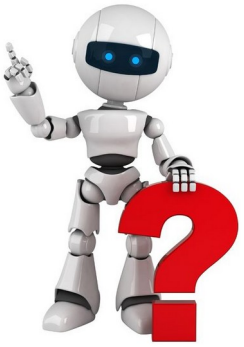
Algoritmos e Estruturas de Dados III

“Lista de Prioridades - Heap”

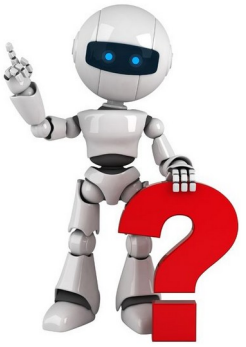
Prof. Dr. Felipe Oliveira

Lista de Prioridades



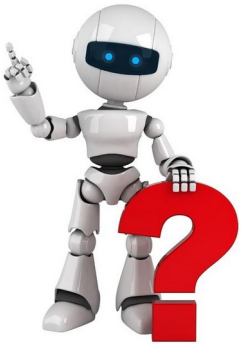
- Em algumas **aplicações**, **dados** de coleções são **acessados** por **ordem de prioridade**:
  - Fila de impressão, escalonamento de tarefas, simulações;
- A **prioridade** associada a um dado pode ser **qualquer coisa**:
  - Tempo, custo, distância (valor escalar) …

Lista de Prioridades



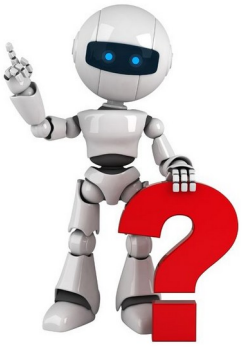
- **Operações** que devem ser **eficientes**:
  - **Seleção** do elemento com **maior/menor prioridade;**
  - **Remoção** do elemento com **maior/menor prioridade;**
  - **Alteração** da **prioridade** de um elemento;
  - **Inserção** de um novo elemento;
  - **Construção** de uma lista de prioridades.

Lista de Prioridades - Heap



- O **Heap** é uma **estrutura de dados** que implementa uma **lista de prioridades**, por meio de um conjunto de operações:
  - **Seleção** (da maior prioridade);
  - **Remoção** (da maior prioridade);
  - **Alteração** (da prioridade);
  - **Inserção** elemento;
  - **Construção** de lista (A partir de dados).

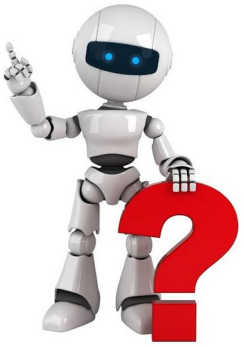
Lista de Prioridades - Heap



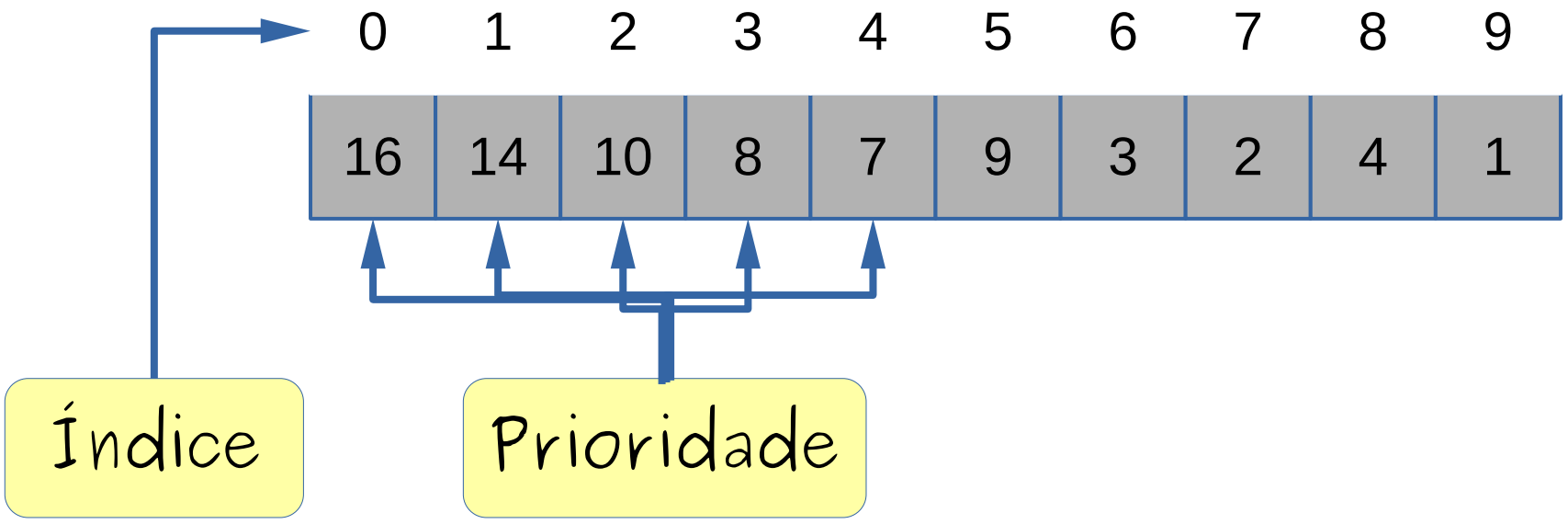
- O **Heap** é uma **estrutura de dados** que implementa uma **lista de prioridades**, por meio de um conjunto de operações:

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 (do menor)	O(n)	O(1)	O(log n)	O(log n)
Alteração de prioridade	O(n)	O(n)	O(log n)	O(log n)
Construção	O(n)	O(n log n)	O(n log n)	O(n)

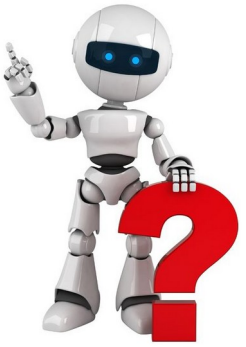
Lista de Prioridades - Heap



- **Heaps** são implementados usando **vetores** ou listas sequenciais.



Lista de Prioridades - Heap

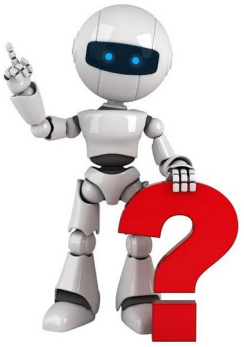


- **Heaps** são implementados usando **vetores** ou listas sequenciais.

0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

- A relação entre as chaves pode ser **MODELADA** por **árvores binárias:**
  - Mas **não** é **implementado** por **AB** e nem **ABB**

Lista de Prioridades - Heap

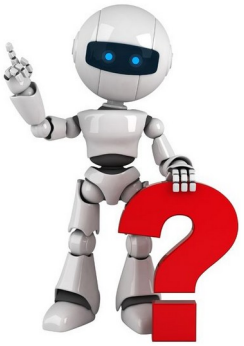


- **Heaps** são implementados usando **vetores** ou listas sequenciais.

Modelar o conjunto de dados como **árvore binária** ajuda na **compreensão** e na **verificação das propriedades** do Heap

- Mas **não** é **implementado** por **AB** e nem **ABB**

Lista de Prioridades - Heap

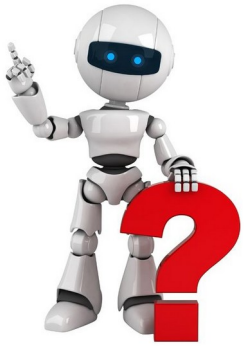


- **Heaps** são implementados usando **vetores** ou listas sequenciais.

