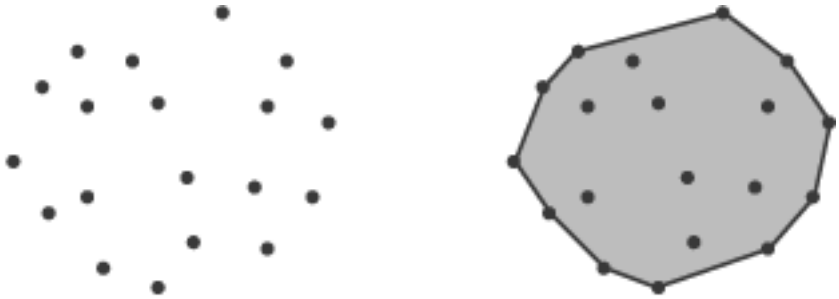
Др Драган Урошевић

КОНВЕКСНИ ОМОТАЧ

1. Увод

У овом тексту ће бити приказано неколико поступака за одређивање конвексног омотача. Конвексни омотач скупа тачака које припадају једној равни је конвексни полигон најмањег обима који садржи све тачке тог скупа. Када кажемо да садржи тачке, подразумевамо да се оне могу налазити у унутрашњости полигона, али и на полигоналној линији. Проблем одређивања конвексног омотача се може формулисати на следећи начин: *Дат је скуп тачака у равни. Одредити егов конвексни омотач.*

Показује се да је конвексни омотач истовремено пресек свих конвексних полигона који садрже тај скуп тачака. На слици 1 је приказан један скуп тачака и конвексни омотача за тај скуп тачака.



Сл. 1. Пример: један скуп тачака и конвексни омотач за тај скуп тачака

Важну улогу у одређивању конвексног омотача игра оријентација. Ако обилазимо темена конвексног полигона у смеру супротном од кретања казаљке на часовнику, онда се при проласку кроз било које теме мора скренути (мање или више) у леву страну (у произвољном полигону то није случај). Ако су темена полигона P_1, P_2, P_3, $\ldots$, P_n, онда за свако теме P_k важи да вектори $\overrightarrow{P_{k-1}P_k}$ и $\overrightarrow{P_kP_{k+1}}$ образују лево оријентисани систем (овде је за $k = 1$, $P_{k-1} = P_n$ и за $k = n$, $P_{k+1} = P_1$). За нас је врло битно да умемо да одредимо када два вектора образују лево оријентисани систем. Показује се да се то може једноставно утврдити. Тако вектори $\vec{a} = (x_a, y_a)$ и $\vec{b} = (x_b, y_b)$ образују лево оријентисани систем

ако је селедећа детерминанта

$$\begin{vmatrix} x_a & y_a \\ x_b & y_b \end{vmatrix}$$

позитивна.

У неким случајевима ћемо уместо провере да ли вектори $\overrightarrow{P_{k-1}P_k}$ и $\overrightarrow{P_kP_{k+1}}$ (што значи да се крај првог поклапа са почетком другог) образују лево оријентисан систем, проверавати да ли вектори $\overrightarrow{P_{k-1}P_k}$ и $\overrightarrow{P_{k-1}P_{k+1}}$ (што значи да се почеци поклапају) образују лево оријентисан систем. Ако последња два вектора образују десно оријентисан систем, теме P_k „нарушава конвексност“.

Најједноставнији, али уједно и најмање ефикасан начин за одређивање конвексног омотача јесте да за сваки пар тачака P_i и P_j утврдимо да ли оријентисана дуж $\overrightarrow{P_iP_j}$ припада конвексном омотачу. Да би оријентисана дуж $\overrightarrow{P_iP_j}$ припадала конвексном омотачу за сваку трећу тачку P_k, мора важити да вектори $\overrightarrow{P_iP_j}$ и $\overrightarrow{P_iP_k}$ образују лево оријентисани систем. Ако нека нека тачка P_k (различита од P_i и P_j) припада правој која пролази кроз P_i и P_j, онда свакако два поменута вектора не могу образовати лево оријентисани систем. Тада за тачку P_k мора важити да припада дужи чији су крајеви P_i и P_j, тј. мора се налазити између тачака P_i и P_j.

