Algoritmos e Estruturas de Dados III

3º Período Engenharia da Computação

Prof. Edwaldo Soares Rodrigues
Email: edwaldo.rodrigues@uemg.br

UNIVERSIDADE
DO ESTADO DE MINAS GERAIS | UEMG
UNIDADE DIVINÓPOLIS

Filas de prioridade

- É uma fila onde cada elemento possui uma prioridade;
- Essa prioridade determina a posição de um elemento na fila, portanto, determina quem deve ser o primeiro a ser removido da fila;
- Numa fila sem prioridade, sempre removemos o elemento do início da fila, de acordo com a propriedade FIFO (First in, First out);
- Já na fila de prioridades, o elemento que será removido é determinado pela prioridade;
- A fila de prioridade possui o critério de ordenação de acordo com a prioridade;

Filas de prioridade

- Fila de prioridade: é uma lista de itens na qual cada item possui uma prioridade associada;

- Pode-se determinar o item que tem a maior e a menor prioridade:

  - Filas de prioridade máxima;
  - Filas de prioridade mínima;

Filas de prioridade

- Existem vários tipos de implementações:
  - Lista encadeada;
  - **Heap binária;**
  - Array desordenado;
  - Array ordenado;

Filas de prioridade

- Cada tipo de implementação possui um custo diferente em relação às operações (inserção e remoção):

- Considerando que n é a quantidade de elementos da fila:

  - Lista encadeada: inserção O(n) e remoção O(1);
  - **Heap binária: inserção e remoção O(log n);**
  - Array desordenado: inserção O(1) e remoção O(n);
  - Array ordenado: inserção O(n) e remoção O(1);

Filas de prioridade

- Operações básicas de uma fila de prioridade:
  - Criação da fila;
  - Inserção de um elemento na fila com prioridade;
  - Remoção de um elemento da fila com maior prioridade;
  - Acesso a um elemento do início da fila (maior prioridade);
  - Liberação da fila;
  - Funções como tamanho e verificação de cheia e vazia;

Filas de prioridade - Heap

- É uma estrutura de dados que pode ser visualizada como uma árvore binária quase completa;

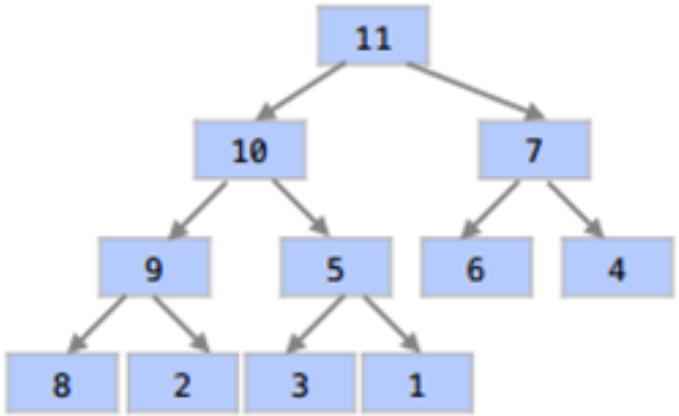
- Cada nó da árvore é ocupado por um elemento e temos as seguintes propriedades;

  - A árvore é completa até o penúltimo nível;
  - No último nível as folhas estão o mais a esquerda possível;
  - O conteúdo de um nó é maior ou igual ao conteúdo dos nós na subárvore enraizada nele (max-heap);
  - O conteúdo de um nó é menor ou igual ao conteúdo dos nós na subárvore enraizada nele (min-heap);

Filas de prioridade - Heap

- Filas de prioridade podem ser implementadas usando heap;

- Heap: um tipo especial de árvore:
  - Cada subárvore da árvore A é um heap;
  - A raiz de A é menor/maior ou igual do que cada subárvore de A;

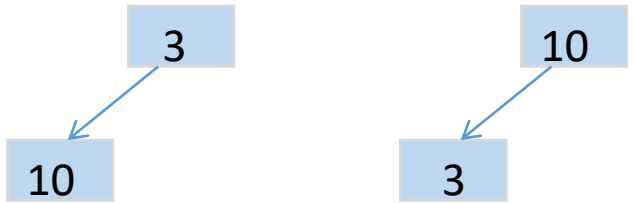


Filas de prioridade - Heap

- Inserção em filas de prioridade usando heap máximo:
  - Inserção do elemento 3;

