Tópicos Avançados em Algoritmos

Hamilton José Brumatto

Bacharelado em Ciências da Computação - UESC

7 de fevereiro de 2019

Corretude e Eficência em Competição

Corretude e eficiência de algoritmos em Competições

- Revisão sobre Corretude e Eficiência em Algoritmos - PAA
- Eficiência em Competição.
- Prova de Corretude?
- A prática faz a perfeição. Só escreve correto quem escreve sempre.
- C++?! Por quê?

Árvore de Busca e Poda - Jogos

- Algoritmos de BackTracking - Árvore de Busca
- Técnicas de Poda
  - MinMax;
  - Antecipação Limitada;
  - Alfa-Beta

Algoritmos Genéricos

- Classes STL - Standard Template Library
- O que é programação genérica.
- O que são Classes Containers e como usá-las
- Exemplos básicos com estruturas simples: Vetores, Listas, Filas e Pilhas

Iterators

- Compreendendo Iterators e como utilizá-los para manipular listas
- Input, Output e Forward Iterators
- Categorias de Iterators
- Manipulando objetos em uma lista (sobrepondo operações através de iterators)
- Adaptors e Allocators

Estruturas de Sequência - 1

- Funções que são parâmetros:
  - Copy,
  - Foreach,
  - Sort,
  - Search,
  - BSearch, ...

Estruturas de Sequência - 2

- Vectors
- Deques
- Lists
- Stack
- Queue
- Priority Queue

Teoria dos Números

- Noções de teoria elementar dos números
- Números primos
- Bases e Conversão de Bases
- Bignums

Teoria dos Números

- Máximo Divisor Comum
- Minimo Múltiplo Comum
- Aritmética Modular
- Podências de um número

Geometria Computacional

- Modelagem: Ponto, reta, segmento.
- Propriedades de Segmentos de linha.
- Par mais próximo.
- Localização de um ponto.

Retas

- Interseção de retas e segmentos
- Poligonos e Ângulos
- Envoltória Convexa

Conjuntos

- Conjuntos e Implementação
- Operações e Relações sobre Conjuntos
- Strings e Substrings

Conjuntos

- Cobertura em Conjuntos
- Conjuntos Disjuntos e Operações
- Florestas de Conjuntos disjuntos

Árvores

- Implementação de árvores
- Pair
- Algoritmos em Árvores

Algoritmos em Árvores

- Árvores de Segmentos
- Árvores de Indexação Binária

Grafos

- Implementação de Grafos
- Manipulação de Grafos
- Propriedade de grafos
- Grafos de arestas ponderadas

Algoritmos em Grafos

- BFS (busca em Largura)
- Vizinhos mais próximos
- DFS (busca em profundidade)
- Grafos Cíclicos e Acíclicos
- Ordenação topológica (Fonte e Sorvedouro)
- Componentes Fortemente Conexos

Algoritmos em Grafos

- Caminhos Mínimos de única origem
- Caminhos Mínimos de todas origens
- Árvore Geradora Mínima (Floresta de Conjuntos)

Redes

- Corte em Grafos
- Redes e Fluxos em Redes
- Fluxo Máximo e Corte Mínimos
- Algoritmo e implementação

Grafos Planares

- Grafos planares e representação planar
- Teorema da Curva de Jordan
- Dualidade
- Fórmula de Euler
- Teorema de Kuratowski
- Algoritmo de Reconhecimento de Planaridade de Grafos

Coloração em Grafos

- Coloração de Grafos Planares
  - Coloração de Faces
  - Coloração de Vértices
  - Coloração de Arestas
- O teorema da coloração de 5 cores
- A coloração de 4 cores

Emparelhamento em Grafos Bipartidos

- Grafos Bipartidos
- Emparelhamento em Grafos Bipartidos
  - Teorema de Hall
  - Emparelhamento e Cobertura
- Algoritmos de Emparelhamento
  - Caminhos Aumentantes
  - Algoritmo de Egerváry

Atividades e Critérios

- Atividades valendo nota:
  - Listas de Exercícios Práticos (P)
  - Competição na CPU/PROGBASE (CPU)
  - Maratonas Mensais (MM)
- Atividades Obrigatórias:
  - Presença!!!
- Critérios:
  - Nota Semestral: (MM + P + CPU)/3
  - Presença obrigatória: 75%

Datas Importantes

- CPU/PROGBASE: 6/5
- Maratonas Mensais: 20/3, 17/4, 6/5(CPU), 19/6
- Avaliação Final: 3/7
- Datas sem encontro presencial:
  - Feriados: 4/3, 6/3, 22/4, 1/5, 24/6
  - : ERBASE: 6/5 (CPU, presença obrigatória) e 8/5
  - Emendas: 18/3, 26/6, 1/7

Bibliografia

- LEISERSON, Charles E. **Algoritmos** – Trad. 2ª Ed. Americana., Editora Campus, 2002.
- SKIENA, Steven S. **The Algorithm Design manual.** Telos / Springer – Velag, 1997.
- SKIENA, Steven S. e Revilla, Miguel A. **Programming Challenges - The Programming Contest Training Manual**. Springer 2002.
- BONDY, J. A e MURTY, U. S. R. **Graduate Texts in Mathematics - Graph Theory**. Springer 2008.
- MUSSER, David R. e Saini, Atul. **STL Tutorial and Reference Guide - C++ Programming with the Standard Template Library**. Addison-Wesley Publishing Company, 2nd Ed. 1996

Código em competição

- Um código para competição não é necessariamente um código para reuso, logo toda a parte sobre criar um código legível e organizado pode ser posta de canto, o objetivo é a rapidez e a corretude.
- Nada impede que após o término da competição não vale a pena rever o código e organizá-lo de forma a ficar legível para posterior uso.
- Na competição não é preciso se preocupar com a segurança do código, ou com o mau uso que um usuário irá destinar ao código, logo não é necessário um conjunto de testes nos valores de entrada.

