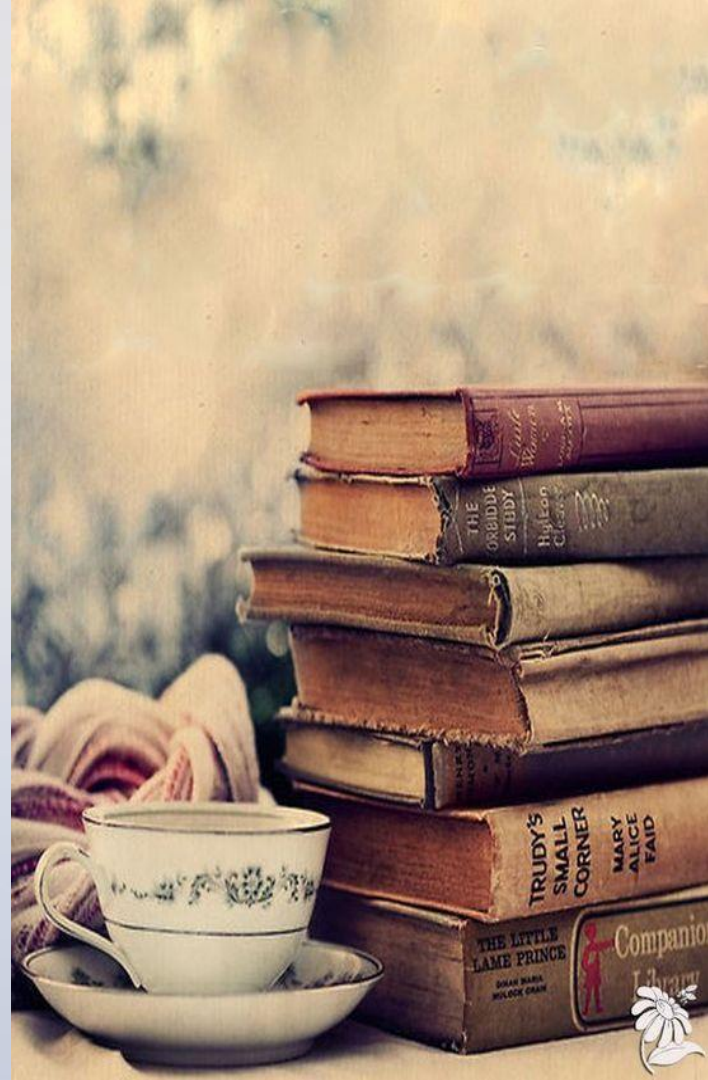


LABORATÓRIO DE PROGRAMAÇÃO II

Fila de Prioridades - Heap





Agenda

Objetivos: Realizar operações de armazenamento e de recuperação de dados em filas de prioridades



Fila de Prioridades	
Heap e Árvore Binária	
Inserção e Remoção em Heap	
Propriedades de um Heap	
Exemplo de Inserção/Remoção em Heap	
Condições	

Mecanismos de Acesso à Dados

Há 4 mecanismos que controlam o modo como os dados podem ser acessados por um programa de computador

A escolha do mecanismo depende do problema abordado

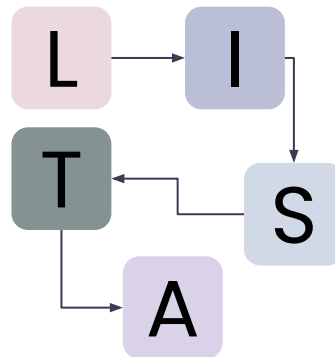
Pilha



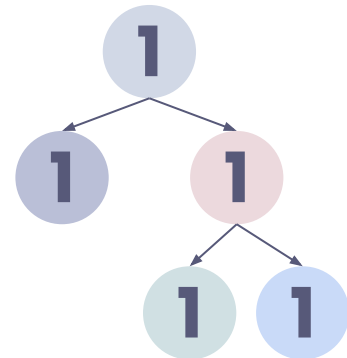
Fila



Lista Encadeada



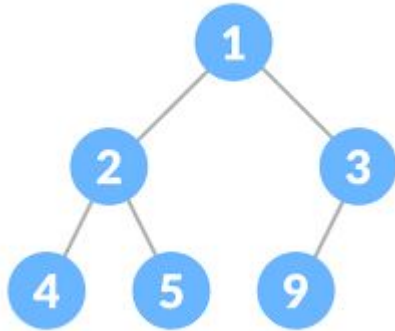
Árvore Binária



...