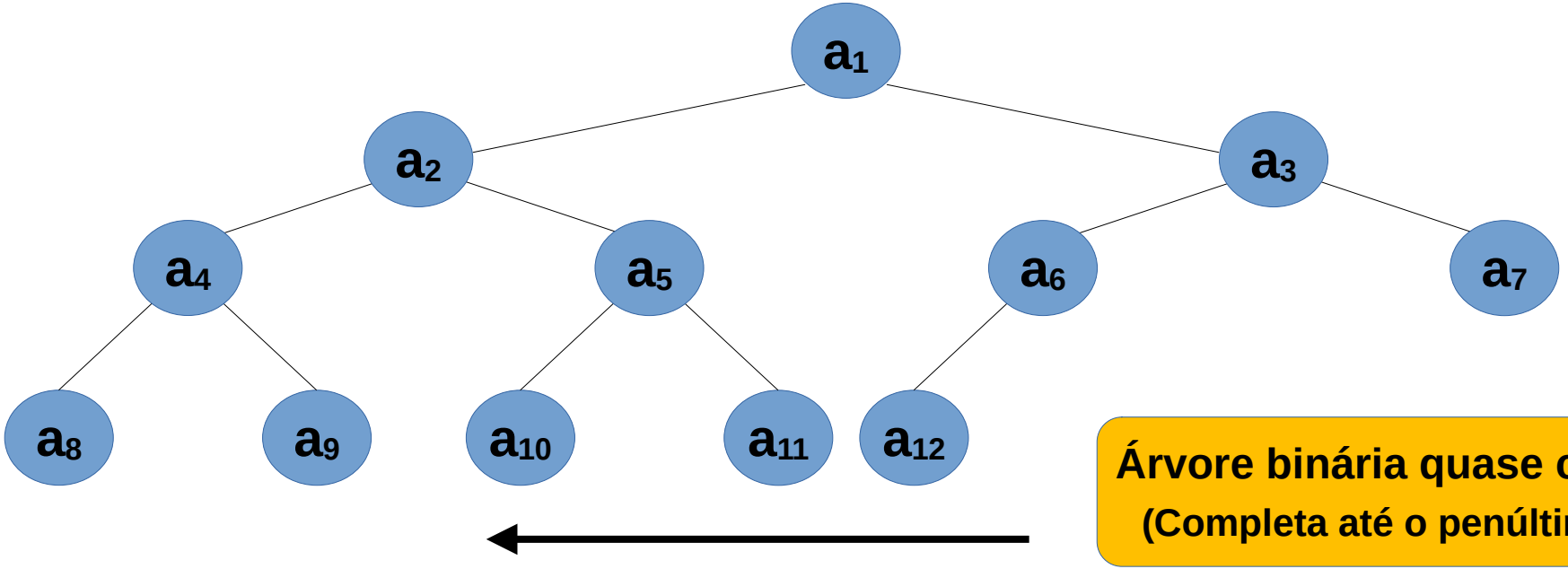
Sim!!! O Heap tem propriedades para verificar.

- Mas **não** é **implementado** por **AB** e nem **ABB**

Lista de Prioridades - Heap

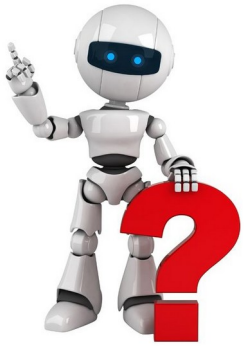


0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

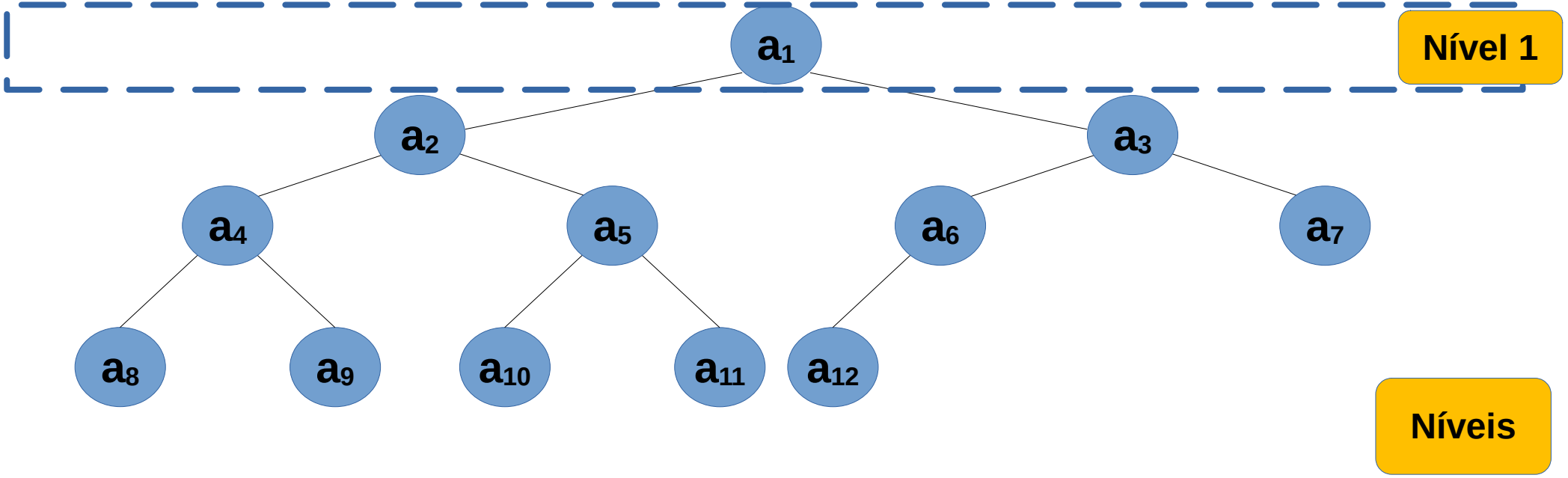


Árvore binária quase completa.
(Completa até o penúltimo nível)

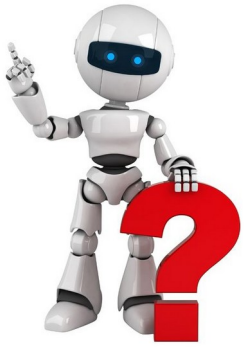
Lista de Prioridades - Heap



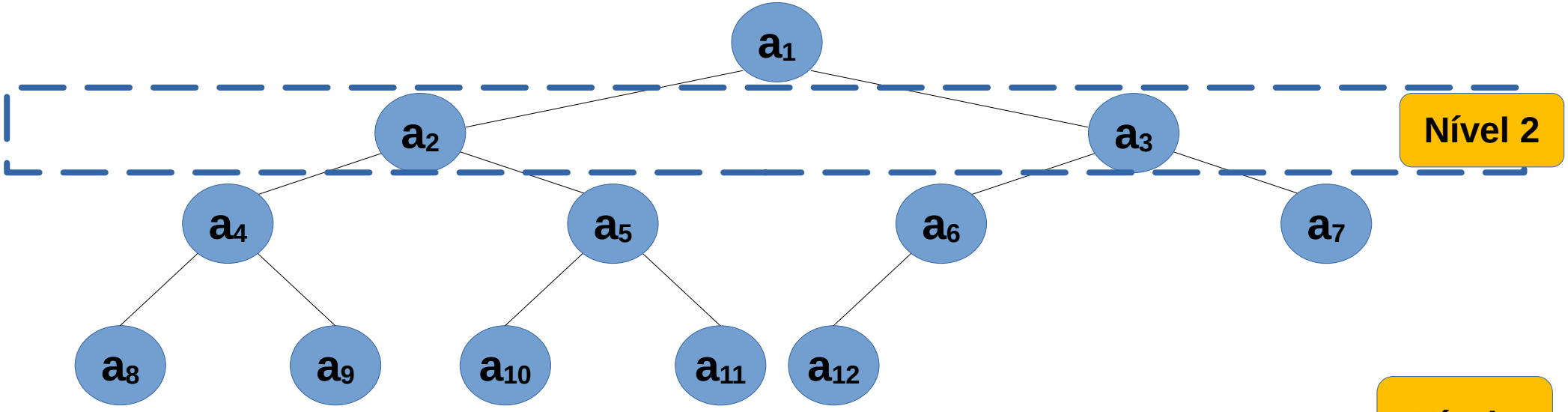
0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}



