Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO-01
Vanessa Souza

Atividade Avaliativa

Assunto : TAD Heap

Lista de Prioridades

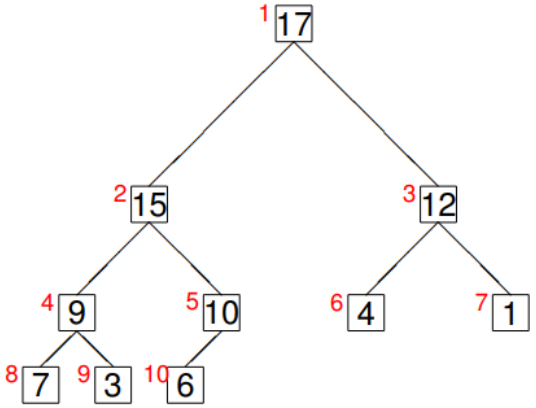
Já estudamos a estrutura de dados **fila**, que organiza os elementos seguindo o princípio FIFO (*First In, First Out*), ou seja, o primeiro elemento a entrar é também o primeiro a sair. No entanto, em diversas aplicações, pode ser necessário remover o elemento com maior prioridade, independentemente da ordem de chegada. Por exemplo, em um sistema de atendimento hospitalar, pacientes em estado crítico devem ser atendidos antes dos demais, mesmo que tenham chegado depois. A melhor representação de uma fila de prioridade é através de uma estrutura de dados chamada **HEAP**.

Heap é uma árvore binária completa e de prioridade. Uma árvore binária diz-se completa quando os seus níveis estão cheios, com possível excepção do último, o qual está preenchido da esquerda para a direita até um certo ponto. A estrutura Heap é composta por um conjunto finito de dados, cada qual com uma chave que determinará sua prioridade dentro da lista. Logo, essa estrutura é um vetor e pode ser vista como uma árvore binária completa.

Há dois tipos de Heap, a de máximo e a de mínimo. Em um Heap de máximo, o maior elemento encontra-se na raiz e o pai é sempre maior que seus filhos. A Figura 1 ilustra uma Heap de máximo, representada em árvore binária e abaixo o vetor.

No vetor, a relação entre os nós pai, filho da esquerda e filho da direita se dá da seguinte forma, onde *i* é a posição do elemento no vetor.

$Pai(i) = \left\lfloor \frac{i}{2} \right\rfloor$
$FEsq(i) = 2 * 1$
$FDir(i) = 2 * i + 1$



	17	15	12	9	10	4	1	7	3	6					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Então, no vetor, quando i=1, o filho da esquerda é a elemento do vetor na posição 2*i = 2 (15). Já o filho da direita é o elemento do vetor posição 2*i + 1 = 3 (12).

Considerando quando i=2 (15), o filho da esquerda é o elemento do vetor na posição 4 (9) e o filho da direita é o elemento 5 do vetor (10).

Na heap de máximo, os filhos são sempre menores que seus pais.

Uma nova inserção nessa heap deve acontecer à direita do nó 10, que corresponde à posição 11 do vetor. Por isso se diz que a estrutura Heap é sempre preenchida da esquerda para a direita.

Para implementação dessa heap do exemplo, o tamanho do vetor que representa a heap é de 16 posições. A posição zero é desprezada, uma vez que não é possível calcular os filhos com base nas equações.

A heap tem um atributo chamado *tail*, que marca a próxima posição onde uma nova chave será inserida. No exemplo, o *tail* é igual a 11.

A heap possui três operações importantes: inserir, remover e imprimir.

A inserção acontece sempre a posição tail do vetor. Posteriormente, o valor é comparado com seu pai. No caso da heap de máximo, se ele é maior que o pai, há uma troca. Essa troca continua até que o pai seja menor que o valor inserido, ou que se alcance a raiz da árvore (posição 1 do vetor).

A remoção sempre acontece na ordem de prioridade. Então, num heap de máximo, remove-se sempre a raiz da árvore, que é quem tem a maior prioridade. Essa remoção é feita substituindo o último elemento da árvore (posição tail – 1) com o primeiro (posição 1). Após a substituição, verifica-se a ordenação da heap, num processo conhecido como *heapifiy*.

Já a impressão é feita pelo vetor. Basta imprimir o vetor da posição 1 até o tail – 1.

Objetivo Pedagógico

O objetivo pedagógico da atividade é proporcionar aos alunos uma compreensão prática e aprofundada sobre a estrutura de dados Heap de Máximo, destacando sua aplicação e funcionamento. Ademais, espera-se traduzir conceitos abstratos sobre heaps em código funcional, reforçando o entendimento sobre as propriedades de uma árvore binária completa e a lógica de comparação entre nós para garantir a estrutura de heap.

Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO-01
Vanessa Souza

Descrição Geral
Estudo e Implementação de uma Heap de Máximo.

Objetivo

1. **Pesquisa Teórica:**

- Estude o conceito de **Heap de Máximo**, suas propriedades e as operações básicas.
- Responda:
  a) O que é uma Heap de Máximo e como ela é representada em memória?
  b) Como funciona o processo de manutenção da propriedade de heap (*heapify*) após a inserção ou remoção de elementos?

Para essa etapa, sugiro o livro ***Estrutura de Dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++***. Ascencio e Araújo. Pearson. 2010.

2. **Implementação:**

- Escreva um programa em C para implementar uma Heap de Máximo utilizando uma lista de inteiros.
- Inclua as seguintes operações:
  - Inserção de um elemento.
  - Remoção do maior elemento (raiz).
  - Impressão do estado atual da heap (representada na lista).

Requisitos do Programa

1. Carregamento Inicial:
   - O programa deve ler um arquivo de inteiros, onde cada linha contém um número inteiro. Esse número representa a prioridade na lista.
   - Seu programa deve implementar o conceito de heapify-up. Ou seja, a cada nova inserção, o elemento é inserido no final da heap (posição tail) e a estrutura é reorganizada para garantir que a propriedade da heap seja mantida.
   - O tamanho máximo da heap é informado no programa principal.

Exemplo de Funcionamento
O mesmo do livro da Ascencio.

1 10 2 15 3 7 4 23 5 18 6 4 7 36 *Figura 1: Entrada.txt*	Heap Gerada: 36 18 23 10 15 4 7 Heap após duas remoções: 18 15 7 10 4