2 – Parte 2

Heap



Fila de prioridades

Uma fila é uma estrutura que segue a política **FIFO**, tal como uma fila de banco ou de supermercado.

Entretanto, pessoas idosas ou gestantes, por exemplo, possuem prioridade no atendimento em uma fila, bem como pacientes que chegam a um hospital e passam pela triagem. Logo, o mecanismo FIFO não deve ser adotado neste tipo de fila.

Este tipo de fila é chamado de fila de prioridades.

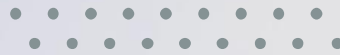


ATENDIMENTO PRIORITÁRIO



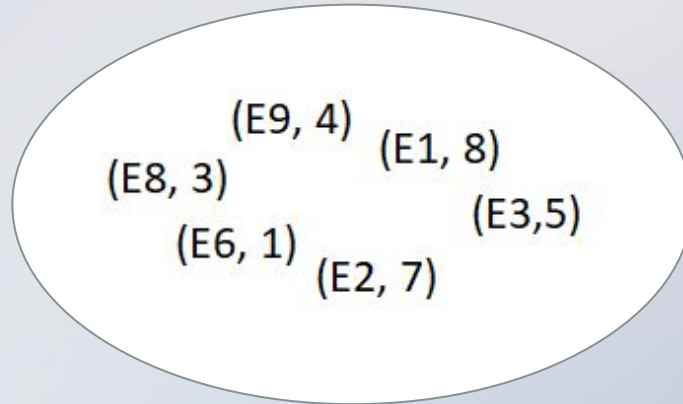
60⁺





Fila de Prioridades


Uma fila de prioridades é uma coleção de **e**lementos com as respectivas prioridades (**E,P**), tal que a ordem de acesso a eles é definida pela prioridade dos mesmos.





Fila de prioridades

Algumas aplicações:

- Sistemas operacionais mantêm uma fila de prioridade de processos a serem executados
 - Algoritmo de ordenação heapsort
 - Pessoas em uma fila de atendimento
 - Sempre que for necessário remover repetidamente o elemento com prioridade
- 



Fila de prioridades

Uma **fila de prioridades** pode ser representada por uma estrutura linear, tal como um vetor.

5	8	10	12	18	22	25	28
0	1	2	3	4	5	6	7

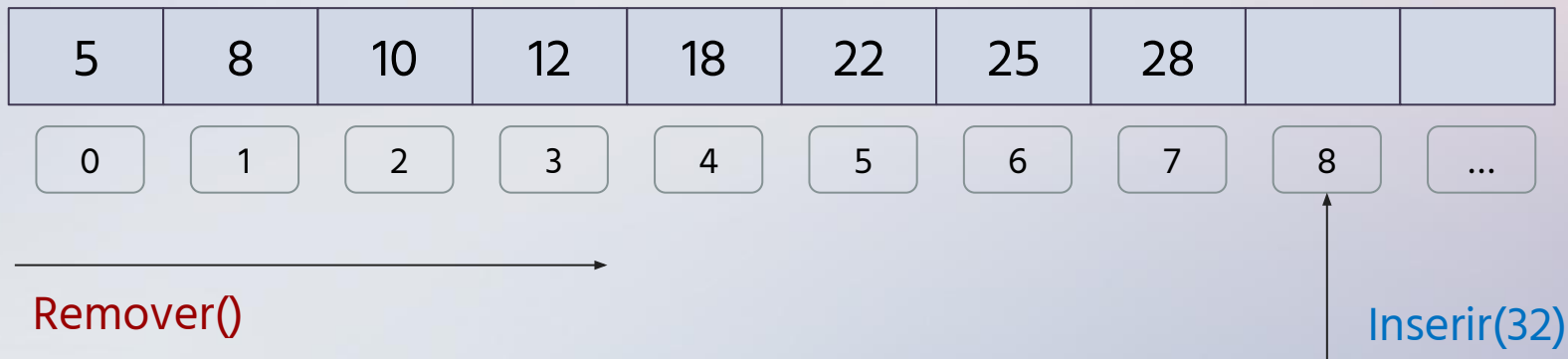
As prioridades são representadas por números inteiros e positivos e, dependendo do problema, as prioridades podem ser as de maior valor (ordem decrescente) ou as de menor valor (ordem crescente).