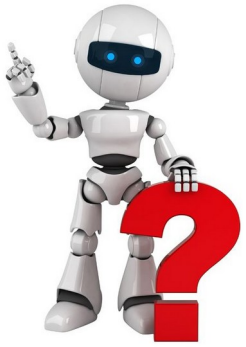
Lista de Prioridades - Heap



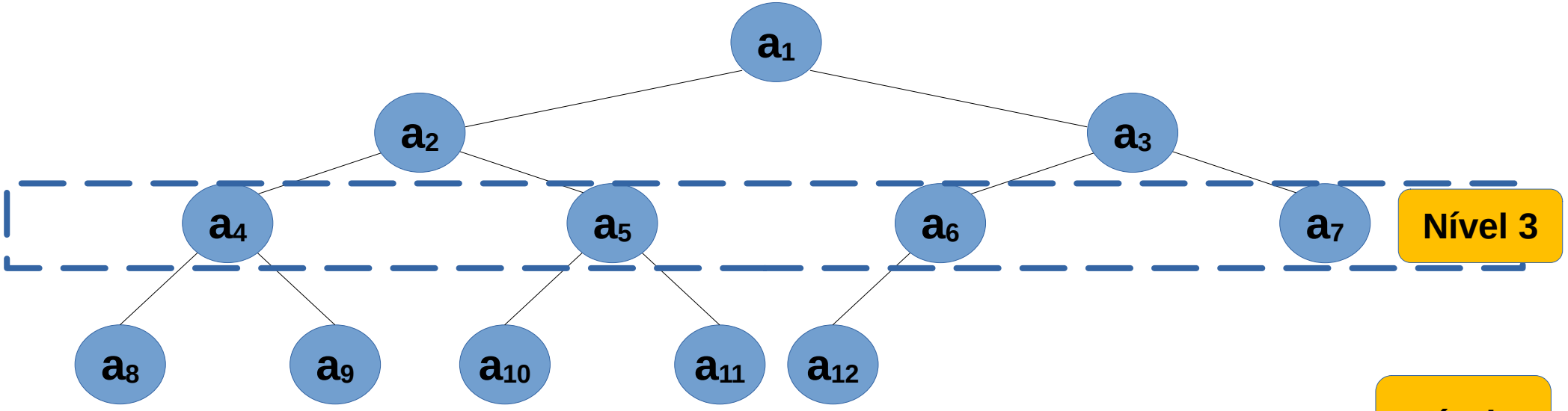
0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}



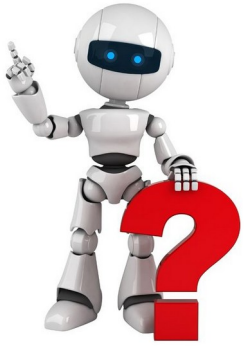
Lista de Prioridades - Heap



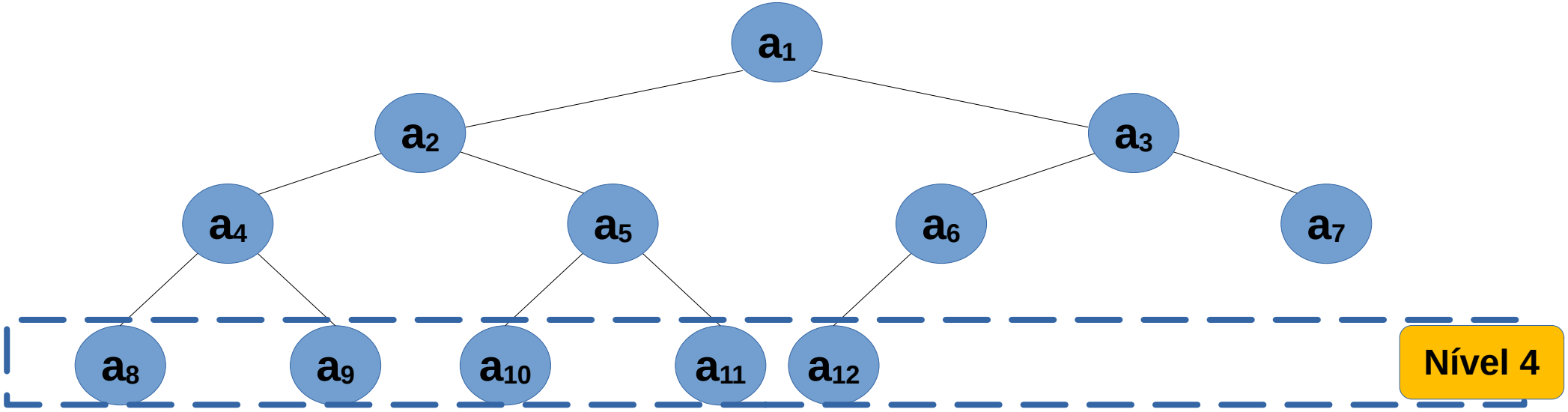
0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}



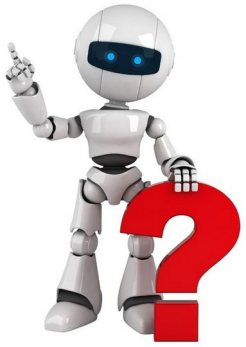
Lista de Prioridades - Heap



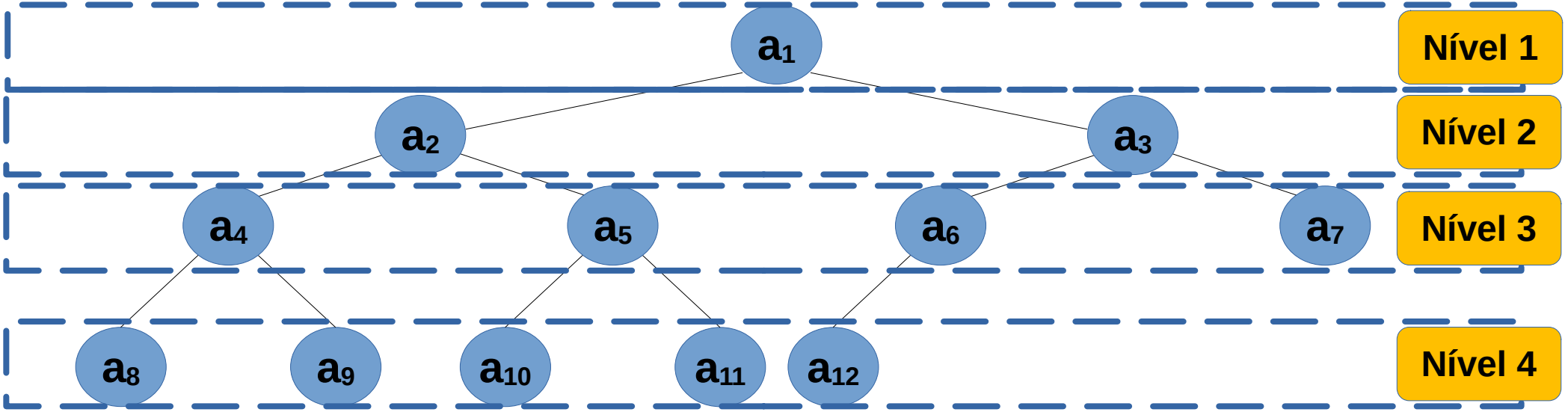
0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}



