UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

acm International Collegiate Programming Contest

Team reference

Challenge Accepted !

Sumário

Tabelas, fórmulas e teoremas

2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^10	1.024
2^11	2.048
2^12	4.096
2^13	8.192
2^14	16.384
2^15	32.768
2^16	65.536
2^17	131.072
2^18	262.144
2^19	524.288
2^20	1.048.576

3^2	9
3^3	27
3^4	81
3^5	243
3^6	729
3^7	2.187
3^8	6.561
3^9	19.683
3^10	59.049
3^11	177.147
3^12	531.441
3^13	1.594.323
3^14	4.782.969
3^15	14.348.907
3^16	43.046.721
3^17	129.140.163
3^18	387.420.489
3^19	1.162.261.467
3^20	3.486.784.401

2!	2
3!	6
4!	24
5!	120
6!	720
7!	5.040
8!	40.320
9!	362.880
10!	3.628.800
11!	39.916.800
12!	[limite int] 479.001.600
13!	6.227.020.800
14!	87.178.291.200
15!	1.307.674.368.000
16!	20.922.789.888.000
17!	355.687.428.096.000
18!	6.402.373.705.728.000
19!	121.645.100.408.832.000
20!	2.432.902.008.176.640.000 [limite unsigned long long]

Catalan numbers. Catalan numbers are defined by the recurrence:

$$C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i}$$

A closed formula for Catalan numbers is:

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n+1}$$

Stirling numbers of the first kind. These are the number of permutations of I_n with exactly k disjoint cycles. They obey the recurrence:

$$\begin{bmatrix} n \\ k \end{bmatrix} = (n-1)\begin{bmatrix} n-1 \\ k \end{bmatrix} + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$$

Stirling numbers of the second kind. These are the number of ways to partition I_n into exactly k sets. They obey the recurrence:

$$\begin{Bmatrix} n \\ k \end{Bmatrix} = k\begin{Bmatrix} n-1 \\ k \end{Bmatrix} + \begin{Bmatrix} n-1 \\ k-1 \end{Bmatrix}$$

Rational Expansion Theorem for Distinct Roots.

If $R(z) = P(z)/Q(z)$, *where* $Q(z) = q_0(1-\rho_1 z)\dots(1-\rho_l z)$ *and the numbers* $(\rho_1, \dots, \rho_l)$ *are distinct, and if* $P(z)$ *is a polynomial of degree less than* l, *then*

$$[z^n]\,R(z) = a_1\rho_1^n + \cdots + a_l\rho_l^n, \qquad \text{where } a_k = \frac{-\rho_k P(1/\rho_k)}{Q'(1/\rho_k)}. \quad (7.29)$$

Proof: Let $a_1, \dots, a_l$ be the stated constants. Formula (7.29) holds if $R(z) = P(z)/Q(z)$ is equal to

$$S(z) = \frac{a_1}{1-\rho_1 z} + \cdots + \frac{a_l}{1-\rho_l z}.$$
$$Q(z) = q_0 + q_1 z + \cdots + q_m z^m, \qquad \text{where } q_0 \neq 0 \text{ and } q_m \neq 0.$$

The "reflected" polynomial

$$Q^R(z) = q_0 z^m + q_1 z^{m-1} + \cdots + q_m$$

has an important relation to $Q(z)$:

$$Q^R(z) = q_0(z-\rho_1)\dots(z-\rho_m)$$
$$\iff \quad Q(z) = q_0(1-\rho_1 z)\dots(1-\rho_m z).$$

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$
$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$
$$\binom{n}{k} = \frac{n}{n-k}\binom{n-1}{k}$$
$$\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$$
$$\binom{n+1}{k} = \frac{n+1}{n-k+1}\binom{n}{k}$$
$$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

$$x^m = \sum_{k=0}^{m} \begin{Bmatrix} m \\ k \end{Bmatrix} x^{\underline{k}},$$

$\sum_{k=0}^{n} k = n(n+1)/2$
$\sum_{k=0}^{n} k^2 = n(n+1)(2n+1)/6$
$\sum_{k=0}^{n} k^4 = (6n^5+15n^4+10n^3-n)/30$
$\sum_{k=0}^{n} x^k = (x^{n+1}-1)/(x-1)$

$\sum_{k=a}^{b} k = (a+b)(b-a+1)/2$
$\sum_{k=0}^{n} k^3 = n^2(n+1)^2/4$
$\sum_{k=0}^{n} k^5 = (2n^6+6n^5+5n^4-n^2)/12$
$\sum_{k=0}^{n} kx^k = (x-(n+1)x^{n+1}+nx^{n+2})/(x-1)^2$

$$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$$
$$\sum_{k=1}^{n} k^2\binom{n}{k} = (n+n^2)2^{n-2}$$
$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$
$$\binom{m+n}{r} = \sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k}$$
$$\binom{n}{k} = \prod_{i=1}^{k}\frac{n-k+i}{i}$$

Table 55 What's the difference?

$f = \Sigma g$	$\Delta f = g$	$f = \Sigma g$	$\Delta f = g$
$x^{\underline{0}} = 1$	0	2^x	2^x
$x^{\underline{1}} = x$	1	c^x	$(c-1)c^x$
$x^{\underline{2}} = x(x-1)$	$2x$	$c^x/(c-1)$	c^x
$x^{\underline{m}}$	$mx^{\underline{m-1}}$	cf	$c\Delta f$
$x^{\underline{m+1}}/(m+1)$	$x^{\underline{m}}$	$f+g$	$\Delta f+\Delta g$
H_x	$x^{\underline{-1}} = 1/(x+1)$	fg	$f\Delta g + Eg\Delta f$

$f(x)$	$\Delta f(x)$	$\sum f(x)\,\delta x$
$x^{\underline{m}}$	$mx^{\underline{m-1}}$	$\frac{x^{\underline{m+1}}}{m+1}$
$x^{\underline{-1}}$	$-x^{\underline{-2}}$	H_x
2^x	2^x	2^x
c^x	$(c-1)c^x$	$\frac{c^x}{c-1}$
$\binom{x}{m}$	$\binom{x}{m-1}$	$\binom{x}{m+1}$
$u(x)+v(x)$	$\Delta u(x)+\Delta v(x)$	$\sum u(x)\,\delta x + \sum v(x)\,\delta x$
$u(x)v(x)$	$u(x)\Delta v(x)+v(x+1)\Delta u(x)$	
$u(x)\Delta v(x)$		$u(x)v(x) - \sum v(x+1)\Delta u(x)\,\delta x$

$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$
$\sin(\alpha-\beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta$
$\tan(\alpha+\beta) = \frac{\tan\alpha+\tan\beta}{1-\tan\alpha\tan\beta}$
$\cos^2\alpha = \frac{1}{2}(1+\cos 2\alpha)$
$\sin\alpha+\sin\beta = 2\sin\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2}$
$\sin\alpha-\sin\beta = 2\sin\frac{\alpha-\beta}{2}\cos\frac{\alpha+\beta}{2}$
$\tan\alpha+\tan\beta = \frac{\sin(\alpha+\beta)}{\cos\alpha\cos\beta}$
$\sin\alpha\sin\beta = \frac{1}{2}[\cos(\alpha-\beta)-\cos(\alpha+\beta)]$
$\sin\alpha\cos\beta = \frac{1}{2}[\sin(\alpha+\beta)+\sin(\alpha-beta)]$
Law of sines: $\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = 2R_{out}$.
Law of cosines: $c^2 = a^2+b^2-2ab\cos C$.
Law of tangents: $\frac{a+b}{a-b} = \frac{\tan[\frac{1}{2}(A+B)]}{\tan[\frac{1}{2}(A-B)]}$

$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$
$\cos(\alpha-\beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta$
$\sin 2\alpha = 2\sin\alpha\cos\alpha$, $\cos 2\alpha = \cos^2\alpha - \sin^2\alpha$
$\sin^2\alpha = \frac{1}{2}(1-\cos 2\alpha)$
$\cos\alpha+\cos\beta = 2\cos\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2}$
$\cos\alpha-\cos\beta = -2\sin\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2}$
$\cot\alpha+\cot\beta = \frac{\sin(\alpha+\beta)}{\sin\alpha\sin\beta}$
$\cos\alpha\cos\beta = \frac{1}{2}[\cos(\alpha-\beta)+\cos(\alpha+\beta)]$
$\sin' x = \cos x$, $\cos' x = -\sin x$
Inscribed/outscribed circles: $R_{out} = \frac{abc}{4S}$, $R_{in} = \frac{2S}{a+b+c}$
Heron: $\sqrt{s(s-a)(s-b)(s-c)}$, $s = \frac{a+b+c}{2}$.
Δ's area, given side and adjacent angles: $\frac{c^2}{2(\cot\alpha+\cot\beta)}$

