Pripadnost točke poligonu

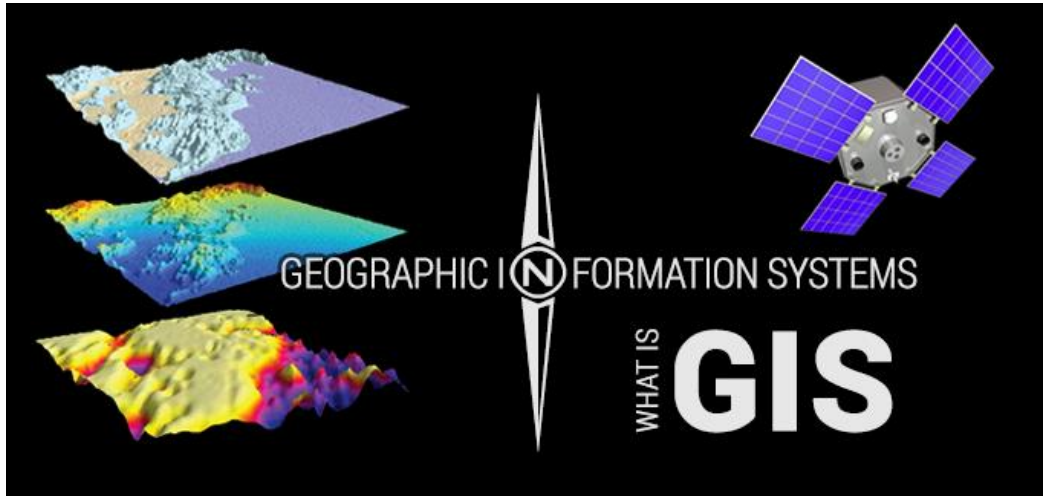
Nikola Jelić, 23. siječnja 2018.

Kolegij: Oblikovanje i analiza podataka

Sadržaj

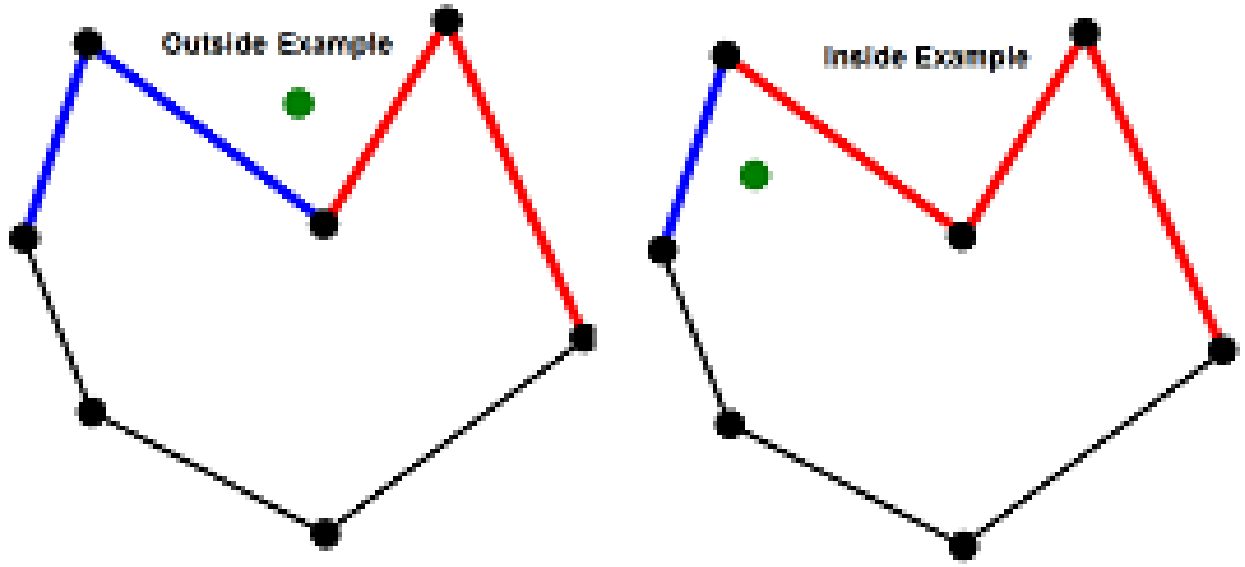
Motivacija

- Ovaj problem pripada računalnoj geometriji.
- Veliki je spektar primjene.
- Korisnost:
  - Računalna grafika
  - Računalni vid
  - Geografski informacijski sistemi (GIS).



Uvod

- Prvo, što je poligon, kako ga opisati?
- Kako modelirati problem?

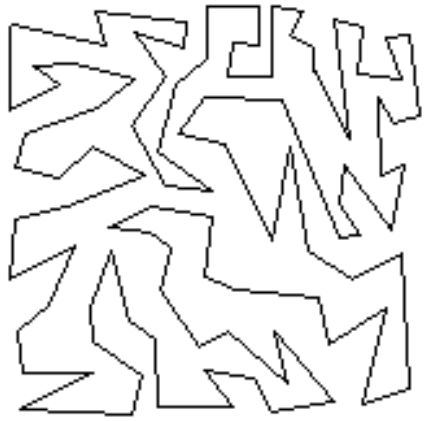


Uvod – definicije

- Poligon je zatvoren planaran put sastavljen od konačnog broja sekvencijalnih segmenata.
- Poligonski lanac je niz vrhova i bridova koji spajaju susjedne vrhove.
- Kažemo da je poligon P jednostavan ili Jordanov ako je sastavljen od nepresjecajućih bridova, a inače kažemo da je P složen.
- Na poligon P ćemo gledati kao na slijedni niz vrhova (P = $\{v_0 , v_1 , \ldots v_n = v_0 \}$), dok je točka q zadana svojim x i y koordinatama (q = (x, y)).

Uvod – definicije

- Poligon je zatvoren planaran put sastavljen od konačnog broja sekvencijalnih segmenata.
- Poligonski lanac je niz vrhova i bridova koji spajaju susjedne vrhove.
- Kažemo da je poligon P jednostavan ili Jordanov ako je sastavljen od nepresjecajućih bridova, a inače kažemo da je P složen.
- Na poligon P ćemo gledati kao na slijedni niz vrhova (P = $\{v_0 , v_1 , \ldots v_n = v_0 \}$), dok je točka q zadana svojim x i y koordinatama (q = (x, y)).