Према томе, поступак се састоји у утврђивању скупа оријентисаних дужи који образују конвексни омотач, да би се након тога дужи тог скупа поређале у редоследу појављивања у омотачу.

Ако у нашем скупу тачака има n тачака, онда постоји $\binom{n}{2} = \frac{n(n-1)}{2}$ парова (и двапут више оријентисаних дужи) за које треба утврдити да ли припадају омотачу. Провера да ли једна дуж припада омотачу захтева проверу да ли све преостале тачке (њих $n-2$) припадају једној полуравни одређеној правом која пролази кроз крајеве те дужи. Све скупа, то је $O(n^3)$ неких елементарних операција и то представља сложеност овог поступка.

Препуштамо читаоцу да сам испише програм за ову варијанту.

2. Грахамово скенирање

Други поступак (значајно ефикаснији) носи назив Грахамово сканирање. По овом поступку се:

- одређује најзападнија тачка скупа (тј. тачка са најмањом вредношћу x-координате); ако таквих тачака има више, онда се бира она која има најмању вредност y-координате; приметимо да је то тачка која се сигурно налази у конвексном омотачу; означимо ту тачку са P_1;
- све преостале тачке P_i се сортирају у неопадајућем поретку по углу које вектор чији је почетак у P_1, а крај у P_i заклапа са позитивним смером x-осе; тај угао се креће између $-\pi/2$ (и негативан је за тачке чија је y-координата мања од y-координате тачке P_1) и $\pi/2$ (позитиван је за тачке које се налазе „изнад“ P_1); нека је $P_2, P_3, \ldots, P_n$ редослед у сортираном низу;
- тачке P_1 и P_2 се постављају у конвексни омотач, а све преостале тачке се анализирају у редоследу појављивања у низу;

- претпоставимо да је последња анализирана тачка P_{i-1}, односно да је наредна за разматрање P_i; од анализираних тачака образован је део конвексног омотача и њега образују тачке P_{j_1}, P_{j_2}, ... , P_{j_m}; тачка P_i се додаје конвексном омотачу, али то може довести до нарушавања конвексности; да бисмо утврдили да ли се то догодило, посматрамо тројку тачака $P_{j_{m-1}}$, P_{j_m}, P_i; ако оне образују десно оријентисан систем, тачка P_{j_m} се избацује из конвексног омотача; међутим не мора бити само тачка P_{j_m} „проблематична“; то исто може важити и за $P_{j_{m-1}}$, $P_{j_{m-2}}$, ... ; стога се поступак провере мора продужити тако што се бирају две последње (након брисања) и тачка P_i, а прекида се у тренутку када остане само једна тачка (наравно, без P_i) или када посматрана тројка образује лево оријентисани систем.

