

# HEAPS

## DCE792 - AEDs II (Prática)

Atualizado em: 20 de novembro de 2023

Iago Carvalho

Departamento de Ciência da Computação



Na aula passada, nós vimos o conceito de filas de prioridades

- Uma estrutura de fila
- Engloba itens de diferentes prioridades
  - Útil para simular filas de bancos, mercados e processadores, dentre outras aplicações

As implementações que vimos nas aulas passadas não eram muito eficientes

Nesta aula, vamos implementar filas de prioridades utilizando **heaps**

Uma **heap** é a implementação mais eficiente existente de uma fila de prioridades

- Existem duas diferentes implementações
  1. min-heap
  2. max-heap

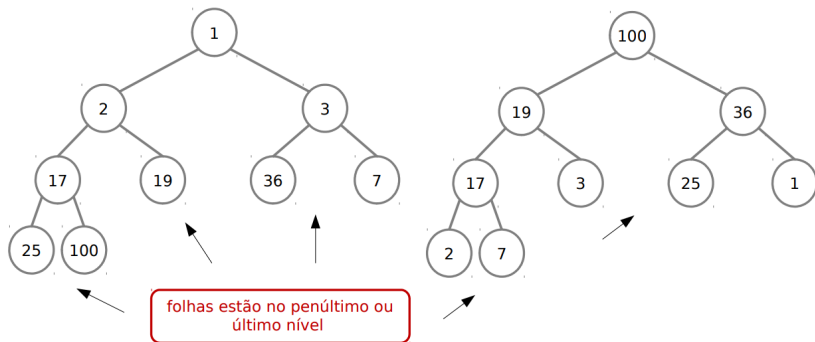
Operações que devem ser eficientes

- Seleção/remoção de elemento com maior/menor prioridade
- Inserção de um novo elemento

# HEAP COMO ÁRVORE BINÁRIA

Idealmente, uma heap é representada como uma árvore binária balanceada

- min-heap: os pais são **menores** que seus filhos
- max-heap: os pais são **maiores** que seus filhos



# OPERAÇÕES COM HEAPS

Uma heap é extremamente eficiente quando comparado a outras estruturas de dados

Operação	Lista	Lista Ordenada	Árvore (Balanceada)	Heap
Seleção	$O(n)$	$O(1)$	$O(\log n)$	$O(1)$
Inserção	$O(1)$	$O(n)$	$O(\log n)$	$O(\log n)$
Remoção	$O(n)$	$O(1)$	$O(\log n)$	$O(\log n)$
Construção	$O(n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$

# INSERÇÃO DE ELEMENTOS

Novos elementos sempre são inseridos como folhas

- Depois de inseridos, eles tem que *subir* na árvore até alcançar sua posição correta

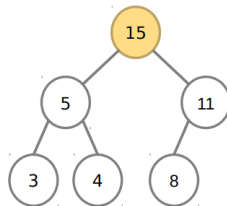
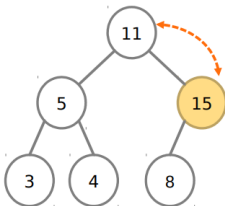
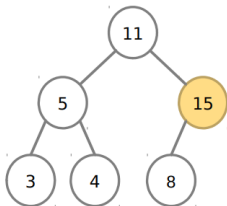
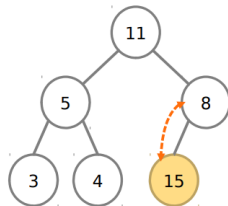
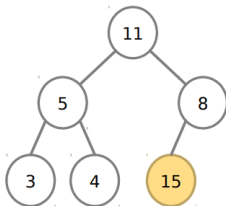
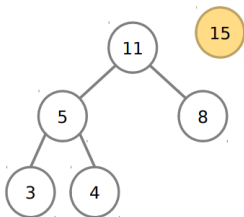
Para o elemento *subir* na árvore, deve-se

1. Comparar o elemento com seu pai
2. Se estiver em ordem, a inserção foi realizada
3. Caso contrário, troque o elemento com seu pai