Identities:

$$\sin x = \frac{1}{\csc x}, \qquad \cos x = \frac{1}{\sec x}, \qquad \sin 2x = 2\sin x\cos x,$$
$$\tan x = \frac{1}{\cot x}, \qquad \sin^2 x + \cos^2 x = 1, \qquad \cos 2x = \cos^2 x - \sin^2 x,$$
$$1+\tan^2 x = \sec^2 x, \qquad 1+\cot^2 x = \csc^2 x, \qquad \cos 2x = 1 - 2\sin^2 x,$$
$$\sin x = \cos\left(\tfrac{\pi}{2}-x\right), \qquad \sin x = \sin(\pi - x), \qquad \tan 2x = \frac{2\tan x}{1-\tan^2 x},$$
$$\cos x = -\cos(\pi - x), \qquad \tan x = \cot\left(\tfrac{\pi}{2}-x\right), \qquad \sin 2x = \frac{2\tan x}{1+\tan^2 x},$$
$$\cot x = -\cot(\pi - x), \qquad \csc x = \cot\tfrac{x}{2} - \cot x, \qquad \cos 2x = 2\cos^2 x - 1,$$
$$\sin(x\pm y) = \sin x\cos y \pm \cos x\sin y, \qquad \cos 2x = \frac{1-\tan^2 x}{1+\tan^2 x},$$
$$\cos(x\pm y) = \cos x\cos y \mp \sin x\sin y, \qquad \cot 2x = \frac{\cot^2 x - 1}{2\cot x},$$
$$\tan(x\pm y) = \frac{\tan x \pm \tan y}{1 \mp \tan x\tan y},$$
$$\cot(x\pm y) = \frac{\cot x\cot y \mp 1}{\cot x \pm \cot y},$$
$$\sin(x+y)\sin(x-y) = \sin^2 x - \sin^2 y,$$
$$\cos(x+y)\cos(x-y) = \cos^2 x - \sin^2 y.$$

θ	$\sin\theta$	$\cos\theta$	$\tan\theta$
0	0	1	0
$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$
$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$
$\frac{\pi}{2}$	1	0	∞

Derivatives:

1. $\frac{d(cu)}{dx} = c\frac{du}{dx}$,
2. $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$,
3. $\frac{d(uv)}{dx} = u\frac{dv}{dx} + v\frac{du}{dx}$,
4. $\frac{d(u^n)}{dx} = nu^{n-1}\frac{du}{dx}$,
5. $\frac{d(u/v)}{dx} = \frac{v\left(\frac{du}{dx}\right) - u\left(\frac{dv}{dx}\right)}{v^2}$,
6. $\frac{d(e^{cu})}{dx} = ce^{cu}\frac{du}{dx}$,
7. $\frac{d(c^u)}{dx} = (\ln c)c^u\frac{du}{dx}$,
8. $\frac{d(\ln u)}{dx} = \frac{1}{u}\frac{du}{dx}$,
9. $\frac{d(\sin u)}{dx} = \cos u\frac{du}{dx}$,
10. $\frac{d(\cos u)}{dx} = -\sin u\frac{du}{dx}$,
11. $\frac{d(\tan u)}{dx} = \sec^2 u\frac{du}{dx}$,
12. $\frac{d(\cot u)}{dx} = \csc^2 u\frac{du}{dx}$,
13. $\frac{d(\sec u)}{dx} = \tan u\sec u\frac{du}{dx}$,
14. $\frac{d(\csc u)}{dx} = -\cot u\csc u\frac{du}{dx}$,
15. $\frac{d(\arcsin u)}{dx} = \frac{1}{\sqrt{1-u^2}}\frac{du}{dx}$,
16. $\frac{d(\arccos u)}{dx} = \frac{-1}{\sqrt{1-u^2}}\frac{du}{dx}$,
17. $\frac{d(\arctan u)}{dx} = \frac{1}{1+u^2}\frac{du}{dx}$,
18. $\frac{d(\operatorname{arccot} u)}{dx} = \frac{-1}{1+u^2}\frac{du}{dx}$,
19. $\frac{d(\operatorname{arcsec} u)}{dx} = \frac{1}{u\sqrt{1-u^2}}\frac{du}{dx}$,
20. $\frac{d(\operatorname{arccsc} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}}\frac{du}{dx}$,
21. $\frac{d(\sinh u)}{dx} = \cosh u\frac{du}{dx}$,
22. $\frac{d(\cosh u)}{dx} = \sinh u\frac{du}{dx}$,
23. $\frac{d(\tanh u)}{dx} = \operatorname{sech}^2 u\frac{du}{dx}$,
24. $\frac{d(\coth u)}{dx} = -\operatorname{csch}^2 u\frac{du}{dx}$,
25. $\frac{d(\operatorname{sech} u)}{dx} = -\operatorname{sech} u\tanh u\frac{du}{dx}$,
26. $\frac{d(\operatorname{csch} u)}{dx} = -\operatorname{csch} u\coth u\frac{du}{dx}$,
27. $\frac{d(\operatorname{arcsinh} u)}{dx} = \frac{1}{\sqrt{1+u^2}}\frac{du}{dx}$,
28. $\frac{d(\operatorname{arccosh} u)}{dx} = \frac{1}{\sqrt{u^2-1}}\frac{du}{dx}$,
29. $\frac{d(\operatorname{arctanh} u)}{dx} = \frac{1}{1-u^2}\frac{du}{dx}$,
30. $\frac{d(\operatorname{arccoth} u)}{dx} = \frac{1}{u^2-1}\frac{du}{dx}$,
31. $\frac{d(\operatorname{arcsech} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}}\frac{du}{dx}$,
32. $\frac{d(\operatorname{arccsch} u)}{dx} = \frac{-1}{|u|\sqrt{1+u^2}}\frac{du}{dx}$.

Integrals:

1. $\int cu\,dx = c\int u\,dx$,
2. $\int (u+v)\,dx = \int u\,dx + \int v\,dx$,
3. $\int x^n\,dx = \frac{1}{n+1}x^{n+1}, \quad n \neq -1$,
4. $\int \frac{1}{x}dx = \ln x$,
5. $\int e^x\,dx = e^x$,
6. $\int \frac{dx}{1+x^2} = \arctan x$,
7. $\int u\frac{dv}{dx}dx = uv - \int v\frac{du}{dx}dx$,
8. $\int \sin x\,dx = -\cos x$,
9. $\int \cos x\,dx = \sin x$,
10. $\int \tan x\,dx = -\ln|\cos x|$,
11. $\int \cot x\,dx = \ln|\cos x|$,
12. $\int \sec x\,dx = \ln|\sec x + \tan x|$,
13. $\int \csc x\,dx = \ln|\csc x + \cot x|$,
14. $\int \arcsin\frac{x}{a}dx = \arcsin\frac{x}{a} + \sqrt{a^2-x^2}, \quad a > 0$,
15. $\int \arccos\frac{x}{a}dx = \arccos\frac{x}{a} - \sqrt{a^2-x^2}, \quad a > 0$,
16. $\int \arctan\frac{x}{a}dx = x\arctan\frac{x}{a} - \frac{a}{2}\ln(a^2+x^2), \quad a > 0$,
17. $\int \sin^2(ax)dx = \frac{1}{2a}\big(ax - \sin(ax)\cos(ax)\big)$,
18. $\int \cos^2(ax)dx = \frac{1}{2a}\big(ax + \sin(ax)\cos(ax)\big)$,
19. $\int \sec^2 x\,dx = \tan x$,
20. $\int \csc^2 x\,dx = -\cot x$,

Dilworth's theorem

In any partially ordered set, the maximum number of elements in any antichain equals the minimum number of chains in any partition of the set into chains.

Matrix-tree theorem (Kirchhoff's theorem)

Let matrix T = [tij], where tij is the number of multiedges between i and j, for i!=j, and tii = −degi . Number of spanning trees of a graph is equal to the determinant of a matrix obtained by deleting any k-th row and k-th column from T.

Gale-Shapley algorithm (SMP)

While there is a free man m: let w be the most-preferred woman to whom he has not yet proposed, and propose m to w. If w is free, or is engaged to someone whom she prefers less than m, match m with w, else deny proposal.

Pick's theorem

A of this polygon in terms of the number i of lattice points in the interior located in the polygon and the number b of lattice points on the boundary placed on the polygon's perimeter. A=i+b⁄2-1

Catalan number, Narayana number

{ 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, ...}

$$C(n) = \binom{2n}{n} - \binom{2n}{n+1}$$

narayana is the number of paths from (0, 0) to (2n, 0), with steps only northeast and southeast, not straying below the x-axis, with k peaks.

$$N(n,k) = 1/n * \binom{n}{k} * \binom{n}{k-1}$$

$$N(n,1) + N(n,2) + \ldots + N(n,n) = C(n)$$

Lucas's theorem

For non-negative integers m and n and a prime p

$$\binom{n}{m} = \prod \binom{ni}{mi} \ (mod\ p)$$

Wilson's theorem

$$n > 1\ primo \Leftrightarrow (n-1)! = -1 \ (mod\ n)$$

Derangements

Number of permutations of the elements of a set such that none of the elements appear in their original position. 1, 0, 1, 2, 9, 44, 265, 1854, 14833,...

$$D(n) = (n-1) * \big(D(n-1) + D(n-2)\big) = n * D(n-1) + (-1)^n.$$

Fermat primes

A Fermat prime is a prime of form $2^{2^{\wedge}n} + 1$. The only known Fermat primes are 3, 5, 17, 257, 65537. A number of form $2n + 1$ is prime only if it is a Fermat prime.

Perfect numbers

$n > 1$ is called perfect if it equals sum of its proper divisors and 1. Even n is perfect i $n = 2^{p-1}(2^p - 1)$ and $2^p - 1$ is prime (Mersenne's). No odd perfect numbers are yet found.

