Estrutura de Dados (CC4652)

Aula 08 - Filas de Prioridades e Heaps

Prof. Luciano Rossi

Ciência da Computação
Centro Universitário FEI

2º Semestre de 2025

Fila de Prioridades

centro universitário FEI

Fila de Prioridades

Definição

Fila de Prioridades

- Uma **fila de prioridades** é uma estrutura de dados abstrata que mantém um conjunto de elementos, onde cada elemento possui uma prioridade associada.
- Os elementos são servidos de acordo com sua prioridade, não necessariamente na ordem de inserção.
- O elemento de maior prioridade é sempre o primeiro a ser removido.

centro universitário FEI

Fila de Prioridades

Propriedades

Propriedades

- **Ordenação por Prioridade:** Elementos com maior prioridade são servidos antes dos de menor prioridade.
- **Inserção Dinâmica:** Novos elementos podem ser inseridos a qualquer momento com suas respectivas prioridades.
- **Acesso Restrito:** Apenas o elemento de maior prioridade é diretamente acessível para remoção.

centro universitário FEI

Fila de Prioridades

Operações Fundamentais

Operações Principais

- `Insert(elemento, prioridade)`: Insere um elemento com prioridade especificada
- `ExtractMax()`: Remove e retorna o elemento de maior prioridade
- `Peek()`: Consulta o elemento de maior prioridade sem removê-lo
- `IsEmpty()`: Verifica se a fila está vazia

Operações Opcionais

- `ChangeKey(elemento, nova_prioridade)`: Modifica a prioridade de um elemento
- `Delete(elemento)`: Remove um elemento específico

Fila de Prioridades

Implementações

Estruturas de Implementação

- **Heap Binário:** Implementação mais eficiente e comum
- **Array Não-Ordenado:** Inserção $O(1)$, extração $O(n)$
- **Array Ordenado:** Inserção $O(n)$, extração $O(1)$
- **Lista Ligada:** Variações ordenadas ou não-ordenadas

Complexidade com Heap

- Inserção: $O(\log n)$
- Extração: $O(\log n)$
- Consulta: $O(1)$

centro universitário FEI

Fila de Prioridades

Aplicações

Aplicações Práticas

- **Algoritmos de Grafos:** Dijkstra, Prim, A*
- **Simulação de Eventos:** Sistemas de eventos discretos
- **Agendamento:** Escalonamento de processos em SO
- **Algoritmos Gulosos:** Problemas de otimização
- **Algoritmos de Ordenação:** HeapSort

Exemplo Prático

- Sistema de atendimento hospitalar: pacientes são atendidos por ordem de gravidade (prioridade), não por ordem de chegada

centro universitário FEI

Heaps

Heaps

Definição

Heaps

- Em ciência da computação, um heap é uma estrutura de dados que é usada para armazenar um conjunto de elementos, geralmente em forma de árvore binária.

Heaps

Propriedades

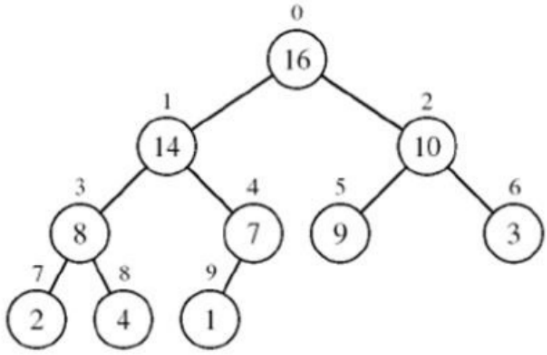
Propriedades

- Balanceamento: Uma árvore binária quase completa, com eventual excessão do último nível. Quando o último nível não está completo, as folhas devem estar mais a esquerda da árvore.
- Estrutural: O valor armazenado em cada nó não é menor que os de seus filhos.

centro universitário FEI

Heaps

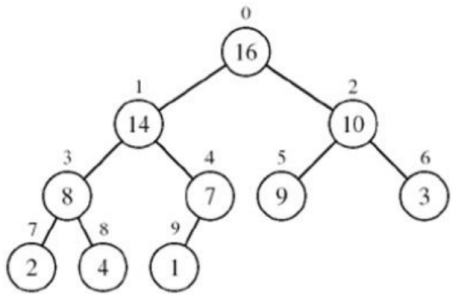
Exemplo

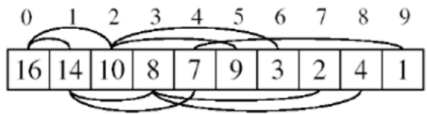


Heaps

Exemplo – Representação de Heap em um Vetor

- Normalmente, um heap é **armazenado em um vetor**, onde cada posição deste vetor é um nó da árvore.