Ево и комплетног програма за одређивање конвексног омотача применом Грахамовог скенирања:

```
program GrahS;
const
  MAXN = 10000;
type
  tacka = record
    x, y: real;
  end;
  nizt = array[1..MAXN] of tacka;
var
  fn: string;
  nt, np: integer;
  at, ap: nizt;
function vpro(a, b, c: tacka): real;
begin
  vpro := (b.x-a.x)*(c.y-a.y) - (b.y-a.y)*(c.x-a.x);
end;
procedure citaj(fn: string);
var
  f: text;
  kt: integer;
begin
  assign(f, fn); reset(f);
  read(f, nt);
  for kt := 1 to nt do read(f, at[kt].x, at[kt].y);
  close(f);
end;
procedure qsort(ks, ke: integer; var at: nizt);
var
  b: tacka;
  km, kt: integer;
begin
  if ks+1 = ke then begin
    if vpro(at[1], at[ks], at[ke]) < 0 then begin
      b := at[ks]; at[ks] := at[ke]; at[ke] := b;
    end
  end else begin
```

```
      km := ks;
      for kt := ks+1 to ke do
        if vpro(at[1], at[ks], at[kt]) < 0 then begin
          km := km+1;
          b := at[km]; at[km] := at[kt]; at[kt] := b;
        end;
      b := at[ks]; at[ks] := at[km]; at[km] := b;
      if ks < km-1 then qsort(ks, km-1, at);
      if km+1 < ke then qsort(km+1, ke, at);
    end;
end;
procedure sort(nt: integer; var at: nizt);
var
  b: tacka;
  kt, jt, it: integer;
begin
  kt := 1;
  for jt := 2 to nt do
    if (at[jt].x < at[kt].x) or
      ((at[jt].x = at[kt].x) and (at[jt].y < at[kt].y)) then kt := jt;
  b := at[kt]; at[kt] := at[1]; at[1] := b;
  qsort(2, nt, at);
end;
function konvom(nt: integer; at: nizt; var bt: nizt): integer;
var
  mt, kt: integer;
begin
  bt[1] := at[1];
  bt[2] := at[2];
  mt := 2;
  for kt := 3 to nt do begin
    while (mt > 1) and (vpro(bt[mt-1], bt[mt], at[kt]) <= 0) do mt := mt-1;
    mt := mt+1; bt[mt] := at[kt];
  end;
  konvom := mt;
end;
procedure stampaj(nt: integer; at: nizt);
var
  kt: integer;
begin
  for kt := 1 to nt do
    writeln(at[kt].x:8:3, ' ', at[kt].y:8:3);
end;
begin
  write('Naziv datoteke: '); read(fn);
  citaj(fn);
  sort(nt, at);
  np := konvom(nt, at, ap);
  stampaj(np, ap);
end.
```

3. Одређивање конвексног омотача спајањем горњег и доњег руба

Трећи поступак одређивања конвексног омотача се састоји у одређивању *доњег* и *горњег руба* конвексног омотача. Теме конвексног омотача са најмањом x-координатом и теме са највећом x-координатом деле конвексни омотач на два дела и они носе назив *доњи* и *горњи руб* полигона. На слици 2 је приказан један конвексни полигон и горњи и доњи руб тог полигона.

Да бисмо одредили та два руба, тачке се сортирају по вредности своје x-координате у растућем поретку. Наравно, ако има тачака са истом x-координатом, онда се у низ прво ставља тачка са најмањом вредношћу y-координате. Затим се ради одређивања горњег руба анализира једна по једна тачка из тог низа и издвајају тачке које ће чинити горњи руб. За почетак се у руб стављају прве две тачке из уређеног низа тачака. Претпоставимо да смо већ обрадили првих $k-1$ тачака P_1, P_2, ... , P_{k-1} и да тренутно горњи руб образују тачке P_{j_1}, P_{j_2}, ... , P_{j_m}.

Додавање тачке P_k може довести до избацивања одређеног броја тачака које се тренутно налазе у рубу. Да би последња тачка у тренутном рубу била избачена, морају бити испуњени следећи услови:

- број тачака у рубу мора бити бар 2 ($m \geqslant 2$);
- угао $\angle P_{j_m} P_{j_{m-1}} P_k$, тј. угао чије је теме у тачки $P_{j_{m-1}}$, први крак пролази кроз P_{j_m} док други крак пролази кроз P_k мора бити позитивно оријентисан, тј. оријентисан у смеру супротном од кретања казаљке на часовнику.

Ако су ова два услова испуњена, онда се тачка P_{j_m} избацује из руба, број m се смањује за 1 и комплетан поступак провере понавља за нови руб (тј. нову вредност m). У тренутку када не буду више испуњени услови за уклањање последње тачке са руба, тачка P_k се додаје на руб.

Ако се описани поступак изведе за све тачке, у редоследу у коме се оне налазе у сортираном низу, биће одређен горњи руб конвексног полигона.