Carmichael numbers

A positive composite n is a Carmichael number ($a^{n-1} \equiv 1 \pmod{n}$ for all a: $\gcd(a, n) = 1$), i n is square-free, and for all prime divisors p of n, $p - 1$ divides $n - 1$.

Precedência

Lv	Operator	Description	Grouping
1	`::`	scope	Left-to-right
2	`() [] . -> ++ -- casts typeid`	postfix	Left-to-right
3	`++ -- ~ ! sizeof new delete`	unary (prefix)	Right-to-left
	`* &`	indirection and ref	
	`+ -`	unary sign operator	
4	`(type)`	type casting	Right-to-left
5	`.* ->*`	pointer-to-member	Left-to-right
6	`* / %`	multiplicative	Left-to-right
7	`+ -`	additive	Left-to-right
8	`<< >>`	shift	Left-to-right
9	`< > <= >=`	relational	Left-to-right
10	`== !=`	equality	Left-to-right
11	`&`	bitwise AND	Left-to-right
12	`^`	bitwise XOR	Left-to-right
13	`|`	bitwise OR	Left-to-right
14	`&&`	logical AND	Left-to-right
15	`||`	logical OR	Left-to-right
16	`?:`	conditional	Right-to-left
17	`= *= /= %= += -= >>= <<= &= ^= |=`	assignment	Right-to-left
18	`,`	comma	Left-to-right

Heavy-Light Tree

```
int to[maxe], ant[maxe], m;
int adj[maxv], n;
vector< vector<int> > path;
int parent[maxv];
int path_id[maxv], path_index[maxv];
int subtree[maxv];
int niv[maxv];
int in[maxv], out[maxv], tempo;
int top_sort[maxv], pos;

inline void init() {
  memset(adj,-1,sizeof adj), n = m = 0; }

inline void add(int u, int v) {
  to[m] = v, ant[m] = adj[u], adj[u] = m++;
  to[m] = u, ant[m] = adj[v], adj[v] = m++;
  n = _max(n, _max(++u,++v)); }

void dfs(int no = 0, int pai = -1, int nivel = 0) {
  in[no] = tempo++;
  subtree[no] = 1;
  niv[no] = nivel;
  parent[no] = pai;
  for(int i = adj[no]; i >= 0; i = ant[i]) {
    int prox = to[i];
    if(prox == pai) continue;
    dfs(prox,no,nivel+1);
    subtree[no] += subtree[prox]; }
  out[no] = tempo++;
  top_sort[pos++] = no; }

void heavy_light(int root = 0) {
  pos = 0, dfs(root);
  path.clear();
  vector<bool> parent_edge_processed(n, false);
  parent_edge_processed[root] = true;
  for(int i = 0; i < pos; i++) {
    int x = top_sort[i];
    if(parent_edge_processed[x])
      continue;
    vector<int> this_path;
    this_path.push_back(x);
    for(;;) {
      int is_heavy = (2*subtree[x] >= subtree[parent[x]]);
      parent_edge_processed[x] = true;
      x = parent[x];
      this_path.push_back(x);
      if(!is_heavy || parent_edge_processed[x]) break; }
    path.push_back(this_path); }
  for(int i = path.size()-1; i >= 0; i--)
    for(int j = path[i].size()-2; j >= 0; j--)
      path_id[ path[i][j] ] = i,
      path_index[ path[i][j] ] = j;
}

inline bool is_ancestor(int x, int y) {
  return in[y] <= in[x] && out[x] <= out[y]; }

int lca(int u, int v) {
  for(;;) {
    if(is_ancestor(v,u))
      return u;
    else if(is_ancestor(u,v))
      return v;
    int fu = path[path_id[u]].back();
    int fv = path[path_id[v]].back();
    if(niv[fu] >= niv[fv])
      u = fu;
    else
      v = fv;
  }
}
```

LIS

```
int dolis(int z = 0) {
  rep(i,n) { // se puder repetir, trocar por upper_bound
    int x = lower_bound(lis,lis+z,i,cmp)-lis;
    if( x == z ) z++;
    lis[x] = i;
    pai[i] = x ? lis[x-1] : -1; }
  return z;
}
```

KD-Tree

```
const int maxn = (1<<20);
struct P {
  int x, y, id;
  ll dist(P p) { return sq(x - p.x) + sq(y - p.y); }
} p[maxn];
ll resp[maxn];
int N;

bool cmpX(P p1, P p2) {
  return (p1.x != p2.x ? p1.x < p2.x : p1.y < p2.y); }
bool cmpY(P p1, P p2) {
  return (p1.y != p2.y ? p1.y < p2.y : p1.x < p2.x); }

void doit(int ini = 0, int fim = N-1, char flip = 0) {
  if(ini < fim) {
    int raiz = (ini + fim) / 2;
    nth_element(p+ini, p+raiz, p+fim+1, flip ? cmpY : cmpX);
    doit(ini, raiz - 1, flip^1);
    doit(raiz + 1, fim, flip^1); }
}

void query(P ponto, ll& best, int ini = 0,
           int fim = N-1, char flip = 0) {
  if(ini > fim) return;

  int raiz = (ini + fim) / 2;
  ll dist = ponto.dist(p[raiz]);
  if(dist > 0) { best = min(best, dist); }
  ll D = flip? sq(ponto.y-p[raiz].y) :sq(ponto.x-p[raiz].x);

  if((flip ? ponto.y <= p[raiz].y : ponto.x <= p[raiz].x)) {
    query(ponto, best, ini, raiz - 1,flip^1);
    if(D <= best) query(ponto, best, raiz+1, fim,flip^1);
  } else {
    query(ponto, best, raiz + 1, fim,flip^1);
    if(D <= best) query(ponto, best, ini, raiz - 1,flip^1);
  }
}
```

Treap

```
struct TNode {
  int x, y, z;
  TNode *L, *R;
  TNode() {}
  TNode(int x, TNode *L, TNode *R) {
    this->x = x;
    this->y = rand();
    this->z = 0;
    this->L = L;
    this->R = R;
  }
} *nil = new TNode(0,NULL,NULL), node[maxn];
typedef TNode* Node; Node arv[maxn];

void fix(Node &root) {
  if(root != nil) root->z = root->L->z + root->R->z + 1; }

void split(Node root, Node &L, Node &R, int x) {
  if(root == nil)
    return L = nil, R = nil, void();
  if(root->x <= x)
    return L = root, split(root->R,L->R,R,x), fix(L);
  return R = root, split(root->L, L,R->L,x), fix(R);}

Node merge(Node L, Node R) {
  if(L == nil) return R; else if(R == nil) return L;
  if(L->y >= R->y) return L->R = merge(L->R,R), fix(L), L;
  else return R->L = merge(L,R->L), fix(R), R; }

void insert(Node &root, Node &add) {
  if(root == nil || add->y >= root->y)
    split(root, add->L, add->R, add->x), root = add;
  else if(add->x < root->x) insert(root->L, add);
  else insert(root->R, add);
  fix(root); }

void remove(Node &root, int x) {
  if(root->x == x)
    return root = merge(root->L, root->R), fix(root);
  if(x < root->x) remove(root->L,x), fix(root);
  else remove(root->R,x), fix(root); }
```

```
42
43 int kth(Node root, int x) {
44   int myx = root->L->z + 1;
45   if(x < myx) return kth(root->L, x);
46   if(x > myx) return kth(root->R, x - myx);
47   return root->x;
48 }
```

Pontos de Articulação, Componentes Biconexos e Pontes

```
1  void gerar(int no) {
2    vector<int> vet;
3    while(q--) {
4      int no2 = from[pilha[q]];
5      vet.push_back(no2);
6      if(from[pilha[q]] == no) break;
7    }
8    // vet.size() == 2 => ponte
9  }
10
11 void dfs(int no, int dad) {
12   pai[no] = dad, ord[no] = low[no] = tempo++;
13   markv[no] = 1;
14
15   for(int i = adj[no]; i >= 0; i = ant[i]) {
16     int prox = to[i];
17     if(!marke[i]) {
18       marke[i] = marke[i^1] = 1;
19       pilha[q++] = i;  }
20     if(!markv[prox]) {
21       dfs(prox,no);
22       low[no] = min(low[no],low[prox]); }
23     else if(prox != dad) {
24       low[no] = min(low[no],ord[prox]); }
25     if(q && pai[prox] == no && low[prox] >= ord[no]) {
26       gerar(no); }
27   }
28 }
29
30 bool articulation(int no) {
31   if(pai[no] == -1) {
32     int ct = 0;
33     for(int i = adj[no]; i >= 0; i = ant[i]) {
34       int prox = to[i];
35       ct += (pai[prox] == no);
36     }
37     return ct >= 2;
38   }
39   else {
40     for(int i = adj[no]; i >= 0; i = ant[i]) {
41       int prox = to[i];
42       if(pai[prox] == no && low[prox] >= ord[no]) {
43         return 1;
44       }
45     }
46     return 0;
47   }
48 }
49
50 bool bridge(int x, int y) {
51   if(ord[x] > ord[y]) swap(x,y);
52   return pai[y] == x && low[y] > ord[x]; // freq[x][y] == 1?
53 }
```

