Conhecimentos Básicos:

**Conteúdo**

[Problem Solving Paradigms:](#h.gjdgxs)

[∙ Ternary Search](#h.30j0zll)

[Strings:](#h.1fob9te)

[∙ From Infix to postfix (rpn) or prefix notation](#h.3znysh7)

[Programação Dinâmica:](#h.2et92p0)

[∙ Longest Arithmetic Progression Subsequence](#h.tyjcwt)

[Teoria dos Números:](#h.3dy6vkm)

[Geometria:](#h.1t3h5sf)

[∙ Ponto de Fermart](#h.4d34og8)

[∙ Interseção de duas retas](#h.2s8eyo1)

[Algebra Linear:](#h.17dp8vu)

[∙ Fórmulas explícitas para sistemas de 2 equações](#h.3rdcrjn)

[Combinatória:](#h.26in1rg)

[Probabilidade:](#h.lnxbz9)

[Física:](#h.35nkun2)

[∙ Equações dos movimentos retilíneos](#h.1ksv4uv)

# Problem Solving Paradigms:

## Ternary Search

Encontrar o ponto máximo de uma função com único máximo local ( ou seja, sendo x o ponto de máximo, a função é monotonamente crescente até x e depois monotonamente decrescente).Função abaixo para quando intervalo (left – right) for menor que EPS ou então depois de “cont” iterações.

double ter(double left, double right, int cont ){

if( right - left < eps || cont == 0)

return (left + right)/2.0;

double lt = ((2\*left) + right)/3.0;

double rt = (left + (2\*right))/3.0;

if( func(lt) < func(rt)) return ter(lt,right, cont--);

else return ter( left, rt , cont--);

}

# Strings:

## From Infix to postfix (rpn) or prefix notation

Algorithm to postfix (to transform to prefix just reverse the input string [ex: (P + Q \* C-D) / (E \* F) and it’s reverse (F \* E) / (D – C \* Q + P), you can do this reading the input backward], and output the inverse of the string returned by the procedure)

1. Clean out your stack

2. Get the next token from the expression.

3. If token is left paren, push it to the stack

4. If token is right paren, keep popping from stack and

appending to a Reverse Polish Notation string (or an RPN

vector, if you prefer) until you hit the ( symbol on the stack.

5. If token is an operator then:

5a. If stack is empty or operator is higher precedence than that on top of stack

then add operator to the stack

5b. If operator is lower precedence than the one on stack,

keep popping from

the stack and appending to a RPN string (or RPN vector), until you hit

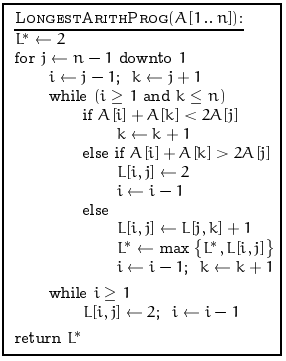
an operator with lower precedence.

6. If token is a number then simply append the number to the

RPN string (or the RPN vector)

# Programação Dinâmica:

## Longest Arithmetic Progression Subsequence

**Problema**: Dado um conjunto A[1..N] , ache o maior subconjunto de A onde seus elementos estejam numa progressão aritmética. O estado L[i][j] armazena o tamanho do maior subconjunto que inicia com os elementos A[i] e A[j]. L\* armazena a resposta. Para conseguir os elementos da sequência é só guardar no em uma matriz auxiliar onde AUX[i][j] tem valor de k para o caso em que A[i] + A[k] == 2\*[j]. Considerando o A[i], A[j] da última vez que L\* troca de valor, os primeiros elementos da sequencia, e os próximos elementos da sequencia serão A[AUX[i][j]] , A[AUX[j][AUX[i][j]]], etc.

# Teoria dos Números:

## Modular multiplicative inverse:

In [modular arithmetic](http://en.wikipedia.org/wiki/Modular_arithmetic), the **modular multiplicative inverse** of an [integer](http://en.wikipedia.org/wiki/Integer) *a* [modulo](http://en.wikipedia.org/wiki/Modular_arithmetic) *m* is an integer *x* such that

a\,x \equiv 1 \pmod{m}.

The multiplicative inverse of *a* modulo *m* exists [if and only if](http://en.wikipedia.org/wiki/Iff) *a* and *m* are [coprime](http://en.wikipedia.org/wiki/Coprime) (i.e., if [gcd](http://en.wikipedia.org/wiki/Greatest_common_divisor)(*a*, *m*) = 1).

### Extended Euclidean algorithm

### The modular multiplicative inverse of *a* modulo *m* can be found with the [extended Euclidean algorithm](http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm). The algorithm finds solutions to [Bézout's identity](http://en.wikipedia.org/wiki/B%C3%A9zout%27s_identity)

ax + by = \gcd(a, b)\,

where *a* and *b* are given and *x*, *y* and gcd(*a*, *b*) are the integers that the algorithm discovers. So, since the modular multiplicative inverse is the solution to

ax \equiv 1 \pmod{m},

by the definition of congruence, *m* | *ax* − 1, which means that m is a [divisor](http://en.wikipedia.org/wiki/Divisor) of *ax* − 1. This, in turn, means that

ax - 1 = qm.\,

Rearranging produces

ax - qm = 1,\,

with *a* and *m* given, *x* the inverse, and *q* an integer multiple that will be discarded. This is the exact form of equation that the extended Euclidean algorithm solves—the only difference being that gcd(*a*, *m*) = 1 is predetermined instead of discovered. Thus, *a* needs to be [coprime](http://en.wikipedia.org/wiki/Coprime) to the modulus, or the inverse won't exist.

