Математика / Геометрија

дискретна математика, едноставни и напредни геометриски алгоритми, примери

Бојан Костадинов [bojankostadinov@gmail.com]
Факултет за електротехника и информациски технологии, Скопје

Содржина

- **Математика:**
  - прости броеви
  - НЗД / НЗС
  - бројни системи
  - степенување

- **Геометрија:**
  - точки, прави, отсечки
  - прости алгоритми
  - посложени алгоритми

- **Многу примери**

1.1. Прости броеви

- Прост број е природен број кој има точно два (различни) природни броја за делители, тоа се 1 и самиот тој број.

```
bool isPrime(int n)
{
    for (int i=2; i<n; i++)
      if (n%i == 0) return false;

    return true;
}
```

1.1. Прости броеви (2)

- Потребно е да проверуваме деливост само со броеви помали или еднакви на sqrt(N). Доколку N е делив со број поголем од sqrt(N), тогаш резултатот од делењето е број помал од sqrt(N).

```
bool isPrime(int n)
{
  if (n <= 3) return true;
  if (n%2 == 0) return false;

  for (int i=3; i<=sqrt(n); i+=2)
    if (n%i == 0) return false;

  return true;
}
```

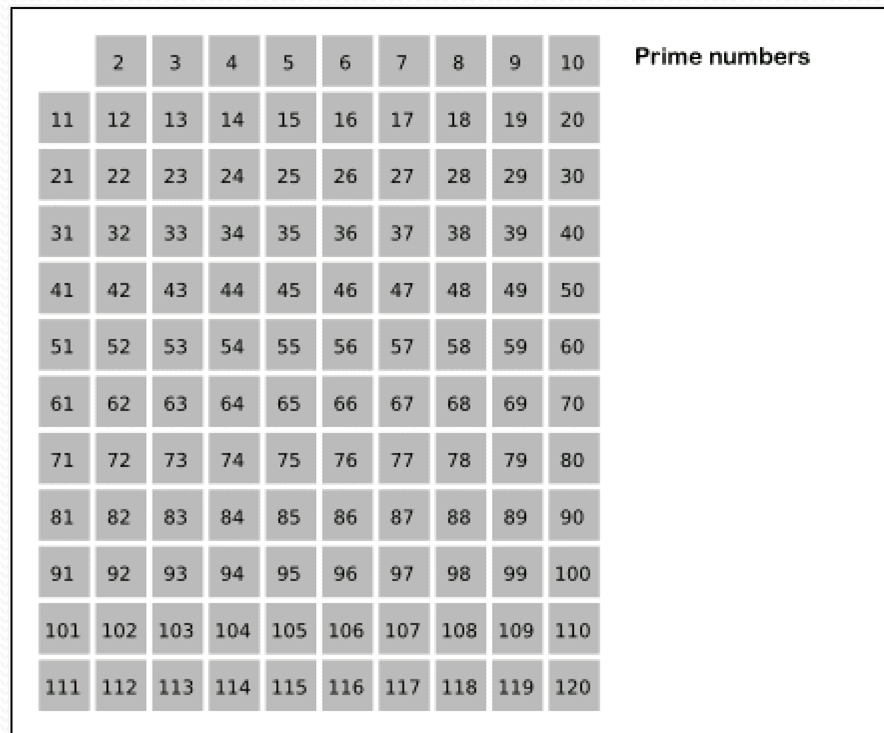
1.2. Ератостеново сито

- едноставен алгоритам за пронаоѓање на сите прости броеви до одреден број N [сито на Аткин].

```
vector<bool> sieve(int n)
{
    vector<bool> prime(n+1, true);

    for (int i=2; i<=sqrt(n); i++)
      if (prime[i])
         for (int k=i*i; k<=n; k+=i)
            prime[k] = false;

   return prime;
}
```



1.3. Најголем заеднички делител

- НЗД на два цели позитивни броја A и B е најголемиот цел број кој е делител и на A и на B.

```
int gcd(int a, int b)
{
    for (int i=min(a,b); i>=1; i--)
        if (a%i==0 && b%i==0) return i;
}

/* very slow, runs in linear time */
```