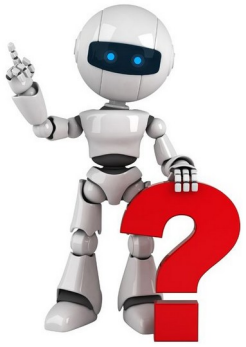
Lista de Prioridades - Heap



0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

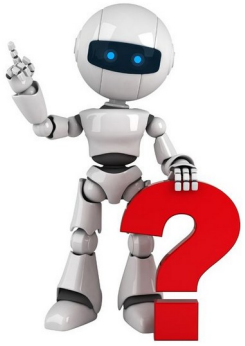


Lista de Prioridades - Heap

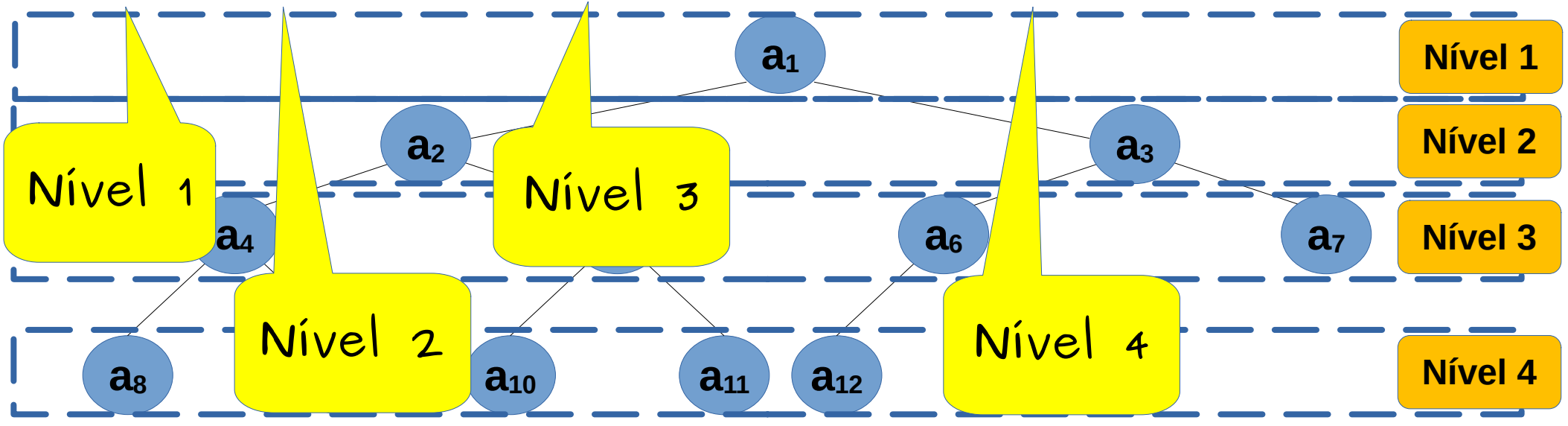


0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

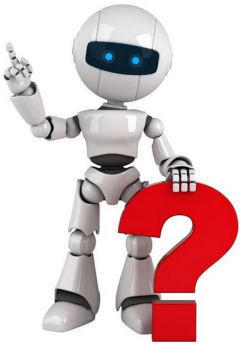
Lista de Prioridades - Heap



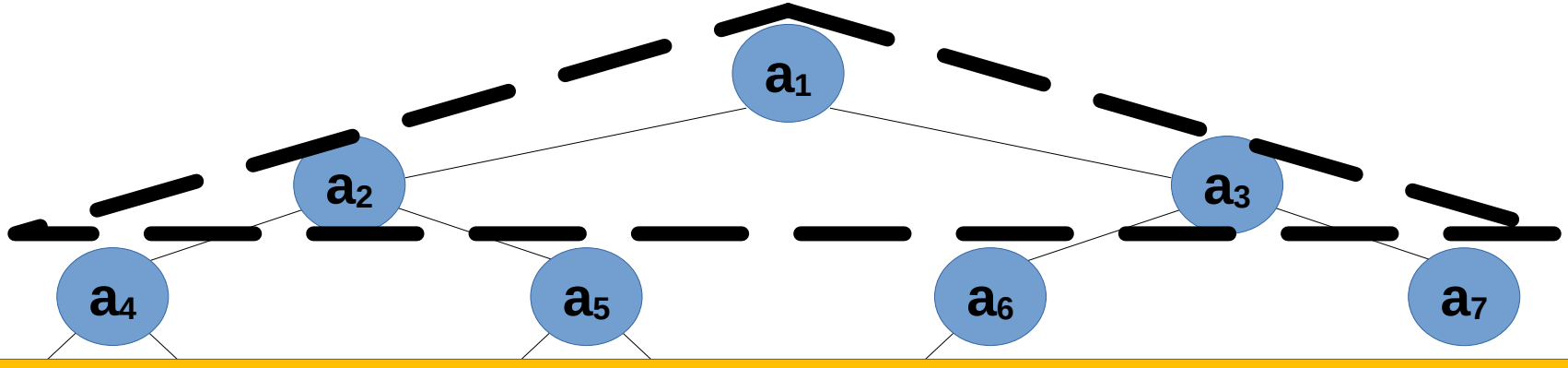
0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}



Lista de Prioridades - Heap

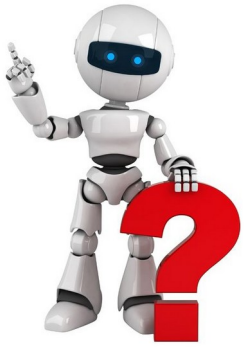


0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

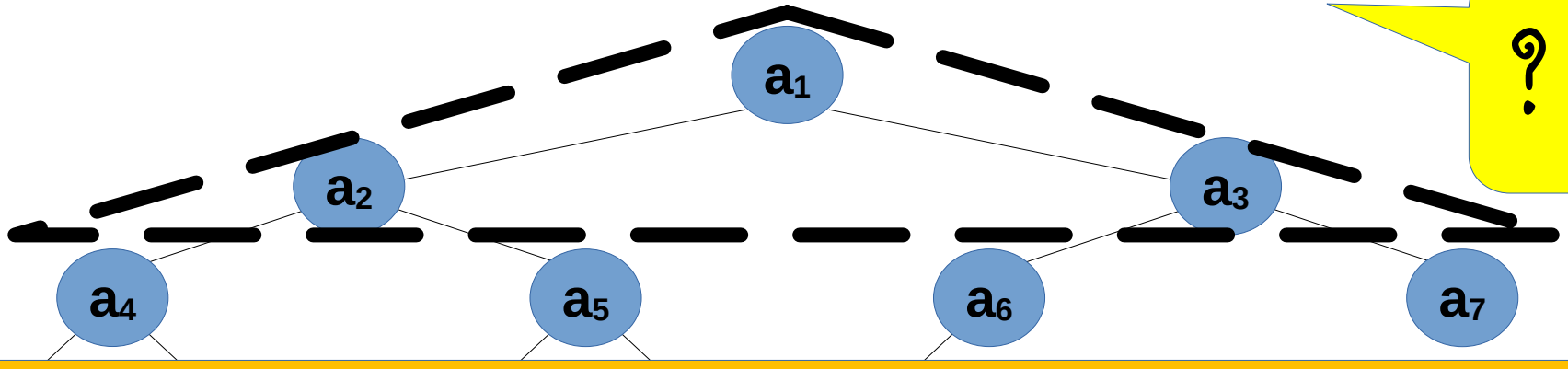


- **Chave do nó i > chave à esquerda (se houver);**
- **Chave do nó i > chave à direita (se houver);**
- **A raiz contém a chave de maior prioridade.**

Lista de Prioridades - Heap

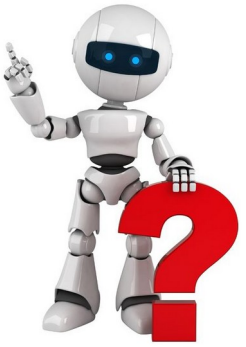


0	1	2	3	4	5	6	7	8	9	10	11
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

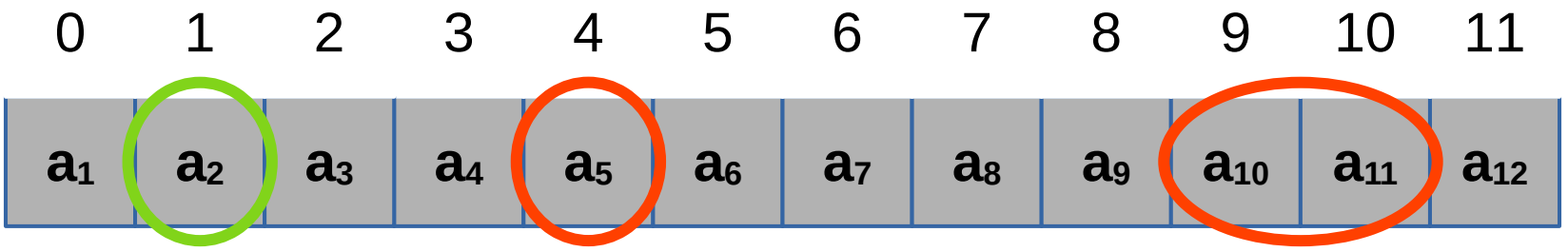


- **Chave do nó i > chave à esquerda (se houver);**
- **Chave do nó i > chave à direita (se houver);**
- **A raiz contém a chave de maior prioridade.**

Propriedades Heap



- **Heaps** são implementados usando **vetores** ou listas sequenciais.



- Dado um nó armazenado no índice i, o índice
  - **Filho esquerdo** de i >> 2i
  - **Filho direito** de i >> 2i + 1
  - Nó **pai** de i >> i div 2

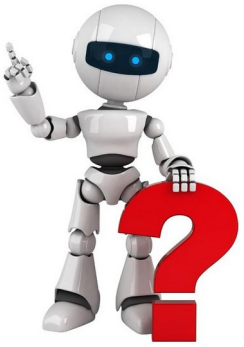
Propriedades Heap

Em um **heap**, **todo nó** deve ter **prioridade maior ou igual** à prioridade de seus **filhos**, se eles existirem.

- Dado um nó armazenado no índice i, o índice
  - **Filho esquerdo** de i >> 2i
  - **Filho direito** de i >> 2i + 1
  - Nó **pai** de i >> i div 2

Exercícios

Propriedades Heap
Exercícios:



Os vetores são heap?