KonigtheoremHopcroftkarpDilworththeorem

```
1  public class KonigtheoremHopcroftkarpDilworththeorem {
2
3    public static void main(String[] args) throws Exception {
4      new KonigtheoremHopcroftkarpDilworththeorem().solve();
5    }
6
7    int a[][]; // adjacent matrix
8    int n;
9
10   int match[], pai[], vis[], passo = 0;
11
12   void path(int no) {
13     while(no >= 0) {
14       match[ pai[no] ] = no;
15       match[ no ] = pai[no];
16       no = pai[pai[no]];
17     }
18   }
```

```
19
20    int fila[], pos, q;
21
22    int hopcroftKarp() {
23      fila = new int[2*n];
24      pai = new int[2*n];
25      vis = new int[2*n];
26      match = new int[2*n]; Arrays.fill(match,-1);
27      for(int i = 0; i < n; i++) { // left
28        pos = q = 0;
29        fila[q++] = i; vis[i] = ++passo; pai[i] = -1;
30        boolean found = false;
31        while(pos < q && !found) {
32          int no = fila[pos++];
33          for(int j = 0; j<n; j++) if(a[no][j] != 0) {//right
34            int prox = j + n;
35            if(match[prox] == -1) {
36              pai[prox] = no;
37              path(prox);
38              found = true;
39              break;
40            }
41            else if(vis[ match[prox] ] != passo) {
42              vis[ match[prox] ] = passo;
43              pai[prox] = no;
44              pai[ match[prox] ] = prox;
45              fila[q++] = match[prox];
46            }
47          }
48        }
49      }
50      int mf = 0;
51      for(int i = 0; i < n; i++) mf += (match[i] != -1 ?1:0);
52      return mf;
53    }
54
55    boolean cover[];
56
57    void minVertexCover() {
58      cover = new boolean[2*n];
59      for(int i = 0; i < n; i++) if(match[i] != -1) {
60        cover[i] = true;
61      }
62      ++passo;
63      for(int i = 0; i < n; i++)
64            if(match[i] == -1 && vis[i] != passo) { // left
65        pos = q = 0;
66        fila[q++] = i; vis[i] = passo;
67        while(pos < q) {
68          int no = fila[pos++]; cover[no] = false;
69          for(int j = 0; j<n; j++) if(a[no][j] != 0) {//right
70            int prox = j + n;
71            if(match[no] == prox) continue;
72            else cover[prox] = true;
73            if(match[prox]!= -1 &&vis[match[prox]] != passo){
74              vis[match[prox]] = passo;
75              fila[q++] = match[prox];
76            }
77          }
78        }
79      }
80      for(int i = 0; i < n; i++) if(cover[n + i]) {
81        cover[i] = true;
82      }
83    }
84
85    void solve() throws Exception {
86      int h = hopcroftKarp(); int foi = 0;
87      System.out.printf("%d\n",n - h);
88      minVertexCover();
89      for(int i = 0; i < n; i++) if(!cover[i] && !cover[n+i]){
90        if(foi++ != 0) System.out.printf(" ");
91        System.out.printf("%d",i+1);
92      }
93      System.out.printf("\n");
94    }
95  }
```

MaxFlow e Gomory-Hu

```
int level[maxv], fila[maxv], copy_adj[maxv];

int bfs(int source, int sink) {
  memset(level,-1,sizeof level);
  level[source] = 0;
  int pos = 0, tam = 0;
  fila[tam++] = source;
  while(pos < tam) {
    int now = fila[pos++];
    for(int i = adj[now]; i >= 0; i = ant[i])
          if(cap[i] && level[to[i]] == -1) {
      level[to[i]] = level[now] + 1;
      fila[tam++] = to[i];
    }
  }
  return level[sink] != -1;
}

int dfs(int no, int sink, int flow) {
  if(no == sink) return flow;
  for(int &i = copy_adj[no]; i >= 0; i = ant[i])
        if(cap[i] && level[no] + 1 == level[to[i]]) {
    int nflow = dfs(to[i],sink,min(flow,cap[i]));
    if(nflow) {
      cap[i] -= nflow, cap[i^1] += nflow;
      return nflow;
    }
  }
  return 0;
}

long long maxflow(int source, int sink) {
  long long mf = 0;
  while(true) {
    if(bfs(source,sink)) {
      memcpy(copy_adj,adj,sizeof adj);
      while(true) {
        int add = dfs(source,sink,1<<30);
        if(add) mf += add; else break;
      }
    } else break;
  }
  return mf;
}

int parent[maxv];
int answer[maxv][maxv];

void gomory_hu() {
  static int cap2[maxe]; rep(i,m) cap2[i] = cap[i];
  memset(parent,0,sizeof parent);
  memset(answer,0x3f,sizeof answer);
  for(int i = 1;i < n; i++){
    rep(k,m) cap[k] = cap2[k];
    int f = maxflow(i, parent[i]);
    bfs(i,parent[i]);
    for (int j = i+1; j < n; j++) {
      if (level[j] != -1 && parent[j]==parent[i]) {
        parent[j] = i;
      }
    }
    answer[i][parent[i]]=answer[parent[i]][i] = f;
    for (int j=0;j<i;++j) {
      answer[i][j] = answer[j][i] =
           min(f,answer[parent[i]][j]);
    }
  }
}
```

Mincost MaxFlow

```
int dist[maxv], pot[maxv], pai[maxv];
set< pair<int,int> > heap;
void update(int no, int ndist, int p) {
  if(ndist >= dist[no]) return;
  if(dist[no] < inf) heap.erase(pair<int,int>(dist[no],no));
  dist[no] = ndist, pai[no] = p;
  heap.insert(pair<int,int>(dist[no],no)); }

pair<int,int> top() {
  pair<int,int> ret = *heap.begin();
  heap.erase(heap.begin());
  return ret; }

int djikstra(int source, int sink) {
  heap.clear(), memset(dist,inf,sizeof dist);
  update(source,0,-1);
  while(heap.size()) {
    pair<int,int> p = top();
    for(int i = adj[p.second]; i>=0; i = ant[i]) if(cap[i])
      update(to[i],p.first+w[i]+pot[p.second]-pot[to[i]],i);
  }
  return dist[sink] < inf; }

pair<int,int> mcmf(int source, int sink) {
  //need bellman-ford?
  //memset(pot,0x3f,sizeof pot), pot[source] = 0;
  //for(int k = 0; k < n; k++) for(int i = 0; i < n; i++)
  //     for(int j = adj[i]; j >= 0; j = ant[j]) if(cap[j])
  //     pot[to[j]] = min(pot[to[j]], pot[i] + w[j]);
  memset(pot,0,sizeof pot);
  pair<int,int> p(0,0); // cost,flow
  while(djikstra(source,sink)) {
    int cost = 0, flow = inf;
    for(int x = sink; x != source; x = from[pai[x]])
      if(cap[pai[x]] < flow) flow = cap[pai[x]];
    for(int x = sink; x != source; x = from[pai[x]])
      cap[pai[x]] -= flow, cap[pai[x]^1] += flow,
      cost += w[pai[x]]*flow;
    for(int x = 0; x < n; x++) pot[x] += dist[x];
    p.first += cost, p.second += flow;
  }
  return p; }
```

Pape

```
int mate[maxn], gf[maxn], n;
char inner[maxn];
vector<int> adj[maxn]; // bidirectional

int func() {
  memset(mate,-1,sizeof(mate));
  for(int i = 0; i < n; i++) {
    if(mate[i] != -1) continue;
    for(int j = adj[i].size()-1; j >= 0; j--) {
      int prox = adj[i][j];
      if(mate[prox] != -1) continue;
      mate[i] = prox, mate[prox] = i;
      break; }
  }
  for(int i = 0; i < n; i++) {
    if(mate[i] != -1) continue;
    memset(inner,0,sizeof(inner));
    queue<int> q;
    q.push(i), inner[i] = 1, gf[i] = -1;
    while(!q.empty()) {
      int u = q.front(), v; q.pop();
      for(int j = adj[u].size()-1; j >= 0; j--) {
        int prox = adj[u][j];
        if(inner[prox]) continue;
        if(mate[prox] == -1) {
          while(u >= 0) { // aumente
            int old = mate[u];
            mate[u] = prox, mate[prox] = u;
            prox = old, u = gf[u]; }
          goto next; }
        for(v = u; v >= 0; v = gf[v]) if(v == prox) break;
        if(v != prox) {
          inner[prox] = 1;
          q.push(mate[prox]), gf[ mate[prox] ] = u; }
      }
    }
    next:;
  }
  int tot = 0;
  for(int i = 0; i < n; i++) tot += (mate[i] != -1);
  return tot / 2; }
```

Stable Marriage

```
1  struct S {
2    // sorted list of indices by
3    int lista[maxn], pref[maxn]; preference
4  } h[maxn], m[maxn];
5  int prox[maxn], quem[maxn], n;
6
7  void find(int H, int M) {
8    if(quem[M] == -1) {
9      quem[M] = H;
10   }
11   else if( m[M].pref[H] < m[M].pref[quem[M]] ) {
12     int now = quem[M];
13     quem[M] = H;
14     find(now,h[now].lista[prox[now]++]);
15   }
16   else {
17     find(H,h[H].lista[prox[H]++]);
18   }
19 }
20
21 void process() {
22   memset(prox,0,sizeof(prox));
23   memset(quem,-1,sizeof(quem));
24   rep(i,n) if(i) find(i,h[i].lista[prox[i]++]);
25   rep(i,n) if(i) printf("%d %d\n",quem[i],i);
26 }
```