1.4. Евклидов алгоритам

- Едноставно се имплементира како рекурзивна функција.
- Се користи и за решавање на Диофантови равенки.

```
int gcd(int a, int b)
{
    if (b==0) return a;
    else return gcd(b, a%b);
}

/* very fast, runs in logarithmic time */
```

1.5. Најмал заеднички содржател

- се користи за собирање / одземање на дропки ?

```
int lcm(int a, int b)
{
    return b*(a/gcd(a,b));
}

fraction addFractions(fraction a, fraction b)
{
    int d = lcm(a.d, b.d);
    fraction result;
    result.n = a.n*(d/a.d) + b.n*(d/b.d);
    result.d = d;
    return result;
}
```

1.6. Претворување во декаден систем

- $1011_2 = 1 + 1 * 2 + 0 * 2 * 2 + 1 * 2 * 2 * 2 = 1 + 2 + 8 = 11_{10}$

```
int toDecimal(int n, int b)
{
    int result = 0, mult = 1;

    while (n > 0)
    {
        result += (n % 10) * mult;
        mult *= b;
        n = n / 10;
    }

    return result;
}
```

1.7. Претворување од декаден систем

- Во секој чекор, го делиме бројот со b, и го паметиме остатокот.

```
int fromDecimal(int n, int b)
{
	int result = 0, mult = 1;

	while (n > 0)
	{
		result += (n % b) * mult;
		mult *= 10;
		n = n / b;
	}

	return result;
}
```

```
11 / 2 = 5 + остаток 1
5  / 2 = 2 + остаток 1
2  / 2 = 1 + остаток 0
1  / 2 = 0 + остаток 1
```

```
43 / 2 = 21 + остаток 1
21 / 2 = 10 + остаток 1
10 / 2 = 5  + остаток 0
5  / 2 = 2  + остаток 1
2  / 2 = 1  + остаток 0
1  / 2 = 0  + остаток 1
```

1.8. Претворување од декаден систем (2)

- Доколку b > 10, користиме не-нумерички карактери

```
string fromDecimal(int n, int b)
{
	string result = "";
	string chars = "0123456789ABCDEFGHIJ";

	while (n > 0)
	{
		result = chars[n%b] + result;
		n = n / b;
	}

	return result;
}
```

```
109 / 16 = 6 + остаток 13
6   / 16 = 0 + остаток 6
```

пример. $109_{10} = 6D_{16}$

1.9. Степенување

- 3^37 = 3 * 3 * 3 * 3 * 3 * 3 * 3 * 3 * 3 .... * 3 * 3 (37 пати)

```
int pow(int base, int power)
{
    int result = 1;

    for (int i=1; i<=power; i++)
        result *= base;

    return result;
}
```

многу бавно!

1.10. Степенување (2)

- 3^37 = 3 * 3 * 3 * 3 * 3 * 3 * 3 * 3 * 3 .... * 3 * 3 (37 пати)

```
int pow(int base, int power)
{
    if (power == 0) return 1;
    else if (power % 2 == 0)
        return sqr(pow(base, power/2));
    else
        return base*(pow(base, power-1));
}
```

$$
\begin{aligned}
3 &= 3 \\
3^2 &= 3 * 3 \\
3^4 &= 3^2 * 3^2 \\
3^8 &= 3^4 * 3^4 \\
3^{16} &= 3^8 * 3^8 \\
3^{32} &= 3^{16} * 3^{16}
\end{aligned}
$$

$$3^{37} = 3^{32} * 3^4 * 3^1$$

2.1. Точки, прави, отсечки

- репрезентација на основни геометриски објекти

```
struct point
{
        int x; //double x;
        int y; //double y;
};

struct line
{
        point p1, p2;
};
```

type point = record **x**, *y*: integer **end;**
line = **record** *p1*, *p2*: point **end;**

```
……
point p1, p2;
p1.x = p1.y = 0;
p2.x = p2.y = 1;

line segment1;
segment1.p1 = p1;
segment1.p2 = p2;
```

2.2. Основни алгоритми

- signed_triangle_area() ја враќа плоштината на триаголникот ABC, со позитивна вредност доколку точките ABC се CCW ориентирани, или со негативна вредност доколку точките ABC се CW ориентирани!!!

```
double signed_triangle_area(point a, point b, point c)
{
    return ( (a.x*b.y - a.y*b.x + a.y*c.x
              - a.x*c.y + b.x*c.y - c.x*b.y) / 2.0);
}
```