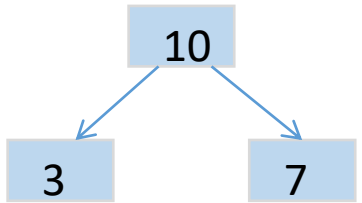
Filas de prioridade - Heap

- Inserção em filas de prioridade usando heap máximo:
  - Inserção do elemento 10;



Filas de prioridade - Heap

- Inserção em filas de prioridade usando heap máximo:
  - Inserção do elemento 7;

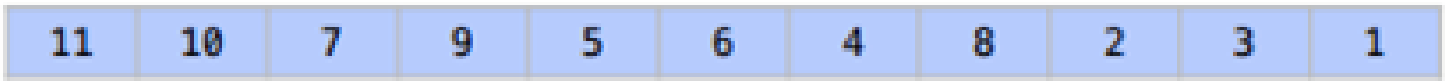


Filas de prioridade - Heap

- Inserção em filas de prioridade usando heap máximo:
  - Inserção do elemento 5;

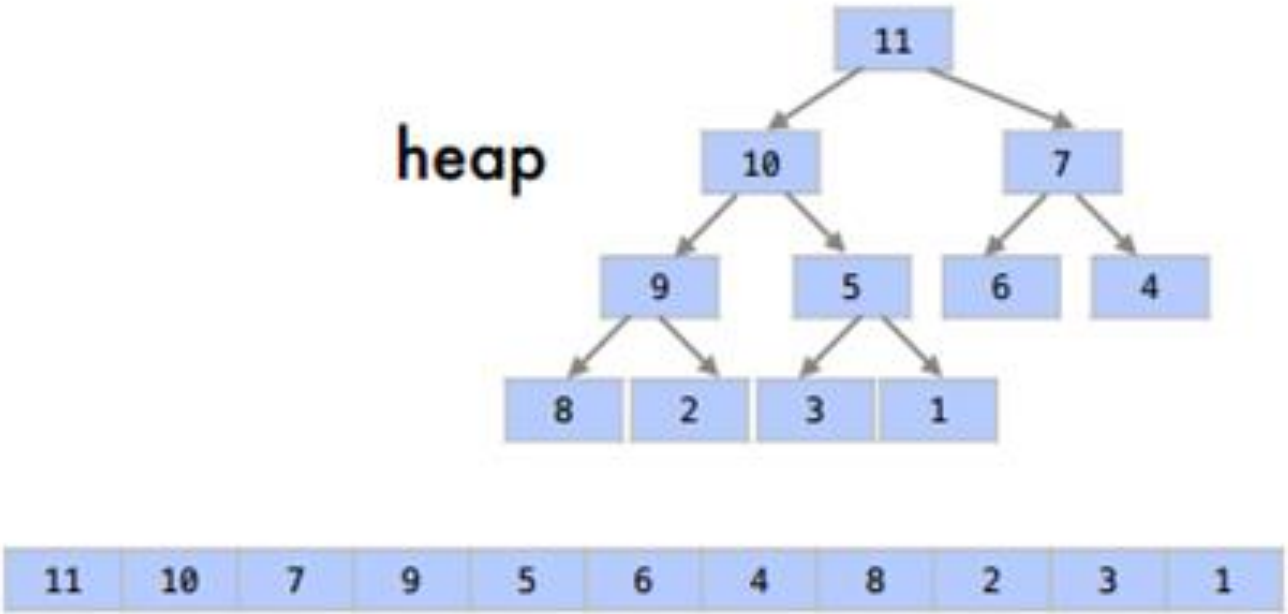
Filas de prioridade -Heap

- Filas de prioridade pode ser implementada usando um arranjo, não necessitando assim trabalhar com ponteiros