Código em competição

- Da mesma forma o código não precisa rodar em um *SandBox* que garanta uma execução segura para a máquina (Ele irá rodar automaticamente em um SandBox na máquina do juiz), assim a linguagem escolhida deve ser a mais simples, sem desnecessários *tries* e *catches*
- De preferência utilize a linguagem que mais ofereça recursos de bibliotecas prontas para que você não precise escrever a biblioteca.
- Vamos adotar o C++

Entrada e Saída

- Vamos pegar um problema bem simples:

Problema URI 1003 - Soma Simples

Leia dois valores inteiros, no caso para variáveis A e B. A seguir, calcule a soma entre elas e atribua à variável SOMA. A seguir escrever o valor desta variável.
Entrada: O arquivo de entrada contém 2 valores inteiros.
Saída: Imprima a variável SOMA com todas as letras maiúsculas, com um espaço em branco antes e depois da igualdade seguido pelo valor correspondente à soma de A e B.

Entrada e Saída

Solução: ERRADA

```
#include <iostream>
using namespace std;

int main() {
  int n1, n2;

  cout <<"Digite 2 números:  "<<endl;
  cin >>n1 >>n2;
  cout << "A soma é:  "<<(n1 + n2) <<endl;

  return 0;
}
```

Entrada e Saída

Solução: CORRETA

```
#include <iostream>
using namespace std;

int main() {
  int n1, n2;

  cin >>n1 >>n2;
  cout << "SOMA = "<<(n1 + n2) <<endl;

  return 0;
}
```

Entrada e Saída

- Não precisa pedir os valores de entradas, é só ler, eles estarão lá.
- Só pode mandar imprimir aquilo que é informado, na forma que é especificada.
- Não precisa testar as entradas, se é dito que a entrada é um inteiro n, com $0 \leqslant n \leqslant 10$, então não haverá, com certeza, nenhum número negativo ou maior que 10 sendo fornecido.
- A proposta é construir um algoritmo que resolva aquele problema apresentado em específico, não precisa resolver nenhum outro problema, pois inexiste.

Limites

- Muita atenção aos limites, pois os limites que são indicados na entrada pode estabelecer o tipo numérico que você deve utilizar.
  - Se a entrada é um inteiro $0 \leqslant n \leqslant 10^5$ você pode usar o tipo `int`
  - Se a entrada é um inteiro $0 \leqslant n \leqslant 10^{12}$ você deve usar o tipo `long long int`
- Também cuidado com o tamanho da alocação estática. Um vetor pode ocupar no máximo 1 milhão de inteiros, se passar disto haverá erro de alocação de memória

Problema 2534 do URI Online Judge: Exame Geral

Todo ano bissexto é realizado o exame geral de matemática da Nlogônia. Todos os cidadões da nação são avaliados a fim de se estudar o desenvolvimento lógico e matemático do país ao longo dos anos. Após as correções, os cidadões são ordenadados de acordo com suas notas (quanto maior, melhor) e recebem descontos no imposto de renda de acordo com sua qualificação. O Escritório Central de Estatística (ECE) é encarregado de processar os dados das notas obtidas no exame. Entretanto este ano, Vasya, um dos responsáveis, está internado no hospital com gripe H1N1 e você foi contratado para realizar o seu trabalho. Escreva um programa que dado o número de habitantes da Nlogônia e todas as notas obtidas, responda as consultas para retornar a nota do cidadão que ficou em determinada posição.

Entrada:

A entrada contém vários casos de teste. A primeira linha de cada caso contém dois inteiros **N** ($1 \leqslant \mathbf{N} \leqslant 100$), **Q** ($1 \leqslant \mathbf{Q} \leqslant 100$), o número de habitantes do país e o número de consultas, respectivamente.
As **N** linhas seguintes contém, cada uma, a nota $\mathbf{n}_i$ obtida pelo i-ésimo cidadão ($0 \leqslant \mathbf{n}_i \leqslant 30000$).
As próximas **Q** linhas contém cada uma uma consulta, a posição $\mathbf{p}_i$ ($1 \leqslant \mathbf{p}_i \leqslant \mathbf{N}$) a qual a ECE está interessada em saber a nota.
A entrada termina com fim-de-arquivo (EOF).

Saída:

Para cada caso de teste, imprima, para cada consulta, uma linha contendo a nota do cidadão que ficou classificado na posição $\mathbf{p}_i$.

Solução:

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main () {
 vector<int> v;
 int n, q, k;
 while(cin >> n >> q) {
  for(int i=0; i<n; i++) { cin >> k; v.push_back(k); }

  sort(v.rbegin(),v.rend());

  for(int i=0; i<q; i++) { cin >> k; cout << v[k-1] << endl; }
  v.clear();
 }
 return 0;
}
```