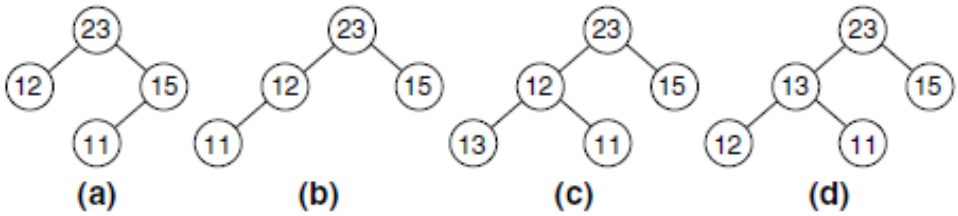
UENF

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Curso:	Ciência de Computação e Informática	**Data:**	19./.09./2022
Lista:	Questões sobre Heaps	**Período:**	4º
Disciplina:	Estrutura de Dados II		
Professor:	Fermín A. Tang	**Turno:**	Diurno

Heaps

1. Indique qual das estruturas são heaps e quais não. Em cada caso, explique porque:



Resposta 1.- Considerando heaps máximos.

(a) Não é *heap*, porque não satisfaz a condição de ser uma árvore quase completa.
(b) É um *Heap*, qualquer nó é sempre maior que os seus filhos. A estrutura é uma árvore quase completa, preenchida de esquerda para direita.
(c) Não é *heap*. Embora seja uma árvore quase completa, não satisfaz a condição de que para qualquer nó, ele sempre será maior que seus filhos. Falha na subárvore com raiz 12.

É um *Heap*, qualquer nó é sempre maior que os seus filhos. A estrutura é uma árvore quase completa, preenchida de esquerda para direita.

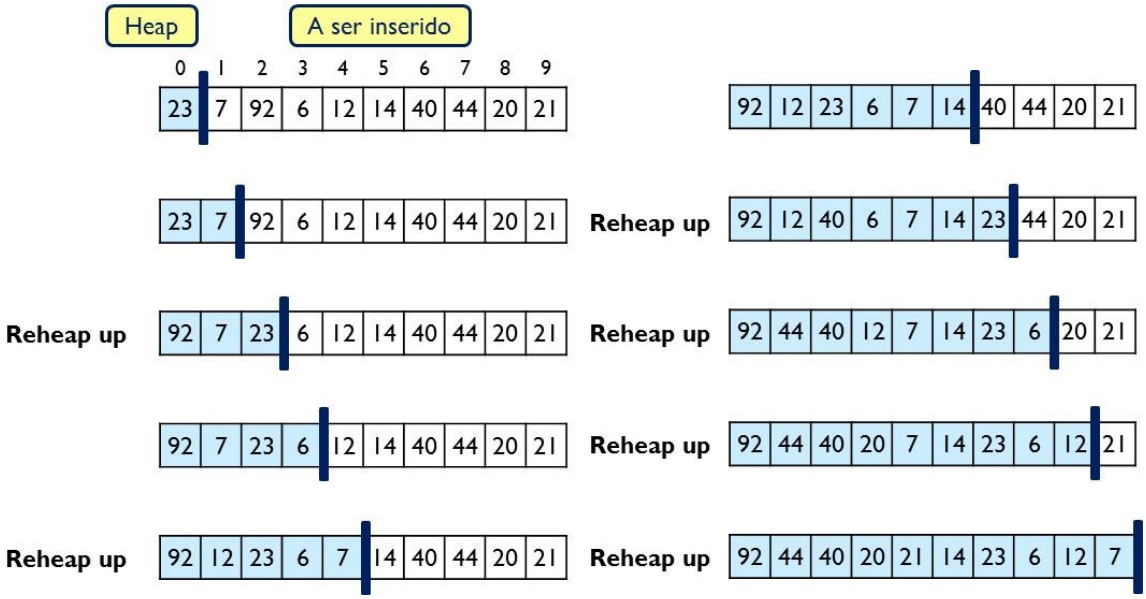
2. Construa um heap a partir da seguinte sequência de dados:

```
23 7 92 6 12 14 40 44 20 21
```