Uvod – osnovni teorem

- (*Jordanov teorem za jednostavni poligon*):

  Za svaki jednostavan poligon P, komplement R^2\P jednak je disjunktnoj uniji točno dvaju povezanih skupova *Int*(P) i *Ext*(P), pri čemu je *Int*(P) omeđen, a *Ext*(P) nije, presjek zatvarača im je P, a svaka poligonska linija s jednim krajem u interioru, a drugim u eksterioru siječe P.

Uvod – osnovni teorem

- (*Jordanov teorem za jednostavni poligon*):

  Za svaki jednostavan poligon P, komplement R^2\P jednak je disjunktnoj uniji točno dvaju povezanih skupova *Int*(P) i *Ext*(P), pri čemu je *Int*(P) omeđen, a *Ext*(P) nije, presjek zatvarača im je P, a svaka poligonska linija s jednim krajem u interioru, a drugim u eksterioru siječe P.

- Posljedice:
  - Ako su proizvoljne točke $p_1 \in$ *Ext*(P) i $p_2 \in$ *Int*(P) spojene poligonskim lancem, onda taj lanac mora sjeći P.

Uvod – osnovni teorem

- (*Jordanov teorem za jednostavni poligon*):

  Za svaki jednostavan poligon P, komplement R^2\P jednak je disjunktnoj uniji točno dvaju povezanih skupova *Int*(P) i *Ext*(P), pri čemu je *Int*(P) omeđen, a *Ext*(P) nije, presjek zatvarača im je P, a svaka poligonska linija s jednim krajem u interioru, a drugim u eksterioru siječe P.

- Posljedice:
  - Ako su proizvoljne točke $p_1 \in$ *Ext*(P) i $p_2 \in$ *Int*(P) spojene poligonskim lancem, onda taj lanac mora sjeći P.
  - Bilo koje dvije točke istog skupa mogu biti povezane poligonskim lancem koji ne siječe P.

Uvod – osnovni teorem

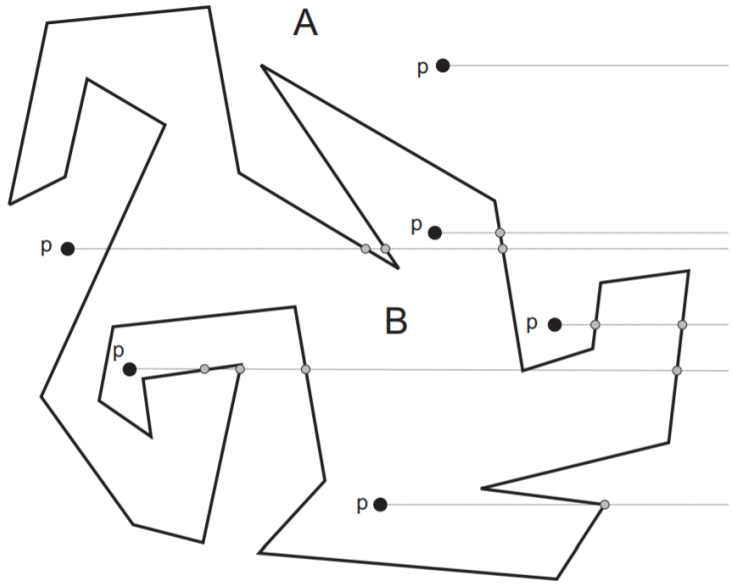
- (*Jordanov teorem za jednostavni poligon*):

  Za svaki jednostavan poligon P, komplement R^2\P jednak je disjunktnoj uniji točno dvaju povezanih skupova *Int*(P) i *Ext*(P), pri čemu je *Int*(P) omeđen, a *Ext*(P) nije, presjek zatvarača im je P, a svaka poligonska linija s jednim krajem u interioru, a drugim u eksterioru siječe P.

- Posljedice:
  - Ako su proizvoljne točke $p_1 \in$ *Ext*(P) i $p_2 \in$ *Int*(P) spojene poligonskim lancem, onda taj lanac mora presjecati P.
  - Bilo koje dvije točke istog skupa mogu biti povezane poligonskim lancem koji ne presjeca P.
  - Koncept parnosti: Promotrimo točku q i zraku čiji je početak u q i koja je fiksnog (odabranog) smjera. Tada je točka q ∈ *Ext*, ako zraka siječe P paran broj puta. Inače, ako zraka siječe P neparan broj puta, tada je q ∈ *Int*.

Uvod – osnovni teorem

- Crtanje zrake iz točke za koju ispitujemo pripadnost poligonu i brojenje sjecišta zrake sa poligonom daje nam ideju za prvi algoritam.



Algoritmi za jednostavne poligone

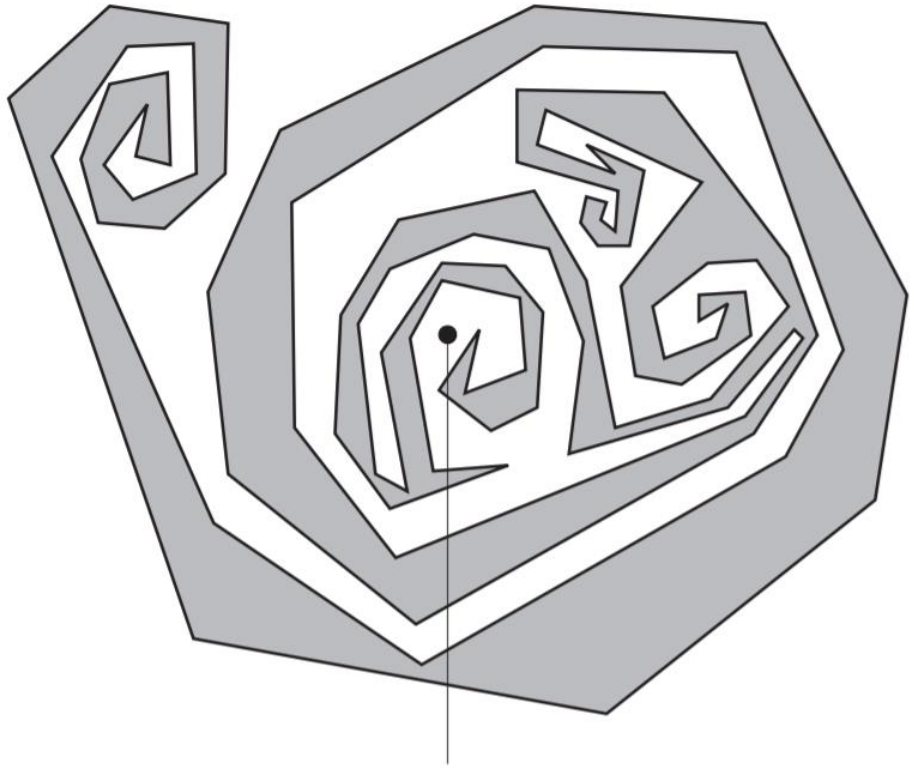
- Crossing number algoritam (CNA) – polazni algoritam
- Algoritam za konveksne poligone
- Aloritam za monotone poligone
- Trapezna dekompozicija jednostavnog poligona
- Triangulacija jednostavnog poligona

