            binary_search_max(coord, A, 0, i);
            binary_search_max(coord, A, j, num-1);
        }
    }
}
```

Slika 1. Procesuiranje

Provera znaka se obavlja funkcijom *check_sign.* Ova funkcija proverava pripadnost dveju tačaka određenoj strani prave. Funkcija je data na S*lici 2*.

```
/* Funkcija provere znaka uglova. Provera se vrsi utvrdjivanjem da li su obe tacke sa iste strane prave*/
/* t1 i t2 odredjuju pravu u odnosu na koju se odredjuje da li su tacke t3 i t4 sa iste strane */
/* Povratna vednost 1 ukoliko su sa iste strane ili je jedna tacka na pravoj, 0 ukoliko nisu */
static int check_sign(struct coordinates t1, struct coordinates t2, struct coordinates t3, struct coordinates t4)
{
    double A;
    double B;

    A = (t3.y - t1.y) * (t2.x - t1.x) - (t2.y - t1.y) * (t3.x - t1.x);
    B = (t4.y - t1.y) * (t2.x - t1.x) - (t2.y - t1.y) * (t4.x - t1.x);

    return (roundf(A * B) >= 0);
}
```

Slika 2. Provera znaka

Binarna pretraga intervala je implementirana u funkciji *binary_search_interval.* Binarnom pretragom kroz celokupan mnogougao, tražimo dve susedne tačke sa suprotnim znakom *(Slika 3.).* Pretraga se vrši u odnosu na ugao P_0AP_1.

```
/* Binarna pretraga intervala gde se znak razlikuje*/
static void binary_search_interval(struct coordinates* coord, struct coordinates A, int* i, int* j, int num)
{
    int start = 0;
    int end = num-1;

    int found = 0;
    while(!found)
    {
        int tmp_i = (end + start) / 2;
        int ret = check_sign(coord[0], A, coord[1], coord[tmp_i]);
        if (ret == 0)
        {
            end = tmp_i;
            if (end == start)
            {
                found = 1;
                *i = tmp_i-1;
                *j = tmp_i;
            }
        }
        else
        {
            start = tmp_i + 1;
        }
    }
}
```

Slika 3. Provera intervala

Pretraga maksimalnog ugla se vrši binarnom pretragom na određenom intervalu. Proverom razlike uglova dva susedna temena sa pravom AP_0 vršimo kretanje po intervalu. Ukoliko je razlika uglova veća od 0, nastavak pretrage se vrši u višem delu intervala, u suprotnom u nižem. Funkcija kojim se obavlja ovaj posao jeste *binary_search_max* (*Slika 4*.).

```
/* Binarna pretraga maksimalnog ugla u odredjenom intervalu u odnosu na P0A */
static void binary_search_max(struct coordinates* coord, struct coordinates A, int i, int j)
{
    int start = i;
    int end = j;
    int found = 0;
    while(!found)
    {
        int tmp_i = (end + start) / 2;
        int tmp_j = tmp_i + 1;
        if (get_angle(coord, A, tmp_i) > get_angle(coord, A, tmp_j))
        {
            if (tmp_i == start)
            {
                found = 1;
                coord[tmp_i].oslonac = 1;
            }
            else
            {
                end = tmp_i;
            }
        }
        else if (get_angle(coord, A, tmp_i) < get_angle(coord, A, tmp_j))
        {
            if (tmp_j == end)
            {
                found = 1;
                coord[tmp_j].oslonac = 1;
            }
            else
            {
                start = tmp_j;
            }
        }
        else
        {
            found = 1;
            coord[tmp_j].oslonac = 1;
            coord[tmp_i].oslonac = 1;
        }