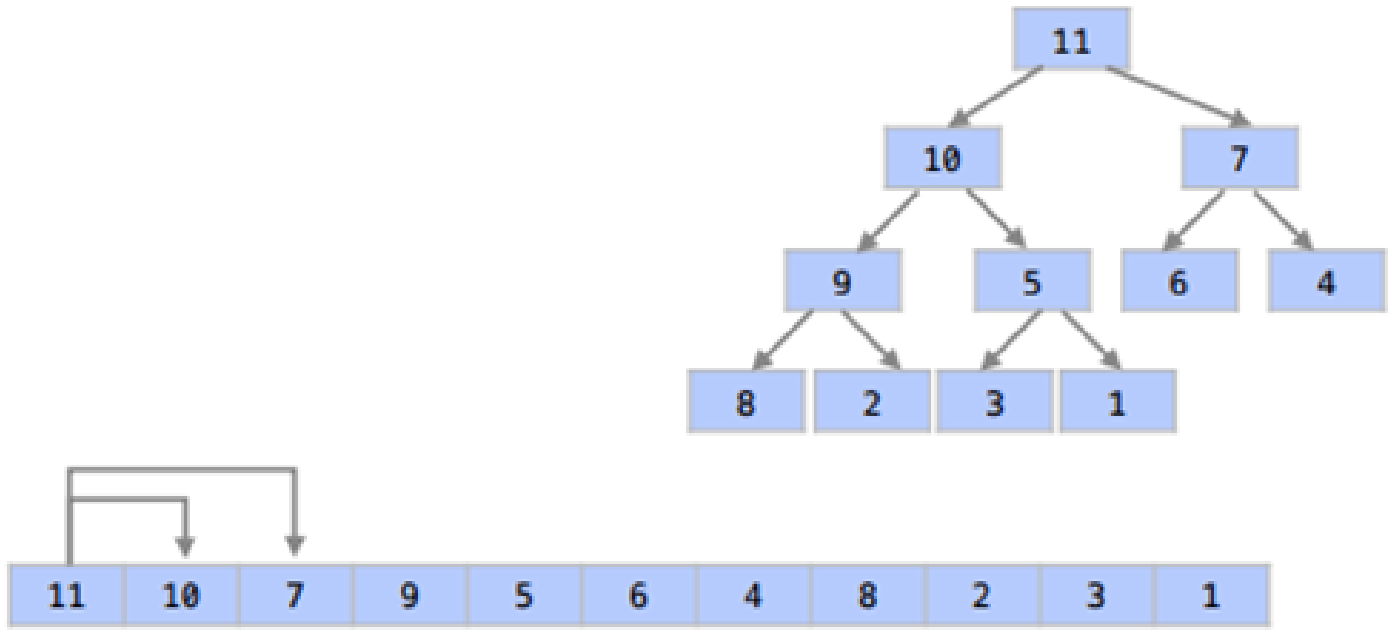
Filas de prioridade - Heap

- Este arranjo representa uma árvore chamada de árvore heap:



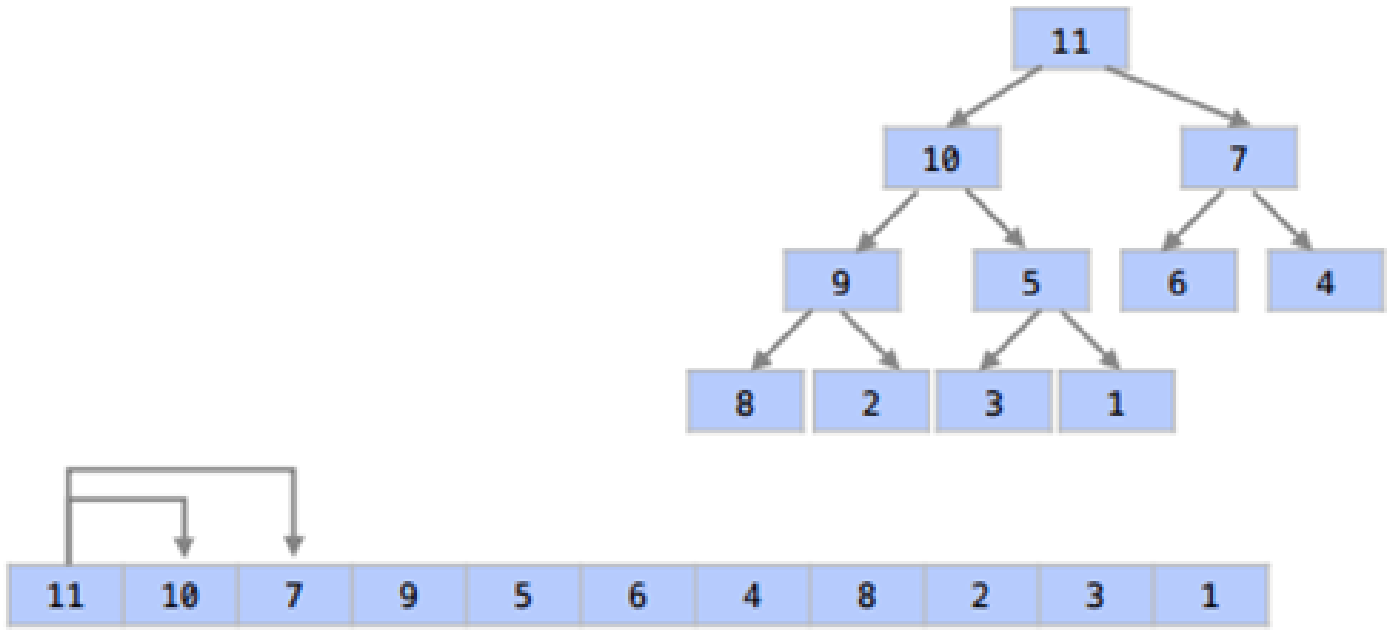
Filas de prioridade - Heap

- O heap é representado pelo arranjo de modo que os filhos de um elemento na posição i estejam na posição 2i+1 e 2i+2:



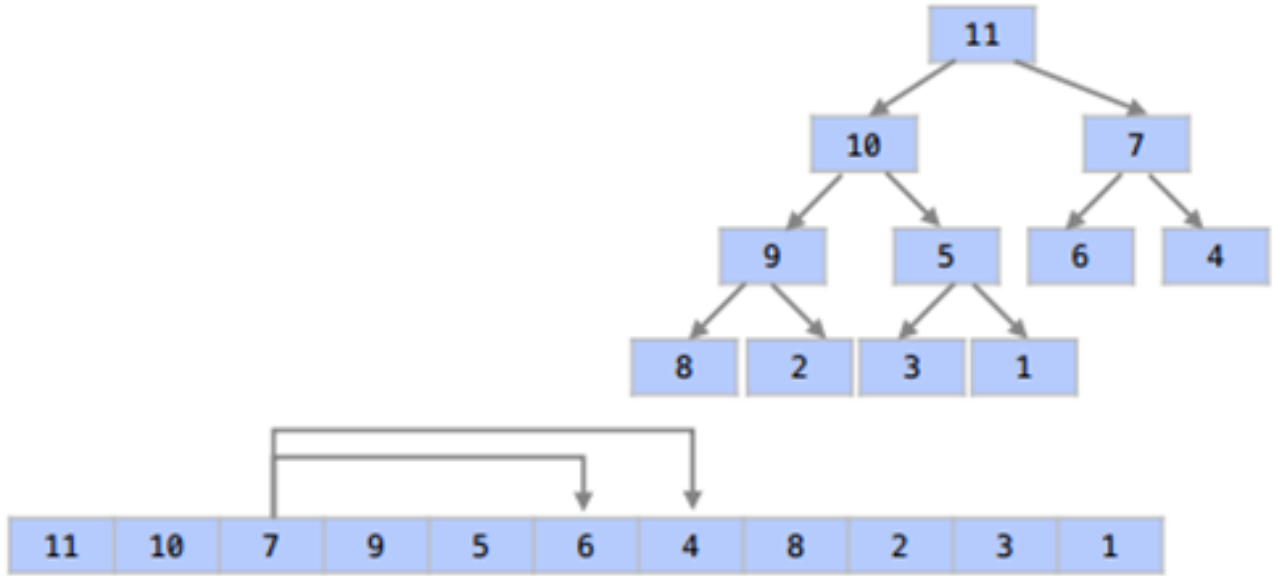
Filas de prioridade - Heap

- Os filhos do elemento 11 da posição 0 são 10 e 7, das posições 1 e 2:



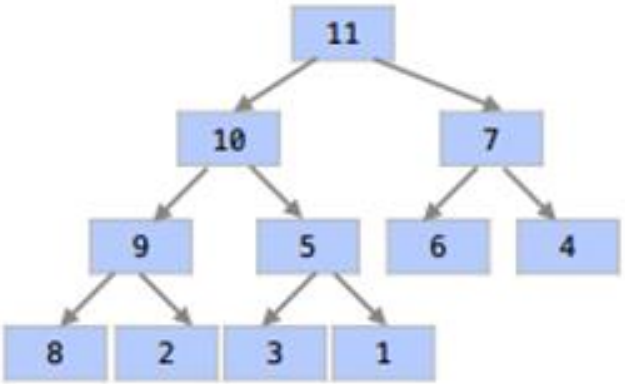
Filas de prioridade - Heap

- Os filhos do elemento 7, da posição 2 são 6 e 4, das posições 5 e 6:



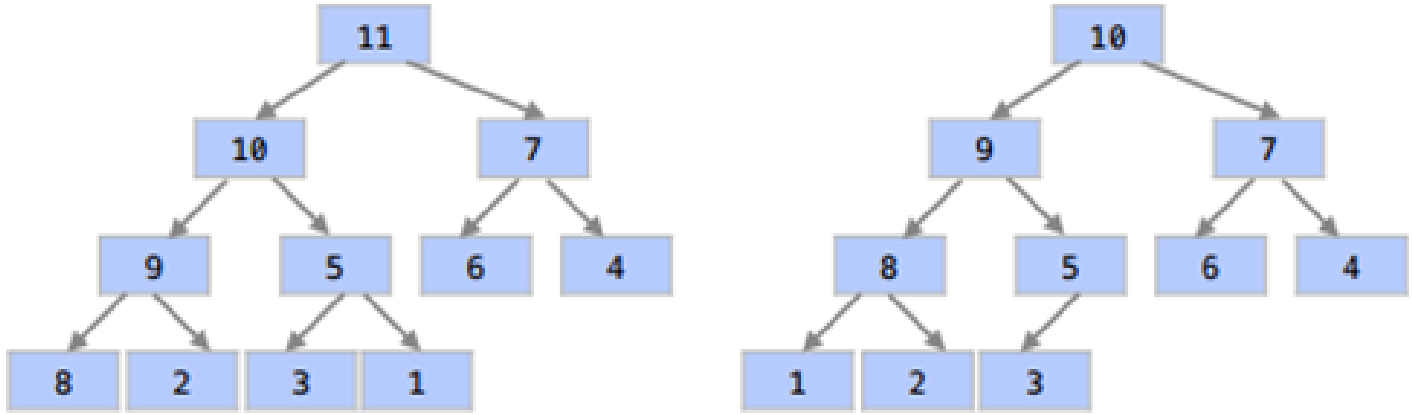
Filas de prioridade - Heap

- Uma propriedade importante destas árvores é que, apesar dos elementos não estarem ordenados, os filhos de um elemento sempre são menores/maiores que o pai:



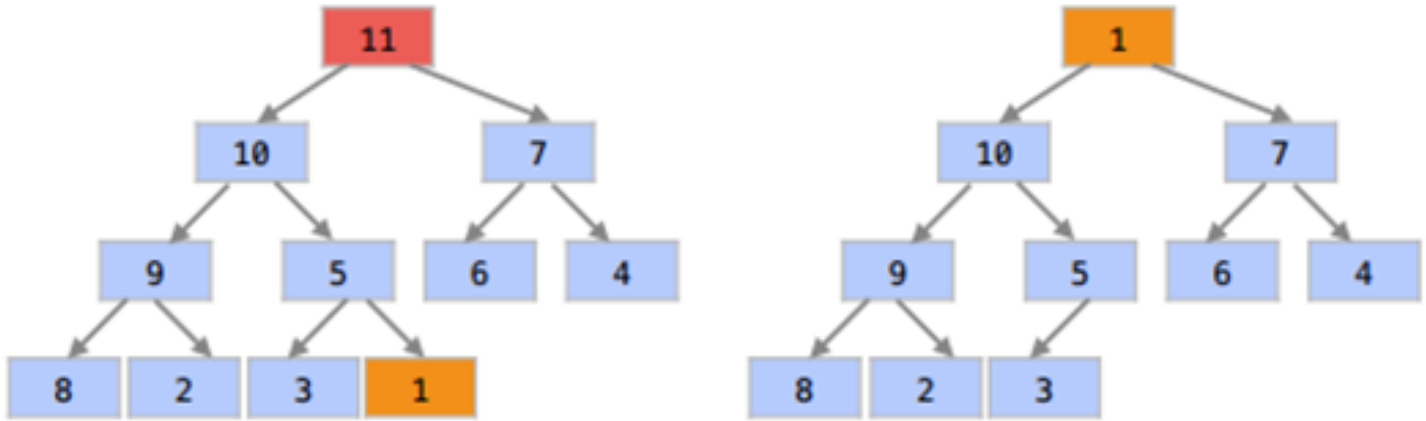
Filas de prioridade - Heap

- Assim, caso o elemento de maior prioridade seja removido, o segundo maior elemento toma seu lugar em tempo O(log n):