Crossing number algoritam (CNA)

- Nacrtamo zraku iz testne točke q (query point) prema dolje (ili u nekom drugom smjeru).
- Svaki puta kad poligon siječe zraku, on razdvaja tu zraku na dio koji je u interioru i dio koji je u eksterioru. Zraka obično ne ide u beskonačnost nego do neke točke za koju smo sigurni da je izvan poligona.
- Nadalje, prebrojimo sjecišta zrake sa poligonom.
- Ako je taj broj neparan, točka q je u interioru od P (ili, jednostavnije, u poligonu P), a ako je taj broj paran, točka q je u eksterioru od P (ili izvan P).

Crossing number algoritam (CNA)

- Zraka siječe poligon u 8 točaka pa je ona izvan poligona.



Crossing number algoritam (CNA)

- Neka je q = (x, y) i neka su (x_1 , y_1), (x_2, y_2) krajevi brida za koji je x_1<x<x_2.
- Presjek zrake i brida poligona:
  - Brid može zapisati kao {(t x_1 + (1 − t) x_2, t y_1 + (1 − t) y_2) : t ∈ [0, 1]}.
  - t = (x − x_2)/(x_1 − x_2)
  - crossing = (x, t y_1 + (1 − t) y_2)

Crossing number algoritam (CNA)

- Iznimne situacije:
  - Točka za koju ispitujemo pripadnost poligonu može biti točno na nekom od bridova.
  - Drugo, zraka može prolaziti točno kroz vrh.

- Rješenje - preturbiramo poligon, tj. točke neznatno pomaknemo u određenom smjeru. Na primjer, pomaknemo q neznatno u desno i to neće promijeniti ispitivanje je li točka u poligonu, ali će spriječiti da zraka prolazi točno kroz vrh poligona. Posebno treba uzeti u obzir kada točka pripada bridu.

Crossing number algoritam (CNA)

```
int crossings = 0
for (int i = 0; i < n; i++)
{
    if ((P(i).x < x && x < P(i+1).x) || (P(i).x > x && x > P(i+1).x))
    {
        t = (x - P(i+1).x) / (P(i).x - P(i+1).x)
        cy = t*P(i).y + (1-t)*P(i+1).y
        if (y == cy) return (bdy)
        else if (y > cy) crossings++;
    }
    if ((P(i).x == x && P(i).y <= y)
    {
        if (P(i).y == y) return (bdy);
        if (P(i+1).x == x)
        {
            if ((P(i).y <= y && y <= P(i+1).y)
               || (P(i).y >= y && y >= P(i+1).y))
               return (bdy);
        }
        else if (P(i+1).x > x) crossings++;
        if (P(i-1).x > x) crossings++;

     }
 }
 if (crossings neparan) return (in);
 else return (out);
```

Crossing number algoritam (CNA)

- Složenost:

Crossing number algoritam (CNA)

- Složenost:
  - Prolazimo for-petljom po svim bridovima.

Crossing number algoritam (CNA)

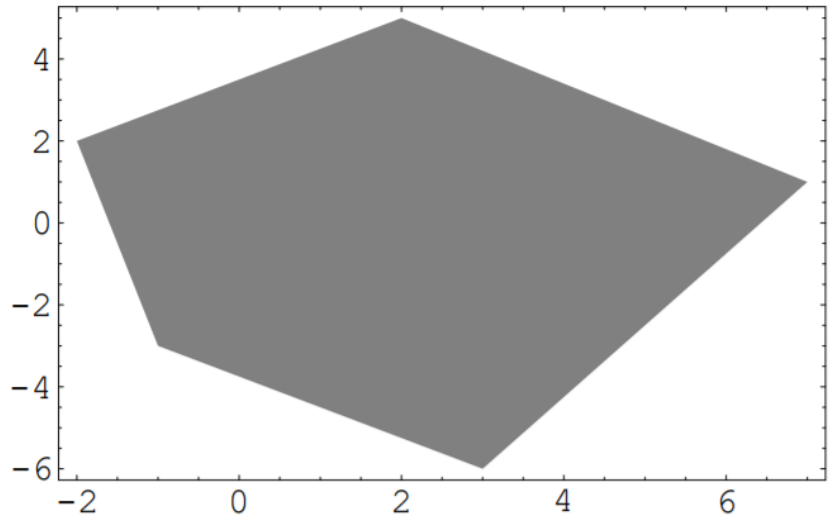
- Složenost:
  - Prolazimo for-petljom po svim bridovima, O(n).
  - Gledamo sve prethodno navedene slučajeve za određeni brid, O(1).

Crossing number algoritam (CNA)

- Složenost:
  - Prolazimo for-petljom po svim bridovima, O(n).
  - Gledamo sve prethodno navedene slučajeve za određeni brid, O(1).
  - Dakle, složenost je O(n).

Algoritam za konveksne poligone

- Za skup S kažemo da je konveksan ako:
  - $\forall x, y \in S \Rightarrow \lambda x + (1 - \lambda)y \in S, \lambda \in [0, 1]$.
- Neka su vrhovi zapisani u negativnom smjeru (kazaljke na satu).



