

ALGORITMOS II

7^a LISTA DE EXERCÍCIOS ALGORITMOS DE ORDENAÇÃO

- 1 Ordene o vetor $v = [20, 12, 28, 05, 10, 18]$ usando o Método de Inserção. Mostre o vetor a cada iteração.

	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	i
v	20	12	28	05	10	18	0

- 2 Ordene o vetor $v = [20, 12, 28, 05, 10, 18]$ usando o Método de Seleção. Mostre o vetor a cada iteração.

	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	i
v	20	12	28	05	10	18	0

- 3 Ordene o vetor $v = [20, 12, 28, 05, 10, 18]$ usando o Método da Bolha (Bubble Sort). Mostre o vetor a cada iteração.

	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	i
v	20	12	28	05	10	18	0

4 A seguinte rotina apresenta um algoritmo de ordenação por inserção:

```
int Insercao (int n, float *vet) {  
    float aux;  
    int i, j;  
    for (i = 1; i < n; i++) {  
        aux = vet[i];  
        j = i - 1;  
        while (j >= 0 && aux < vet[j] ) {  
            vet[j+1] = vet[j];  
            j = j - 1;  
        }  
        vet[j+1] = aux;  
    }  
}
```

Reescreva esta rotina, para que a ordenação ocorra de forma decrescente.

5 Problema: FLIPERAMA

Bebe-bebe é um jogo muito popular de fliperama. E, como a maioria dos jogos de fliperama, ele deve mostrar as maiores pontuações. Para esse fim, a companhia Otori te contratou.

Escreva um programa que, dada a lista de todas as pontuações dos jogos de Bebe-bebe, mostra os melhores placares em ordem decrescente.

Entrada

A entrada é composta de um único caso de teste. A primeira linha consiste de dois inteiros N e M, dizendo quantas partidas foram jogadas de Bebe-bebe e quantas linhas cabem no mostrador de melhores rankings. As N linhas seguintes contem cada uma um inteiro indicando a pontuação obtida em cada jogo.

Saída

Seu programa deve imprimir M linhas, contendo as M maiores pontuações em ordem decrescente.

Restrições

$1 \leq N \leq 10000$

$1 \leq M \leq 500$

$M \leq N$

Exemplo

Entrada

7 4

100

200

200

150

30

524

942

Saída

942

524

200

200

6 Problema: ELEIÇÕES

O prefeito de Piraporinha do Sul foi afastado de seu cargo, devido a acusações de corrupção em contratos da prefeitura, e por isso foram convocadas novas eleições para prefeito. Procurando uma renovação política, a comissão eleitoral permitiu que mesmo candidatos de fora da cidade concorressem ao cargo de prefeito.

Devido a essa nova regra, houve uma quantidade muito grande de candidatos à prefeitura. O software da comissão eleitoral de Piraporinha do Sul não estava preparado para isso, e por isso você foi contratado para escrever um programa que, dados os votos lançados pelos habitantes da cidade, decide qual candidato ganhou.

Entrada:

A entrada é composta de um único caso de teste. A primeira linha contém um inteiro N representando o número de votos. Nas próximas N linhas, haverá um inteiro X_i , que representa o i -ésimo voto (os candidatos são identificados por inteiros).

Saída:

Para cada conjunto de teste da entrada seu programa deve produzir uma única linha, contendo o número do candidato que venceu (aquele que obteve mais votos). Você pode supor que existe apenas um vencedor.

Restrições:

$$1 \leq N \leq 100000$$

$$1 < X_i \leq 1000000000$$

Exemplo:

Entrada:

5

1000

1000

2588

4000

2587

Saída:

1000

Resolução:

Esse problema apesar de, a princípio parecer não precisar de ordenação, pode ser resolvido de forma eficiente usando um algoritmo de ordenação. A primeira ideia que vem à cabeça é simplesmente armazenar a quantidade de votos de cada candidato em uma posição de um vetor e ir vendo quem está com mais votos. O problema é que podem haver até 10^9 candidatos, e não é possível declarar um vetor com esse tamanho.

Porém, apesar de o número de candidatos ser muito grande, o número de votos é bem menor (10^6). Então a solução usando ordenação consiste em ordenar os votos e contar a maior sequência de valores iguais, que será o candidato com mais votos e portanto o vencedor.

- 7** Faça uma função que ordene de forma crescente o vetor [9,8,7,6,5,4,3,2,1] e retorne o número de alterações necessárias para tal utilizando o *Bubble Sort*.

