

ESCOLA DE PRIMAVERA DA MARATONA SBC DE PROGRAMAÇÃO



PROMOÇÃO:



APOIO:



Grupo de Computação Competitiva

ORDENAÇÃO COM SORT



Por: *Gabriel Gallegos*
Ribeiro


CONTEÚDOS

- 01 - Problema motivador
- 02 - Definição do algoritmo
- 03 - Funcionamento do algoritmo
- 04 - Algoritmo
- 05 - Resolução do problema
- 06 - Outras aplicações
- 07 - Problemas

01 - PROBLEMA MOTIVADOR

beecrowd | 2316

Autorama

Por OBI - Olimpíada Brasileira de Informática 2006  Brazil

Timelimit: 1

Seu Diniz possui uma pista de autorama profissional. Nessa pista a marcação de tempo é feita com sensores que fazem leitura da passagem de cada cada carrinho pelo ponto onde o sensor está instalado. K sensores são distribuídos ao longo da pista nos chamados postos de checagem.

Durante uma corrida, os carrinhos devem passar pelos postos de checagem na ordem pré-estabelecida, ou seja, primeiro no posto de checagem 1, depois no 2, até o posto de checagem K, quando ele deve retornar ao posto de checagem 1 para completar uma volta. Entretanto, às vezes, quando os carrinhos saem da pista os competidores os recolocam mais à frente na pista, pulando alguns postos de checagem. Nesse caso, todas as passagens daquele carrinho por postos de checagem devem ser ignoradas até que ele passe pelo posto de checagem correto.

A posição de um carrinho na corrida é determinada pelo número de postos de checagem que ele passou na ordem correta. Caso dois carrinhos tenham passado pelo mesmo número de postos de checagem, a ordem utilizada é a ordem cronológica, ou seja, está mais à frente o carrinho que passou pelo último posto de checagem primeiro.

A pista de autorama do Seu Diniz possui um computador central que recebe os sinais lidos pelos sensores, mas ainda não possui um programa que permita determinar a posição dos carrinhos ao final da corrida.

Escreva um programa que, dado uma lista de leituras feitas pelos sensores, determine a classificação dos carrinhos na corrida.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do dispositivo de entrada padrão (normalmente o teclado). A primeira linha da entrada contém três inteiros, **K**, **N** e **M**. **K** representa o número de postos de checagem ($1 \leq K \leq 100$), **N** o número de carrinhos ($1 \leq N \leq 100$) e **M** o número de leituras feitas pelos sensores ($1 \leq M \leq 10000$). Os carrinhos são identificados por inteiros de 1 a **N** e os postos de checagem por inteiros de 1 a **K**. As **M** linhas seguintes contém cada uma dois inteiros **X** e **Y**, separados por espaço. Eles indicam que o carrinho número **X** ($1 \leq X \leq N$) passou pelo posto de checagem **Y** ($1 \leq Y \leq K$). Os eventos são apresentados na ordem cronológica. Sempre é possível determinar a classificação de todos os pilotos com os dados fornecidos.

Saída

Seu programa deve imprimir, na saída padrão, uma linha contendo **N** inteiros, sendo que o *i*-ésimo inteiro representa o carrinho que ocupa a posição *i* na corrida. Ou seja, o primeiro inteiro é o que ocupa o primeiro lugar, o segundo inteiro é o carrinho que ocupa o segundo lugar, e assim por diante.

Cada inteiro *I* contendo o número do carrinho que ocupa a posição de número *I* na corrida: o primeiro colocado ocupa a posição de número 1, o segundo colocado a posição de número 2, etc.

Exemplos de Entrada	Exemplos de Saída
<pre>3 3 6 3 1 1 1 2 1 3 2 3 3 2 2</pre>	<pre>3 2 1</pre>
<pre>2 2 5 2 1 2 1 1 1 2 1 1 2</pre>	<pre>1 2</pre>

02 - DEFINIÇÃO DO ALGORITMO

-A ordenação é basicamente pegar uma sequência fora de ordem e colocá-la na ordem desejada.

8	6	3	9	15	2
---	---	---	---	----	---

2	3	6	8	9	15
---	---	---	---	---	----

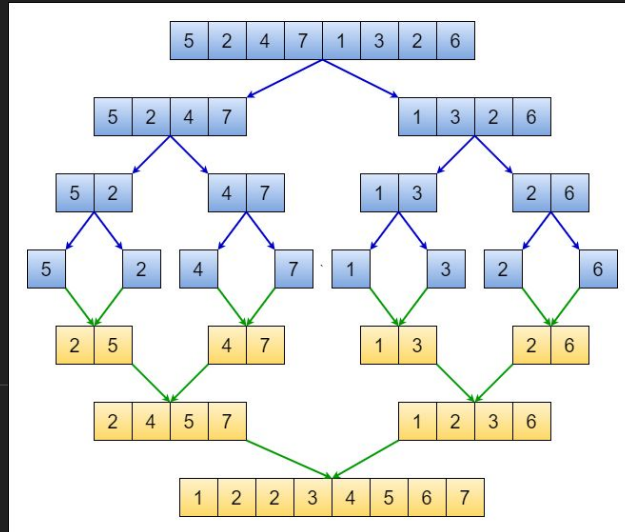
03 - FUNCIONAMENTO DO ALGORITMO

-O algoritmo de sort pode ser feito de várias maneiras, utilizando diversos algoritmos.