Algoritam za konveksne poligone

- Za skup S kažemo da je konveksan ako:
  - $\forall x, y \in S \Rightarrow \lambda x + (1 - \lambda)y \in S, \lambda \in [0, 1]$.
- Neka su vrhovi zapisani u negativnom smjeru (kazaljke na satu).

- Početna ideja za ispitivanje pripadnosti točke konveksnom poligonu može izgledati ovako:
  - Pronađi vrhove s maksimalnom i minimalnom y koordinatom (brute–force).
  - Pomoću binarnog pretraživanja (binary search) pronađi dva brida koja sijeku pravac y = y(q).
  - Utvrdi leži li točka q između ta dva brida.

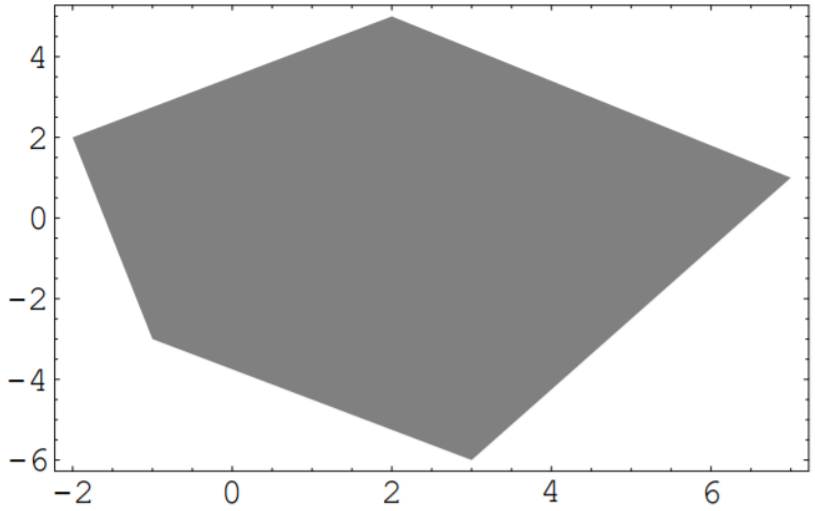
Algoritam za konveksne poligone

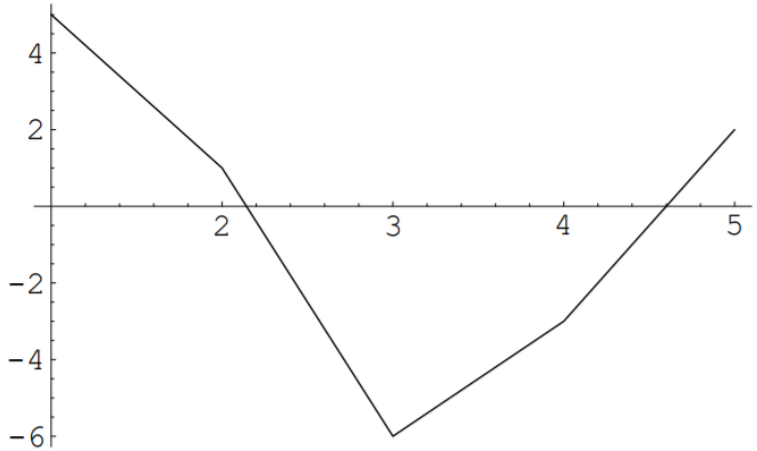
- Za skup S kažemo da je konveksan ako:
  - $\forall x, y \in S \Rightarrow \lambda x + (1 - \lambda)y \in S, \lambda \in [0, 1]$.
- Neka su vrhovi zapisani u negativnom smjeru (kazaljke na satu).

- Početna ideja za ispitivanje pripadnosti točke konveksnom poligonu može izgledati ovako:
  - Pronađi vrhove s maksimalnom i minimalnom y koordinatom (brute–force). O(n)
  - Pomoću binarnog pretraživanja (binary search) pronađi dva brida koja sijeku pravac y = y(q). O(log n)
  - Utvrdi leži li točka q između ta dva brida. O(1)

Algoritam za konveksne poligone

- Cilj: redukcija složenosti prvog dijela algoritma na O(log n).
- Ideja:
  - Neka su y(1), y(2), . . ., y(n) y koordinate vrhova poligona.
  - Izračunaj max{y(1), y(⌊ n/3 ⌋), y(⌊ 2n/3 ⌋), y(n)}.





Algoritam za konveksne poligone

- Cilj: redukcija složenosti prvog dijela algoritma na O(log n).
- Ideja:
  - Neka su y(1), y(2), . . ., y(n) y koordinate vrhova poligona.
  - Izračunaj max{y(1), y(⌊ n/3 ⌋), y(⌊ 2n/3 ⌋), y(n)}.
  - Ako je maksimalna vrijednost y(⌊ n/3 ⌋) (odnosno, y(⌊ 2n/3 ⌋)) tada se maksimum očito nalazi negdje u prvih (zadnjih) 2/3 uzoraka. Tako smo problem skratili za 1/3 .
  - Ako je maksimalna vrijednost y(1) (odnosno, y(n)) tada maksimum mora ležati u prvih (zadnjih) 1/3 uzoraka i ostalih 2/3 otpada.
- Očito je da je sada složenost: $\mathcal{O}(\log_{\frac{3}{2}} n) = \mathcal{O}(\log n)$

Algoritam za konveksne poligone

- Pogledajmo par linija koda:
- //ovako izgleda tipični binary search koji je korišten u drugom koraku //algoritma

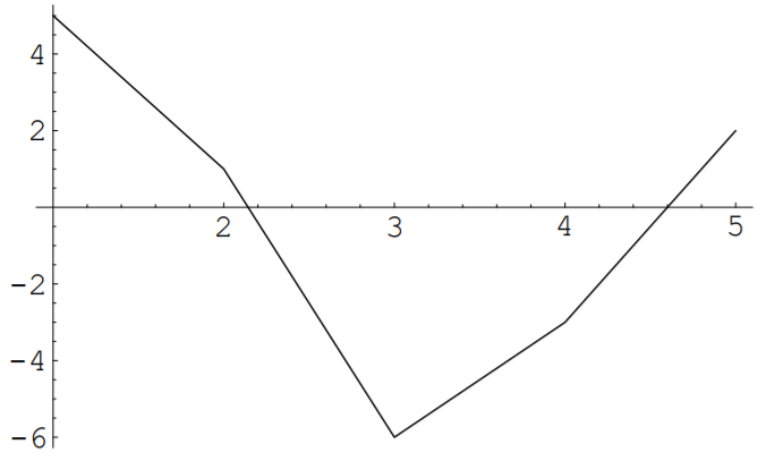
```
while(L1<R1){
        d = ( L1 + R1 )/2;    // d = ( R1 - L1 )/2 +L1
        if(y[d]>Y)
                R1=d-1;
        else if(y[d]<=Y)
                L1=d+1;
}
```