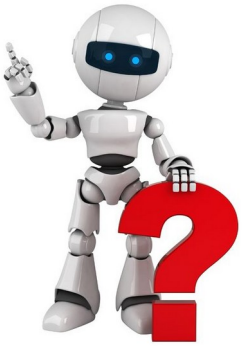
- A_1 =

0	1	2	3	4	5	6	7	8	9
16	14	10	8	7	9	3	2	4	1

- A_2 =

0	1	2	3	4	5	6	7	8	9
15	7	10	8	14	9	3	2	4	1

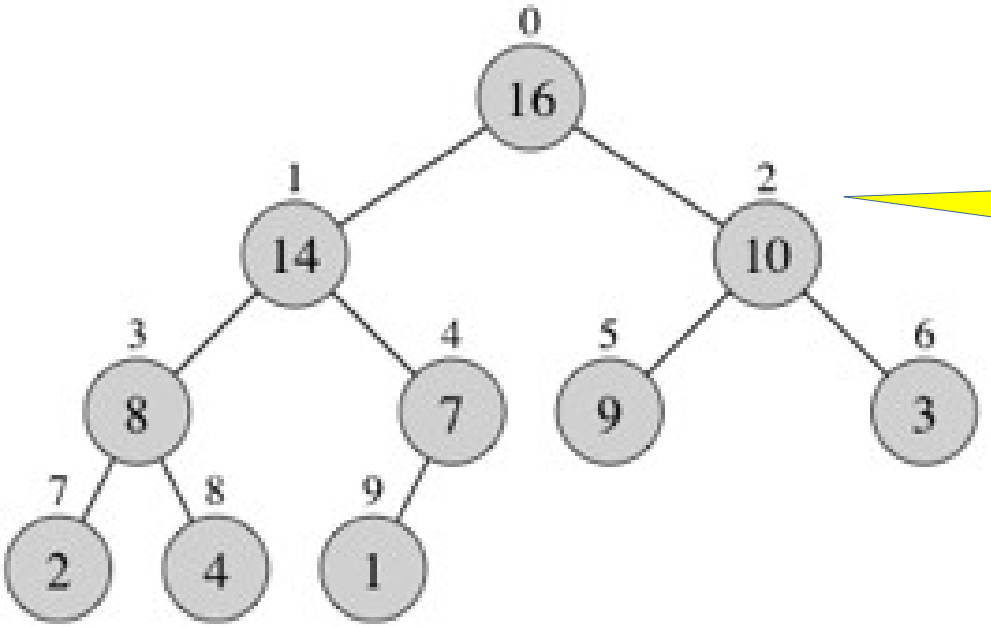
Propriedades Heap
Exercícios:



Os vetores são heap?

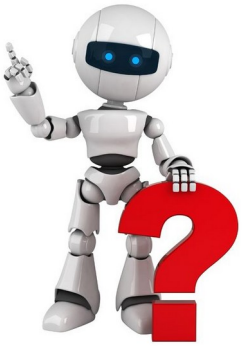
- A_1 =

0	1	2	3	4	5	6	7	8	9
16	14	10	8	7	9	3	2	4	1



Sim

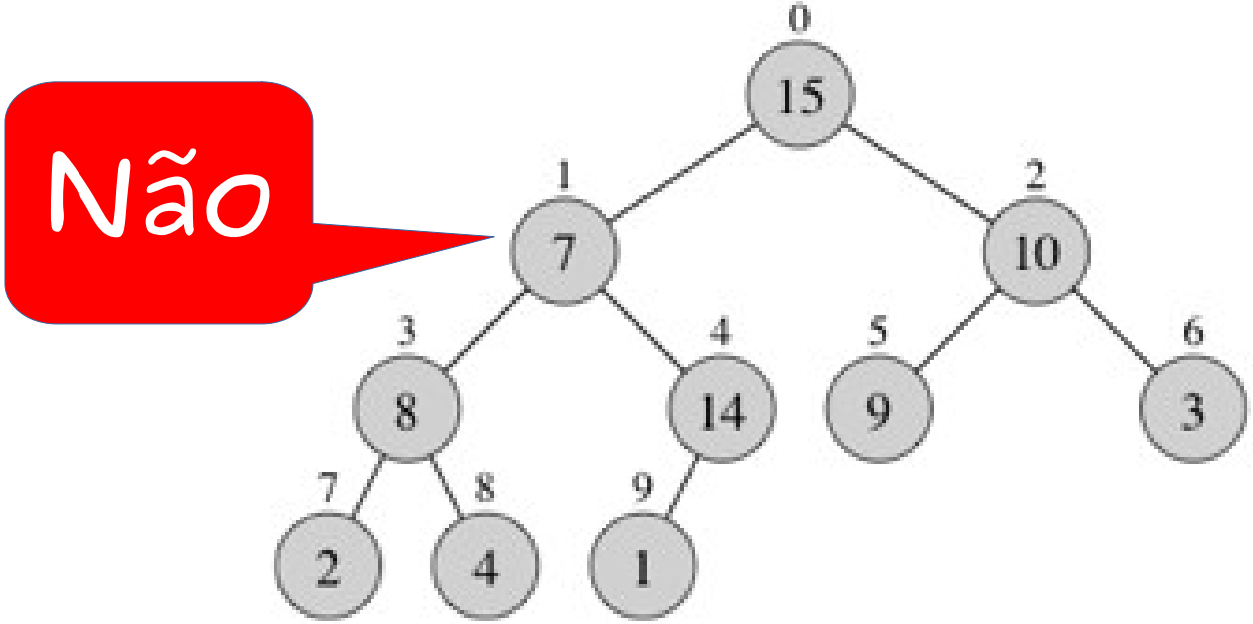
Propriedades Heap
Exercícios:



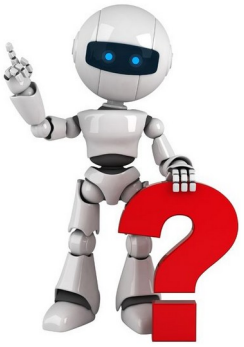
Os vetores são heap?

- A_2 =

0	1	2	3	4	5	6	7	8	9
15	7	10	8	14	9	3	2	4	1



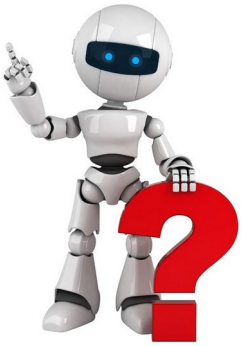
Propriedades Heap Exercícios:



Os vetores são heap?

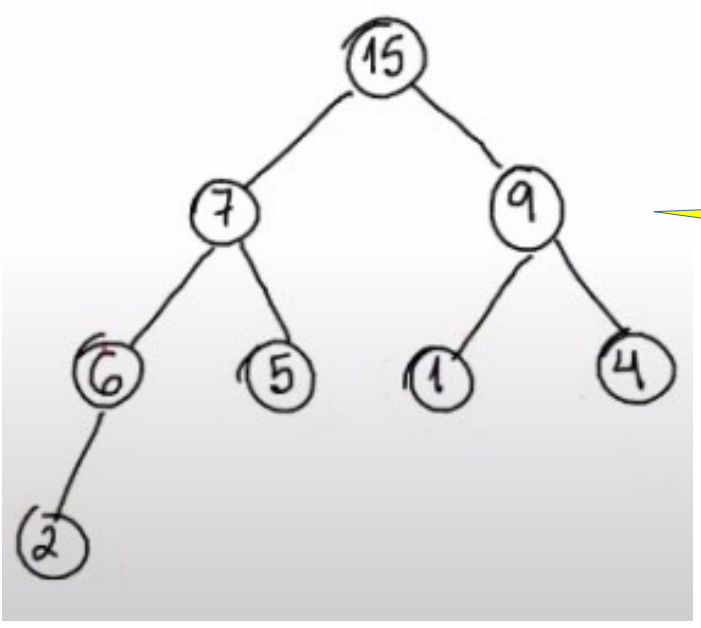
- A_1 = (15, 7, 9, 6, 5, 1, 4, 2)
- A_2 = (15, 7, 9, 8, 5, 10, 4, 2)

Propriedades Heap
Exercícios:



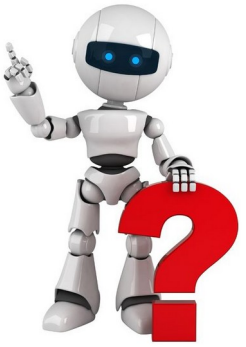
Os vetores são heap?