As principais operações sobre filas de prioridades são *inserir* e *remover* seus elementos.



Fila de prioridades

Como podem ser inseridas/removidas as prioridades em uma fila representada por uma estrutura linear?

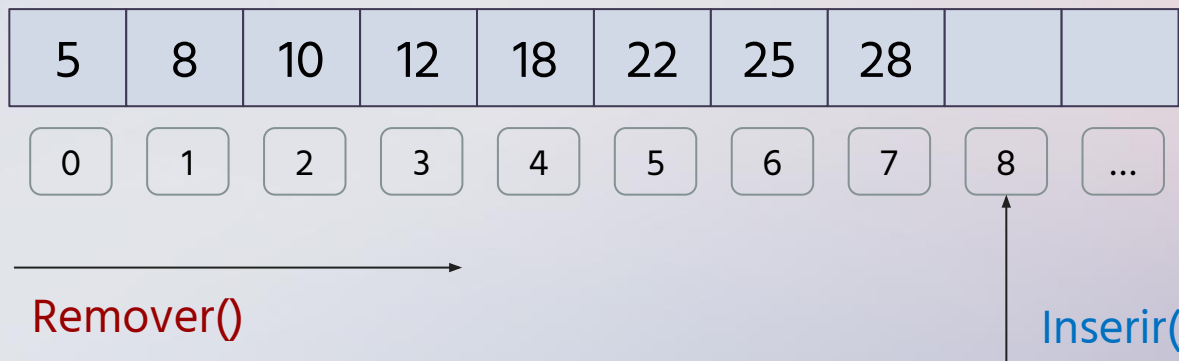


Fila de prioridades

Se a **inserção** for na próxima posição livre, há custo de uma operação, apenas.

Para a **remoção**, seria necessário percorrer essa estrutura, a partir da primeira posição, o que requer $(n-1)$ operações de comparação.

Como podem ser inseridas/removidas as prioridades em uma fila representada por uma estrutura linear?

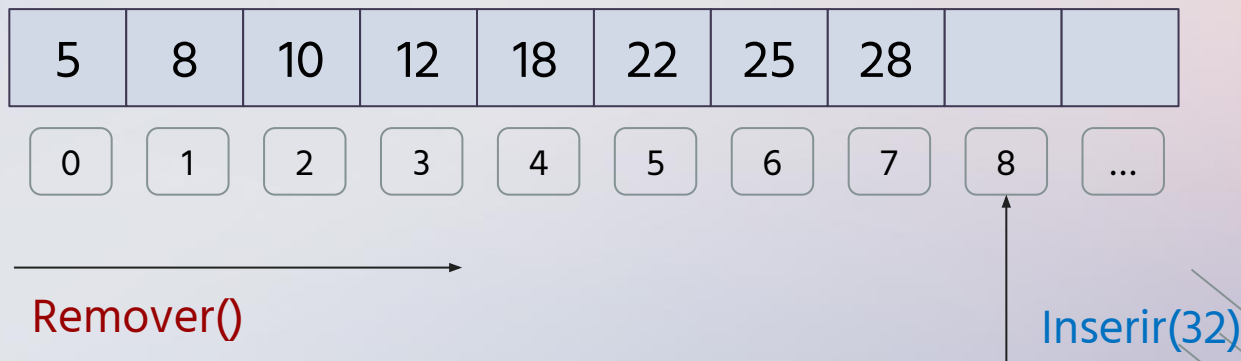


Fila de prioridades

Se a **inserção** for na próxima posição livre, há custo de uma operação, apenas.