Algoritam za konveksne poligone

```
//ovako izgleda trinary search koji je korišten u prvom koraku
//algoritma
 for (;R-L > 4;){            //prije se inicijaliziraju R i L
       b=(R-L)/3+L;          // !!! b=( R + L )/3 NIJE ISTO
       c=2*(R-L)/3+L;
       if( max( max( y[L],y[b] ), max( y[c],y[R] ) ) == y[L] ){ //gledamo koji y je najveći
             if(R==b)
                   R=b-1;
             else R=b;
}
```

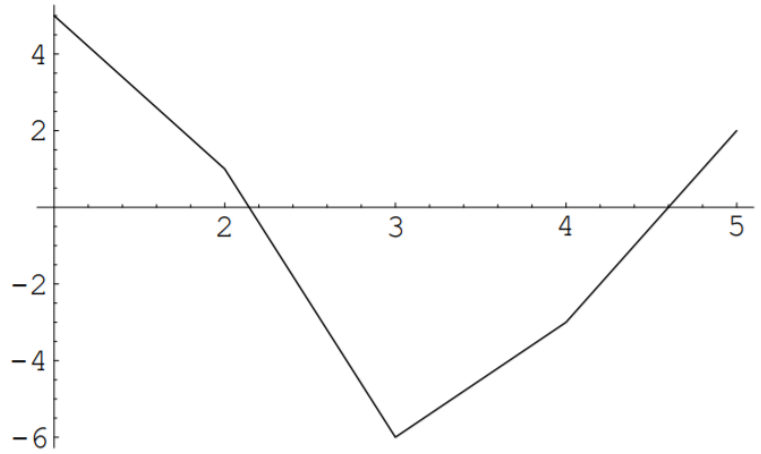
Algoritam za konveksne poligone

```
        if(max( max( y[L],y[b] ), max( y[c],y[R] ) ) == y[b] ) {
            if(R==c)
                R=c-1;
            else R=c;
}
```



Algoritam za konveksne poligone

```
if( max( max( y[L],y[b] ), max( y[c],y[R] ) )  ==y[c] ){
        if(L==b)
            L=b+1;
        else L=b;}
if( max( max( y[L],y[b] ), max( y[c],y[R] ) ) == y[R] ) {
        if(L==c)
            L=c+1;
        else L=c;
    }
}//kraj
```

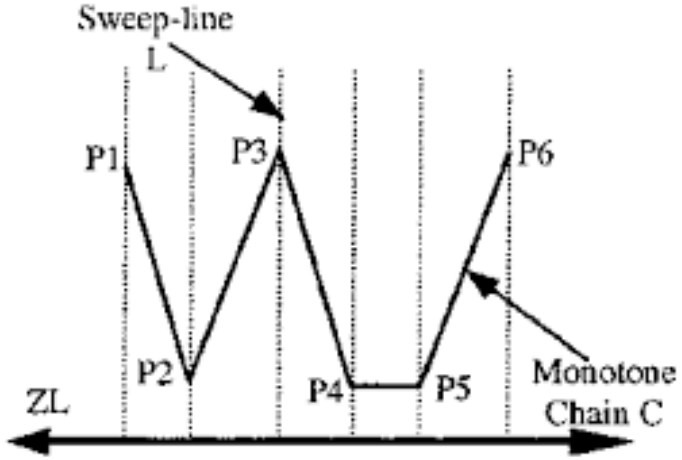


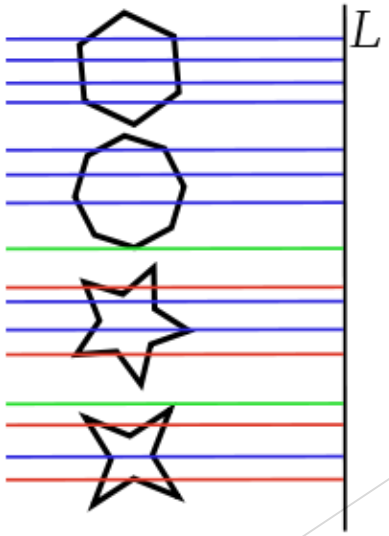
Aloritam za monotone poligone

- Kažemo da je jednostavan poligon monoton obzirom na pravac L ako je njegov presjek s bilo kojim pravcem okomitim na L prazan ili povezan skup.

Aloritam za monotone poligone

- Kažemo da je jednostavan poligon monoton obzirom na pravac L ako je njegov presjek s bilo kojim pravcem okomitim na L prazan ili povezan skup.
- Monotoni lanac je poligonski lanac za koji je istaknut pravac ZL tako da ako projiciramo vrhove na njega, poredak vrhova ostaje očuvan.





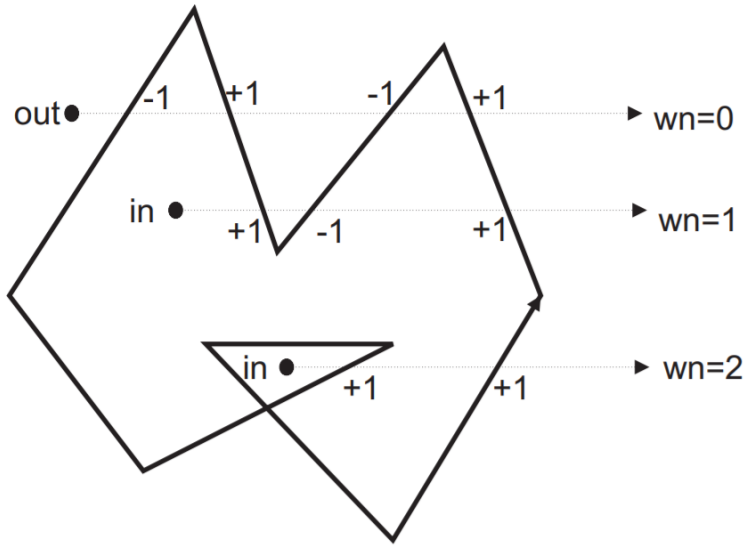
Aloritam za monotone poligone

- Kod poligona koji se “malo” razlikuju od konveksnih mogu se iskoristiti razmatranja kao kod konveksnih poligona.
- Poligon je monoton ako i samo ako se može razdvojiti u dva monotona lanca obzirom na jednu dimenziju.
- Algoritam za konveksne poligone složenosti O(log n) može biti upotrebljen i za monotone poligone obzirom na jednu dimenziju.