Corte Mínimo – Stoer-Wagner

```
1  int adj[110][110];
2  int n,m, inf = 1<<30;
3  bool foi[110], mark[110];
4  int cap[110];
5
6  int corte() {
7    int ret, S, T;
8    memcpy(mark,foi,sizeof foi);
9    fr(i,0,n) if( !foi[i] ) {
10     fr(j,0,n) cap[j] = adj[i][j];
11     mark[i] = true;
12     S = i;
13     break;
14   }
15   while( true ) {
16     int x, y = 0;
17     fr(i,0,n) if( !mark[i] && cap[i] > y ) x = i,y = cap[i];
18     if( !y ) break;
19     ret = y;
20     T = S;
21     S = x;
22     mark[S] = true;
23     fr(i,0,n) if( !mark[i] && adj[S][i] ) {
24       cap[i] += adj[S][i];
25     }
26   }
27   foi[S] = true;
28   fr(i,0,n) {
29     adj[i][T] += adj[i][S];
30     adj[T][i] += adj[S][i];
31   }
32   return ret;
33 }
34 bool read() {
35   scanf("%d%d", &n, &m);
36   memset(foi,false,sizeof foi);
37   memset(adj,0,sizeof adj);
38   fr(i,0,m) {
39     int a,b,c;
40     scanf("%d%d%d", &a, &b, &c); a--,b--;
41     adj[a][b] = adj[b][a] = c;
42   }
43   int res = inf;
44   fr(k,1,n) {
45     res = min(res, corte());
46   }
47   printf("%d\n",res);
48   return true;
49 }
```

Componente Fortemente Conexo

```
1  void dfs(int no, int p) {
2    d[no] = low[no] = tempo++;
3    pai[no] = p, mark[no] = 1;
4    pilha[q++] = no;
5    rep(i,adj[no].size()) {
6      int prox = adj[no][i];
7      if(!mark[prox]) {
8        dfs(prox,no);
9        low[no] = min(low[no],low[prox]);
10     }
11     else if(comp[prox] == -1) {
12       low[no] = min(low[no], d[prox]);
13     }
14   }
15   if(low[no] == d[no]) {
16     while(true) {
17       int x = pilha[--q];
18       comp[x] = ncomp;
19       if(x == no) break;
20     }
21     ncomp++;
22   }
23 }
```

Gauss

```
1  void gauss() {
2    int linha = 0;
3    for(int pivo = 0; pivo < N; pivo++) {
4      int pos = -1;
5      double vmax = 0;
6      for(int i = linha; i < M; i++) {
7        if(cmp(A[i][pivo]) && cmp(fabs(A[i][pivo]),vmax) >0 ){
8          pos = i;
9          vmax = fabs(A[i][pivo]);
10       }
11     }
12     if(pos != -1) {
13       if(pos != linha) {
14         for(int i = 0; i < N; i++) {
15           swap(A[linha][i],A[pos][i]);
16         }
17         swap(B[linha],B[pos]);
18       }
19       double F = A[linha][pivo]; // norm
20       for(int i = 0; i < N; i++) {
21         A[linha][i] /= F;
22       }
23       B[linha] /= F;
24
25       for(int i = linha+1; i < M; i++) {
26         F = A[i][pivo];
27         if( cmp(F) ) {
28           for(int j = pivo; j < N; j++) {
29             A[i][j] = (A[linha][j] * F) –
30                       (A[i][j] * A[linha][pivo]);
31           }
32           B[i] = (B[linha] * F) - (B[i] * A[linha][pivo]);
33         }
34       }
35       linha++;
36     }
37   }
38
39   for(int i = linha; i < M; i++) {
40     if( cmp(B[i]) ) {
41       printf("IMPOSSIBLE\n");
42       assert(0);
43       return; }
44   }
45
46   if(linha < N) {
47     printf("VERY SOLUTIONS\n");
48     assert(0);
49     return; }
50
51   for(int i = linha-1; i >= 0; i--) {
52     assert( cmp(A[i][i]) );
53     B[i] /= (A[i][i]);
54     for(int j = 0; j < i; j++) {
55       B[j] -= (A[j][i] * B[i]); }
56   }
57 }
```

Teroema Chinês do Resto

```
pair<ll,ll> euclides(ll x, ll y) {
  if(!y) return pair<ll,ll>(1,0);
  if(!x) return pair<ll,ll>(0,1);
  pair<ll,ll> p = euclides(y%x,x);
  return pair<ll,ll>(p.second - p.first*(y/x), p.first);
}
ll inverse(ll x,ll mod) {
  pair<ll,ll> p = euclides(x,mod);
  return ((p.first % mod) + mod) % mod;
}

ll chinese(ll *A, ll *B, int n) {
  ll mod = 1, res = 0;
  rep(i,n) mod *= B[i];
  rep(i,n) {
    ll add = A[i] * (mod / B[i]) %
                  mod * inverse(mod / B[i], B[i]) % mod;
    res = (res + add) % mod;
  }
  return res;
}

ll gcd(ll x, ll y) {
  while(x) y %= x, swap(x,y);
  return y;
}

vector<ll> solve(ll a, ll b, ll n) { // a*x=b
  vector<ll> v;
  pair<ll,ll> s = euclides(a, n);
  ll g = gcd(a,b);
  if (b % g == 0) {
    ll x0 = ((s.first*(b/g)) % n + n) % n;
    for(int i = 0; i < g; i++) {
      ll nx = x0 + i*(n/g);
      v.push_back((nx % n + n) % n);
    }
    sort(v.begin(),v.end());
  }
  return v;
}
```

ou

```
int extended_euclid(int a, int b, int &x, int &y) {
  int xx = y = 0;
  int yy = x = 1;
  while (b) {
    int q = a/b;
    int t = b; b = a%b; a = t;
    t = xx; xx = x-q*xx; x = t;
    t = yy; yy = y-q*yy; y = t;
  }
  return a;
}

PII chinese_remainder_theorem(int x, int a, int y, int b) {
  int s, t;
  int d = extended_euclid(x, y, s, t);
  if (a%d != b%d) return make_pair(0, -1);
  return make_pair(mod(s*b*x+t*a*y,x*y)/d, x*y/d);
}


PII chinese_remainder_theorem(const VI &x, const VI &a) {
  PII ret = make_pair(a[0], x[0]);
  for (int i = 1; i < x.size(); i++) {
    ret = chinese_remainder_theorem(
                  ret.first, ret.second, x[i], a[i]);
    if (ret.second == -1) break;
  }
  return ret;
}
```

Pollard Rho

```
ll mulMod(ll a, ll b, ll n) {
  if(a == 0 || b <= (1LL << 62) / a) return a * b % n;
  ll result = 0;
  if(a < b) swap(a,b);
  for(; b != 0; b >>= 1) {
    if(b & 1) {
      result += a; if(result >= n) result -= n; }
    a <<= 1; if(a >= n) a -= n; }
  return result; }

ll powMod(ll a, ll exp, ll n) {
  ll result = 1;
  for(; exp != 0; exp >>= 1) {
    if(exp & 1) {
      result = mulMod(result, a, n); }
    a = mulMod(a, a, n); }
  return result;
}

ll f(ll x, ll c, ll n) {
  ll result = mulMod(x, x, n);
  result += c; if(result >= n) result -= n;
  return result; }

ll gcd(ll u, ll v) {
  while(u) v %= u, swap(u,v); return v; }

bool primeTestMillerRabin(ll n, ll a) {
  ll d = n-1, x;
  int s = 0, r;
  while ((d & 1) == 0 && d != 0) s++, d >>= 1;
  while (a >= n) a >>= 1;
  x = powMod(a, d, n);
  if (x != 1 && x != n-1) {
    r = 1;
    while (r <= s-1 && x != n-1) {
      x = mulMod(x, x, n);
      if (x == 1) return false; else r++; }
    if (x != n-1) return false; }
  return true; }

bool multipleMillerRabin(long long n) {
  if (n <= 1) return false;
  if (n <= 3) return true;
  if ((n & 1) == 0) return false;
  return
    primeTestMillerRabin(n, 2) &&
    primeTestMillerRabin(n, 3) &&
    primeTestMillerRabin(n, 5) &&
    primeTestMillerRabin(n, 7) &&
    (n < 3215031751LL ||
    (primeTestMillerRabin(n, 11) &&
      (n < 2152302898747LL ||
      (primeTestMillerRabin(n, 13) &&
        (n < 3474749660383LL ||
        (primeTestMillerRabin(n, 17) &&
          (n < 341550071728321LL ||
          (primeTestMillerRabin(n, 23)))))))));
}

ll pollardsRho(ll n, bool testPrime) {
  for (int count = 0, c = 1;;) {
    ll x = 2, y = 2, pot = 1, lam = 1, d;
    do {
      ++count;
      if (count == 12) {
        if (testPrime) return multipleMillerRabin(n) ?n:1;
        else if (multipleMillerRabin(n)) return n;
      }
      if(pot == lam) x = y, pot <<= 1, lam = 0;
      y = f(y,c,n), lam++;
      d = gcd(x >= y ? x - y : y - x, n);
    } while (d == 1);
    if (d != n) return d; else c++; }
}

bool tripleMillerRabin(int x) {
  return
    primeTestMillerRabin(x, 2) && primeTestMillerRabin(x, 7)
    && primeTestMillerRabin(x, 61); }

bool isPrime(ll n) {
  if (n <= 0x7fffffff) return tripleMillerRabin((int)n);
  return pollardsRho(n, true) == n;
}
```