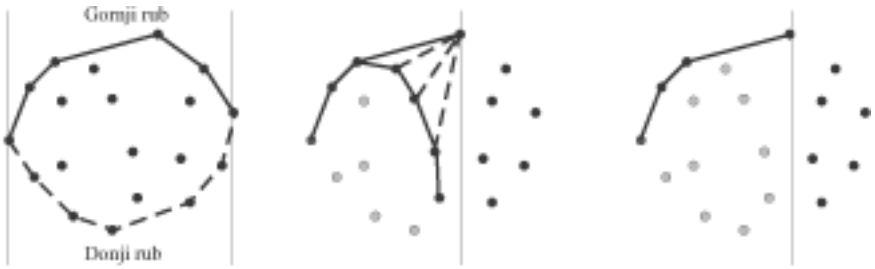
Сличан поступак треба поновити за одређивање доњег руба. Разлика је само у томе што се креће од краја сортираног низа тачака. Последње две тачке се додају рубу, а затим се анализирају једна за другом од $n-2$-ге тачке.

Да би био одређен конвексни омотач, треба спојити два формирана руба, при чему треба водити рачуна да се почетак и крај рубова поклапају и треба их само једном ставити у полигон.

На слици 2 је приказано како се мења изглед горњег руба при додавању нове тачке на крај руба.

Прикажимо и програм, уз напомену да функције које смо раније написали за прву варијанту, а које имају исти изглед нису комплетно исписане, већ је записано само заглавље.

```
program DGRub;
const
  MAXN = 10000;
type
  tacka = record
```



Сл. 2. Пример доњег и горњег руба и додавање једног врха у горњи руб

```
    x, y: real;
  end;
  nizt = array[1..MAXN] of tacka;
var
  fn: string;
  nt, np: integer;
  at, ap: nizt;
function vpro(a, b, c: tacka): real;
procedure citaj(fn: string);
procedure stampaj(nt: integer; at: nizt);
procedure qsort(ks, ke: integer; var at: nizt);
var
  b: tacka;
  km, kt: integer;
begin
  if ks+1 = ke then begin
    if (at[ks].x > at[ke].x) or
      ((at[ks].x = at[ke].x) and (at[ks].y > at[ke].y)) then begin
      b := at[ks]; at[ks] := at[ke]; at[ke] := b;
    end
  end else begin
    km := ks;
    for kt := ks+1 to ke do
      if (at[ks].x > at[kt].x) or
        ((at[ks].x = at[kt].x) and (at[ks].y > at[kt].y)) then begin
        km := km+1;
        b := at[km]; at[km] := at[kt]; at[kt] := b;
      end;
    b := at[ks]; at[ks] := at[km]; at[km] := b;
    if ks < km-1 then qsort(ks, km-1, at);
    if km+1 < ke then qsort(km+1, ke, at);
  end;
end;
procedure sort(nt: integer; var at: nizt);
var
  b: tacka;
  kt, jt, it: integer;
begin
  kt := 1;
  for jt := 2 to nt do
```

```
      if (at[jt].x < at[kt].x) or
           ((at[jt].x = at[kt].x) and (at[jt].y < at[kt].y)) then kt := jt;
    b := at[kt]; at[kt] := at[1]; at[1] := b;
    qsort(2, nt, at);
end;
function konvom(nt: integer; at: nizt; var bt: nizt): integer;
var
    mt, mt1, mt2, kt: integer;
begin
    bt[1] := at[1];
    bt[2] := at[2];
    mt1 := 2;
    for kt := 3 to nt do begin
      while (mt1 > 1) and (vpro(bt[mt1-1], bt[mt1], at[kt]) <= 0) do mt1 := mt1-1;
      mt1 := mt1+1; bt[mt1] := at[kt];
    end;
    bt[mt1+1] := at[nt-1];
    mt2 := mt1+1;
    for kt := nt-2 downto 1 do begin
      while (mt2 > mt1) and
           (vpro(bt[mt2-1], bt[mt2], at[kt]) <= 0) do mt2 := mt2-1;
      if kt > 1 then begin
        mt2 := mt2+1; bt[mt2] := at[kt];
      end;
    end;
    konvom := mt2;
end;
begin
    write('Naziv datoteke: '); read(fn);
    citaj(fn);
    sort(nt, at);
    np := konvom(nt, at, ap);
    stampaj(np, ap);
end.
```

4. Техника „подели па владај“ за одређивање конвексног омотача

Прикажимо технику одређивања конвексног омотача засновану на приступу подели па владај (енгл. *divide and conquer*).