Algoritmi za složene poligone

- Algoritam namotajnog broja
  - Ispitivanje pripadnosti točke proizvoljnom (složenom) zatvorenom poligonu radi tako da računa koliko puta poligon okružuje točku. Točka je izvan jedino ako poligon ne okružuje točku, tj. kada je namotajni broj wn (engl. winding number) jednak nuli. Složenost je O(n).

- Grid algoritam



Grid algoritam

- Ideja je uložiti poligon u pravokutnik. Zatim se pravokutnik particionira u rešetku čije su sve ćelije iste dimenzije.
- Dobivene ćelije klasificiramo na:
  - potpuno unutra (cijela ćelija u interioru)
  - potpuno izvana (cijela ćelija u eksterioru)
  - neodređena (ćelija čiji je jedan dio u interioru, a drugi dio u eksterioru poligona)
- Za neodređene ćelije pamtimo listu bridova koji sijeku danu ćeliju i jedan (ili više) vrh za koji je napravljena klasifikacija, tj. koji je u interioru ili eksterioru.

Grid algoritam

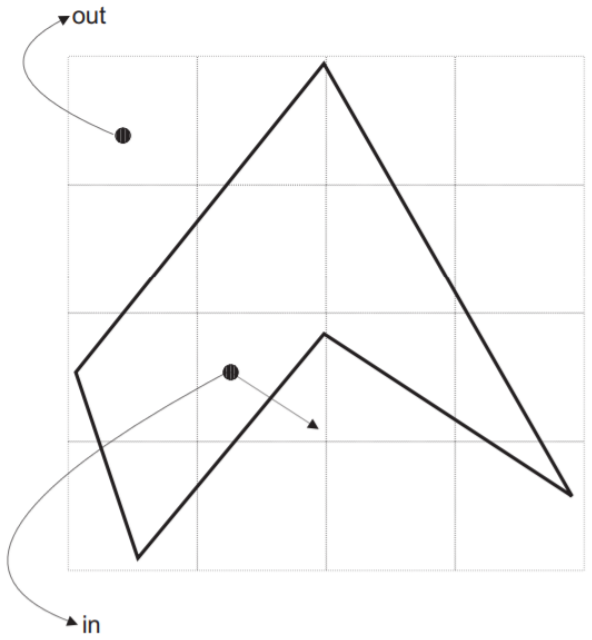
- Ispitivanje je li točka u poligonu svodi se na traženje ćelije kojoj ta točka pripada. Razlikujemo više slučajeva:
  - Ako je točka izvan pravokutnika koji sadrži poligon onda je ona očito u eksterioru poligona.
  - Ako je točka u ćeliji klasificiranoj kao potpuno izvana onda je ona u eksterioru.
  - Ako je točka u ćeliji klasificiranoj kao potpuno unutra onda je ona u interioru.
  - Ako je točka u neodređenoj ćeliji tada:

    Povučemo segment iz točke do određenog vrha ćelije (ili više njih). Kako je svaki vrh već klasificiran (unutra ili izvana), broj sjecišta segmenta s bridovima u listi nam određuje da li je točka u interioru ili eksterioru.

Grid algoritam

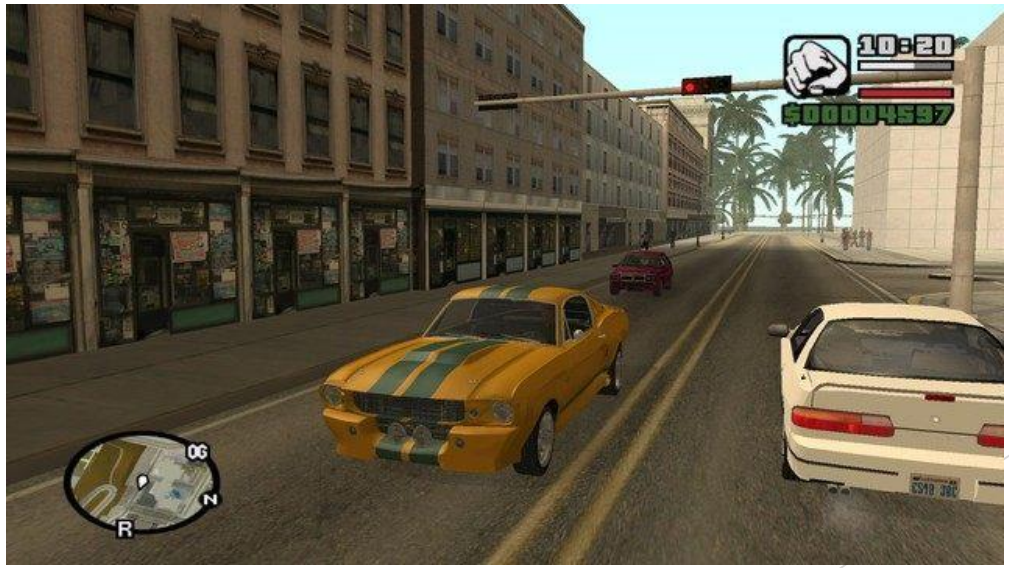
- Problem nastaje ako brid siječe vrh mreže (ili čak dok prolazi “vrlo” blizu vrha).
- Najjednostavnije rješenje je generirati novu mrežu s neznatno većom dimenzijom ćelija.

- Prostorna složenost i vrijeme inicijalizacije su veći nego kod prije navedenih algoritama.
- Za razborite oblike poligona on je dosta brži (bez inicijalizacije) od prije obrađenih algoritama.