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

```
bool ccw(point a, point b, point c)
{
    return (signed_triangle_area(a, b, c) > 0);
}
```

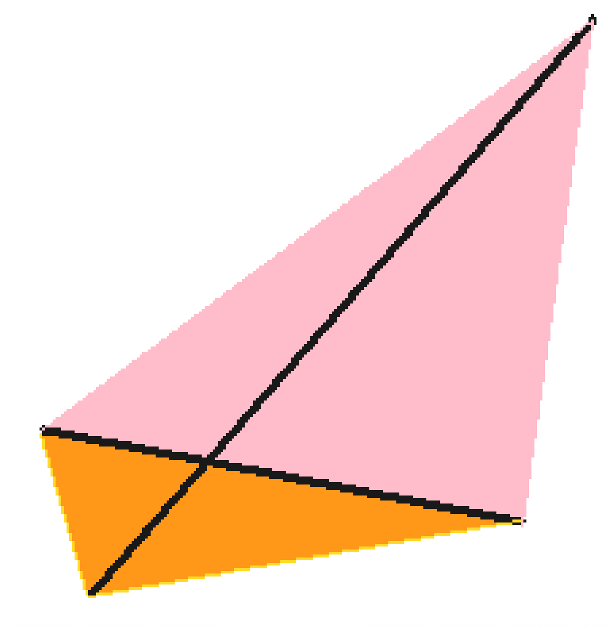
$$\frac{y - y_1}{x - x_1} - \frac{y_2 - y_1}{x_2 - x_1} = 0$$

2.3. Дали две отсечки се сечат?

- две отсечки се сечат доколку краевите на едната отсечка се наоѓаат на различни страни од линијата определена со краевите на другата отсечка и обратно, краевите на другата отсечка...

```
bool intersect(line l1, line l2)
{
    double l1p1 = signed_triangle_area(l1.p1, l1.p2, l2.p1);
    double l1p2 = signed_triangle_area(l1.p1, l1.p2, l2.p2);
    double l2p1 = signed_triangle_area(l2.p1, l2.p2, l1.p1);
    double l2p2 = signed_triangle_area(l2.p1, l2.p2, l1.p2);

    return ((l1p1*l1p2 <= 0) && (l2p1*l2p2 <= 0));
}
```



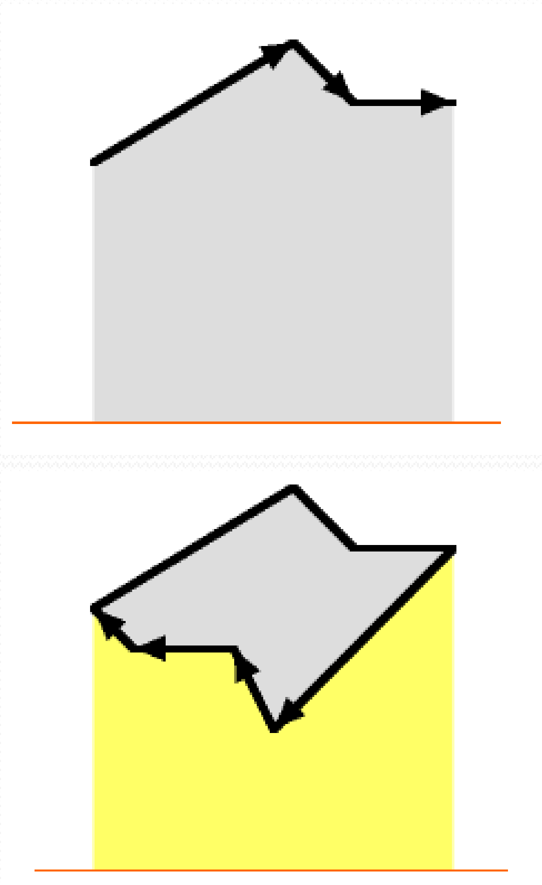
2.4. Плоштина на полигон

- функционира за секаков полигон (не само за конвексен)

```
double signed_area(vector<point> p)
{
        double total = 0.0;

        for (int i=0; i<p.size(); i++)
        {
                int j = (i+1) % p.size();
                total += (p[i].x*p[j].y) - (p[j].x*p[i].y);
        }

      return (total / 2.0);
}
```