O pequeno procedimento acima é repetido até que o passo 2 seja verdadeiro

# INSERÇÃO DE ELEMENTOS

Vamos inserir um elemento com prioridade 15



Como estamos tratando de uma fila de prioridades, sempre realizamos a retirada do elemento com

- Menor prioridade, no caso do min-heap
- Maior prioridade, no caso do max-heap

Por definição, o elemento a ser retirado será sempre a raiz da árvore



# REMOÇÃO DE ELEMENTOS

Após a remoção da raiz, deve-se

- Obter o último elemento da heap
- Inserir-lo na raiz
- Fazer o último elemento *descer* até sua posição correta

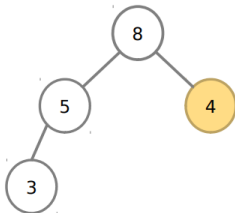
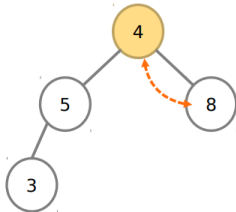
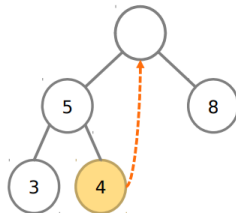
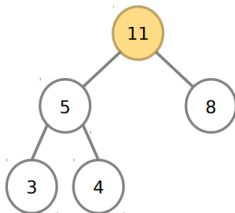
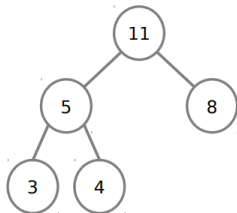
Para o elemento *descer* na árvore, deve-se

1. Comparar o elemento com seus filhos
2. Se estiver em ordem, a remoção foi realizada
3. Caso contrário, troque o elemento com seu maior filho

O pequeno procedimento acima é repetido até que o passo 2 seja verdadeiro ou até que uma folha seja alcançada

# REMOÇÃO DE ELEMENTOS

Vamos remover um elemento da heap



# REPRESENTANDO UMA HEAP COMO UM VETOR

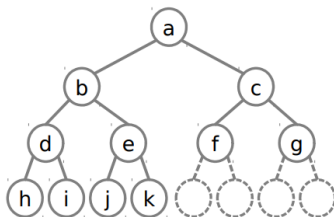
Pode-se, de maneira simples (mas um pouco confusa) representar uma heap utilizando um vetor

Seja  $i$  um nó

- Filho esquerdo de  $i$ : posição  $2i + 1$
- Filho direito de  $i$ : posição  $2i + 2$
- Pai de  $i$ : posição  $\frac{i-1}{2}$

# REPRESENTANDO UMA HEAP COMO UM VETOR

Para armazenar uma heap com altura  $h$ , precisamos de um vetor de tamanho  $2^{(h+1)} - 1$



índice	0	1	2	3	4	5	6	7	8	9	10	...
<b>nó</b>	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>	<b>f</b>	<b>g</b>	<b>h</b>	<b>i</b>	<b>j</b>	<b>k</b>	...
nível	0	1		2				3				

# REPRESENTAÇÃO COMPUTACIONAL DE UMA HEAP

```
1  struct heap {  
2      int max;           /* tamanho maximo do heap */  
3      int pos;           /* proxima posicao disponivel no vetor */  
4      int* prioridade;   /* vetor das prioridades */  
5  };
```

Observe que aqui não estamos representando as chaves dos elementos, somente sua prioridade

# OPERAÇÕES POSSÍVEIS COM HEAPS

- `void heap_inicializa(<tamanho>);`
  - Cria uma nova heap
- `void heap_insere(<prioridade>);`
  - Insere novo item na heap
- `bool ehVazia(void);`
  - Verifica se heap é vazia
- `bool ehCheia(void);`
  - Verifica se heap é cheia
- `<tipo> heap_remove(void);`
  - Remove o item de menor (ou maior) prioridade

# ATIVIDADE PRÁTICA

Implementar a função *main* no código da heap disponível no Github

Implementar as funções *ehVazia* e *ehCheia*

Alterar a min-heap para max-heap



PRÓXIMA AULA:

TABELAS HASH