    }
}
```

Slika 4. Traženje maksimalnog ugla

6. Složenost

Složenost algoritma je određena sa *c* binarnih pretraga, gde je *c* = 1 ili *c* = 3. Kako složenost binarne pretrage iznosi *O(logn)*, tako je za slučaj kada su svi uglovi isti, gde imamo samo jednu binarnu pretragu, složenost reda *O(logn)*. Za slučaj kada imamo uglove suprotnog znaka, algoritam zahteva 3 binarne pretrage, jednu za interval i dve za prave oslonca, pa složenost takodje iznosi *O(logn)*.

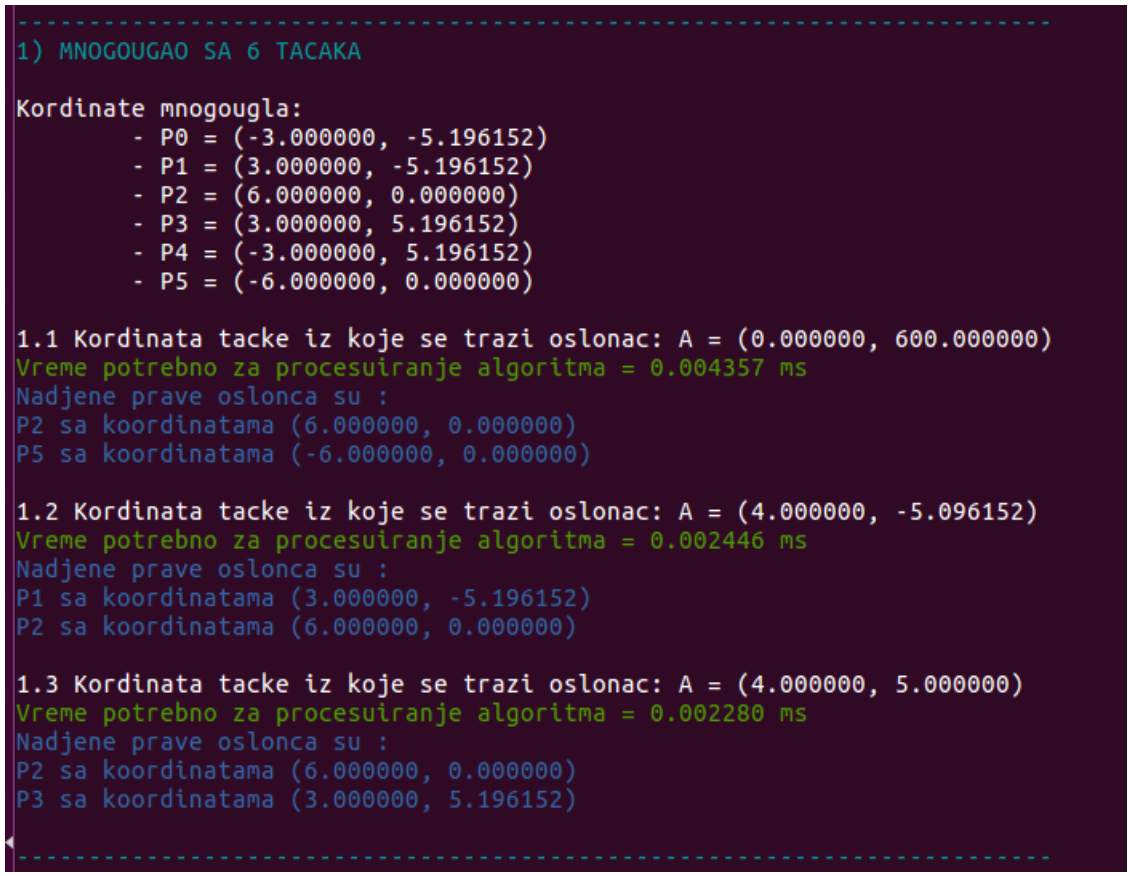
7. Rezultati

Rezultati merenja dokazuju tvrdnju o složenosti prethodno implementiranog algoritma. U *Tabeli 1* možemo videti progres vremena izvršavanja u odnosu na broj tačaka. Udvostručavanjem broja tačaka, vreme izvršavanja se uvećava približno jednako za svaki korak.

Broj tačaka	Jedna binarna pretraga	Tri binarne pretrage
1000	0.006 ms	0.009 ms
2000	0.008 ms	0.012 ms
4000	0.009 ms	0.014 ms
8000	0.01 ms	0.016 ms
16000	0.011 ms	0.0165 ms
32000	0.012 ms	0.017 ms
64000	0.013 ms	0.018 ms
128000	0.014 ms	0.021 ms
256000	0.015 ms	0.024 ms
512000	0.016 ms	0.026 ms
1024000	0.017 ms	0.027 ms
2048000	0.018 ms	0.028 ms

Tabela 1.

Ispravnost rada implementiranog rešenja algoritma je utvrđena testovima za veliki broj temena mnogougla. Na *Slikama 5, 6. i 7.* su dati izlazi algoritma za mnogougao sa brojevima temena 6, 10 i 1000 respektivno. U svakom od ovih slučajeva su računate prave oslonca iz tri različite perspektive radi pokrivanja što većeg broja slučajeva ispravnosti rada algoritma.



Slika 5.