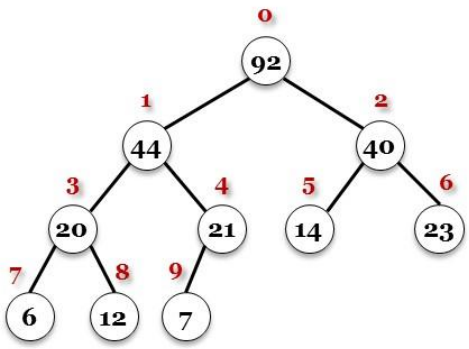
Resposta 2.- A construção de um *heap* pode ser realizada utilizando o algoritmo *build heap*. Inicialmente, considera-se que o *heap* compreende apenas um elemento e insere-se um elemento adicional de cada vez, aumentando o tamanho do *heap* em um, e executando o algoritmo de reconstrução (de baixo para cima) *reheap up*.

O algoritmo *reheap up* compara o último nó inserido com seu pai, e realiza a troca caso não respeitem a hierarquia do *heap*. O processo se repete em seguida com o nó pai até chegar na raiz. Vale lembrar que na representação de vetor, para um nó na posição i, o seu nó pai encontra-se na posição $\lfloor (i-1)/2 \rfloor$.

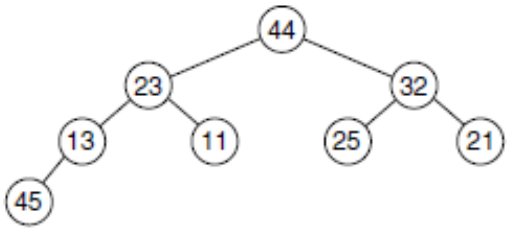
Para cada iteração, mostra-se o vetor que contém o *heap*:



O *heap* representado na forma de árvore é o seguinte:

