**Programação Competitiva**

Passos : [página ref](http://lampiao.ic.unicamp.br/maratona/?name=preparacao)1.~~Escolher a linguagem.~~

* + ~~Dominar sintaxe e comandos básicos.~~
  + Dominar as funções mais usadas em competições.

1. Noções de Complexidade de Algoritmos
   * Calcular complexidade de tempo e memória.
   * Estimativa de tempo.
2. Dominar as estruturas de dados básicas.
   * Vetores, strings.
   * Pilha, fila.
3. Dominar Entrada e Saída
   * Leitura dos diferentes tipos.
   * Saída formatada dos diferentes tipos.
4. Ficar à vontade com Recursão e Backtracking.
   * Saber enumerar permutações, combinações e arranjos.
5. Aprender Grafos.
   * Estruturas para representá-los.
   * Algoritmos de busca: BFS, DFS.
   * Árvore Geradora Mínima.
   * Caminhos mínimos.
6. Aprender Programação Dinâmica.
   * Entender a idéia.
   * Conhecer os problemas clássico.

Anotações Gerais :

#include <bits/stdc++.h>

using namespace std;

int main(){

return 0;

}

tam vetor = sizeof(vetor)/sizeof(vetor[0])

swap(x,y) => troca x por y e y por x

sort(v, v+p) => ordena o vetor apontado por v da posição 0 até p

min(x,y) => retorna o menor valor entre x e y  
max(x,y) => retorna o maior valor entre x e y

reverse(v,v+p) => imprime ao contrário

template<typename T>

struct par{

T x,y;

}

se declararmos   
par<int> p; Substituimos T por int

par<double> q; Substituimos T por double

vector<int> v; Vetor de inteiros v vazio, melhor opção

v.push\_back(k); adiciona o elemento k no final do vetor

string s = “abacaba”

s += “é um palindromo”;

string s = “abacaba é um palíndromo”;

Stack/Pilha :

stack <tipodado> nome\_variavel;

stack <string> cartas;

cartas.size(); tamanho da pilha

cartas.push(“Rei de Copas”); adiciona elemento  
cartas.push(“A de espadas”);

cartas.pop(); retira o elemento do topo, no caso o “A de Espadas”

cartas.top(); imprime a carta no topo

cartas.empty(); retorna true se vazia e false caso contrário

Fila/Queue :

queue <tipodado> nome\_variavel;

queue <string> fila;

fila.size(); tamanho da fila

fila.push(“Thiago”); adiciona o thiago na fila  
fila.push(“Beltrano”); adiciona beltrano na fila

fila.pop() ;remove a pessoa na frente da fila (thiago)

fila.front(); imprime (não remove) a primeira carta

fila.back(); imprime o ultimo elemento d afila

Lista :

list <tipodado> nome\_variavel;

lista.size(); tamanho da fila

lista.push\_front(valor); insere na frente da lista

lista.push\_back(valor); insere na parte de trás da lista

lista.front(); imprime quem está no começo da lista

lista.back(); imprime quem está no final da lista

lista.pop\_front(); remove o elemento que está no começo da lista

lista.pop\_back(); remove o elemento que está no final da lista

lista.clear(); remove todos os elementos

lista.insert(posicao,valor); insere na posicao passada o valor passado

lista.merge(outra\_lista); remove os elementos da outra lista e os adiciona na lista original

list <int> lista(5,50); lista com 5 elementos preenchida com o numero 50

para receber uma linha de codigo contendo espaços e letras

string s;

getline(cin, s);

ou esse caso, recebe ate não ter mais nada para receber

while (cin >> x) {

// code

}

int : − 2^31 a 2^31 − 1 ou − 2 · 10^9 a 2 · 10^9 .

long long int : − 2^63 a 2^63 − 1 ou − 9 · 10^18 a 9 · 10^18 .

( a + b ) mod m = ( a mod m + b mod m ) mod m

( a − b ) mod m = ( a mod m − b mod m ) mod m

( a · b ) mod m = ( a mod m · b mod m ) mod m

Assim, podemos ir tirando o mod nos números, de tal modo que ele nunca sofrerá um overflow, e na resposta final, devemos tirar o mod da resposta final.