Fast Fourier Transform

```
typedef complex<double> T;
typedef vector<T> VT;
const int maxn = 1<<20;
T _A[maxn];

void fft(T* A, int n, bool inv = false) {
  if(n==1) return;
  T *A0=A,*A1=A+n/2; rep(i,n) _A[i]=A[i];
  rep(i,n/2) A0[i]=_A[2*i], A1[i]=_A[2*i+1];
  fft(&A0[0],n/2,inv), fft(&A1[0],n/2,inv);
  T w=exp(T(0,(inv?-2:2)*M_PI/n)), wnow=T(1,0);
  rep(i,n/2)
       _A[i]=A0[i]+wnow*A1[i],
       _A[i+n/2]=A0[i]-wnow*A1[i],wnow*=w;
  rep(i,n) A[i]=_A[i];
}

void multiply(VT& A, VT B) {
  int n = A.size() + B.size();
  while(n & (n-1)) n++;
  A.resize(n), B.resize(n);
  fft(&A[0],n,false), fft(&B[0],n,false);
  rep(i,n) A[i] *= B[i];
  fft(&A[0],n,true);
  rep(i,n) A[i] /= n;
}
```

Josephus

```
ll pilha[1<<21];
ll josephus(ll n, ll k, int st = 0) {
  pilha[st++] = n;
  while( n = pilha[st-1] ) {
    if( n == 1 || k == 1 ) break;
    if( k > n ) pilha[st++] = n-1;
    else pilha[st++] = n-n/k;
  }
  ll res = 0;
  if( k == 1 ) res = n-1;
  while( st ) {
    n = pilha[--st];
    if( k > n ) res = (res + k)%n;
    else res = res - n%k, res += (res < 0 ? n : res/(k-1));
  }
  return res;

  // recursivo
  if( n == 1 ) return 0;
  if( k == 1 ) return n-1;
  if( k > n ) return ( josephus(n-1, k) + k )%n;
  res = josephus(n-n/k, k) - n%k;
  if( res < 0 ) res += n;
  else res += res/(k-1);
  return res;
}
```

Polinômios

```
typedef complex<double> cdouble;
int cmp(cdouble x, cdouble y = 0) {
  return cmp(abs(x), abs(y));
}
const int TAM = 200;
struct poly {
  cdouble poly[TAM]; int n;
  poly(int n = 0): n(n) { memset(p, 0, sizeof(p)); }
  cdouble& operator [](int i) { return p[i]; }
  poly operator ~() {
    poly r(n-1);
    for(int i = 1; i <= n; i++)  r[i-1] = p[i] * cdouble(i);
    return r;
  }
  pair<poly, cdouble> ruffini(cdouble z) {
    if (n == 0) return make_pair(poly(), 0);
    poly r(n-1);
    for (int i = n; i > 0; i--) r[i-1] = r[i] * z + p[i];
    return make_pair(r, r[0] * z + p[0]);
  }
  cdouble operator ()(cdouble z) {
    return ruffini(z).second;
  }
```

```
24    cdouble find_one_root(cdouble x) {
25      poly p0 = *this, p1 = ~p0, p2 = ~p1;
26      int m = 1000;
27      while (m--) {
28        cdouble y0 = p0(x);
29        if (cmp(y0) == 0) break;
30        cdouble G = p1(x) / y0;
31        cdouble H = G * G - p2(x) - y0;
32        cdouble R = sqrt(cdouble(n-1)*(H * cdouble(n) - G*G));
33        cdouble D1 = G + R, D2 = G - R;
34        cdouble a = cdouble(n) / (cmp(D1, D2) > 0 ? D1 : D2);
35          x -= a;
36          if (cmp(a) == 0) break;
37        }
38        return x;
39      }
40    vector<cdouble> roots() {
41      poly q = *this;
42      vector<cdouble> r;
43      while (q.n > 1) {
44        cdouble z(rand()/double(RAND_MAX),
45                  rand()/double(RAND_MAX));
46        z = q.find_one_root(z); z = find_one_root(z);
47        q = q.ruffini(z).first;
48        r.push_back(z);
49      }
50      return r;
51    }
52  };
```

Simplex

```
1   const double eps = 1e-9;
2   typedef long double T;
3   typedef vector<T> VT; typedef vector<int> VI;
4   vector<VT> A; VT b,c,res; VI kt,N; int m;
5
6   inline void pivot(int k,int l,int e){
7       int x=kt[l]; T p=A[l][e];
8       rep(i,k) A[l][i]/=p; b[l]/=p; N[e]=0;
9       rep(i,m)
10          if(i!=l) b[i]-=A[i][e]*b[l],A[i][x]=A[i][e]*-A[l][x];
11      rep(j,k) if (N[j]){
12          c[j]-=c[e]*A[l][j];
13          rep(i,m) if (i!=l) A[i][j]-=A[i][e]*A[l][j];
14      }
15      kt[l]=e; N[x]=1; c[x]=c[e]*-A[l][x];
16  }
17  VT doit(int k){
18      VT res; T best;
19      while (1){
20          int e=-1,l=-1;
21          rep(i,k) if (N[i] && c[i]>eps) {e=i; break;}
22          if (e==-1) break;
23          rep(i,m)
24              if (A[i][e]>eps && (l==-1 || best>b[i]/A[i][e]))
25                  best=b[ l=i ]/A[i][e];
26          if (l==-1) /*ilimitado*/ return VT();
27          pivot(k,l,e);
28      }
29      res.resize(k,0); rep(i,m) res[kt[i]]=b[i];
30      return res;
31  }
32
33  VT simplex(vector<VT> &AA,VT &bb,VT &cc){
34      int n=AA[0].size(),k;
35      m=AA.size(); k=n+m+1; kt.resize(m);
36      b=bb; c=cc; c.resize(n+m); A=AA;
37      rep(i,m){
38         A[i].resize(k); A[i][n+i]=1; A[i][k-1]=-1; kt[i]=n+i;}
39      N=VI(k,1); rep(i,m) N[kt[i]]=0;
40      int pos=min_element(b.begin(),b.end())-b.begin();
41      if (b[pos]<-eps){
42          c=VT(k,0); c[k-1]=-1; pivot(k,pos,k-1); res=doit(k);
43          if (res[k-1]>eps) /*impossivel*/ return VT();
44          rep(i,m) if (kt[i]==k-1)
45              rep(j,k-1)
46                  if (N[j] && (A[i][j]<-eps || eps<A[i][j])){
47                      pivot(k,i,j); break;
48              }
49          c=cc; c.resize(k,0); rep(i,m) rep(j,k) if (N[j])
    c[j]-=c[kt[i]]*A[i][j];
50      }
51      res=doit(k-1); if (!res.empty()) res.resize(n);
52      return res;
53  }
```

Aho-Corasick

```
1  struct Trie {
2    int adj[52], pai, fail, c;
3    bitset<1000> b;
4    void init(char ch, int dad) {
5      c = ch, pai = dad, fail = 0;
6      memset(adj,-1,sizeof(adj)), b.reset(); }
7    int& operator[](int i) { return adj[i]; }
8    int func(int);
9  } no[maxn]; int nno;
10
11 int Trie::func(int x) {
12   int& res = adj[x];
13   if(res == -1) {
14     if(fail == 0 && no[0][x] == -1) res = 0;
15     else res = no[fail].func(x);
16   }
17   return res; }
18
19 int func(char c) {
20   if('a' <= c && c <= 'z') return c - 'a';
21   else return c - 'A' + 26; }
22
23 void add(char *s, int indice) {
24   int atual = 0;
25   for(int i = 0; s[i]; i++) {
26     int& prox = no[atual][func(s[i])];
27     if(prox == -1) {
28       prox = nno++, no[prox].init(s[i],atual);
29     } atual = prox;
30   }
31   no[atual].b[indice] = 1; }
32
33 #define err(w,c,a) (no[w][c]==-1 || no[w][c]==a)
34 void ativar() {
35   queue<int> Q;
36   for(int i = 0; i < 52; i++) {
37     if(no[0][i] != -1) {
38       Q.push(no[0][i]);
39     }
40   }
41   while(!Q.empty()) {
42     int atual = Q.front(); Q.pop();
43     int c = func(no[atual].c);
44     bitset<1000> &b = atual[no].b;
45     int& w = no[atual].fail = no[atual].pai;
46     while(w && err(w,c,atual)) {
47       w = no[w].fail, b |= no[w].b; }
48     if(!err(w,c,atual)) {
49       w = no[w][c], b |= no[w].b; }
50     for(int i = 0; i < 52; i++) {
51       if(no[atual][i] != -1) {
52         Q.push(no[atual][i]);
53       }
54     }
55   }
56 }
57 #undef err
58
59 bitset<1000> func(char texto[]) {
60   bitset<1000> b;
61   int atual = 0;
62   for(int i = 0; texto[i]; i++) {
63     atual = no[atual].func( func(texto[i]) );
64     b |= no[atual].b; }
65   return b; }
```

KMP

```
1  void kmp(char * p, char * s) {
2    static int f[1<<20];
3    f[0] = f[1] = 0;
4    for( int i = 1, j = 0 ; p[i] ; )
5      if( p[i] == p[j] ) f[++i] = ++j;
6      else if( j ) j = f[j];
7      else f[++i] = 0;
8    for( int i = 1, j = *s==*p ; s[i-1] ; ) {
9      if( !p[j] ) printf("%d ", i); //match
10     if( s[i] == p[j] ) ++i, ++j;
11     else if( j ) j = f[j];
12     else ++i;
13   }
14 }
```