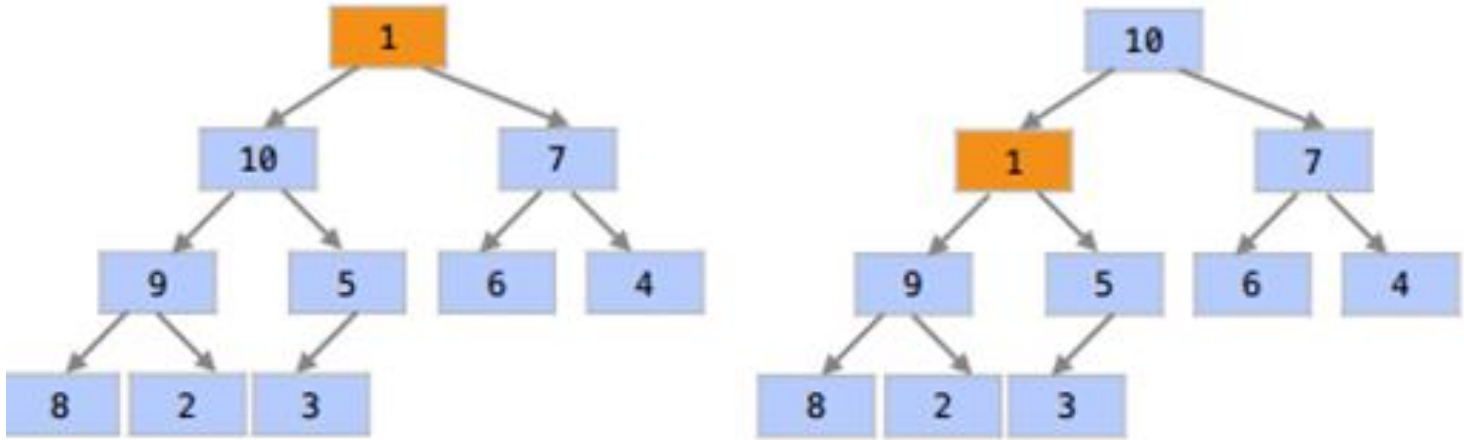
Filas de prioridade - Heap

- Para rearranjar o heap após remover o elemento do topo, o último elemento do heap toma o lugar do elemento removido:



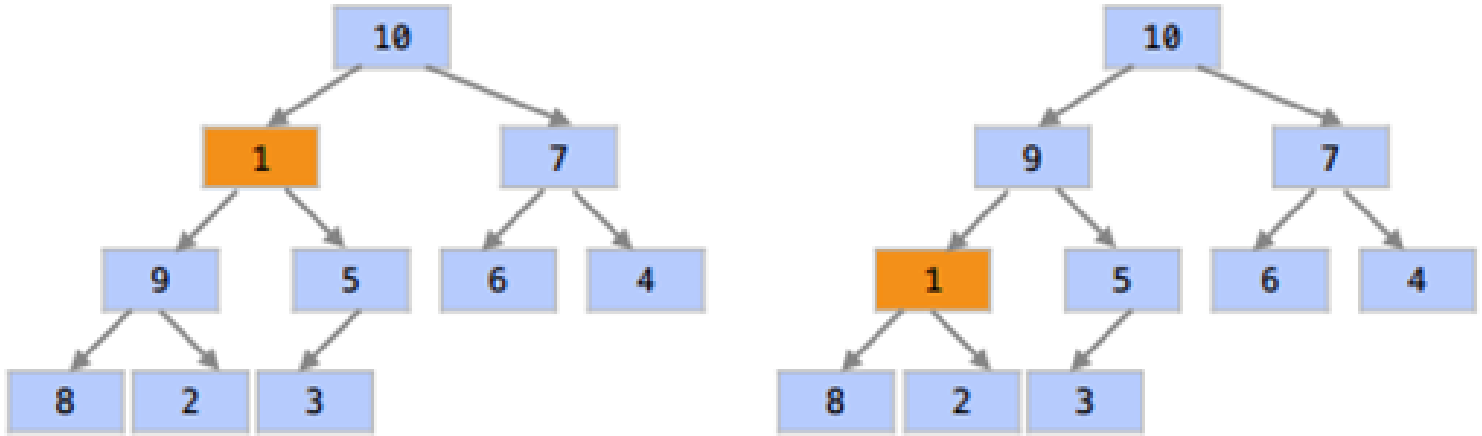
Filas de prioridade - Heap

- O elemento em questão é comparado com seus filhos e troca de posição com o maior deles:



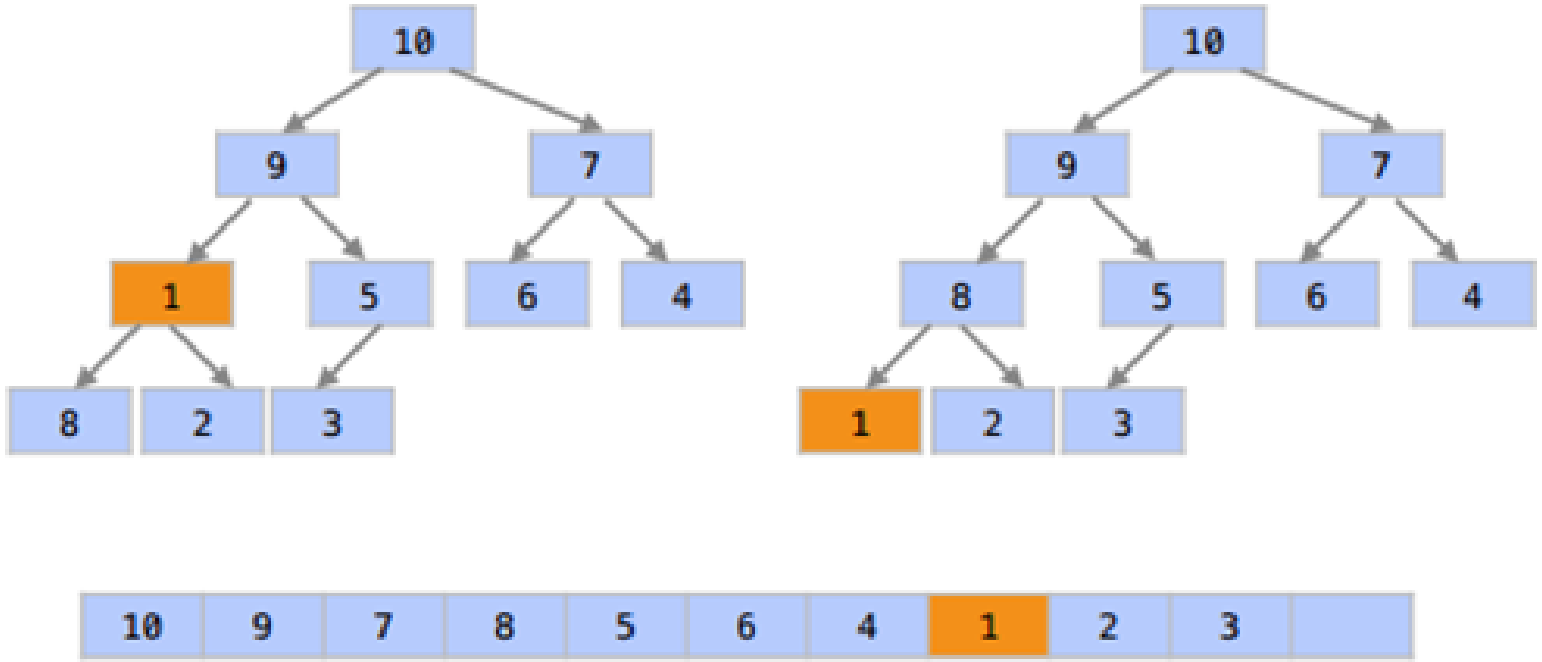
Filas de prioridade - Heap

- Novamente, o elemento é comparado com seus filhos e troca de posição com o maior deles:



Filas de prioridade - Heap

- O processo se repete até que o elemento esteja em uma posição que respeita a condição de ser maior que seus filhos:



Filas de prioridade - Heap

- As operações das filas de prioridades podem ser utilizadas para implementar algoritmos de ordenação;
- Basta utilizar repetidamente a operação Insere para construir a fila de prioridades;
- Em seguida, utilizar repetidamente a operação Retira para receber os itens na ordem reversa;