- A_1 = (15, 7, 9, 6, 5, 1, 4, 2)



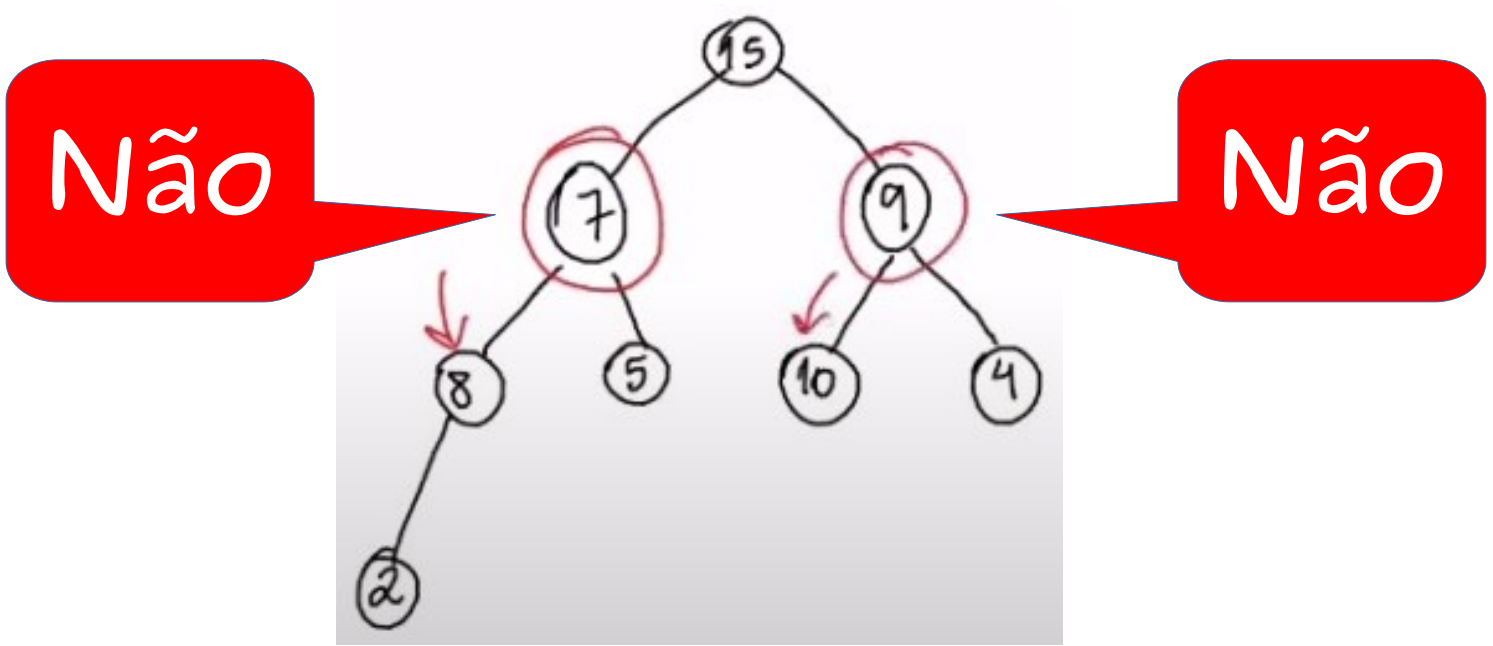
Sim

Propriedades Heap
Exercícios:

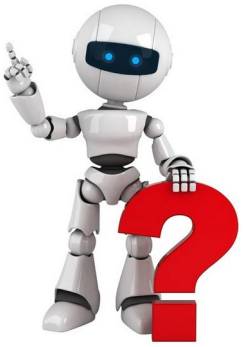


Os vetores são heap?

- A_2 = (15, 7, 9, 8, 5, 10, 4, 2)



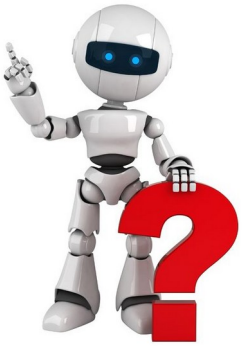
Propriedades Heap Exercícios:



O vetor é heap?

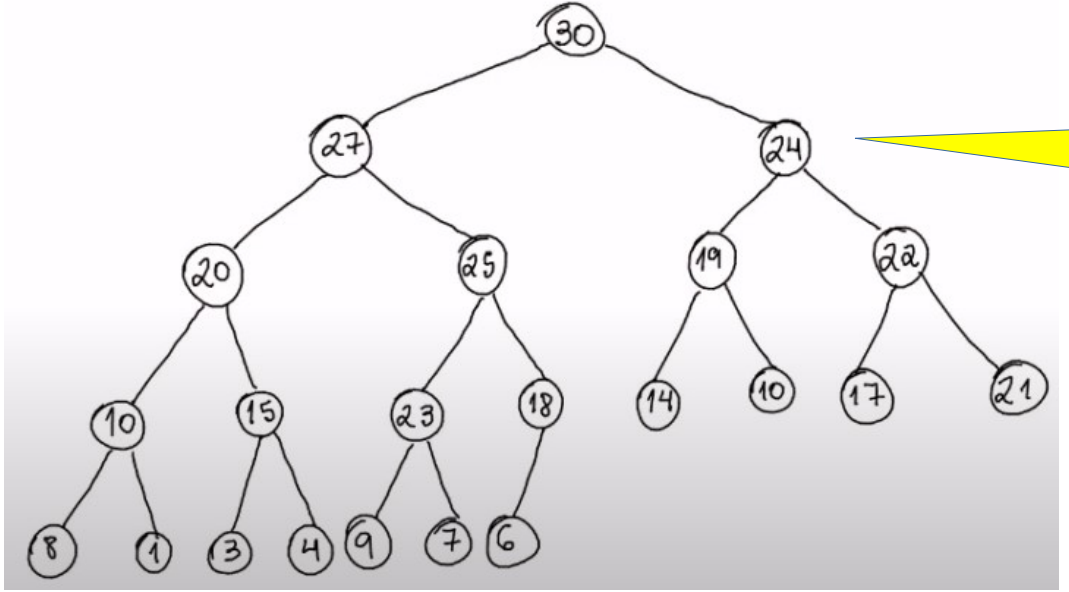
- A_3 = (30, 27, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

Propriedades Heap
Exercícios:



O vetor é heap?

- A_3 = (30, 27, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)



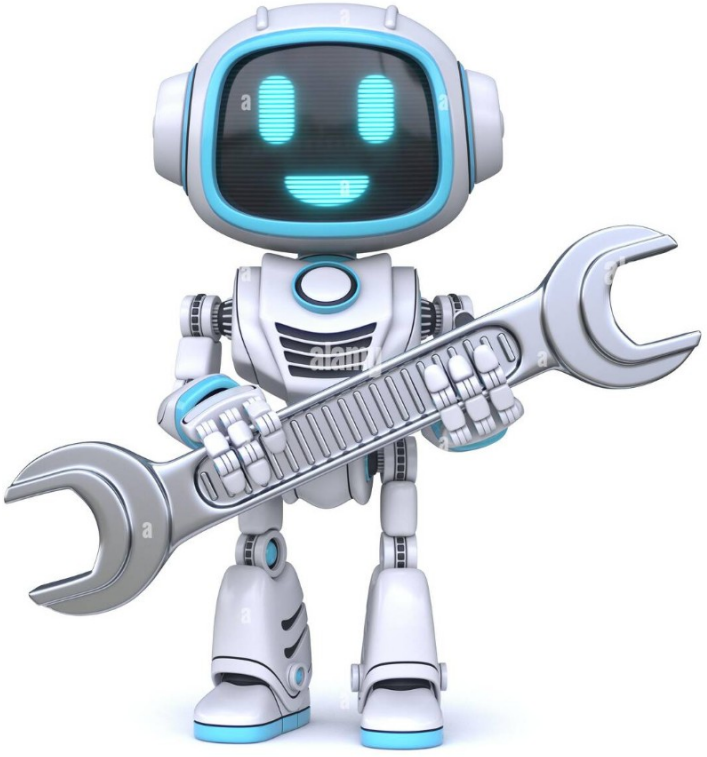
Sim

Propriedades Heap
Exercícios:

Observações:

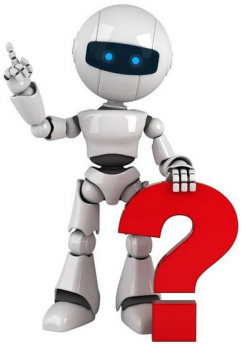
- A **raiz** é o elemento de **maior prioridade**;
- **Consulta** ao elemento **maior** - **O(1)**;
- **Não** há **garantia** que o **2º** e **3º** **maior** elemento estarão no **nível 2**;
- **Heaps não** garantem uma **busca eficiente**;
- **Não** tem a estrutura de uma **árvore (vetor)**.

Alteração - Heap



Prof. Felipe Oliveira ICET / UFAM

Alteração Heap (1)

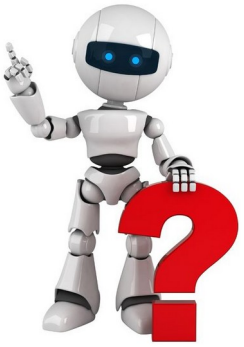


A = (**30**, 27, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

Alteração da prioridade

A = (**12**, 27, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

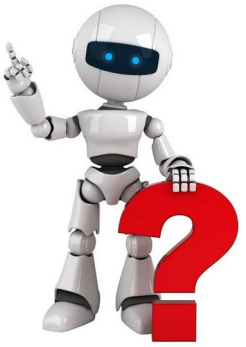
Alteração Heap (1)



O vetor é um heap?