Manacher

```
int manacher() {
  n = strlen(s);
  for(int i = 0, l = 0, r = -1; i < n; i++) {  // even
    int k = (i > r ? 0 : min(even[r + l - i - 1], r - i));
    while( 0<= i-k && i+k+1 < n && s[i-k] == s[i+k+1] ) k++;
    even[i] = k;
    if(i + k > r) l = i - k + 1, r = i + k;
    // 1 <= x <= even[i] : string(s+i-x+1, s+i+x+1)
    // for(int x = 1; x <= even[i]; x++)
    //   cout<< string(s + i - x + 1, s + i + x + 1) <<endl;
  }
  for(int i = 0, l = 0, r = -1; i < n; i++) { // odd
    int k = (i > r ? 0 : min(odd[r + l - i],r - i));
    while(0<= i-k-1 && i+k+1<n && s[i-k-1] == s[i+k+1]) k++;
    odd[i] = k; if(i + k > r) l = i - k, r = i + k;
    // 0 <= x <= odd[i] : string(s + i - x, s + i + x + 1);
    // for(int x = 0; x <= odd[i]; x++)
    //   cout <<  string(s + i - x, s + i + x + 1) << endl;
  }
  int palindromes = 0;
  for(int i = 0; i < n; i++)
    palindromes += even[i], palindromes += (odd[i] + 1);
  return palindromes;
}
```

Array de Sufixo

```
int cmp(int i, int j) {
  return dp[(i += skip) <= n ?i:n][h]
         < dp[(j+=skip) <= n ?j:n][h];
}

void suffixArray(int len = 1<<30) {
  n = strlen(s);
  for(int i = 0; i <= n; i++)
    suffix[i] = i, dp[i][0] = s[i];
  skip = h = 0, sort(suffix,suffix+n+1,cmp);
  for(skip=1;skip;skip<<=1) {
    int g = -1;
    for(int i = 0,j = 0; i <= n;) {
      if(j==n+1 or dp[suffix[i]][h] != dp[suffix[j]][h]) {
        sort(suffix+i,suffix+j,cmp);
        dp[suffix[i]][h+1] = ++g;
        while(++i < j) {
          if(cmp(suffix[i-1],suffix[i])) g++;
          dp[suffix[i]][h+1] = g;
        }
      } else j++;
    }
    if(++h && g >= n) break; else if(skip >= len) break;
  }
}

int lcp(int i, int j) {
  if(i == j) return n - i;
  int len = 0;
  for(int k = h; k >= 0; k--) if(dp[i][k] == dp[j][k])
    len += (1<<k), i += (1<<k), j += (1<<k);
  return len;
}
```

Regex

```
public class RegularExpression {

  /*
   * .        :: Matches any sign
   * [ab]     :: Set definition, can match the letter a or b
   * [^ab]    :: This can match any character except a or b
   * [a-d1-7] :: Ranges, letter between a and d and
   *                figures from 1 to 7, will not match d1
   * X|Z      :: Finds X or Z
   * XZ       :: Finds X directly followed by Z
   * X*       :: X occurs zero or more times
   * X+       :: X occurs one or more times
   * X{Y,Z}   :: X occurs between Y and Z times
   * $        :: Checks if a line end follows
   */

  public static void main(String[] args) throws Exception {
    Pattern pattern = Pattern.compile("a{1,500}b"); // regex
```

```
19      Matcher matcher = pattern.matcher("aaaaab"); // input
20      boolean found = false;
21      while(matcher.find()) {
22        System.out.println(
23           "at ("+ matcher.start() +","+ matcher.end() + ")");
24        found = true;
25      }
26      if(!found) {
27        System.out.println("null");
28      }
29      System.out.println(matcher.matches() ? "yes" : "no");
30    }
31  }
```

Árvore de Sufixo

```
1   const int K = 26, maxn = 500000+10;
2   struct state {
3     int pos, length, link, v, w;
4     int next[K];
5   };
6   state st[maxn];
7   int cnt, last;
8   void init() {
9     cnt = last = 0;
10    st[0].link = -1;
11    st[0].pos = -1;
12    st[0].w = INT_MAX;
13    memset (st[0].next, -1, sizeof st[0].next);
14    ++cnt; }
15
16  void sa_extend (int position, char c) {
17    int nlast = cnt++;
18    st[nlast].length = st[last].length + 1;
19    st[nlast].pos = position;
20    st[nlast].w = INT_MAX;
21    memset (st[nlast].next, -1, sizeof st[0].next);
22    int p;
23    for (p=last; p!=-1 && st[p].next[c]==-1; p=st[p].link)
24      st[p].next[c] = nlast;
25    if (p == -1)
26      st[nlast].link = 0;
27    else {
28      int q = st[p].next[c];
29      if (st[p].length + 1 == st[q].length)
30        st[nlast].link = q;
31      else {
32        st[cnt].length = st[p].length + 1;
33        memcpy (st[cnt].next, st[q].next, sizeof st[0].next);
34        st[cnt].link = st[q].link;
35        st[cnt].pos = -1;
36        st[cnt].w = INT_MAX;
37        for (; p!=-1 && st[p].next[c]==q; p=st[p].link)
38          st[p].next[c] = cnt;
39        st[q].link = st[nlast].link = cnt;
40        ++cnt;
41      }
42    }
43    last = nlast;
44  }
45
46  void build_dfa(char *s, int n){
47    init();
48    for (size_t i=0; i<n; ++i) {
49      sa_extend (i, s[i]-'a');
50    }
51  }
```

Algoritmo Z

```
1   int main() {
2     scanf("%s",str);
3     n = strlen(str);
4     fz[0] = n;
5     for(int i = 1, l = 0, r = 0; i < n; i++) {
6       if(l && i + fz[i-l] < r)
7         fz[i] = fz[i-l];
8       else {
9         int j = min(l ? fz[i-l] : 0, i > r ? 0 : r-i);
10        while(str[i+j] == str[j] && ++j);
11        fz[i] = j, l = i, r = i + j;
12      }
13    }
14  }
```

Poker

```
const char* rank_names = "**23456789TJQKA";
const char* suit_names = "CDHS";

struct card {
  int rank, suit;
  int read() {
    char ch[2];
    if (scanf(" %c%c", &ch[0], &ch[1]) == EOF) return 0;
    for (rank = 0; rank_names[rank] != ch[0]; rank++);
    for (suit = 0; suit_names[suit] != ch[1]; suit++);
    return 1;
  }
  void print() {
    printf("%c%c", rank_names[rank], suit_names[suit]); }
};

struct freq_lt {
  int* freq;
  freq_lt(int* freq): freq(freq) {}
  bool operator ()(const card A, const card B) const {
    if (int t = freq[A.rank] - freq[B.rank]) return t > 0;
    else return A.rank > B.rank;
  }
};

struct hand {
  card C[5];
  int type() {
    int freq[15]; memset(freq, 0, sizeof(freq));
    sort(C, C+5, freq_lt(freq));
    bool flush = true, straight = true;
    for (int i = 0; i < 5; i++) {
      if (i && C[i].suit != C[i-1].suit) flush = false;
      if (i && !(C[i].rank == 5 && C[i-1].rank == 14)
        && C[i].rank != C[i-1].rank - 1) straight = false;
      freq[C[i].rank]++;
    }
    sort(C, C+5, freq_lt(freq));

    int kind[5]; memset(kind, 0, sizeof(kind));
    for (int i = 2; i <= 14; i++) kind[freq[i]]++;
    if (straight && flush) return 8;
    else if (kind[4]) return 7;
    else if (kind[3] && kind[2]) return 6;
    else if (flush) return 5;
    else if (straight) return 4;
    else if (kind[3]) return 3;
    else return kind[2];
  }
  bool operator <(hand H) {
    if (int t = type() - H.type()) return t < 0;
    for (int i = 0; i < 5; i++)
    if (int t = C[i].rank - H.C[i].rank) return t < 0;
    return false;
  }
};
```

Minimum Enclosing Circle

```
typedef pair<Ponto,double> circle;
Ponto circumcenter(Ponto p, Ponto q, Ponto r) {
  Ponto a = p - r, b = q - r,
    c = Ponto(a * (p + r) / 2, b * (q + r) / 2);
  return Ponto(c%Ponto(a.y, b.y), Ponto(a.x, b.x)%c)/(a%b);
}
bool in_circle(circle C, Ponto p){
  return cmpEPS(C.first.dist(p) , C.second) <= 0;
}
circle spanning_circle(Ponto *T, int n) {
  random_shuffle(T, T + n);
  circle C(Ponto(), 0);
  for (int i = 0; i < n; i++) if (!in_circle(C, T[i])) {
    C = circle(T[i], 0);
    for (int j = 0; j < i; j++) if (!in_circle(C, T[j])) {
      C = circle((T[i] + T[j]) / 2, T[i].dist(T[j]) / 2);
      for (int k = 0; k < j; k++) if (!in_circle(C, T[k])) {
        Ponto o = circumcenter(T[i], T[j], T[k]);
        C = circle(o, T[k].dist(o));
      }
    }
  }
  return C;
}
```

