ACH2023 – turmas 04 e 94 – EP3 2º Semestre de 2021

Inserção, Exclusão e Mudança de Prioridade no Heap Máximo

Destaca-se que um Heap Máximo é uma árvore binária na qual a prioridade de um nó sempre será maior ou igual à prioridade de seus filhos. O heap é representando utilizando um arranjo e não é necessário que cada elemento possua um ponteiro para seu filho a esquerda, a direita ou para seu pai, pois, a partir da posição de um elemento no arranjo sabemos onde está seu pai e seus filhos (se existirem). Dado um elemento na posição i do arranjo:
Seu pai (se existir [apenas a raiz não possui pai]) estará na posição: (i-1)/2 (arredondado para baixo, no caso de não ser um número inteiro)
Seu filho à esquerda (se existir) estará na posição 2*i+1
Seu filho à direita (se existir) estará na posição 2*i+2

Destaca-se que, para este EP, o arranjo para representar o heap não será um arranjo de elementos e sim um arranjo de ponteiros para os elementos.

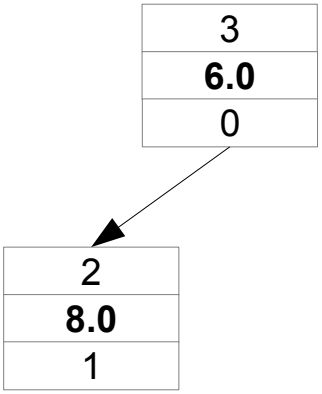
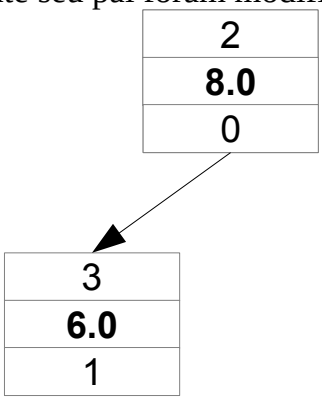
Videoaula "A estrutura de dados heap": https://www.youtube.com/watch?v=hyedRznKnjQ

A **inserção** no heap máximo ocorre, inicialmente, na primeira posição 'livre' do arranjo representado pelo heap (isto é, após o último elemento válido) e deve-se aumentar em um o valor da variável que armazena a quantidade de elementos no heap (*elementosNoHeap*). Porém, o heap pode precisar ser reorganizado para manter a sua propriedade de heap máximo.
A reorganização do heap ocorre da seguinte maneira: se o pai do elemento inserido tiver prioridade menor do que a do elemento inserido, estes dois elementos devem trocar de posição. Isto deve ser feito (iterativamente ou recursivamente) até que o elemento inserido tenha prioridade menor ou igual à prioridade de seu pai ou torne-se a raiz do heap.

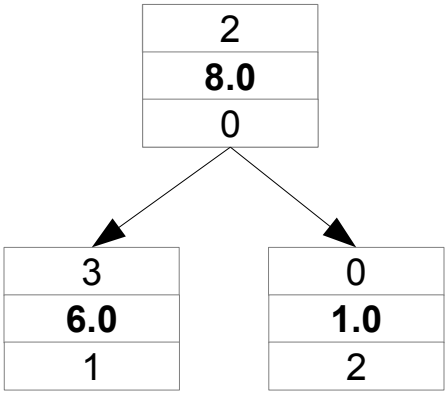
Exemplos de inserção:
Inserção do elemento com id igual a 3 e prioridade igual a **6.0** (ele ficará na posição zero do heap, isto é, do arranjo que presenta o heap, lembrando que o heap é armazenado como um arranjo, mas é visto como uma árvore binária completa ou quase completa):

3
6.0
0

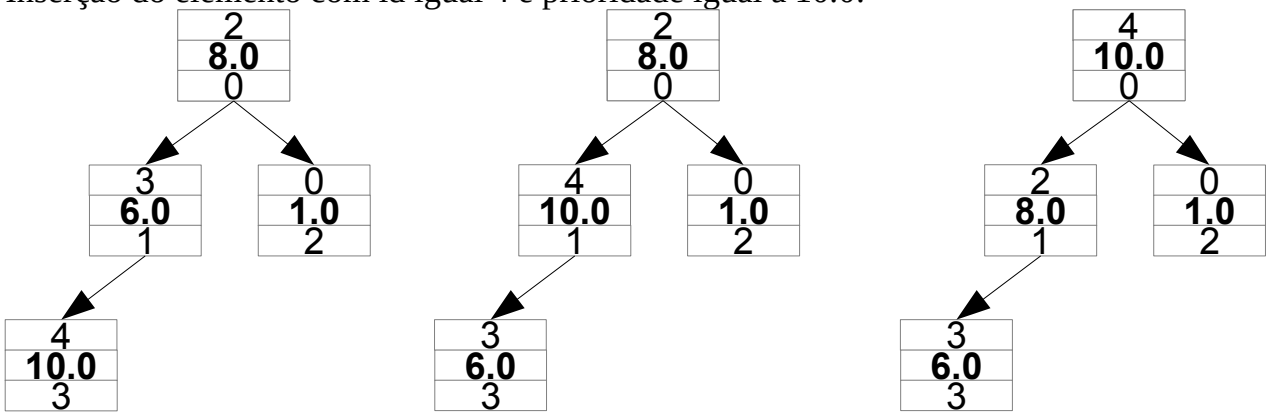
Inserção do elemento com id igual a 2 e prioridade igual a **8.0**:

Inicialmente o elemento é inserido no final do heap (posição 1 do arranjo, neste caso). 	Iterativamente ele é reposicionado (trocando de lugar com seu pai) até que ele tenha prioridade menor ou igual à do seu pai ou se torne a raiz da árvore (notem que o campo posicao do novo elemento e de quem era originalmente seu pai foram modificadas). 