- 8 João diz ter desenvolvido um algoritmo que é capaz de ordenar qualquer conjunto de n números reais, fazendo apenas $O(n^{3/2})$ comparações. Você compraria este algoritmo? Justifique.
- 9 Um amigo lhe diz que é capaz de ordenar qualquer conjunto de 6 números com no máximo 8 comparações. O seu amigo está falando a verdade ou mentindo? Justifique.
- 10 Uma **ordenação por contagem** de um vetor x de tamanho n é executada da seguinte forma: declare um vetor *count* e defina $count[i]$ como o número de elementos menores que $x[i]$. Em seguida, coloque $x[i]$ na posição $count[i]$ de um vetor de saída (leve em consideração a possibilidade de elementos iguais). Escreva uma função para ordenar um vetor x de tamanho n usando esse método.
- 11 Presuma que um vetor contém inteiros entre a e b , inclusive, com vários números repetidos diversas vezes. Uma **ordenação por distribuição (BucketSort)** ocorre da seguinte maneira: declare um vetor *number* de tamanho $b - a + 1$, defina q como o número de vezes que o inteiro i aparece no vetor e, em seguida, redefina os valores no vetor concomitantemente. Escreva uma função para ordenar um vetor x de tamanho n contendo inteiros entre a e b , inclusive, com esse método. Escreva um programa que utilize o programa desenvolvido.
- 12 A ordenação por **transposição de par-ímpar** ocorre da seguinte maneira. Percorra o vetor várias vezes. Na primeira passagem compare $x[i]$ com $x[i+1]$ para todo i ímpar. Na segunda passagem compare $x[i]$ com $x[i+1]$ para todo i par. Toda vez que $x[i] > x[i+1]$ troque os dois. Continue alternando dessa maneira até que o vetor esteja ordenado.
- Qual a condição para o término da ordenação?
 - Escreva uma função para implementar essa ordenação?
 - Qual é o custo médio dessa ordenação?
- 13 Escreva um programa que imprima todos os conjuntos de seis inteiros positivos, a_1, a_2, a_3, a_4, a_5 e a_6 , tais que:

$$\begin{aligned} a_1 &\leq a_2 \leq a_3 \leq 20 \\ a_1 &< a_4 \leq a_5 \leq a_6 \leq 20 \end{aligned}$$

e a soma dos quadrados de a_1, a_2 e a_3 seja igual à soma dos quadrados de a_4, a_5 e a_6 .

(Dica: Gere todas as somas possíveis de três quadrados e use um procedimento de ordenação para localizar repetições).

- 14 A **ordenação por inserção intercalada** é a seguinte:

Passo 1: Para todo i par entre 0 e $n-2$, compare $x[i]$ a $x[i+1]$. Posicione o maior na próxima posição de um vetor *large* e o menor na próxima posição de um vetor *small*. Se n for ímpar, posicione $x[n-1]$ na última posição do vetor *small*. (*large* é de tamanho ind , onde $\text{ind} = (n - 1)/2$; *small* é de tamanho ind ou $\text{ind}+1$, dependendo de n ser ímpar ou par.).

Passo 2: Ordene o vetor *large* usando a inserção intercalada recursivamente. Sempre que

um elemento $large[j]$ for transferido para $large[k]$, $small[j]$ será também movido para $small[k]$. (No final desse passo, $large[i] \leq large[i+1]$ para todo i menor que ind , e $small[i] \leq large[i]$ para todo i menor ou igual a ind).

Passo 3: Copie $small[0]$ e todos os elementos de $large$ em $x[0]$ a $x[ind]$.

Passo 4: Defina o inteiro $num[i]$ como $(2^{i+1} + (-1)^i)/3$. Começando com $i = 0$ e continuando de 1 em 1 enquanto $num[i] \leq (n/2) + 1$, insira os elementos $small[num[i+1]]$ até $small[num[i]+1]$ em x , por vez, usando a inserção binária. (Por exemplo, se $n = 20$, os sucessivos valores de num são $num[0] = 1$, $num[1] = 1$, $num[2] = 3$, $num[3] = 5$ e $num[4] = 11$, que é igual a $(n/2) + 1$. Dessa forma, os elementos de $small$ serão inseridos na seguinte ordem: $small[2]$, $small[1]$; em seguida, $small[4]$, $small[3]$; depois $small[9]$, $small[8]$, $small[7]$, $small[6]$, $small[5]$. Neste exemplo, não existe $small[10]$).

Escreva uma função para implementar essa técnica.

- 15** Considere a seguinte **ordenação por seleção quadrática**: divida os n elementos do vetor em \sqrt{n} grupos de \sqrt{n} elementos cada. Encontre o maior elemento de cada grupo e insira-o num vetor auxiliar. Encontre o maior elemento nesse vetor auxiliar. Esse será o maior elemento do vetor. Em seguida, substitua esse elemento dentro do vetor pelo maior elemento seguinte do grupo a que ele pertencia. Ache novamente o maior elemento do vetor auxiliar. Esse será o segundo maior elemento do vetor. Repita o processo até que o vetor esteja classificado. Escreva uma função para implementar uma ordenação por seleção quadrática o mais eficiente possível.