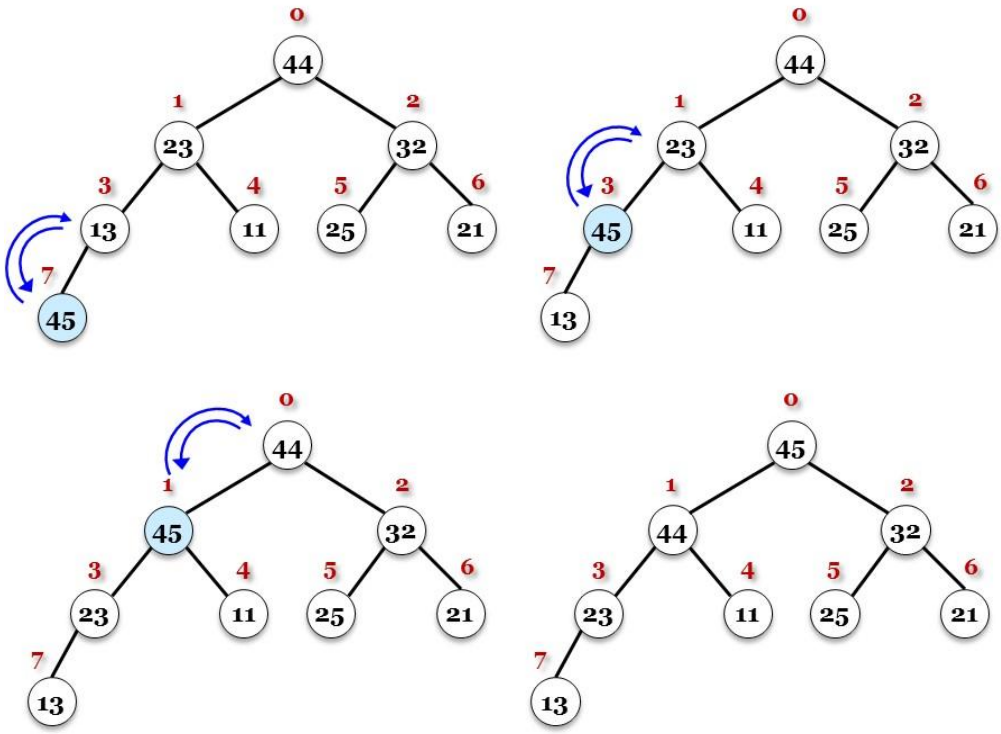


3. Aplique o algoritmo **reheap-up**, reconstrução ascendente, à estrutura mostrada na figura abaixo:

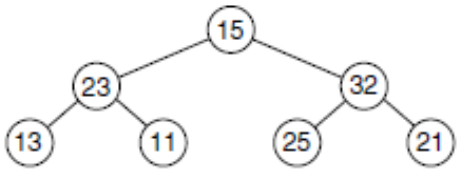


Resposta 3.- O algoritmo *reheap up* compara o último nó do *heap* com seu pai, e realiza a troca caso não respeitem a hierarquia do *heap*. O processo se repete em seguida com o nó pai até chegar na raiz. Vale lembrar que na representação de vetor, para um nó na posição i, o seu nó pai encontra-se na posição $\lfloor (i-1)/2 \rfloor$.

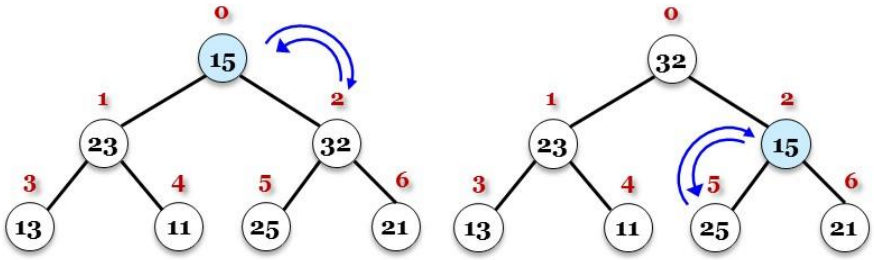
Assim temos:

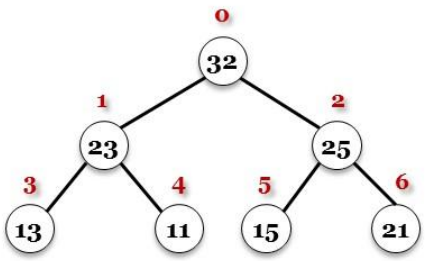


4. Aplique o algoritmo **reheap-down**, reconstrução descendente, à estrutura mostrada na figura abaixo:

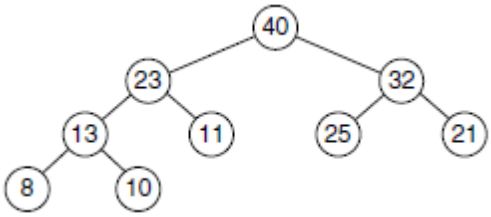


Resposta 4.- O algoritmo *reheap down* compara o nó raiz do *heap* com seu filho de maior valor e realiza a troca caso não respeitem a hierarquia do *heap*. O processo se repete em seguida com o nó filho até chegar a um nó folha. Vale lembrar que na representação de vetor, para um nó na posição i, o seu filho esquerdo encontra-se na posição 2i+1, enquanto o filho direito encontra-se na posição 2i+2. Assim temos:





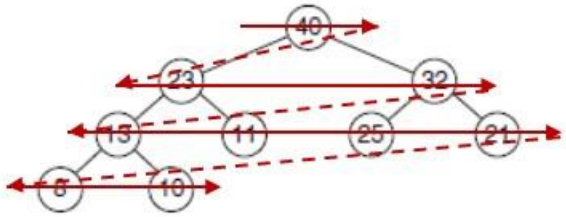
5. Mostre a representação do heap, mostrado na figura abaixo, na implementação de vetor.