Inserção do elemento com id igual a 0 e prioridade igual a 1.0 (ele é inicialmente inserido na posição 2 do arranjo, e já está na posição correta [seu pai tem prioridade maior ou igual a sua prioridade]):



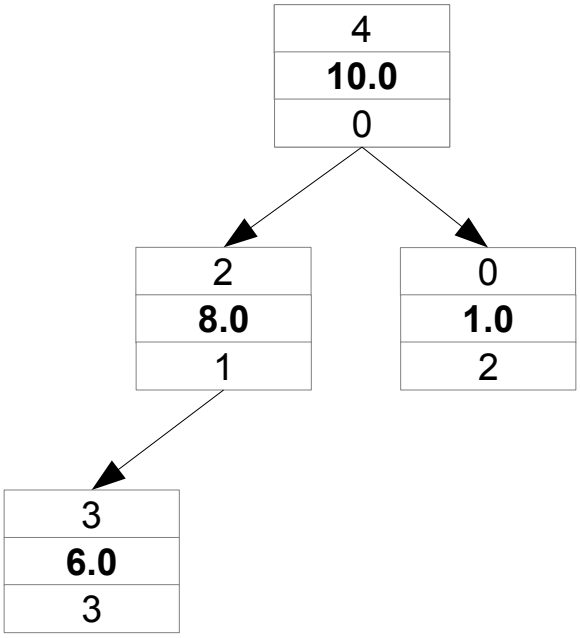
Inserção do elemento com id igual 4 e prioridade igual a 10.0:

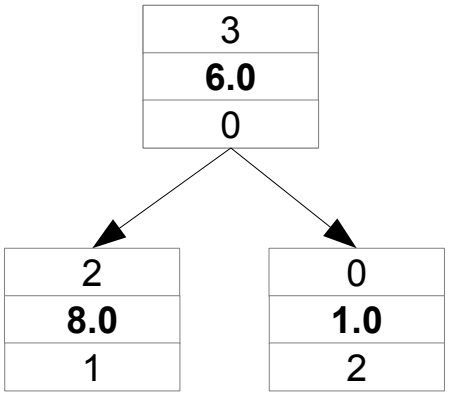
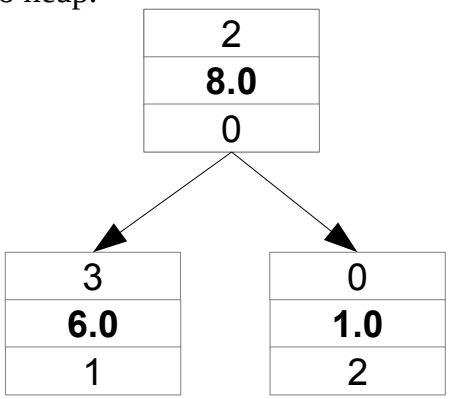


Observe que sempre que mudamos o valor do campo *posicao* de um elemento do heap, precisamos ajustar o ponteiro na respectiva posição do aranjo que representa o heap neste EP.

A **remoção** (ou exclusão) no heap máximo ocorre sempre no primeiro elemento (isto é, o elemento de maior prioridade, que é a raiz). Após a remoção da raiz é colocado o último elemento no lugar da raiz e é diminuído em um o campo que indica o número de elementos do heap. Porém, o heap pode precisar ser reorganizado para manter a sua propriedade de heap máximo.
A reorganização após remoção do heap ocorre da seguinte maneira: se o elemento que foi colocado no lugar da raiz tiver prioridade menor do que a de um de seus filhos ele deverá trocar de lugar com o filho que possuir maior prioridade. Isto deve ser feito (iterativamente ou recursivamente) até o que o elemento que era o último do heap tenha prioridade maior do que a de seus filhos ou torne-se uma folha no heap.

Exemplos de remoção:
No seguinte heap, devemos excluir o elemento com id igual a 4 (pois corresponde à raiz do heap)



O último elemento do heap (o de id igual 3) deve ser colocado no lugar da raiz.	Iterativamente deve-se trocar de posição este último elemento com o seu filho de maior prioridade, até que sua prioridade seja maior do que a de seus filhos ou ele se torne uma folha do heap.
	

Para a mudança de prioridade, a lógica é semelhante a inclusão ou a remoção.
Ao **diminuir a prioridade** de um elemento, ele deve ser, iterativamente, trocado de posição com seu filho de maior prioridade (caso a prioridade do filho de maior prioridade seja maior do que a nova prioridade do elemento atual) até que o elemento tenha prioridade maior do que a de seus filhos ou se torne folha do heap.

Ao **aumentar a prioridade** de um elemento, ele deve ser, iterativamente, trocado de posição com seu pai (caso a prioridade do pai seja menor do que a nova prioridade do elemento atual) até que o elemento tenha prioridade menor do que a de seu pai ou se torne a raiz do heap.

Observe que sempre que mudamos o valor do campo *posicao* de um elemento do heap, precisamos ajustar o ponteiro na respectiva posição do aranjo que representa o heap neste EP.