2.4. Плоштина на полигон

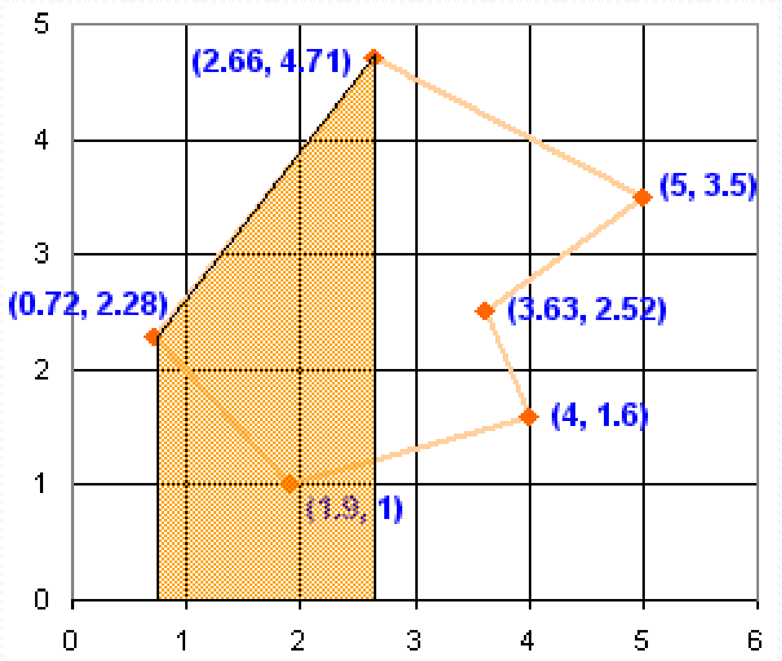
- функционира за секаков полигон (не само за конвексен)

```
double signed_area(vector<point> p)
{
        double total = 0.0;

        for (int i=0; i<p.size(); i++)
        {
                int j = (i+1) % p.size();
                total += (p[i].x*p[j].y) - (p[j].x*p[i].y);
        }

      return (total / 2.0);
}
```

$$P_{вк.} = P_{триа.} + P_{прав.}$$



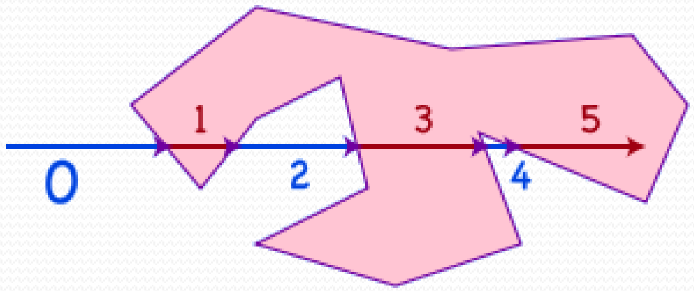
2.5. Точка во / надвор од полигон ?

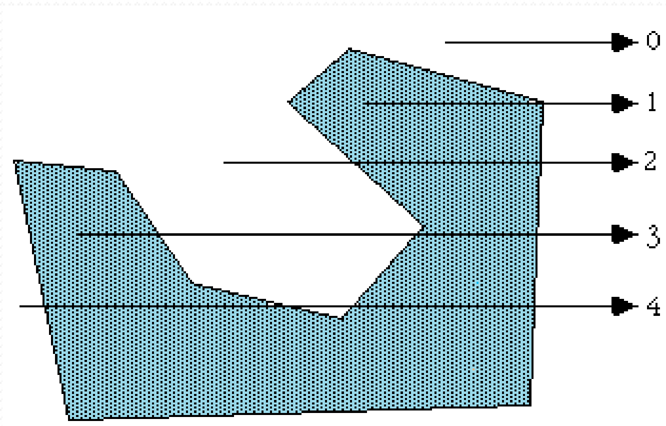
- Ray Casting Algorithm (Algorithms by Robert Sedgewick)

```
bool insidePolygon(vector<point> poly, point x)
{
    line tl; tl.p1=tl.p2=x; tl.p2.x=1e20; tl.p2.y++;
    int intersects = 0;

    for(int i=0; i<poly.size(); i++)
    {
        int j = (i+1) % poly.size();
        line cl; cl.p1 = poly[i]; cl.p2 = poly[j];
        //pl should be point-line (for boundary check)
        //if(intersect(cl, pl)) return true;   //boundary
        if(intersect(cl,tl)) intersects++;  //intersect
    }

    if(intersects%2 == 0) return false; else return true;
}
```