Circulo circulo

```
vector<PT> CircleCInters(PT a, PT b, double r, double R) {
  vector<PT> ret;
  double d = sqrt(dist2(a, b));
  if (d > r+R || d+min(r, R) < max(r, R)) return ret;
  double x = (d*d-R*R+r*r)/(2*d);
  double y = sqrt(r*r-x*x);
  PT v = (b-a)/d;
  ret.push_back(a+v*x + RotateCCW90(v)*y);
  if (y > 0)
    ret.push_back(a+v*x - RotateCCW90(v)*y);
  return ret;
}
```

Distancia 3D

```
double distSP(P a, P b, P c) {
  P pp = a + (c-a)/(b-a);
  if( !comp(!(a-pp) + !(pp-b), !(a-b)) ) return !( c-pp );
  return min(!(a-c), !(b-c));
}

double distSS(P a, P b, P c, P d) {
  P ba = b-a;
  P cd = c-d;
  P ca = c-a;
  P w = ba%cd;
  if( !comp(w*w,0) ) {
    return min(min(distSP(a,b,c), distSP(a,b,d)),
               min(distSP(c,d,a), distSP(c,d,b)));
  }
  double dd = ba%cd*w;
  double x = (ca%cd*w)/dd;
  double y = (ba%ca*w)/dd;
  double z = (ba%cd*ca)/dd;
  if( x >= 0 && x <= 1 && y >= 0 && y <= 1 ) return !(w*z);
  return min(min(distSP(a,b,c), distSP(a,b,d)),
             min(distSP(c,d,a), distSP(c,d,b)));
}

double distPP(P a, P b, P c, P d) {
  P ba = b-a;
  P ca = c-a;
  P da = d-a;
  P w = ba%ca;
  P q = d-da/w;
  double x=(b-a)%(q-a)*w, y=(c-b)%(q-b)*w, z=(a-c)%(q-c)*w;
  if(x <= 0 && y <= 0 && z <= 0||x >= 0 && y >= 0 && z >= 0)
    return !(da/w);
  return min( min(distSP(a,b,d), distSP(b,c,d)),
              distSP(c,a,d));
}

int read() {
  fr(i,0,4) scanf("%lf%lf%lf", &p[i].x, &p[i].y, &p[i].z);
  fr(i,0,4) scanf("%lf%lf%lf", &q[i].x, &q[i].y, &q[i].z);

  double dist = 1./0.  ;
  fr(i,0,4) fr(j,i+1,4) fr(k,j+1,4) fr(l,0,4) {
    double d = distPP(p[i], p[j], p[k], q[l]);
    if( d < dist ) dist = d;
    d = distPP(q[i], q[j], q[k], p[l]);
    if( d < dist ) dist = d;
  }
  fr(i,0,4) fr(j,i+1,4) fr(k,0,4) fr(l,k+1,4) {
    double d = distSS(p[i], p[j], q[k], q[l]);
    if( d < dist ) dist = d;
  }

  printf("%.2lf\n", dist);

  return 1;
}
```

Convex Hull 3D

```
P p[M], co;
vector<int> face[M<<3];
int aresta[M][M], f, n;

bool comp2(P a, P b) {
  double d = a.x*b.y - a.y*b.x;
  return d > 0 || d == 0 &&
    a.x*a.x+a.y*a.y < b.x*b.x+b.y*b.y; }
bool comp3(P a, P b) {
  double d = a.x*b.y - a.y*b.x;
  return d >= 0;}
void ch2d(vector<int>& w, P X, P Y) {
  int n = w.size(); vector<P> q;
  fr(i,0,n) q.push_back(P(p[w[i]]*X, p[w[i]]*Y, w[i]));
  P o = *min_element(q.begin(), q.end());
  o.z = 0;
  fr(i,0,n) q[i] = q[i] - o;
  sort(q.begin(), q.end(), comp2);
  int m = 0;
  for( int i = 1 ; i < n ; ++i ) {
    while( m && comp3(q[i]-q[m-1], q[m]-q[m-1]) ) m--;
    q[++m] = q[i];
  }
  w.resize(++m);
  fr(i,0,m) w[i] = q[i].z+0.5;
}

void go(int a, int b) {
  if( ~aresta[a][b] ) return;
  P A = p[a], v = p[b]-A, w = co-A;
  vector<int> plano;
  fr(i,0,n) if( i != a && i != b ) {
    P u = v%(p[i]-A);
    if( u*w > 0 ) plano = vector<int>(1,i), w = p[i]-A;
    else if( u*w == 0 ) plano.push_back(i);
  }
  plano.push_back(a);
  plano.push_back(b);
  ch2d(plano, v, (co-A)%v);
  face[f++] = plano;
  int m = plano.size();
  fr(i,0,m) aresta[plano[i]][plano[(i+1)%m]] = f;
  fr(i,0,m) go(plano[(i+1)%m], plano[i]);
}

void convex() {
  sort(p,p+n); n = unique(p,p+n) - p;
  int a = min_element(p,p+n)-p, b = (a+1)%n;
  co = p[a];
  fr(i,0,n) if( i != a ) {
    co = co + p[i];
    P v = (p[i]-p[a])%(p[i]-p[b]);
    #define meo(a, b) ({ \
      P v = (a)%(b); \
      v.z != 0 ? v.z < 0 : \
      v.y != 0 ? v.y < 0 : \
      v.x != 0 ? v.x > 0 : !(a) > !(b); \
    })
    if( meo( p[i]-p[a], p[i]-p[b] ) ) b = i;
  }
  co = co*(1./n);
  fr(i,0,n) fr(j,0,n) aresta[i][j] = -1; f = 0;
  //memset(aresta,-1,sizeof aresta); f = 0;
  go(a,b); go(b,a);
}

int main() {
  int t = -1, caso = 1; cin >> t;
  while( t-- && cin >> n && n ) {
    fr(i,0,n) cin >> p[i].x >> p[i].y >> p[i].z;
    convex();
    double area = 0, vol = 0;
    fr(i,0,f) {
      int m = face[i].size();
      P a = p[face[i][0]];
      fr(j,2,m) {
        P b = p[face[i][j-1]];
        P c = p[face[i][j]];
        area += !((b-a)%(c-a))/2;
        vol += c%b*a/6;
      }
    }
    printf("%.6lf %.6lf\n", area, vol);
  }
}
```

LIS 2D

```
int n, m;
map<int,int> coast[MAX];

bool comp(pii pt, const map<int,int>& ct) { // ct > pt
  map<int,int>::const_iterator it = ct.lower_bound(pt.F);
  //if( it != ct.end() && it->F == pt.F ) se x1<=x2
  //return it->S >= pt->S; // > se y1<=y2
  if( it == ct.begin() ) return true;
  return (--it)->S >= pt.S; // > se y1<=y2
}
void add(map<int,int>& ct, int x, int y) {
  map<int,int>::iterator it = ct.lower_bound(x), jt = it;
  if( it != ct.end() && it->F == x && it->S < y )
    return; // sem se x1<=x2
  if( it != ct.begin() && (--it)->S == y )
    return; it = jt; // sem se y1<=y2
  while( jt != ct.end() && jt->S >= y ) ++jt;
  ct.erase(it,jt); ct[x] = y;
}
int main() {
  while( cin >> n ) {
    m = 0; // x1 < x2 && y1 < y2
    fr(i,0,n) {
      int x,y; cin >> x >> y;
      int t = upper_bound(coast, coast+m, pii(x,y), comp)-
              coast;
      if( t == m ) coast[m++].clear();
      add(coast[t], x,y); }
    printf("%d\n", m);
  }
}
```

Convex Hull 2D

```
bool comp1(P a, P b) {
  return a.x < b.x || a.x == b.x && a.y < b.y; }
bool comp2(P a, P b) {
  return a%b > 0 || a%b == 0 && a*a < b*b; }

vector<P> hull(vector<P> p) {
  int n = p.size();
  P o = *min_element(p.begin(), p.end(), comp1);
  fr(i,0,n) p[i] = p[i] - o;
  sort(p.begin(), p.end(), comp2);
  int w = n-1; while( w && p[w]%p[w-1] == 0 ) w--;
  reverse(p.begin()+w, p.end());// remover se sem colineares

  int m = 0;
  for( int i = 1 ; i < n ; ++i ) { // >= elimina colineares
    while( m && (p[i]-p[m-1])%(p[m]-p[m-1]) > 0 ) m--;
    p[++m] = p[i];  }
  p.resize(++m);
  fr(i,0,m) p[i] = p[i] + o;
  return p; }
```

DrawPolygon

```
import java.awt.*;
import java.util.*;
import javax.swing.*;
public class DPP extends JPanel {
  static Polygon pol = new Polygon();
  static int M = 25, E = 10, W = 800, H = 600;
  public void paint(Graphics g) {
    for (int i = 0; i < 50; i++) {
      g.drawLine(i*M+E, 0, i*M+E, H);
      g.drawLine(0, i*M+E, W, i*M+E); }
    g.setColor(Color.blue);
    ((Graphics2D)g).setStroke(new BasicStroke(3f));
    g.drawPolygon(pol); }
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    while( scan.hasNextInt() )
      pol.addPoint(E+scan.nextInt()*M,H-E-scan.nextInt()*M);
    JFrame frame = new JFrame();
    frame.setSize(W, H);
    frame.getContentPane().add(new DPP());
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

Problem	D	F	L	Assunto	Descrição	Idéia
A						
B						
C						
D						
E						
F						
G						
H						
I						
J						
K						
L						