Grid algoritam

- Prednosti:
  - Brzi odgovori na različite upite pripada li točka poligonu.
  - Korisno ako se poligon ne mijenja, odnosno više upita za isti poligon.
  - Funkcionira i ako imamo više poligona.
- Asocijacija:



Testiranje

- Usporedit ćemo:
  - Crossing number algoritam (CNA)
  - Algoritam za konveksne poligone (KA)
- Svaki poligon će biti zapravo pravilni mnogokut.
- Algoritmi su testirani na:
  - Aspire E15, preciznije – E5-572G-57Q1
  - CPU: Intel Core i5-4210M 2,6GHz with Turbo Boost up to 3,2GHz
  - RAM: 4GB DDR3 L
  - UBUNTU17.10

Testiranje

- Prvo testiranje – eksponencijalan rast broja vrhova poligona:
  - N=10 iteracija
  - Za svaku iteraciju broj vrhova = T*2^i
  - Broj iteracije je i (i<=N)
  - T je početni broj vrhova poligona
- Drugo testiranje – linearan rast broja vrhova:
  - N=10 iteracija
  - Za svaku iteraciju broj vrhova = T*I
  - Broj iteracije je i (i<=N)
  - T je početni broj vrhova poligona

Testiranje – eksponencijalan rast broja vrhova

- Prvi redak je N.
- Drugi redak je T.
- Treći X koordinata upitne točke.
- Zatim slijedi Y koordinata.
- Svaki slijedeći redak je i-ti odgovor je li točka unutar poligona.

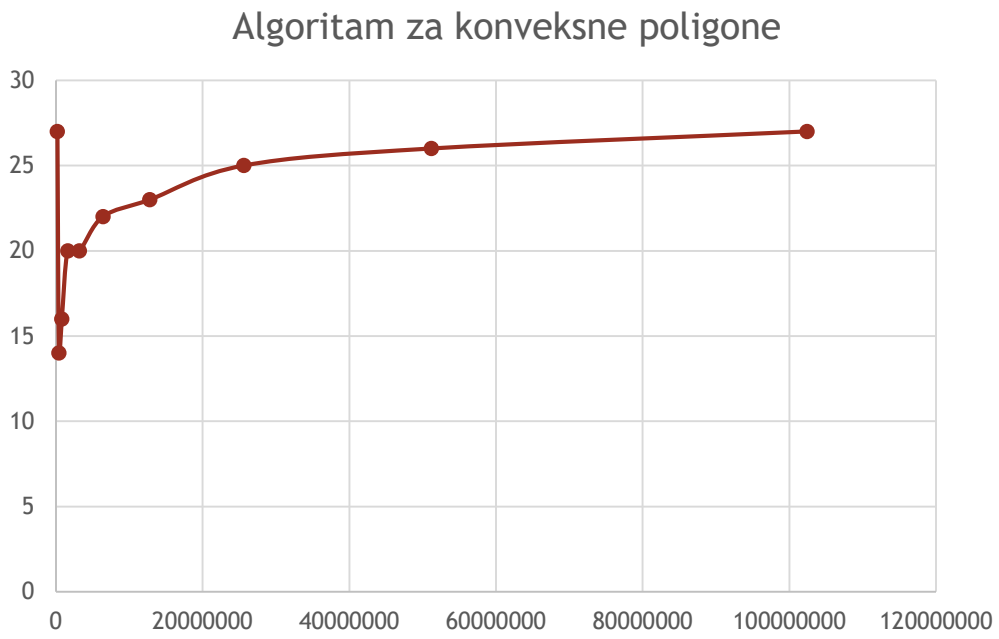
```
10
100000.0
1000000.0
1000000.0
0

0

0

1

1

1

1

1

1

1
```

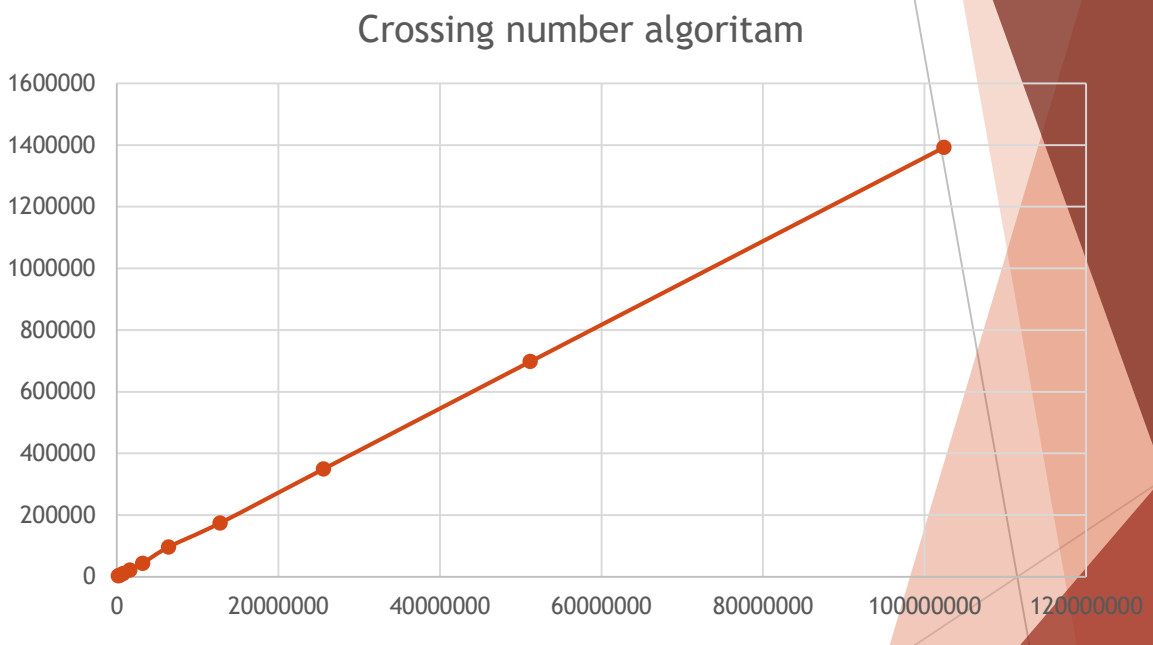
Testiranje – eksponencijalan rast broja vrhova

Algoritam za konveksne poligone		
N	Broj vrhova (T*2^N)	Vrijeme (ms)
1	200000	27
2	400000	14
3	800000	16
4	1.60E+06	20
5	3.20E+06	20
6	6.40E+06	22
7	1.28E+07	23
8	2.56E+07	25
9	5.12E+07	26
10	1.02E+08	27

Crossing number algoritam		
N	Broj vrhova (T*2^N)	Vrijeme (ms)
1	200000	2956
2	400000	5317
3	800000	10599
4	1.60E+06	21743
5	3.20E+06	43776
6	6.40E+06	95959
7	1.28E+07	174207
8	2.56E+07	349334
9	5.12E+07	697681
10	1.02E+08	1392241

Testiranje – eksponencijalan rast broja vrhova





Testiranje – linearan rast broja vrhova

- Prvi red: N
- Drugi: T
- Treći: X koordinata točke
- Četvrti: Y koordinata
- U idućih 10 redova se nalaze rezultati.

