Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação
Departamento de Ciências de Computação
SCC0201 - Introdução à Ciência de Computação II

2022

Professor: Fernando Pereira dos Santos
Monitores:
Adrio Oliveira Alves
Gabriel Natal Coutinho
João Guilherme Jarochinski Marinho

2° Trabalho prático

Motivação:

Ordenar alunos por tabela de notas.

Objetivo do trabalho:

Desenvolva um programa em C para ordenar os alunos por média das notas decrescentes e assim encontrar os melhores alunos da turma.

No entanto, se ocorrer empate, leva-se em consideração para desempate a nota da primeira avaliação, se empatar de novo a 2ª avaliação, depois a 3ª e assim por diante.

Entrada:

- Número de alunos (int)
- Número de avaliações que os alunos fizeram (int)
- Tabela de notas
    - Nome do aluno (string) e notas (int) separadas por ‘ ‘

Saída:

Ranking de alunos ordenados com a maior média da turma e qual avaliação foi necessária para distingui-lo do próximo colocado (‘media’ ou ‘desempate: nota x’).

Exemplo:

ALUNO	NOTA1	NOTA2	NOTA3
Adrio	6	5	5
Natal	5	4	7
Pato	5	7	4

Input:

```
3 3
Adrio 6 5 5
Natal 5 4 7
Pato 5 7 4
```

Output:

```
Maior media: 5,333
1. Adrio - desempate: nota 1
2. Pato - desempate: nota 2
3. Natal
```

Algoritmo de ordenação:

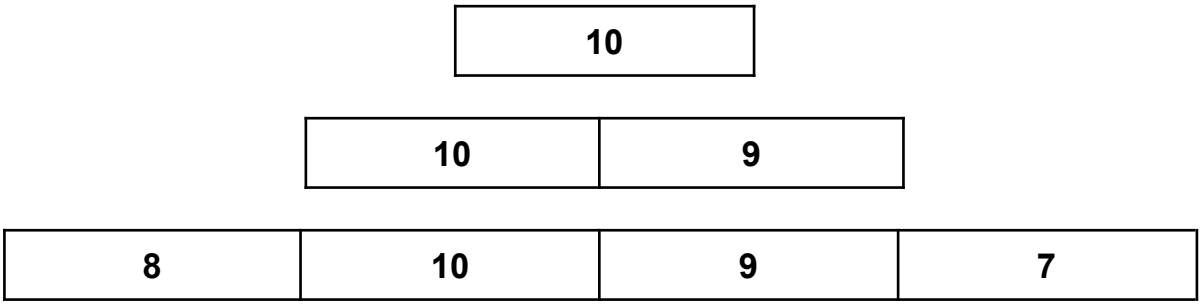
O algoritmo de ordenação desse trabalho deverá seguir os passos do mais novo: "***torneioGeralSort***".

O *torneioGeralSort* se baseia em organizar uma árvore binária de torneio em que o elemento pai é o vencedor do torneio dos filhos, em que vencer significa ser o maior dos competidores.

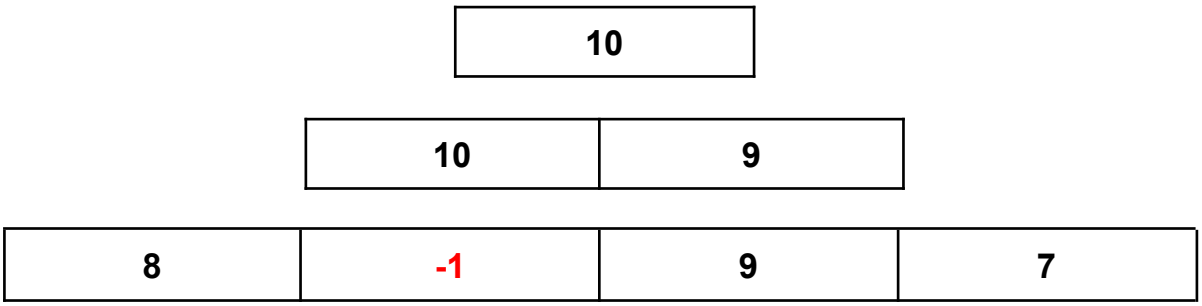
Primeiro é necessário preencher as folhas da árvore com os valores que se deseja ordenar (8,10,9,7).