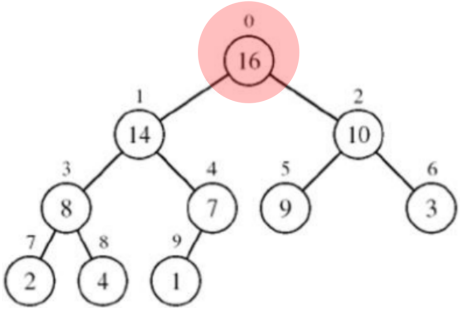


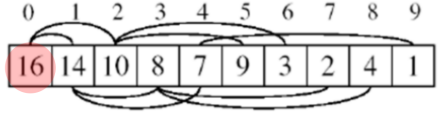


Heaps

Exemplo – Representação de Heap em um Vetor

- A **raiz da árvore** está sempre localizada na posição $V[0]$

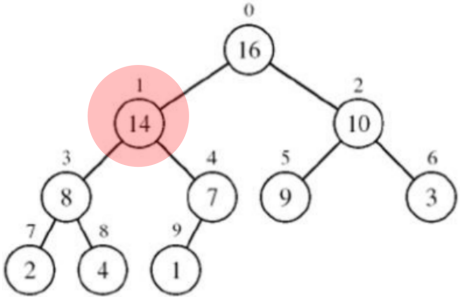


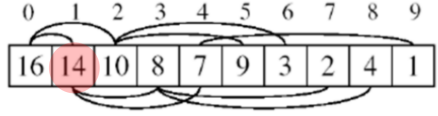


Heaps

Exemplo – Representação de Heap em um Vetor

- O **filho esquerdo** de um nó de índice i é dado por: $fe = 2.i + 1$

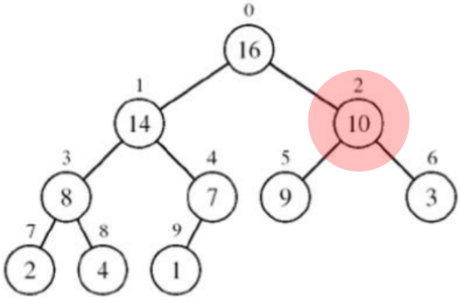


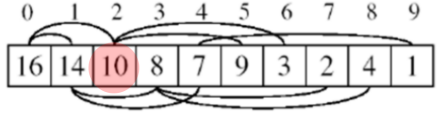


Heaps

Exemplo – Representação de Heap em um Vetor

- O **filho direito** de um nó de índice i é dado por: $fd = 2.i + 2$

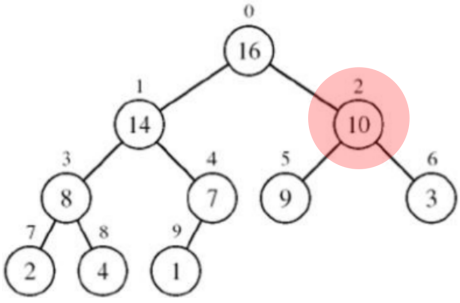


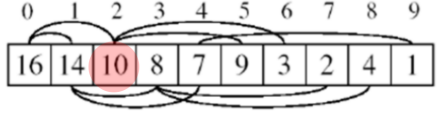


Heaps

Exemplo – Representação de Heap em um Vetor

- O **pai** de um nó de índice i é dado por: $p = \lfloor (i-1)/2 \rfloor$

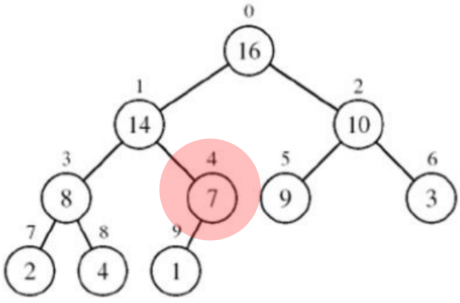


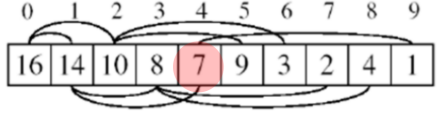


Heaps

Exemplo – Representação de Heap em um Vetor

- O **último pai** de um nó de índice i é dado por: $up = \lfloor n/2 \rfloor - 1$

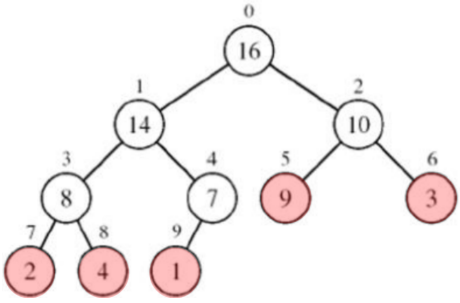


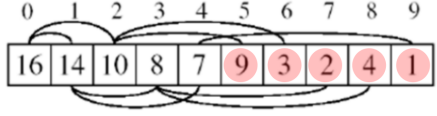


Heaps

Exemplo – Representação de Heap em um Vetor

- O intervalo das **folhas** de um heap é dado por: $\lfloor n/2 \rfloor \leqslant f < n$