This algorithm runs in time O(log(*m*)2), assuming |*a*| < *m*, and is generally more efficient than exponentiation.

### Using Euler's theorem

As an alternative to the extended Euclidean algorithm, Euler's theorem may be used to compute modular inverse:[[1]](http://en.wikipedia.org/wiki/Modular_multiplicative_inverse#cite_note-1)

According to [Euler's theorem](http://en.wikipedia.org/wiki/Euler%27s_theorem), if *a* is [coprime](http://en.wikipedia.org/wiki/Coprime) to *m*, that is, [gcd](http://en.wikipedia.org/wiki/Greatest_common_divisor)(*a*, *m*) = 1, then

a^{\varphi(m)} \equiv 1 \pmod{m}

where φ(*m*) is [Euler's totient function](http://en.wikipedia.org/wiki/Euler%27s_totient_function). This follows from the fact that *a* belongs to the [multiplicative group](http://en.wikipedia.org/wiki/Multiplicative_group_of_integers_modulo_n) (**Z**/*m***Z**)× [iff](http://en.wikipedia.org/wiki/If_and_only_if) *a* is [coprime](http://en.wikipedia.org/wiki/Coprime) to *m*. Therefore the modular multiplicative inverse can be found directly:

a^{\varphi(m)-1} \equiv a^{-1} \pmod{m}

In the special case when *m* is a prime, the modular inverse is given by the below equation as:

a^{-1} \equiv a^{m-2} \pmod{m}

This method is generally slower than the extended Euclidean algorithm, but is sometimes used when an implementation for modular exponentiation is already available.

## Composição de quadrados perfeitos:

Verifica se um determinado inteiro pode ser escrito como soma de dois ou três quadrados perfeitos. O teorema de Lagrange afirma que todos os números não negativos podem ser formados pela soma de quatro quadrados perfeitos.

bool three\_square(ll x){

while (!(x&3)) x>>=2;

return (x+1)&7; // 0 == nao eh formado por 3 quadrados

}

bool twosquare(ll x){

for(ll i = 2; i\*i <= x; ++i){

ll tot = 1;

while(x%i == 0){

tot \*= i; x /= i;

}

if(!((tot+1)&3)) return false;

}

return (x+1)&3;

}

# Geometria:

## Ponto de Fermart

é o ponto tal que a distância total dos três vértices do triângulo até esse ponto é a menor possível (i.e. a soma das distâncias deste ponto aos vértices é mínima

In order to locate the Fermat point of a triangle with largest angle at most 120°:

1. Construct two [equilateral triangles](http://en.wikipedia.org/wiki/Equilateral_triangle) on any of the three sides of the given triangle.
2. For each new [vertex](http://en.wikipedia.org/wiki/Vertex_(geometry)) of the equilateral triangle, draw a line from it to the opposite triangle's vertex.
3. The two lines intersect at the Fermat point.

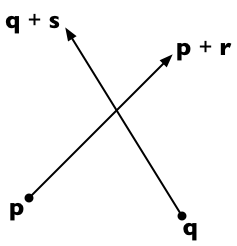
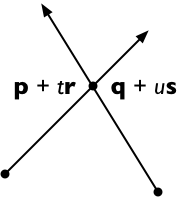
An alternate method is the following:

1. On any two of the three sides, construct two isosceles triangles, with base the side in question, 30-degree angles at the base, and vertices lying outside the original triangle.
2. Draw two circles, each with a center on the vertex of the just constructed isosceles triangles and radius the identical side of the isosceles triangles.
3. The intersection inside the original triangle between the two circles is the Fermat point.

When a triangle has an angle greater than 120°, the Fermat point is sited at the obtuse-angled vertex.

## Interseção de duas retas

Para duas retas A (no pronto p e vetor diretor r) e a reta B (no ponto q e vetor diretor s) temos o ponto de interseção está em: **p** + *t* **r** = **q** + *u* **s**, assim:

Se transforma em: 

Conseguimos derivar o t para encontrar o ponto de interseção:

***t* =** **(q** **−** **p) ×** **s** **/** **(r** **×** **s)** ou então ***u*** **=** (**q** − **p)** **×** **r** **/ (r** **×** **s)**

## Coordenadas cartesianas do circuncentro

Dado três pontos achar o centro do círculo onde os pontos estejam sobre seu perímetro. As coordenadas cartesianas do círculo são:

U_x = ((A_x^2 + A_y^2)(B_y - C_y) + (B_x^2 + B_y^2)(C_y - A_y) + (C_x^2 + C_y^2)(A_y - B_y)) / D,

U_y = ((A_x^2 + A_y^2)(C_x - B_x) + (B_x^2 + B_y^2)(A_x - C_x) + (C_x^2 + C_y^2)(B_x - A_x)) / D

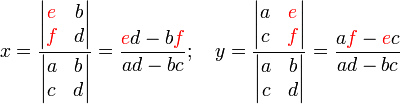
Onde D = 2( A_x(B_y - C_y) + B_x(C_y - A_y) + C_x(A_y - B_y)).\,

# Algebra Linear:

## Fórmulas explícitas para sistemas de 2 equações

    \begin{cases}        a{\color{blue}x} + b{\color{blue}y} = {\color{red}e}\\        c{\color{blue}x} + d{\color{blue}y} = {\color{red}f}    \end{cases} 

tem solução:



# Combinatória:

# Probabilidade:

# Física:

## Equações dos movimentos retilíneos

Equações do MRUV  
s=s_o+v_ot+\frac{at^2}{2} e \Delta s=\frac{v+v_o}{2}t *Equação de Torricelli*: v^2=v_o^2+2a\Delta s