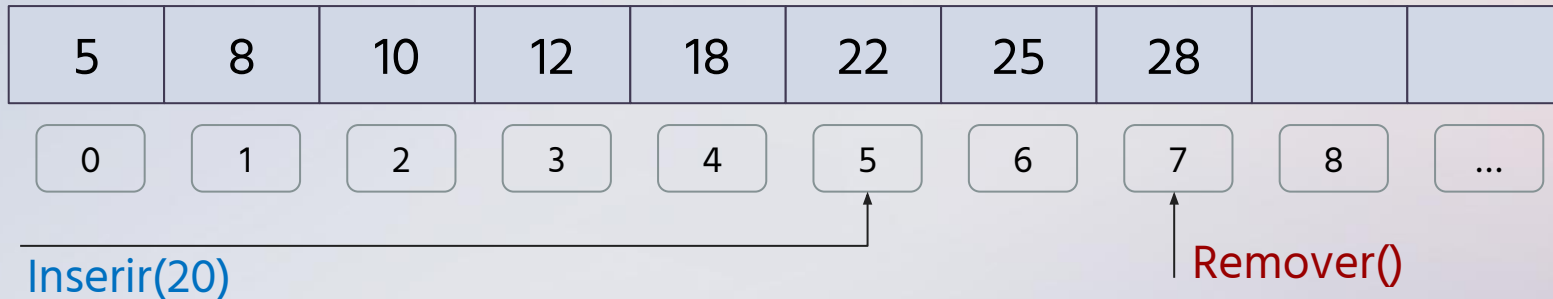
Para a **remoção**, seria necessário percorrer essa estrutura, a partir da primeira posição, o que requer (n-1) operações de comparação.

Como podem ser inseridas/removidas as prioridades em uma fila representada por uma estrutura linear?



Uma possibilidade de reduzir o número de comparações seria aplicar um algoritmo de ordenação eficiente para cada prioridade adicionada à estrutura, e a remoção teria custo 1.

Fila de prioridades



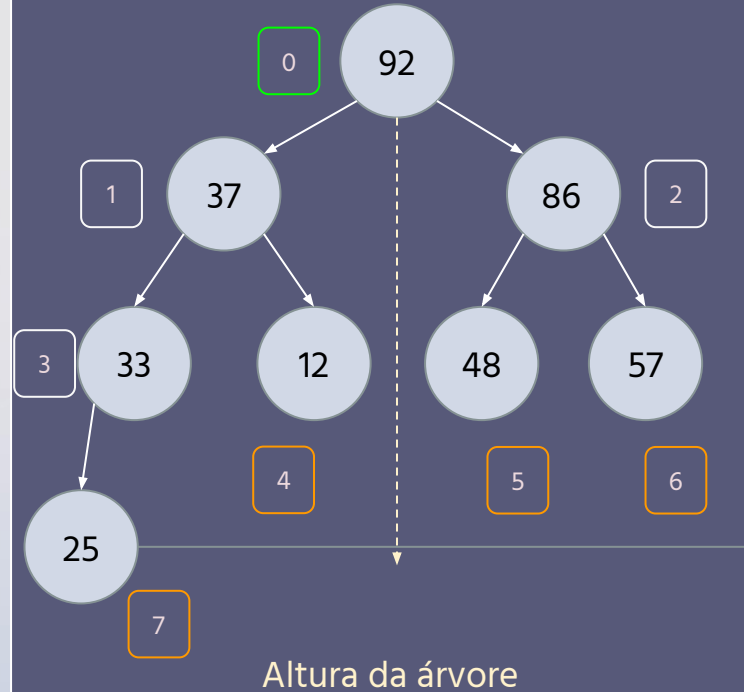
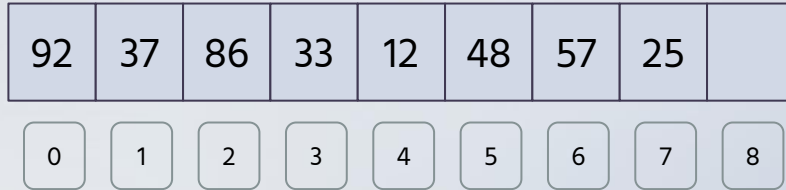
Outra possibilidade é a **inserção** de cada elemento em ordem de valor.