2.6. Convex Hull

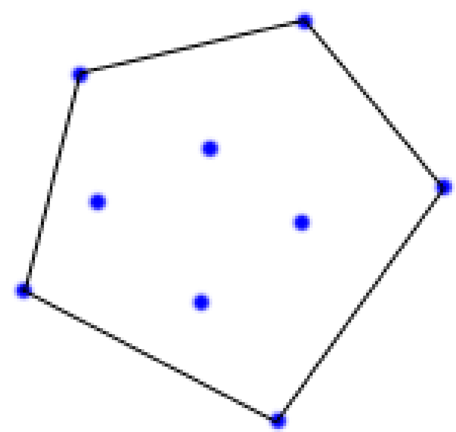
- Convex hull за множество од точки е најмалиот конвексен полигон кој ги содржи сите точки. Овој полигон секогаш е дефиниран од дадените точки во почетното (оригиналното) множество од точки [влезот].

- Алгоритми:

  Package Wraping [$O(N^2)$]

  Graham Scan [$O(N * \log N)$]

  Quick Hull [$O(N * \log N)$]

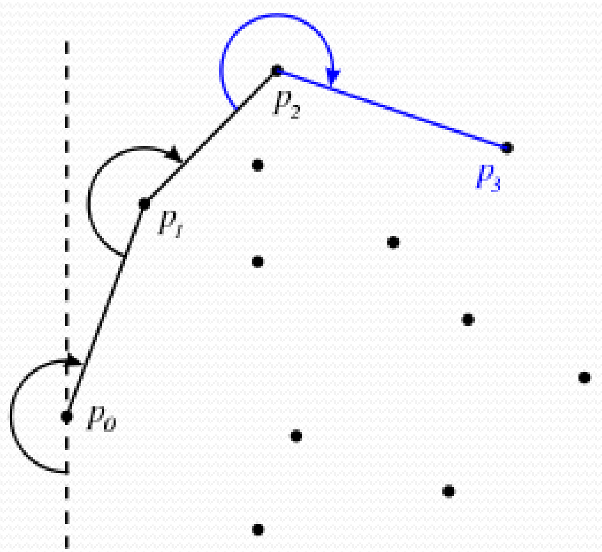


- Quick Hull најчесто не се користи како алгоритам за наоѓање на финалниот convex hull, туку за елиминирање на некои од почетните точки.

2.7. Package Wraping

- најбавен, но наједноставен алгоритам за наоѓање на convex hull

```
i = 0
p[0] = leftmost point of P
do
      p[i+1] = point such that all other points
                        in P are to the right
                        of the line p[i]p[i+1]
      i = i + 1
 while p[i] != p[0]
 return p
```