Resposta 5.- A representação do *heap* usando um vetor é a seguinte:

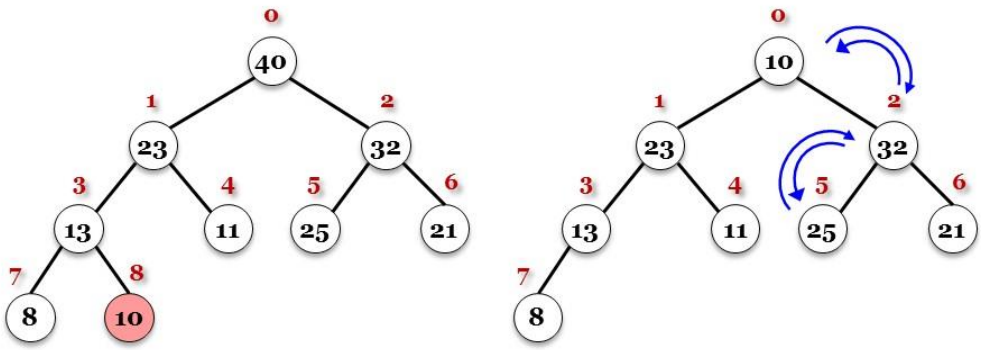
0	1	2	3	4	5	6	7	8
40	23	32	13	11	25	21	8	10

Observe que a árvore é lida de cima para baixo e de esquerda para direita, como mostrado na figura.

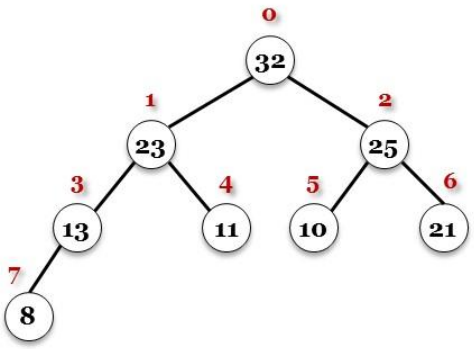


6. Aplique a operação de remoção ao heap da figura acima. Repare o heap após a remoção.

Resposta 6.- A remoção de um elemento do *heap* é uma operação restrita que sempre elimina o elemento no topo da árvore. Esta eliminação acontece por substituição. O elemento do topo é substituído pelo último elemento do *heap*. Temos assim:

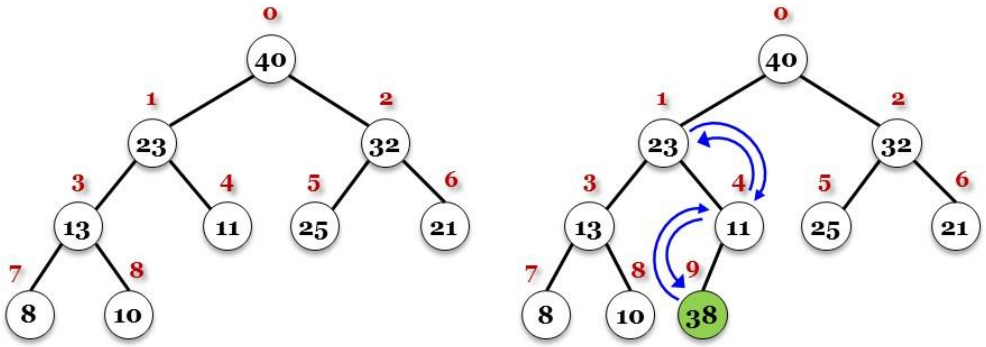


Após a substituição da raiz, aplica-se o algoritmo *reheap down*. Temos assim:

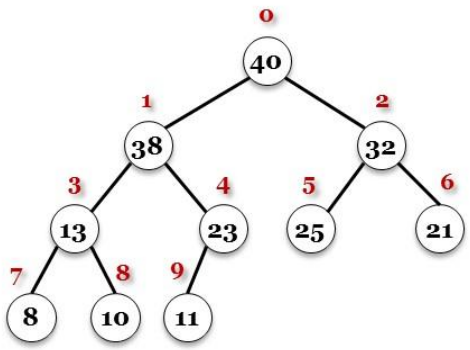


7. Insira o elemento com valor 38 no heap da questão 5. Repare o heap após a inserção.

   Resposta 7.- A inserção de um elemento no *heap* é uma operação restrita que sempre é realizada na primeira posição livre do último nível do *heap*. Na representação usando vetor, a inserção é realizada após a última posição ocupada. Temos assim:

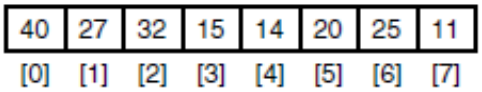


Após a inserção do novo nó, aplica-se o algoritmo *reheap up*. Temos assim:



8. No heap mostrado na figura abaixo, responda as seguintes questões:

   i) Quais são os filhos esquerdo e direito para os nós com valor 32 e 27;
   ii) Qual é o filho esquerdo para os nós com valor 14 e 40.



Resposta 8.- Na representação de um *heap* usando vetor, os filhos esquerdo e direito de um nó na posição i, se encontram nas posições 2i+1 e 2i+2, respectivamente.

i) Como o nó com valor 32 se encontra na posição 2, seus filhos esquerdo e direito se encontram nas posições, 5 e 6, respectivamente. O filho esquerdo tem valor 20 e o direito valor 25.
   Como o nó com valor 27 se encontra na posição 1, seus filhos esquerdo e direito se encontram nas posições, 3 e 4, respectivamente. O filho esquerdo tem valor 15 e o direito valor 14.

ii) O nó 14 não tem filho esquerdo;
   O nó 40 tem como filho esquerdo o nó 27.

9. No heap mostrado na figura anterior, indique:

   i) Qual é o pai do elemento com valor 11;
   ii) Qual é o pai do elemento com valor 20;
   iii) Qual é o pais do elemento com valor 25.

Resposta 9.- Na representação de um *heap* usando vetor, o nó pai de um nó na posição i, se encontram na posição $\lfloor (i-1)/2 \rfloor$. Assim:

i) Como o elemento com valor 11 se encontra na posição 7, o seu pai se encontra na posição $\lfloor (7-1)/2 \rfloor = 3$. O pai do elemento com valor 11 é o elemento com valor 15.
ii) Como o elemento com valor 20 se encontra na posição 5, o seu pai se encontra na posição $\lfloor (5-1)/2 \rfloor = 2$. O pai do elemento com valor 20 é o elemento com valor 32.
iii) Como o elemento com valor 25 se encontra na posição 6, o seu pai se encontra na posição $\lfloor (6-1)/2 \rfloor = \lfloor 2{,}5 \rfloor = 2$. O pai do elemento com valor 25 também é o elemento com valor 32.

10. Se o nó de um heap, implementado usando um vetor, se encontra no índice 25, considerando que os índices começam em 0, responda:

    i) Qual é o índice do seu filho direito?
    ii) Qual é o índice do seu filho esquerdo?

11. Se um nó se encontra no índice 37, responda quais são os índices de seus ancestrais até a raiz.

12. Responda qual das seguintes sequências são heaps:

    a. 42 35 37 20 14 18 7 10
    b. 42 35 18 20 14 30 10
    c. 20 20 20 20 20 20

13. Mostre quais elementos seriam eliminados do seguinte heap, após aplicar o algoritmo de remoção três vezes seguidas.

```
50 30 40 20 10 25 35 10 5
```

14. Mostre o heap resultante após inserir elementos com valores 33, 22 e 8 ao seguinte heap:

```
50 30 40 20 10 25 35 10 5
```

15. É possível a existência de uma árvore que seja ao mesmo tempo um heap e uma árvore de busca binária?. Caso exista desenhe essa árvore.