Esta operação requer até n comparações e o possível deslocamento de elementos na estrutura para ceder espaço ao novo elemento e, a **remoção** teria o custo de uma única operação (remoção do último elemento)



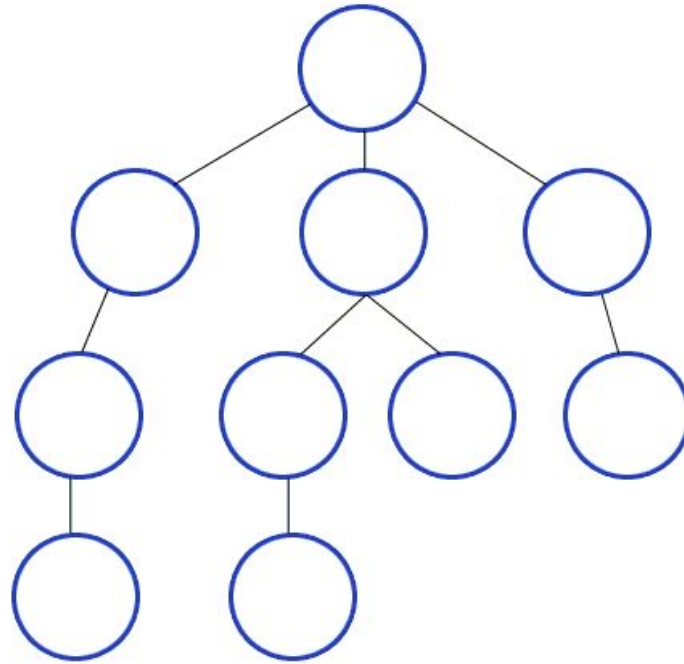
Fila de prioridades – HEAP

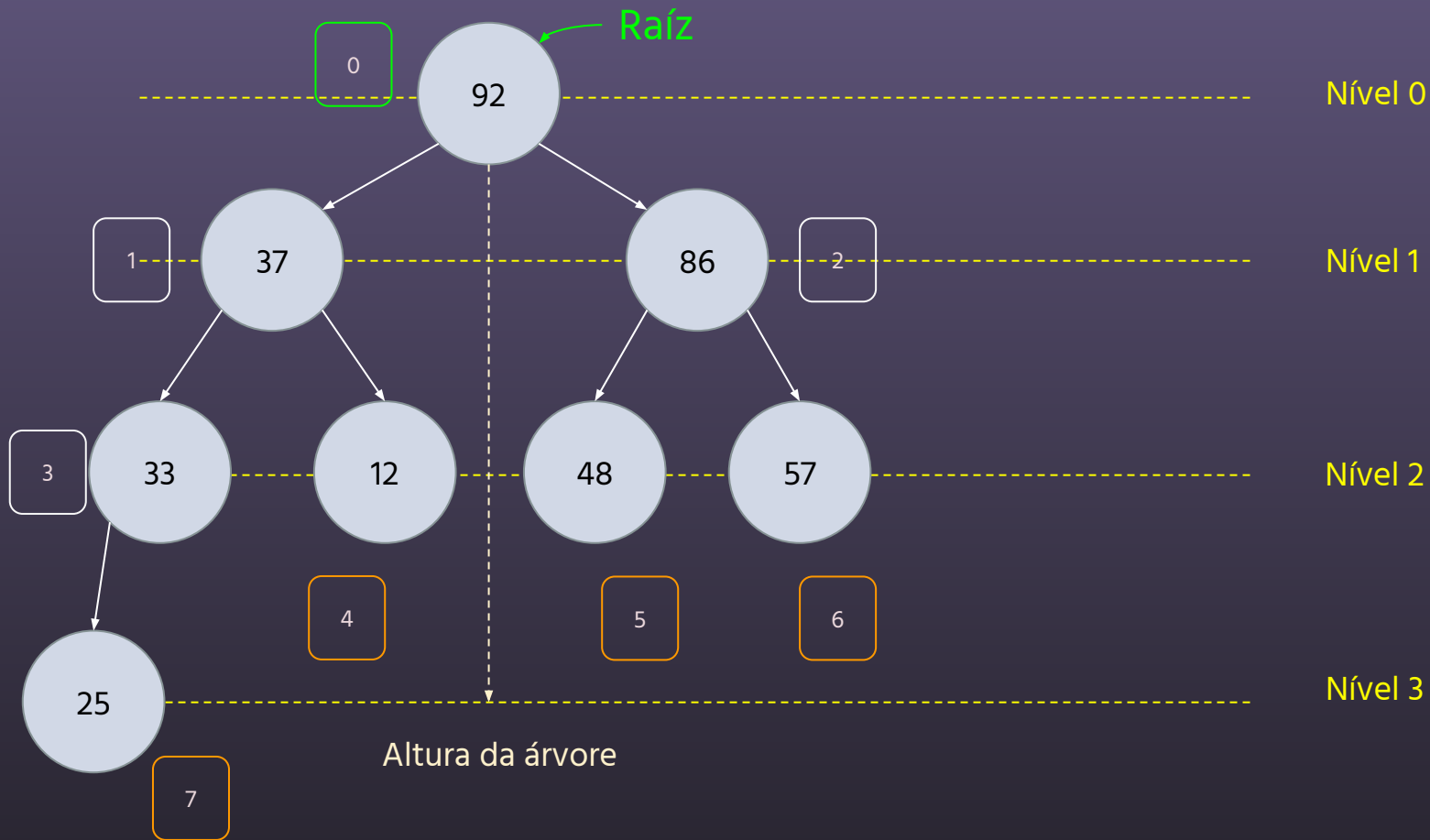
Outra maneira de implementar uma fila de prioridades é utilizar a estrutura de dados denominada heap, que é uma estrutura linear cujos elementos (prioridades) são **manipulados seguindo a estrutura de uma árvore binária** (não requer implementar a árvore).