Тачке треба сортирати у растућем поретку по вредности x-координате. Претпоставимо да треба одредити конвексни омотач тачака P_l, P_{l+1}, P_{l+2}, ... , P_{k-2}, P_{k-1}, P_k, тј. конвексни омотач од тачака из једног подскупа. Тада су могући следећи случајеви:

- $k-l \leqslant 2$; тада имамо највише три тачке и конвексни омотач чини свих $k-l+1$ тачака; битно је поређати тачке тако да се конвексни $k-l+1$-тоугао обилази у смеру супротном од кретања казаљке на часовнику (због остатка поступка);
- $k-l > 2$; тачке треба поделити у две групе; разлика бројева елемената у тим скуповима је највише један; прву групу чини првих $\left\lceil \dfrac{k-l+1}{2} \right\rceil$ елемената у

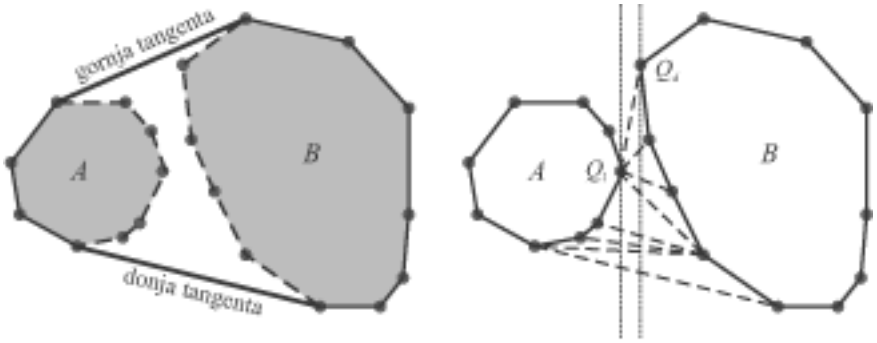
низу тачака, другу групу чине преостали елементи низа; сетимо се да је низ тачака сортиран по x-координатама; за те две половине се одређују конвексни омотачи, а затим се од тих конвексних омотача формира јединствен омотач за свих $k - l + 1$ тачака.

Опишимо како ћемо два формирана конвексна омотача спојити у један. Приметимо да се све тачке првог омотача налазе лево од тачака другог омотача (због чињенице да су скупови над којима су они формирани) исто тако распоређени.

Два омотача се спајају у један тако што се пронађу две заједничке тангенте (доња и горња) та два омотача, а затим формира полигон од тих тангенти и делова та два полигона од темена у којима тангенте додирују полигоне (погледајте слику испод).

Како одредити доњу заједничку тангенту? У левом полигону се одреди тачка са највећом вредношћу x-координате (нека је то теме Q_l, а у десном полигону теме са најмањом вредношћу x-координате (нека је то Q_d). Затим се проверава да ли је права која пролази кроз Q_l и Q_d доња тангета десног полигона. Ако није, Q_d се помера на наредно теме десног полигона (у смеру супротном од кретања казаљке на часовнику) и за то теме проверава да ли је тангента. Поступак се понавља све док не одредимо Q_d такво да је права кроз темена Q_l и Q_d доња тангента десног полигона (приметимо да је све време теме Q_l фиксирано). Када одредимо такво Q_d, проверавамо да ли је права кроз темена Q_l и Q_d доња тангента левог полигона. Ако није, померамо се по левом полигону на наредно теме али у смеру кретања казаљке на часовнику и за ново теме Q_l проверавамо исто. После одређеног броја померања одредићемо Q_l такво да је права која пролази кроз Q_l и Q_d доња тангента левог полигона. Међутим, та права више не мора бити доња тангента десног полигона па се мора поново кориговати Q_d. У сваком случају поступак се понавља све док права кроз темена Q_l и Q_d не постане тангента оба полигона. Сличним поступком се одређује и горња тангента.

На слици је визуелно приказан поступак одређивања доње тангенте за два конвексна омотача добијена рекурзивним поступком.