A = (**12**, 27, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

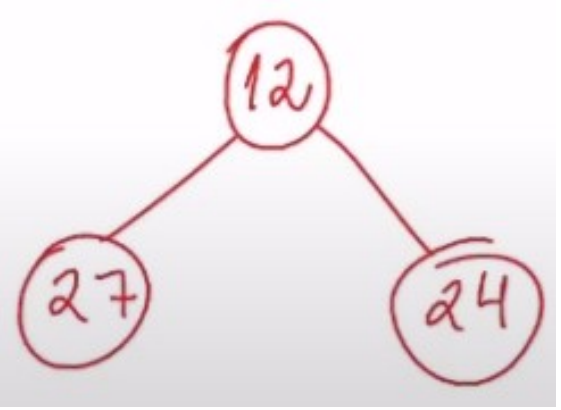
Alteração Heap (1)



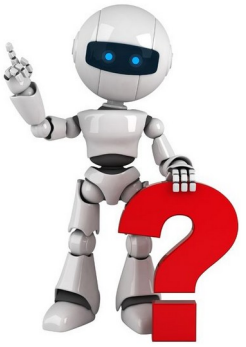
O vetor é um heap?

A = (**12**, 27, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

Não

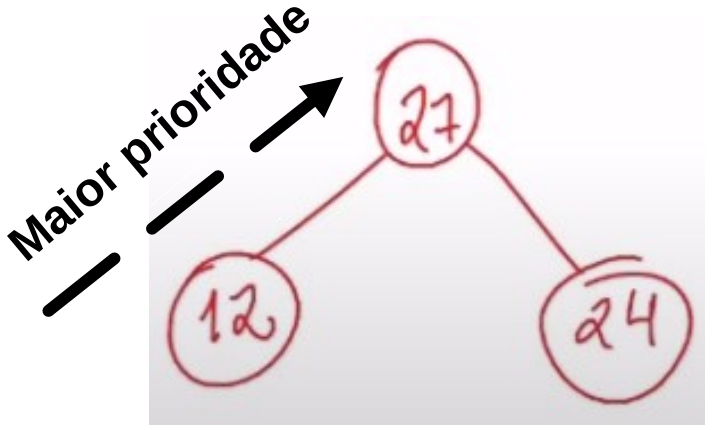


Alteração Heap (1)

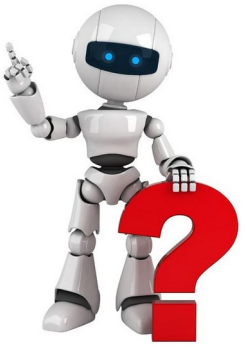


O vetor é um heap?

A = (27, **12**, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

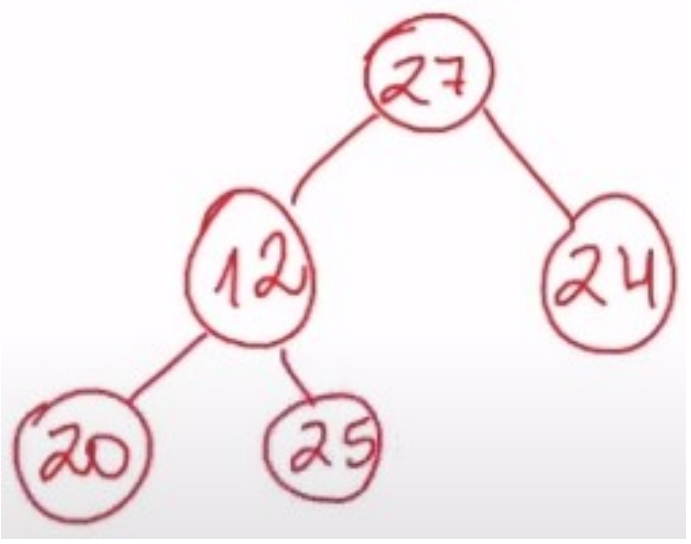


Alteração Heap (1)

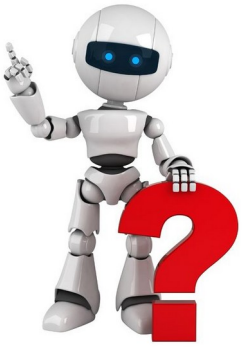


O vetor é um heap?

A = (27, **12**, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

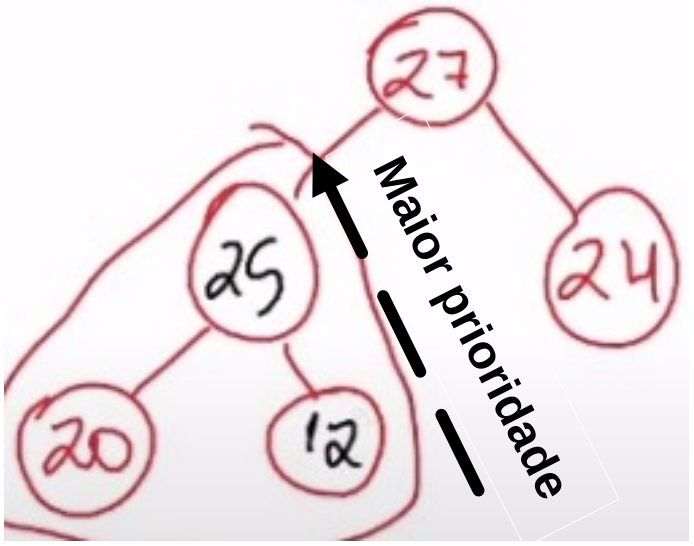


Alteração Heap (1)

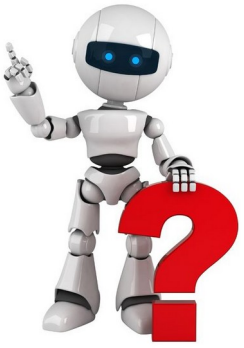


O vetor é um heap?

A = (27, **12**, 24, 20, 25, 19, 22, 10, 15, 23, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)

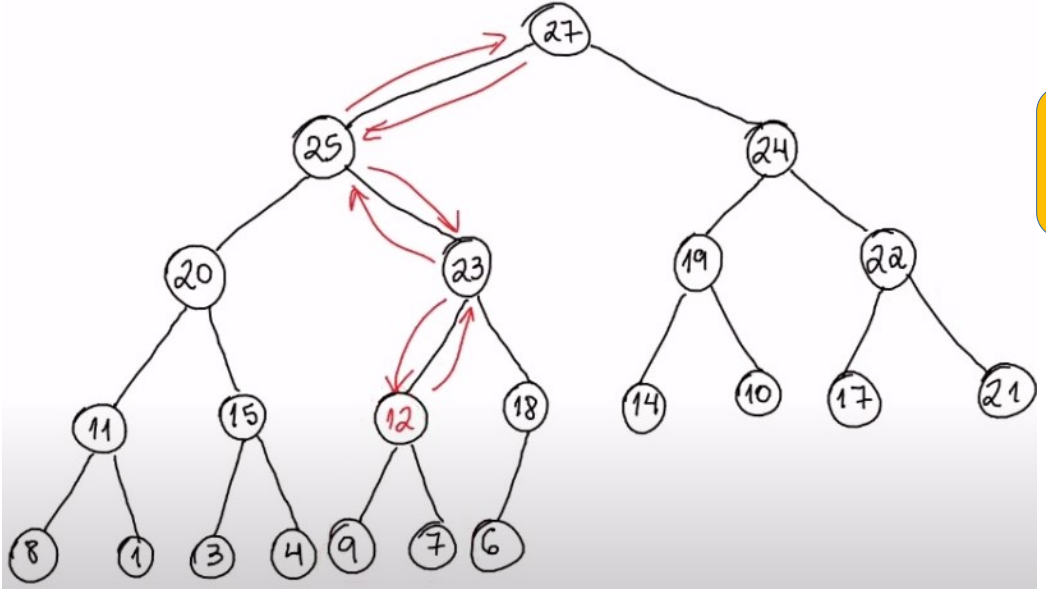


Alteração Heap (1)



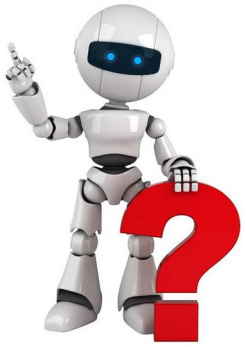
O vetor é um heap?