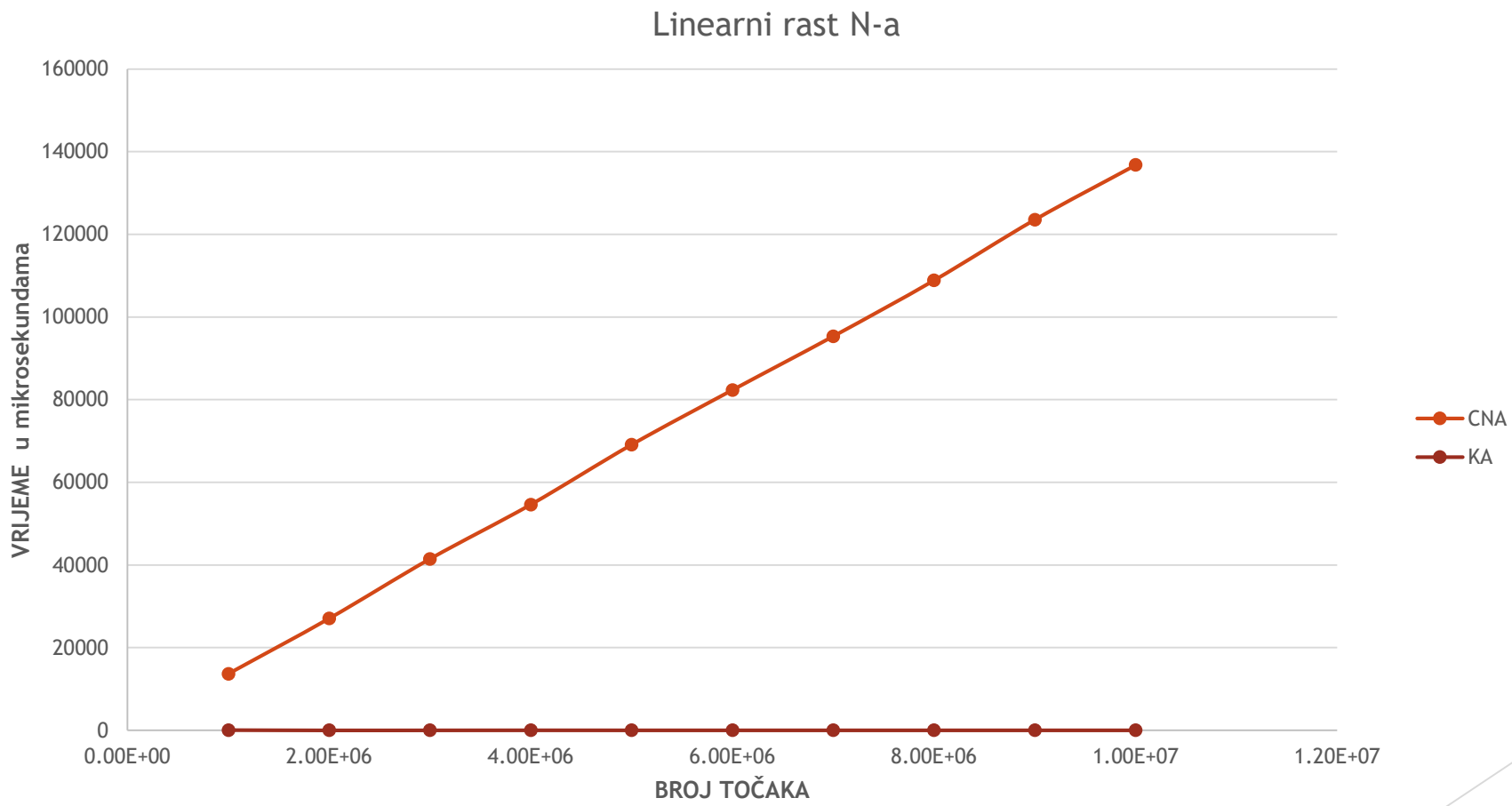
```
10
1000000.0
1000000.0
1000000.0
0

1

1

1

1

1

1

1

1

1
```

Testiranje – linearan rast broja vrhova

Algoritam za konveksne poligone		
N	Broj vrhova (N*T)	Vrijeme (ms)
1	1.00E+06	33
2	2.00E+06	19
3	3.00E+06	21
4	4.00E+06	24
5	5.00E+06	22
6	6.00E+06	22
7	7.00E+06	22
8	8.00E+06	22
9	9.00E+06	22
10	1.00E+07	22

Crossing number algoritam		
N	Broj vrhova (N*T)	Vrijeme (ms)
1	1.00E+06	13631
2	2.00E+06	27021
3	3.00E+06	41434
4	4.00E+06	54595
5	5.00E+06	69090
6	6.00E+06	82321
7	7.00E+06	95323
8	8.00E+06	108855
9	9.00E+06	123545
10	1.00E+07	136774

Testiranje – linearan rast broja vrhova



Literatura

- Ivan Peharda, Algoritmi za ispitivanje pripadnosti točke poligonu, diplomski rad, http://degiorgi.math.hr/oaa/oaa_lit/Peharda_dipl.pdf (Zagreb, ožujak 2006.)

- R. Seidel, A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons, https://www.cs.princeton.edu/courses/archive/fall05/cos528/handouts/A%20Simple%20and%20fast.pdf (Berkeley, ožujak 1991.)