Сл. 3. Спајање два конвексна омотача и одређивање доње тангенте

При имплементацији се користи измењен поступак репрезентације конвексног омотача. Будући да се у поступку спајања два полигона, одређен број темена

избацује, најзгодније је полигон репрезентовати помоћу двоструко-повезне листе. У чворовима те листе су записана темена конвексног полигона.

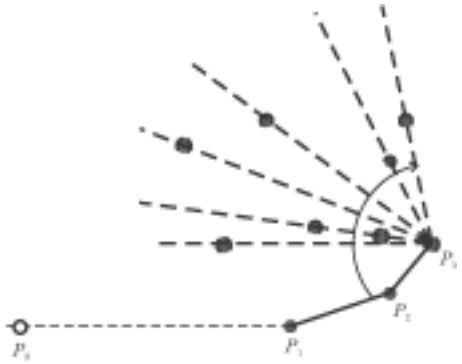
5. Џарвисов алгоритам

Ако поуздано знамо да конвексни омотач има релативно мало темена (у односу на укупан број тачака у скупу чији омотач одређујемо), онда је најефикаснији Џарвисов (*Jarvis*) алгоритам.

По овом поступку се за почетак одређује једно теме које сигурно припада полигону. То може бити теме са најмањом x-координатом или најмањом y-координатом. Узмимо да је то теме са најмањом y-координатом. У омотач се укључује то теме и једно „лажно“ теме које има исту y-координату и x-координату мању од било које друге тачке скупа чији конвексни омотач одређујемо. Затим се омотачу додаје једно по једно теме и то је она тачка из нашег скупа која има својство да је угао између последње странице тренутно оформљеног полигона и полуправе која спаја последње теме полигона са том тачком максималан. Да бисмо одредили ту тачку, морамо за сваку тачку скупа израчунати угао који заклапа полуправа кроз последње додато теме и ту тачку са последњом додатом страницом. Понављањем овог поступка у једном тренутку ће бити поново изабрано оно почетно теме за наредно теме полигона. То ће значити да је полигон затворен и да је поступак одређивања конвексног омотача окончан.

Ако конвексни омотач има релативно много темена, онда ће таквих пролаза кроз низ тачака бити више и поступак неће бити ефикасан.

На слици је приказан избор наредног темена омотача.



Сл. 4. Додавање следећег врха у конвексни омотач применом Џарвисовог поступка

Програм се може записати на следећи начин:

```
program JarvAlg;
const
  MAXN = 10000;
type
  tacka = record
    x, y: real;
```

```
  end;
  nizt = array[0..MAXN] of tacka;
var
  fn: string;
  nt, np: integer;
  at, ap: nizt;
procedure citaj(fn: string);
procedure stampaj(nt: integer; at: nizt);
function vcos(a, b, c: tacka): real;
var
  pr1, pr2: real;
begin
  pr1 := (a.x-b.x)*(c.x-b.x) + (a.y-b.y)*(c.y-b.y);
  pr2 := sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));
  pr2 := pr2 * sqrt(sqr(c.x-b.x)+sqr(c.y-b.y));
  if pr2 <> 0.0 then vcos := pr1/pr2 else vcos := 1;
end;
function konvom(nt: integer; at: nizt; var ap: nizt): integer;
var
  mx: real;
  j, k, m: integer;
  mcos, tcos: real;
begin
  m := 1; mx := at[1].x;
  for k := 2 to nt do begin
    if at[k].y < at[m].y then m := k;
    if at[k].x > mx then mx := at[k].x;
  end;
  ap[0].y := at[m].y; ap[0].x := mx-1;
  ap[1] := at[m];
  np := 1;
  repeat
    k := 1;
    mcos := vcos(ap[np-1], ap[np], at[1]);
    for j := 2 to nt do begin
      tcos := vcos(ap[np-1], ap[np], at[j]);
      if tcos < mcos then begin
        mcos := tcos; k := j;
      end;
    end;
    if k <> m then begin
      np := np+1;
      ap[np] := at[k];
    end;
  until k = m;
  konvom := np;
end;
begin
  write('Naziv datoteke: '); read(fn);
  citaj(fn);
  np := konvom(nt, at, ap);
  stampaj(np, ap);
end.
```