Árvore binária obtida a partir do heap. A ordem dos elementos é verificada pela busca em largura

Busca em largura





Fila de prioridades – HEAP

Acesso aos nós (pai e seus filhos) na árvore:

- Relativamente ao índice i , tem-se:

$$\text{pai} = (i-1)/2,$$

$$\text{filho à esquerda} = 2*i+1,$$

$$\text{filho à direita} = 2*(i+1)$$

92	37	86	33	12	48	57	25
0	1	2	3	4	5	6	7



Fila de prioridades – HEAP

0	1	2	3	4	5	6	7
92	37	86	33	12	48	57	25

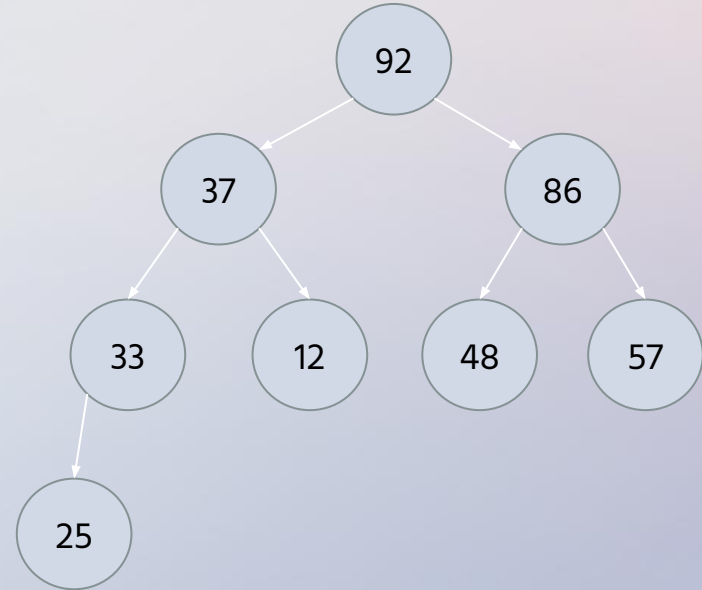
Acesso aos nós (pai e seus filhos)
na árvore:

- Relativamente ao índice i ,
tem-se:

$$\text{pai} = (i-1)/2,$$

$$\text{filho à esquerda} = 2*i+1,$$

$$\text{filho à direita} = 2*(i+1)$$



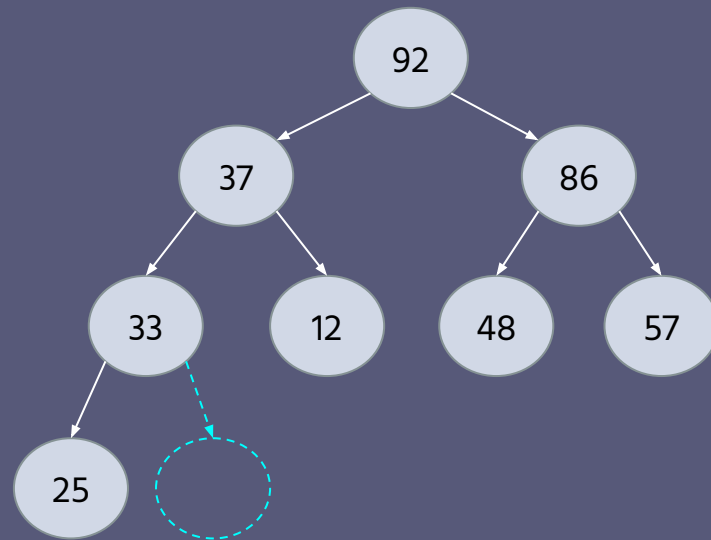
Fila de prioridades – HEAP

O nome “heap”, “monte” em inglês, faz referência à abstração da estrutura de dados não linear chamada árvore, tal que a prioridade a ser *removida* do heap está sempre na raiz da árvore (primeira posição do heap), enquanto a *inserção* ocorre na próxima posição livre do heap.



Fila de prioridades – HEAP

Como **inserir** um elemento em um heap? O elemento é inserido **na próxima posição livre no heap**, o que produz custo de 1 operação.



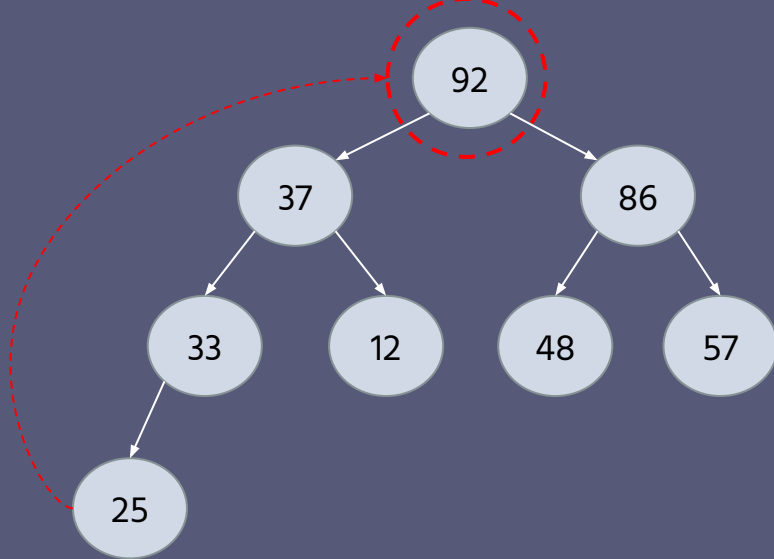
A inserção na próxima posição livre do heap garante que a árvore se mantenha completa, ou quase completa da esquerda para a direita. Por que isso é importante?

Porque quanto mais próximo a árvore estiver de sua altura mínima, mais eficientes são as operações sobre ela

Fila de prioridades – HEAP

Como **remover** um elemento em um heap? O elemento é removido da raiz (primeira posição no heap), o que produz custo de 1 operação. A nova raiz será o último elemento inserido no heap

92	37	86	33	12	48	57	25	
0	1	2	3	4	5	6	7	8



Deslocar o último elemento para o topo da árvore mantém a árvore completa, ou quase completa da esquerda para a direita, garantindo a altura mínima da árvore, e libera a última posição ocupada no heap para inserir um novo elemento com custo de uma única operação.