```
--------------------------------------------------------------------------
2) MNOGOUGAO SA 10 TACAKA

Kordinate mnogougla:
        - P0 = (-3.090170, -9.510565)
        - P1 = (3.090170, -9.510565)
        - P2 = (8.090170, -5.877853)
        - P3 = (10.000000, 0.000000)
        - P4 = (8.090170, 5.877853)
        - P5 = (3.090170, 9.510565)
        - P6 = (-3.090170, 9.510565)
        - P7 = (-8.090170, 5.877853)
        - P8 = (-10.000000, 0.000000)
        - P9 = (-8.090170, -5.877853)

2.1 Kordinata tacke iz koje se trazi oslonac: A = (0.000000, 1000.000000)
Vreme potrebno za procesuiranje algoritma = 0.002519 ms
Nadjene prave oslonca su :
P3 sa koordinatama (10.000000, 0.000000)
P8 sa koordinatama (-10.000000, 0.000000)

2.2 Kordinata tacke iz koje se trazi oslonac: A = (4.000000, -9.510565)
Vreme potrebno za procesuiranje algoritma = 0.002462 ms
Nadjene prave oslonca su :
P0 sa koordinatama (-3.090170, -9.510565)
P1 sa koordinatama (3.090170, -9.510565)
P2 sa koordinatama (8.090170, -5.877853)

2.3 Kordinata tacke iz koje se trazi oslonac: A = (10.000000, 6.000000)
Vreme potrebno za procesuiranje algoritma = 0.002913 ms
Nadjene prave oslonca su :
P3 sa koordinatama (10.000000, 0.000000)
P5 sa koordinatama (3.090170, 9.510565)

--------------------------------------------------------------------------
```

Slika 6.

```
--------------------------------------------------------------------------
3) MNOGOUGAO SA 1000 TACAKA

Kordinate mnogougla: (Pogledati fajl koordinate.txt)
3.1 Kordinata tacke iz koje se trazi oslonac: A = (0.000000, 10000000.000000)
Vreme potrebno za procesuiranje algoritma = 0.009421 ms
Nadjene prave oslonca su :
P250 sa koordinatama (1000.000000, 0.000000)
P750 sa koordinatama (-1000.000000, 0.000000)

3.2 Kordinata tacke iz koje se trazi oslonac: A = (1005.000000, -1000.000000)
Vreme potrebno za procesuiranje algoritma = 0.006009 ms
Nadjene prave oslonca su :
P0 sa koordinatama (-0.000000, -1000.000000)
P251 sa koordinatama (999.980261, 6.283144)

3.3 Kordinata tacke iz koje se trazi oslonac: A = (1300.000000, 1300.000000)
Vreme potrebno za procesuiranje algoritma = 0.009094 ms
Nadjene prave oslonca su :
P217 sa koordinatama (978.580904, -205.862609)
P533 sa koordinatama (-205.862609, 978.580904)
```

Slika 7.

8. Literatura

- Nikola Ajzenhamer. Konstrukcija i analiza algoritama 2. http://nikolaajzenhamer.rs/pdf/kiaa2-v.pdf
- Vesna Marinković. Konstrukcija i analiza algoritama 2. http://poincare.matf.bg.ac.rs/~vesnap/kaa2/kaa2.pdf