Depois divida em pares e copie o maior de cada par como o pai desses elementos.

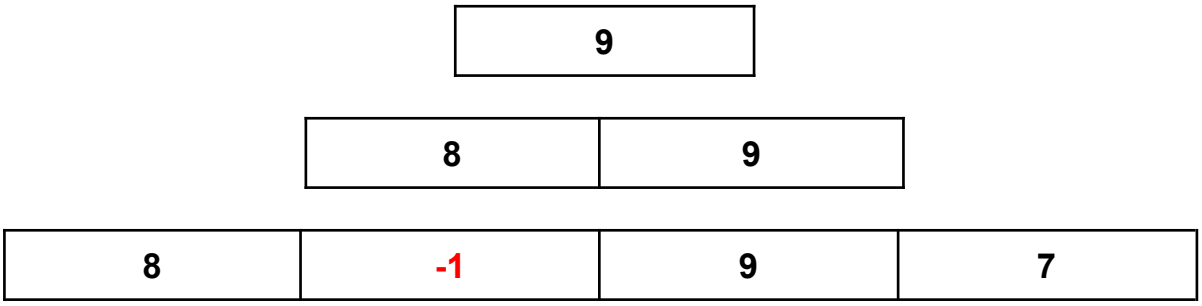
A estrutura da árvore pode ser um vetor em que a posição ***i*** é pai das posições ***2i*** e ***2i+1*** (recomendo não utilizar o índice 0 do vetor para isso. Portanto, a raiz da árvore é o índice 1).

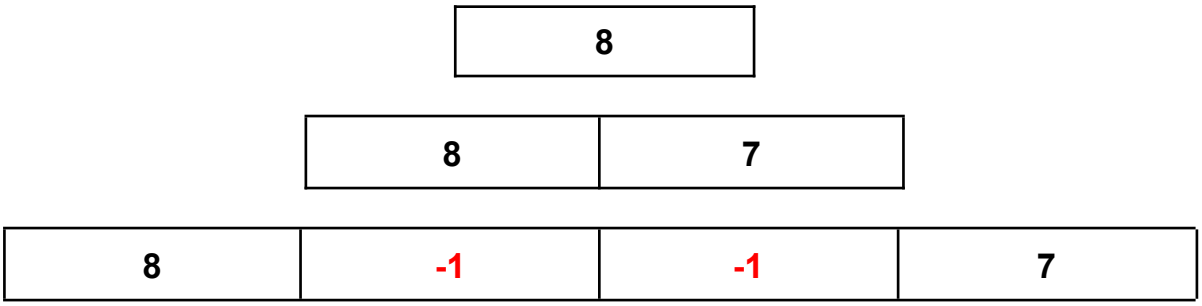


Depois de preencher toda a árvore, vamos retirar o primeiro elemento e colocar como primeiro elemento do novo vetor ordenado. Para isso, preenchemos a posição em que ele é filho com ***-1*** e organizamos a árvore de novo.



E repetimos esse passo de tirar o maior pai até a árvore ficar vazia e terminar de ordenar o vetor.





...

Nesse caso, está ordenando um vetor de *ints,* mas no trabalho será ordenado provavelmente uma *struct* em que cada nó da árvore será uma dessas *structs* e a comparação deve atender os critérios de desempate citados anteriormente.

Observações:

A estrutura da árvore utilizada pelo algoritmo deverá ter capacidade de conter **k** elementos em sua folha, cujo **k** é a primeira potência de 2 maior ou igual ao número de elementos do vetor a se ordenar dado. Portanto, considerando a inutilização do índice 0 da árvore também, seu vetor da árvore terá tamanho total de **2k**.

O *torneioGeralSort* deverá ter complexidade na ordem de **n.log(k)** no pior e melhor caso pelo menos (se atente na hora de atualizar os ramos da árvore após a remoção de um elemento, pois você precisará atualizar apenas os ramos ligados ao que foi removido).

Assim como descrito no trabalho 1, modularize e comente devidamente seu código.

Não plagie seu trabalho, sujeito a nota 0 em ambos os envolvidos.