

Universidade Federal do Ceará - Campus Quixadá
QXD0010 – Estruturas de Dados – Turma 02A
Prof. Atílio Gomes

SEGUNDO TRABALHO

A solução do problema descrito neste documento deve ser entregue até a meia-noite do dia **03/11/2019** pelo SIPPA.

Leia atentamente as instruções abaixo.

Instruções:

- Este trabalho pode ser feito em **dupla** ou **individualmente** e deve ser implementado usando a linguagem de programação C++
- O seu trabalho deve ser compactado (.zip, .rar, etc.) e enviado para o SIPPA na atividade correspondente ao **Trabalho 4** da disciplina.
- Identifique o seu código-fonte colocando o seu **nome** e **matrícula** como comentário no início de seu código.
- Indente corretamente o seu código para facilitar o entendimento.
- As estruturas de dados que forem usadas devem ser implementadas por você como um TAD.
- O código-fonte deve estar devidamente organizado e documentado.
- Observação: Lembre-se de desalocar os endereços de memória alocados quando os mesmos não forem mais ser usados.
- Observação: Qualquer indício de plágio resultará em nota **ZERO** para todos os envolvidos.

DICA: COMECE O TRABALHO O QUANTO ANTES.

Questão 1: Seja $V = [0 \dots n - 1]$ um vetor de n números inteiros arbitrários. Uma árvore binária pode ser usada para se ordenar os elementos de V em ordem não decrescente. A fim de obter tal feito, primeiro, devemos criar uma árvore binária cheia¹ cuja maioria dos seus nós (senão todos) terão como dados os números do vetor. A árvore binária cheia que será criada terá altura $h = \lceil \log_2 n \rceil + 1$.

Essa árvore binária é criada de baixo para cima (*bottom-up fashion*) do seguinte modo: para cada elemento $V[i]$ do vetor, crie uma folha e armazene o número $V[i]$ na respectiva folha. Deste modo, todos os elementos do vetor V serão armazenados nas primeiras n folhas da árvore. Em cada folha restante que não for preenchida, armazene um elemento \mathcal{E} maior do que qualquer outro no vetor. A Figura 1 ilustra o exemplo para $n = 5$, $V = [8, 20, 41, 7, 2]$, $h = \lceil \log_2 5 \rceil + 1 = 4$ e $\mathcal{E} = 42$.

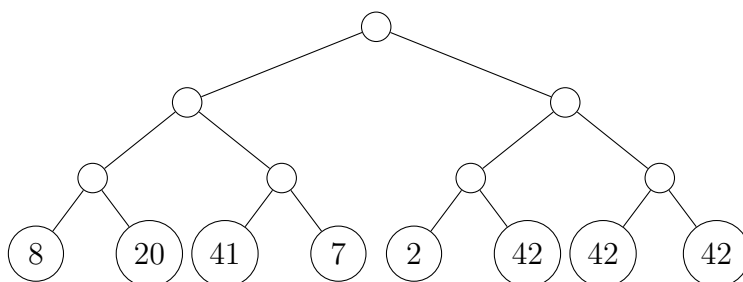


Figura 1: Primeira etapa na criação da árvore binária: as primeiras n folhas da árvore (da esquerda para a direita) contêm os n primeiros elementos do vetor V .

Dada a árvore binária construída de modo que todas as folhas tenham um número inteiro como dado, começando a partir da base da árvore, atribua a cada nó o menor dos valores de seus dois filhos, como mostrado na Figura 2, de modo que o menor elemento na árvore, e_{min} , seja atribuído à raiz.

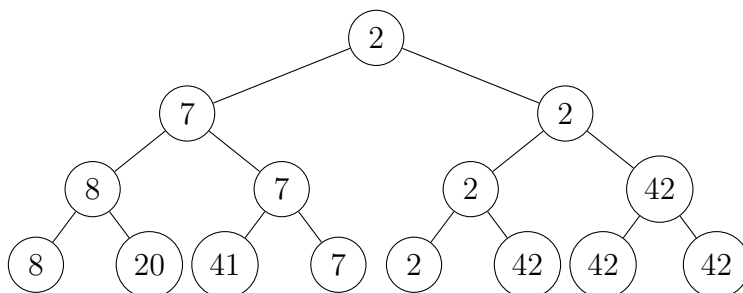


Figura 2: Segunda etapa: preenchimento dos nós da árvore binária. Partindo do último nível até o primeiro, atribuímos a cada nó em um mesmo nível o valor do menor entre os seus dois filhos. Desse modo, garantimos que, ao chegar na raiz da árvore, será atribuída à raiz o menor inteiro de toda a árvore. Este menor inteiro é denotado por e_{min} e nesta árvore da figura temos $e_{min} = 2$.