Heaps

Operações Básicas

- Operações Principais:
  - Inserir Item no Heap
  - Remover Item no Heap (remover o máximo)
- Operações Secundárias
  - Peneirar (para baixo e para cima)
  - Construir (obtém heap a partir de qualquer vetor)

centro universitário FEI

Heap

Procedimento para inserção de elementos

Procedimento para adição de novos elementos a um heap:

- Se houver espaço no vetor, então
  - Deve ser folha no último nível, na primeira posição disponível mais à esquerda
  - Se este nível estiver cheio, comece um novo nível.
- Se houver violação da estrutura de heap, então
  - Invoque a **Peneirar para Cima** para o heap.

centro universitário FEI

Heap

Procedimento para remoção de elementos

Procedimento para remoção de elementos de um heap:

- Troca-se o elemento da raiz, índice 0, com o último elemento do heap;
- Decrementa-se a quantidade;
- Invoca-se **Peneirar** para o heap.

centro universitário FEI

Heap

Procedimento: Peneirar para Cima (Heapify-Up)

Peneirar para Cima

- Caso um nó, em um heap, tenha valor maior que seu pai, podemos recuperar a propriedade trocando-o com seu pai.
- Se ainda houver violação da propriedade estrutural, então
  - Continue subindo na árvore comparando com o novo pai
  - Repita até que não haja mais violações ou chegue à raiz
- Chamamos este procedimento de **Peneirar para Cima** (Sift-Up).

centro universitário FEI

Heap

Procedimento: Peneirar para Baixo (Heapify-Down)

Peneirar para Baixo

- Caso um nó, em um heap, perca sua propriedade estrutural, podemos recuperá-la trocando-o com seu filho de maior valor.
- Se ainda houver violação da propriedade estrutural, então
  - Continue descendo na árvore comparando com os novos filhos
  - Sempre troque com o filho de maior valor
  - Repita até que não haja mais violações ou chegue a uma folha
- Chamamos este procedimento de **Peneirar** (Sift-Down).

centro universitário FEI

Simulação de Heap
Inserir e remover os valores 1...10

Algoritmos

centro universitário FEI

Heap

Pseudocódigo: Peneirar (Heapify Down)

$PENEIRAR(h, pai)$

1 $esq \leftarrow FILHO\_ESQ(pai)$
2 $dir \leftarrow FILHO\_DIR(pai)$
3 se $esq < h.qtde$ e $h.valores[esq] > h.valores[pai]$ então
4 $\quad maior \leftarrow esq$
5 senão
6 $\quad maior \leftarrow pai$
7 se $dir < h.qtde$ e $h.valores[dir] > h.valores[maior]$ então
8 $\quad maior \leftarrow dir$
9 se $pai \neq maior$ então
10 $\quad$ trocar $h.valores[pai]$ com $h.valores[maior]$
11 $\quad PENEIRAR(h, maior)$

Heap

Pseudocódigo: Peneirar para cima (Heapify Up)

$PENEIRAR\_PARA\_CIMA(h, filho)$

1 se $filho \leq 0$ então retorne
2 $pai\_idx \leftarrow PAI(filho)$
3 se $h.valores[filho] > h.valores[pai\_idx]$ então
4 trocar $h.valores[filho]$ com $h.valores[pai\_idx]$
5 $PENEIRAR\_PARA\_CIMA(h, pai\_idx)$

Heap

Pseudocódigo: Construir heap

$CONSTRUIR(h)$

1 para $i \leftarrow ULTIMO\_PAI(h)$ até 0 passo -1 faça

2 $\quad PENEIRAR(h, i)$

Heap

Pseudocódigo: Inserir elemento

$INSERIR(h, valor)$

```
1  se h.qtde < LEN então
2      h.valores[h.qtde] ← valor
3      PENEIRAR_PARA_CIMA(h, h.qtde)
4      h.qtde ← h.qtde + 1
```

Heap

Pseudocódigo: Remover elemento máximo

$REMOVER(h)$

1 se $h.qtde > 0$ então
2 trocar $h.valores[0]$ com $h.valores[h.qtde - 1]$
3 $h.qtde \leftarrow h.qtde - 1$
4 se $h.qtde > 0$ então
5 $PENEIRAR(h, 0)$

Estrutura de Dados (CC4652)

Aula 08 - Filas de Prioridades e Heaps

Prof. Luciano Rossi

Ciência da Computação
Centro Universitário FEI

2º Semestre de 2025