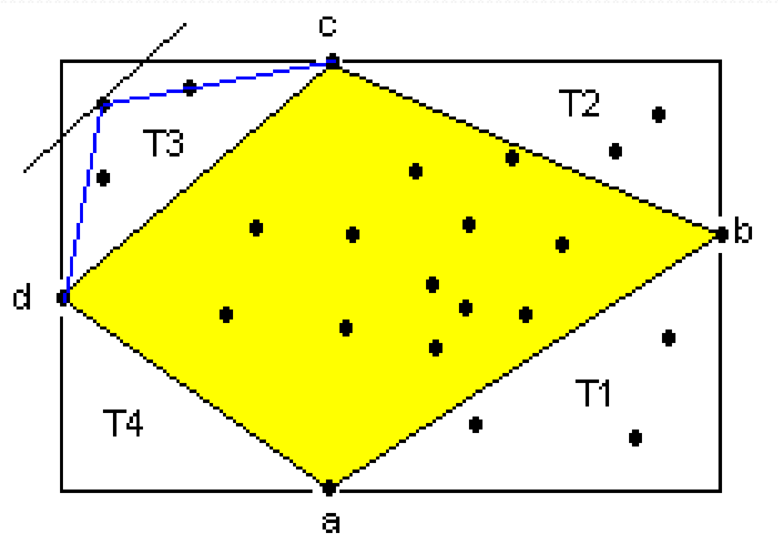


2.8. Package Wraping (2)

```
function  wrap:  integer;
  var i, min, M: integer;
      minangle,  v:  real;
      t: point;
  begin
  min:=1;
  for  i:=2  to  Ndo
     if  p[i].y<p[min].y  then  min:=i;
  M:=0;  p[N+1]:=p[min];  minangle:=0.0;
  repeat
     M:=M+1;  t:=p[M];  p[M]:=p[min];  p[min]:=t;
     min:=N+1;  v:=minangle;  minangle:=360.0;
     for  i:=M+1  to  N+1  do
        if theta(p[M], p[i])>v  then
           if theta(p[M], p[i])<minangle then
              begin  min:=i;  minangle:=theta(p[M], p[min])  end;
  until  min= N+1;
  wrap:=M;
  end ;
```

2.9. Quick Hull

- едноставно и брзо се елиминираат голем број од почетните точки
- Алгоритам:
  1. се одбираат 4 (3) точки (обично min_x, min_y, max_x, max_y)
  2. сите точки кои се наоѓаат во четириаголникот се елиминираат, бидејќи тие сигурно нема да се наоѓаат во конечниот convex hull



- Алгоритамот работи во линеарно време, па во никој случај не влијае на конечната комплексност на решението на проблемот.

П Р И М Е Р И

Пример 1

[TopCoder PointInPolygon] Дадена е точка и темиња на едноставен полигон. Сите страни од полигонот се или хоризонтални или вертикални. Отпечатете дали точката е во внатрешноста на полигонот [INTERIOR], дали лежи на границите на полигонот [BOUNDARY] или е надвор од полигонот [EXTERIOR].

Влез:

```
4
0 0
0 10
10 10
10 0
5 5
```

Излез:

```
INTERIOR
```

Пример 2

[Ниш - Trougao] Дадени се N (3 <= N <= 60) различни точки. Определете триаголник чии темиња се дел од N-те точки дадени во влезот и во чија внатрешност се наоѓаат најмногу точки (во резултатот се бројат и темињата на триаголникот).

Влез:

```
4
0 0
3 0
2 1
2 2
```

Излез:

```
4
```

Пример 3

[Ниш - Prave] Дадени се N (2 <= N <= 100) различни точки. Треба да се најде и отпечати (на стандарден излез) максималниот број на точки кои лежат на една права.

```
Влез:
4
1 1
2 2
2 1
3 1

Излез:
3
```

```
Влез:
4
1 1
2 2
-1 -1
3 3

Излез:
4
```

Пример 4

[Ниш - НЗД] Дадени се N ($2 <= N <= 10^5$) различни природни броеви помали од 10^9. Определете го најголемиот заеднички делител (НЗД) и најмалиот заеднички содржател (НЗС) за сите N броеви.

```
Влез:
5
12
15
12
6
21

Излез:
3 420
```

```
Влез:
5
3
5
7
11
13

Излез:
1 15015
```

Решение 4

```
int n, c; cin >> n;
int nzd, nzs;
cin >> nzd; nzs = nzd;

for (int i=1; i<n; i++)
{
    cin >> c;
    nzd = gcd(nzd, c);
    nzs = lcm(nzs, c);
}

cout << nzd << " " << nzs << endl;
```

Пример 5

[Ниш - НЗД] Дадени се три природни броја A, N и M ($1 <= A, N <= 10^9$, $2 <= M <= 10^4$). Пресметајте и отпечатете го (на стандарден излез) A^N по модул M.

```
Влез:
A=3 N=5 M=100

Излез:
43
```

```
Влез:
A=2 N=10 M=1001

Излез:
23
```

(A + B) mod M = ((A mod M) + (B mod M)) mod M
(A * B) mod M = ((A mod M) * (B mod M)) mod M

Решение 5

```
int sqr(int a)
{
    return a*a;
}

int pow(int base, int power, int modul)
{
    base = base % modul;
    if (power == 0) return 1;
    else if (power % 2 == 0)
        return sqr(pow(base, power/2, modul)) % modul;
    else
        return base*(pow(base, power-1, modul)) % modul;
}
```

Пример 6

[Ниш - Poligon] Даден е полигон составен од N ($2 <= N <= 10^5$) различни точки (точките се дадени во правец на стрелките на часовникот). Определете и отпечатете дали е полигонот конвексен.