-Existem algoritmos de fácil implementação, porém menos eficientes de complexidade $O(n^2)$, e algoritmos de difícil implementação, mas de alta eficiência, que são de complexidade $O(n \log n)$.

-Para representação vamos mostrar os exemplos do bubblesort(de complexidade $O(n^2)$) e o mergesort, de complexidade $O(n \log n)$

03.1 - MERGESORT

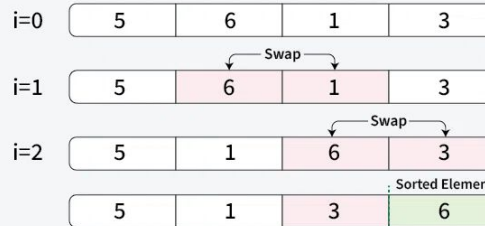


03.2 - BUBBLESORT

```
static void bubbleSort(int[] arr) {
    int n = arr.length;
    int temp = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 1; j < (n - i); j++) {
            if (arr[j - 1] > arr[j]) {
                temp = arr[j - 1];
                arr[j - 1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

01
Step

Placing the 1st largest element at its correct position



Bubble sort

03 - FUNCIONAMENTO DO ALGORITMO

Mesmo tendo todos esses tipos de sort, a linguagem C++ possui uma função própria, que ordena o algoritmo de forma eficiente, usando o insertion sort para casos pequenos e o quicksort para casos grandes.

-A função pode ser auxiliada por uma função ordena, a qual o usuário pode escolher como ordenar a estrutura em questão.

primeiro menor que segundo quando crescente e segundo menor que primeiro decrescente

04 - ALGORITMO











```
int main(){  
    vector<int> v(5,0);  
    for(int i=0;i<5;i++){  
        cin>>v[i];  
    }  
    sort(v.begin(),v.end(),compara);  
    for(int i=0;i<5;i++){  
        cout<<v[i]<<" ";  
    }  
}
```

```
bool compara(int x, int y){  
    if(x<y)  
        return true;  
    else  
        return false;  
}
```

04 - ALGORITMO

```
bool compara(int x, int y) {  
    if (voltas[x] == voltas[y]) {  
        if (posto[x] == posto[y]) {  
            if (tempo[x] == tempo[y]) return x < y;  
            return tempo[x] < tempo[y];  
        }  
        return posto[x] > posto[y];  
    }  
    return voltas[x] > voltas[y];  
}
```

04 - ALGORITMO

Club	MP	W	D	L	GF	GA	GD	Pts	Last 5
1  Atlético Mineiro	38	26	6	6	67	34	33	84	— ✓ ✓ ✓ ✓ ✗
2  Flamengo	38	21	8	9	69	36	33	71	— ✓ — ✗ ✗
3  Palmeiras	38	20	6	12	58	43	15	66	✗ — ✓ — ✓
4  Fortaleza	38	17	7	14	44	45	-1	58	✓ ✗ ✓ ✗ ✓
5  Corinthians	38	15	12	11	40	36	4	57	✓ ✗ ✓ — ✗
6  Bragantino	38	14	14	10	55	46	9	56	✗ — ✗ ✗ ✓
7  Fluminense	38	15	9	14	38	38	0	54	✓ ✓ ✗ ✗ ✓
8  América-MG	38	13	14	11	41	37	4	53	✗ — ✓ — ✓
9  Atlético Goian...	38	13	14	11	33	36	-3	53	— ✓ ✓ ✓ ✓
10  Santos	38	12	14	12	35	40	-5	50	✗ ✓ — ✓ —

05 - RESOLUÇÃO DO PROBLEMA MOTIVADOR

-O problema motivador envolve classificar os competidores em uma corrida de Kart, na qual existem pontos de checagem para saber onde cada piloto está na pista, e ao passar por todos os pontos de checagem se completa uma volta.

-A questão está em que se os competidores estiverem na mesma volta, deve-se checar em que ponto eles estão, e se estão no mesmo, checar quem chegou a esse ponto primeiro, ou seja, uma comparação de 3 características.

05 - RESOLUÇÃO DO PROBLEMA MOTIVADOR

06 - OUTRAS APLICAÇÕES

-Existem problemas que podem usar algoritmos de sort como auxiliares à resolução, como exemplo problemas que pedem em quantos passos se pode ordenar uma sequência, não necessariamente é necessária a sequência ordenada, mas sim quantos passos foram necessários para deixá-la assim, é um exemplo fictício e simples, mas serve para ilustrar a idéia.

07 - PROBLEMAS

Título 1 - <https://judge.beecrowd.com/pt/problems/view/2433>

Título 2 - <https://judge.beecrowd.com/pt/problems/view/2381>

Título 3 - <https://judge.beecrowd.com/pt/problems/view/1802>

OBRIGADO PELA ATENÇÃO

Grupo de Computação Competitiva