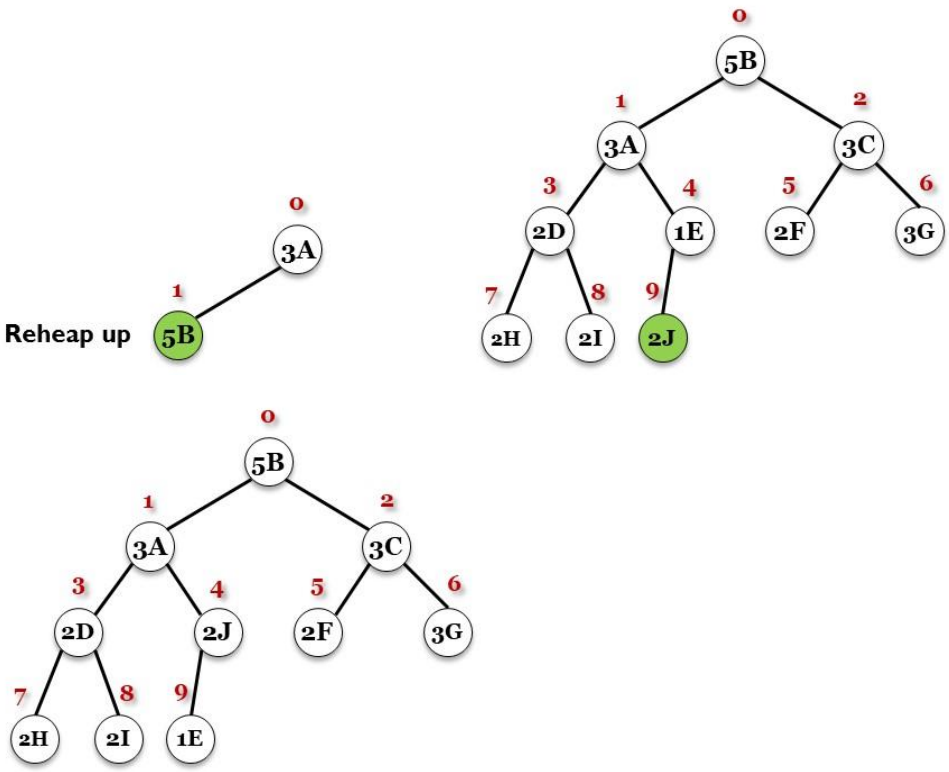
16. Crie uma fila de prioridade usando os seguintes dados:

```
3-A 5-B 3-C 2-D 1-E 2-F 3-G 2-H 2-I 2-J
```

    O primeiro número corresponde a prioridade e a letra corresponde ao dado. Considere que quanto maior o número maior a prioridade.

    Resposta 16.- Uma fila de prioridade pode ser representada usando *heaps*. Para isso é necessário utilizar dois códigos. Um primeiro código deve representar a prioridade do elemento na fila, enquanto que o segundo elemento deve representar a ordem de chegada dos elementos na fila. Um número representará a prioridade, enquanto que uma letra representa a ordem de chegada.

    Assume-se que quanto menor a letra mais cedo a chegada. A tarefa é construir um *heap*. Temos:



A fila de prioridade será obtida ao eliminar o elemento da raiz sucessivamente, e restaurar o *heap* usando *reheap down*.

17. Mostre o conteúdo da fila de prioridade no exercício 16, após eliminar três elementos da fila.

18. Mostre o conteúdo da fila de prioridade no exercício 16, após as seguintes operações:

    i) Inserir 4-K;
    ii) Inserir 3-L;
    iii) Remover;
    iv) Inserir 2-M;
    v) Remover.