→ Fatorial de um número tirando mod m

long long x = 1;

for (int i = 2; i <= n; i++) {

x = (x\*i)%m;

}

cout << x%m << "\n";

Em C++ e outras linguagens, o resto de um número negativo é zero ou negativo, então d emodo a contornar tal detalhe, basta adicionarmos m ao final.

x = x%m;

if (x < 0) x += m;

Comparar dois números floats com == pode resultar em erros, numa vez que a precisão pode ser diferente entre eles. Uma maneira de contornar tal problema é definir um épslon tal que E = 10⁻⁹ e verificar se a subtração entre eles é menor que E. Se for, quer dizer que eles são iguais em uma precisão de até 9 casas, o que garante que possivelmente são iguais em todas as outras.

if (abs(a-b) < 1e-9) {

// a and b are equal

}

Somatorio indo de x=1 a x=n, 1+2+3+...+n = n(n+1)/2

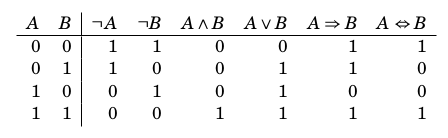
Somatorio indo de x=1 a x=n, 1²+2²+3²+...+n² = n(n+1)(2n+1)/6

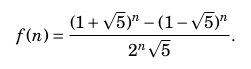
Somatorio PA indo de a até b com n números = n(a+b)/2

Somatorio PG indo de a até b com razao k = (bk-a)/k-1

Seja um conjunto S com K elementos , seu número de subconjuntos é 2^K

A^B = A e B AvB = A ou B ~A = Negação A



Formula termo geral fibonacci 

log(x) k é igual ao número de vezes que temos que dividir o x por k, até encontrarmos o valor 1

ex : log(32)2 = 5, porque temos que dividir 32 cinco vezes até chegarmos no 1

log(a)b = log de a na base b

k^a = x <=> log(x)k = a

log(ab)k = log(a)k + log(b)k

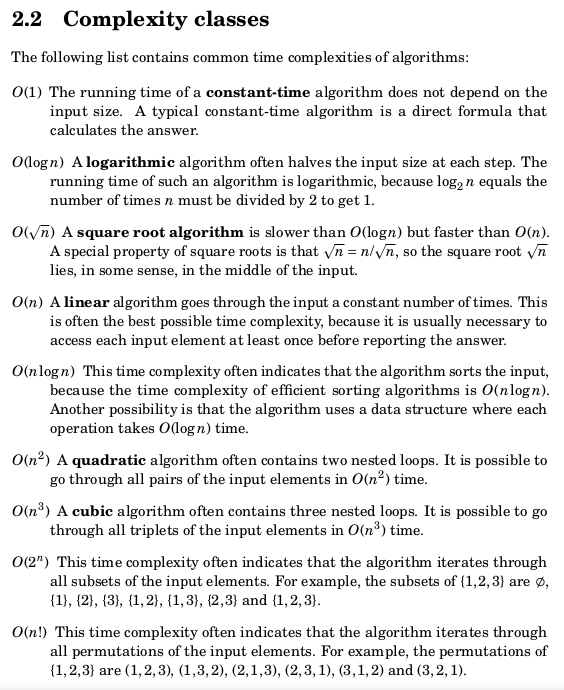
log(a/b)k = log(a)k – log(b)k

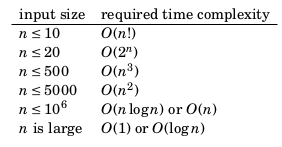
log(x^n)k = nlog(x)k

log(x)u = log(x)k/log(u)k

Número de digitos de um inteiro x na base b : log(x)b + 1 arrendondado pra baixo

ex : 123 na base 2 é 1111011 e log(123)2 + 1 = 7 (arredondado pra baixo)





Sort em vector: vector<int> v

ordem crescente => sort(v.begin(), v.end())

ordem decrescente => sort(v.rbegin(), v.rend())

Sort em string : string s = “thiago” é igual ao do vector sort(s.begin(), s.end())

Array Dinâmico : é um vetor no qual seu tamanho pode ser alterado durante a execução de um programa. → vector<tipo\_do\_dado> nome\_vetor;

ex: vector<int> v(10); 10 elementos com o valor 0

vector<int> v(10,5); 10 elementos com o valor 5

v.back() retorna o último elemento do vetor, v.push\_back() adiciona o elemento passado e v.pop\_back() remove o ultimo elemento do vetor.