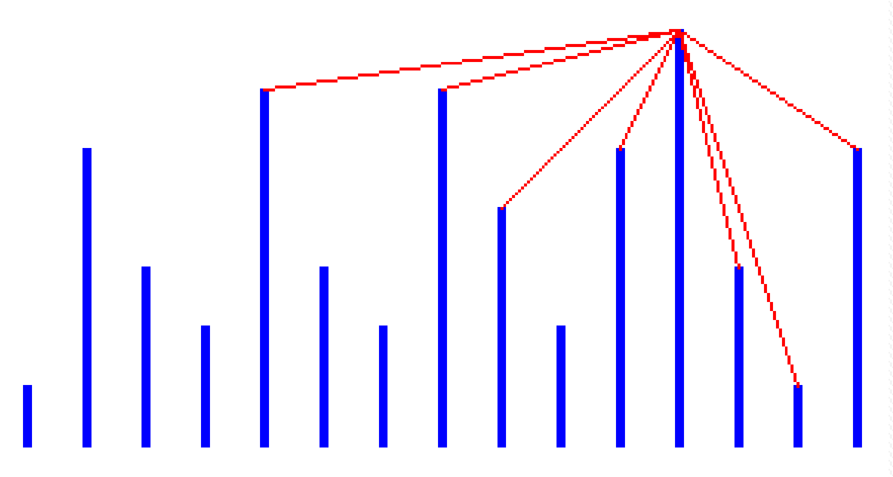
Влез:	Влез:
4	4
0 0	0 0
0 5	0 5
5 5	2 2
5 0	4 2
Излез:	Излез:
DA	NE

Пример 7

[TopCoder - BestView] Дадени се висините на N (1 <= N <= 50) облакодери, наредени во ред. i-тиот облакодер може геометриски да се претстави како отсечка со крајни точки (i,0) и (i, height[i]). Отпечатете го максималниот број на облакодери кои можат да се видат од покривот на еден облакодер.

```
Влез:
15
1 5 3 2 6 3 2 6
4 2 5 7 3 1 5

Излез:
7
```

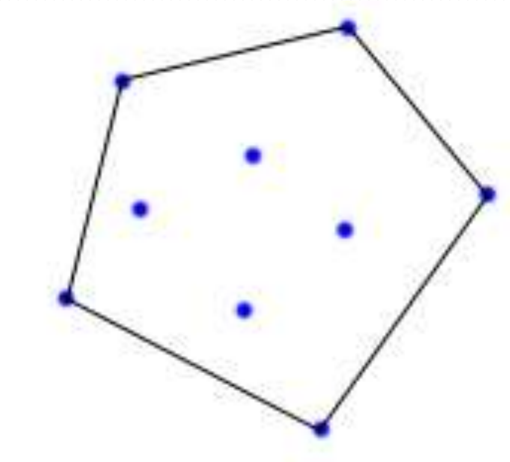


Пример 8

[USACO - Cow Herding] Трпе сака да изгради ограда за да ги заштити кравите од локалните студенти кои ги буцкаат додека спијат. Секоја крава си има омилена позиција за спиење, и Трпе сака да ги огради сите овие места со одредена конвексна фигура, бидејќи така ќе им овозможи на кравите полесно да доаѓаат до својата позиција. Пресметајте и отпечатете ја плоштината на најмалата област која ги вклучува сите омилени позиции за спиење на кравите.

```
Влез:
3
0 0
2 0
1 1
```

```
Излез:
1.000
```



Пример 9

[Sofia 08 - Polygon] Полигон со страни паралелни со координатните оски е претставен само преку своите N (2 <= N <= 1000) хоризонтални страни. Напишете програма која ќе ја пресмета и отпечати (на стандарден излез) плоштината на дадениот полигон.

```
Влез:
3
0 10 10
10 20 0
0 20 20

Излез:
300
```