Filas de prioridade - Heap

- Aplicações:
  - Uma fila de pacientes esperando transplante de algum órgão:
    - A prioridade poderia ser o quão grave é o estado de saúde desse paciente;
  - Em Sistemas operacionais, um exemplo é a fila de prioridade de processos aguardando o processador para execução:
    - A prioridade poderia ser o processo que possui o menor tempo restante;
  - Caminhos mínimos (Algoritmo de Dijkstra):
    - A prioridade seria o menor custo;
  - Fila de pouso de aviões em um aeroporto:
    - Prioridade por combustível disponível;

Filas de prioridade - Heap

- Implementação:

```
#define TAM 10

typedef struct cadastro{
    //dados do usuário ou do que for ser cadastrado
}cadastro;

typedef struct filaP{
    int quant;
    cadastro c[TAM];
}filaP;
```

Filas de prioridade - Heap

- Implementação (inserção):

```
Insere_filaP(fp, d)
    se (fp != NULL && fp != cheia(fp))
        fp->dados[quant] = d
        verifica_filaP_insercao(fp, quant)
        quant++

verifica_filaP_insercao(fp, f)
    p = (f-1)/2
    enquanto(f > 0 && fp->dados[p].priori <= fp->dados[f].priori)
        aux = fp->dados[p]
        fp->dados[p] = fp->dados[f]
        fp->dados[f] = aux
        f = p
        p = (p-1)/2
```

Filas de prioridade - Heap

- Implementação (remoção):

```
remove_filaP(fp)
        se (fp != NULL && fp != vazia(fp))
            quant--
            fp->dados[0] = fp->dados[quant]
            verifica_filaP_remocao(fp, 0)

verifica_filaP_remocao(fp, p)
        f = 2*p +1
        enquanto(f  < quant)
            se(f < quant -1)
                se(fp->dados[f].priori < fp->dados[f+1].priori)
                    f++
            se(fp->dados[p].priori >= fp->dados[f].priori)
                break
            aux = fp->dados[p]
            fp->dados[p] = fp->dados[f]
            fp->dados[f] = aux
            p = f
            f = 2*p +1
```

Filas de prioridade - Heap

- Exercício:
  - Faça o passo a passo para a inserção dos elementos a seguir em um Heap Máximo representado por uma árvore binária. Ao final da inserção, apresente o arranjo correspondente: 12, 54, 7, 45, 32, 22, 13, 4, 8, 19, 36, 72, 1, 67, 15, 42;
  - Faça o passo a passo para a inserção dos elementos a seguir em um Heap Mínimo representado por uma árvore binária. Ao final da inserção, apresente o arranjo correspondente: 12, 54, 7, 45, 32, 22, 13, 4, 8, 19, 36, 72, 1, 67, 15, 42;
  - Apresente o passo a passo do processo de remoção dos 5 elementos mais prioritários em cada um dos Heaps, reajustando o Heap. No final apresente o arranjo correspondente após a remoção.

Algoritmos e Estruturas de Dados III

- Bibliografia:
  - Básica:
    - ASCENCIO, Ana C. G. Estrutura de dados. Rio de Janeiro: Pearson. 2011.
    - CORMEN, Thomas; RIVEST, Ronald; STEIN, Clifford; LEISERSON, Charles. Algoritmos. Rio de Janeiro: Elsevier, 2002.
    - ZIVIANI, Nívio. Projeto de algoritmos com implementação em Pascal e C. São Paulo: Cengage Learning, 2010.
  - Complementar:
    - EDELWEISS, Nina, GALANTE, Renata. Estruturas de dados. Porto Alegre: Bookman. 2009. (Coleção Livros didáticos de informática UFRGS, 18).
    - PINTO, W.S. Introdução ao desenvolvimento de algoritmos e estrutura de dados. São Paulo: Érica, 1990.
    - PREISS, Bruno. Estruturas de dados e algoritmos. Rio de Janeiro: Campus, 2000.
    - TENEMBAUM. Aaron M. Estruturas de Dados usando C. São Paulo: Makron Books. 1995.
    - VELOSO, Paulo A. S. Complexidade de algoritmos: análise, projeto e métodos. Porto Alegre: Sagra Luzzatto, 2001.

Algoritmos e Estruturas de Dados III