Construída a árvore, a seguir, executamos um laço até que o elemento \mathcal{E} seja atribuído à raiz. Em cada iteração desse laço, armazenamos o número \mathcal{E} na folha com valor igual a e_{min} e, então, repetimos a segunda etapa do preenchimento da árvore binária até que

¹Lembre que uma árvore binária é **cheia** quando todas as suas folhas estão no último nível.

o novo menor elemento ocupe o nó raiz. A Figura 3 exibe a árvore da Figura 2 após a primeira iteração do laço.

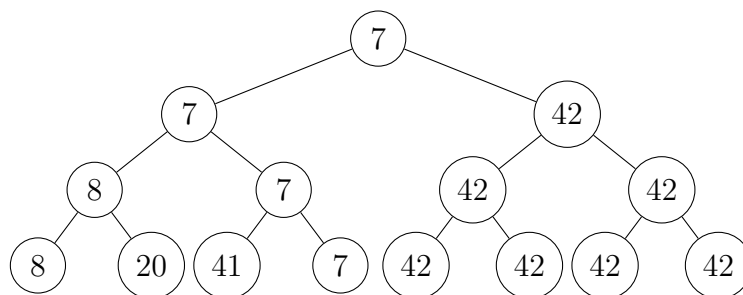


Figura 3: Conteúdo da árvore após a primeira iteração do laço. Neste momento, o novo menor elemento da árvore é o número $e_{min} = 7$.

Trabalho: Construa um programa em linguagem C++ que implemente o algoritmo acima para ordenar um conjunto de vetores de inteiros.

Entrada

O seu programa deverá ler os vetores a partir de um arquivo-texto² chamado `vetores.txt`. Haverá vários casos no arquivo de entrada. Cada caso terá duas linhas. A primeira linha contém o tamanho do vetor, N . A segunda linha contém uma lista dos N elementos do vetor, separados por espaço. O final da entrada é sinalizado pelo caso $N = 0$. Esse caso não deve ser processado e determina que a leitura deve ser finalizada.

Saída

A saída do seu programa deve ser gravada em um arquivo-texto chamado `ordenados.txt` e deve ser formada da seguinte maneira: Para cada vetor lido no arquivo de entrada, escreva uma linha no arquivo de saída contendo os elementos do vetor em ordem não decrescente.

Exemplo de entrada

```

10
23 3 45 6 1 9 37 99 0 30
3
345 2 1
5
98 34 2 1 76
0
  
```

Exemplo de saída

```

0 1 3 6 9 23 30 37 45 99
1 2 345
1 2 34 76 98
  
```

²A fim de ler e escrever em arquivos-texto usando a linguagem C++, pesquise sobre as classes nativas `ofstream` (escrita) e `ifstream` (leitura). Um tutorial pode ser encontrado neste link: <http://www.cplusplus.com/doc/tutorial/files/>

Além do código-fonte deve ser apresentado um relatório com:

- (a) Listagem dos programas em C++
- (b) Listagem dos testes executados.
- (c) Descrição sucinta das estruturas de dados usadas e decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
- (d) Estudo da complexidade do tempo de execução dos métodos implementados e do programa como um todo (Notação O).

Informações adicionais para este trabalho:

- Um dos parâmetros utilizados na avaliação da qualidade de uma implementação consiste na constatação da presença ou ausência de comentários. Comente o seu código. Mas também não comente por comentar, forneça bons comentários.
- Outro parâmetro de avaliação de código é a *portabilidade*. Dentre as diversas preocupações da portabilidade, existe a tentativa de codificar programas que sejam compiláveis em qualquer sistema operacional. Como testarei o seu código em uma máquina que roda Linux, não use bibliotecas que só existem para o sistema Windows como, por exemplo, a biblioteca `conio.h` e outras tantas.
- Este trabalho vale 10 pontos.