A = (27, 25, 24, 20, 23, 19, 22, 10, 15, 12, 18, 14, 10, 17, 21, 8, 1, 3, 4, 9, 7, 6)



Corrige Descendo

Alteração Heap (2)

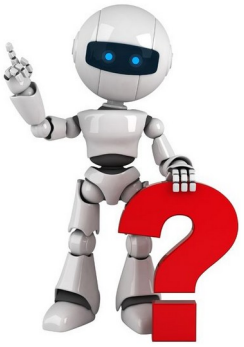


A = (27, 25, 22, 20, 23, 19, 21, 11, **15**, 12, 18, 14, 10, 17, 5, 8, 1, 3, 4, 9, 7, 6)

Alteração da prioridade

A = (27, 25, 22, 20, 23, 19, 21, 11, **45**, 12, 18, 14, 10, 17, 5, 8, 1, 3, 4, 9, 7, 6)

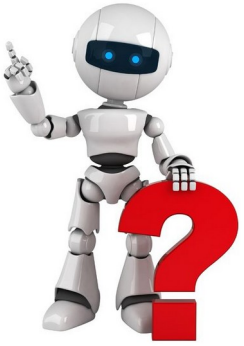
Alteração Heap (2)



O vetor é um heap?

A = (27, 25, 22, 20, 23, 19, 21, 11, **45**, 12, 18, 14, 10, 17, 5, 8, 1, 3, 4, 9, 7, 6)

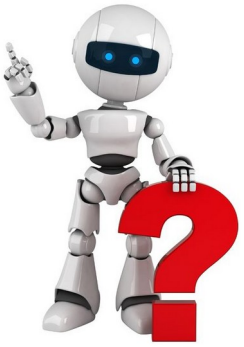
Alteração Heap (2)



O vetor é um heap?

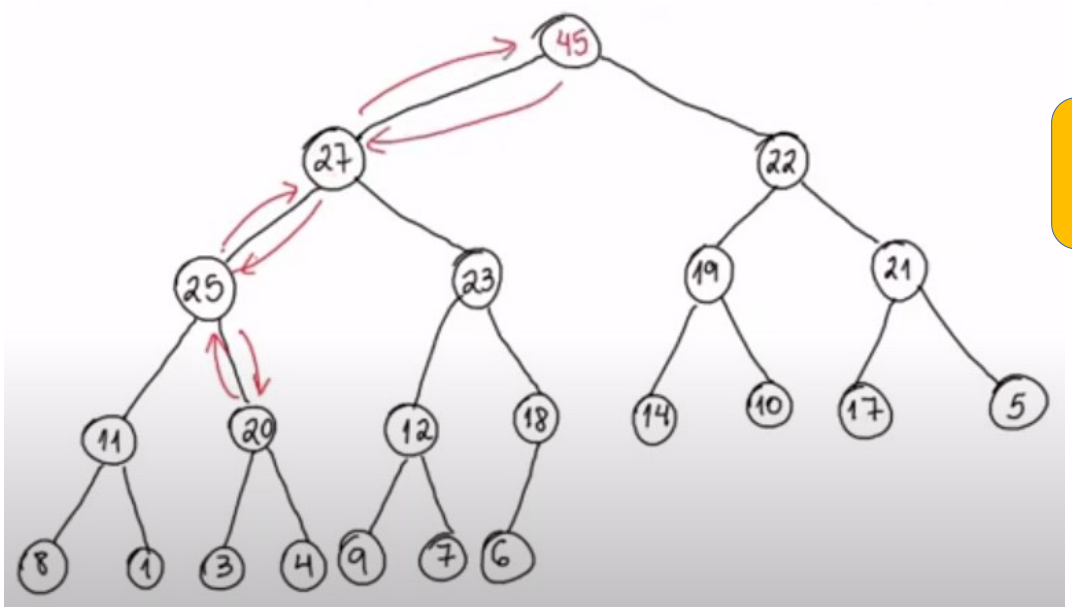
A = (27, 25, 22, 20, 23, 19, 21, 11, **45**, 12, 18, 14, 10, 17, 5, 8, 1, 3, 4, 9, 7, 6)

Alteração Heap (2)



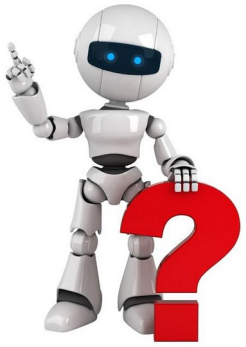
O vetor é um heap?

A = (45, 27, 22, 25, 23, 19, 21, 11, 20, 12, 18, 14, 10, 17, 5, 8, 1, 3, 4, 9, 7, 6)



Corrige Subindo

Alteração Heap (3)

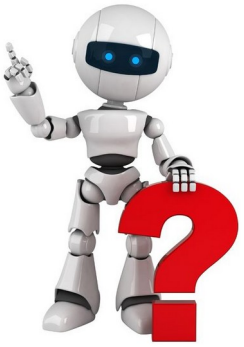


A = (45, 27, 22, 25, 23, 19, 21, 11, 20, 12, 18, 14, **10**, 17, 5, 8, 1, 3, 4, 9, 7, 6)

Alteração da prioridade

A = (45, 27, 22, 25, 23, 19, 21, 11, 20, 12, 18, 14, **30**, 17, 5, 8, 1, 3, 4, 9, 7, 6)

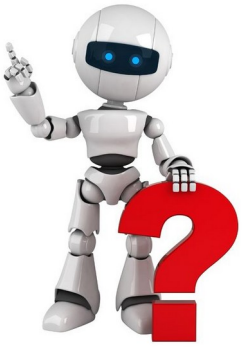
Alteração Heap (3)



O vetor é um heap?

A = (45, 27, 22, 25, 23, 19, 21, 11, 20, 12, 18, 14, **30**, 17, 5, 8, 1, 3, 4, 9, 7, 6)

Alteração Heap (3)

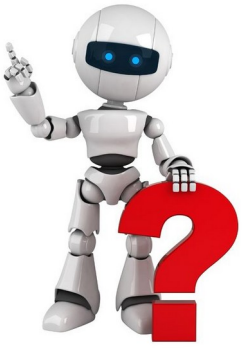


O vetor é um heap?

A = (45, 27, 22, 25, 23, 19, 21, 11, 20, 12, 18, 14, **30**, 17, 5, 8, 1, 3, 4, 9, 7, 6)

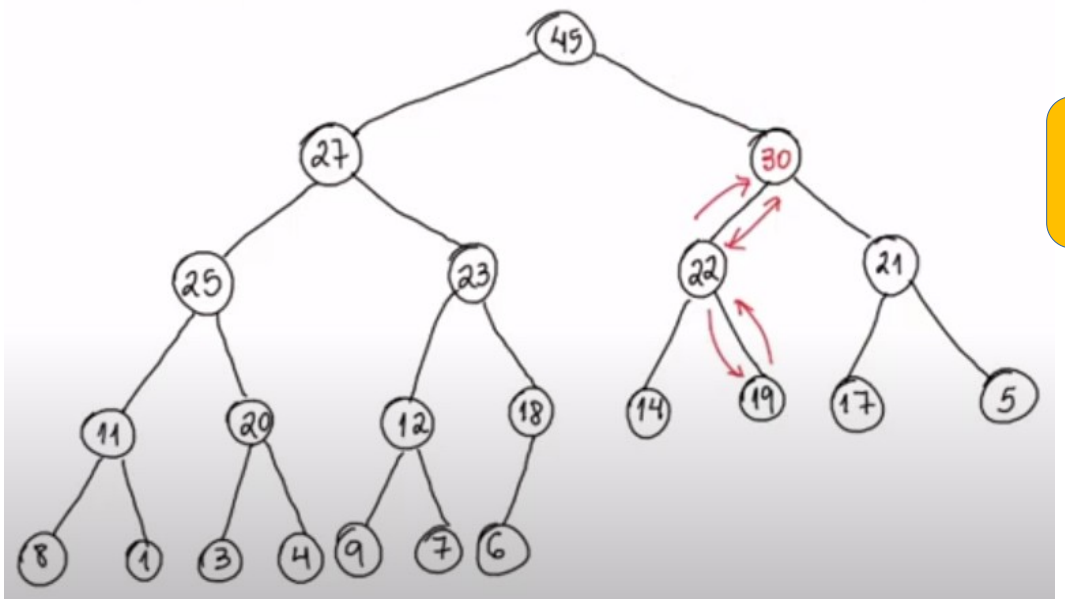
Não

Alteração Heap (3)



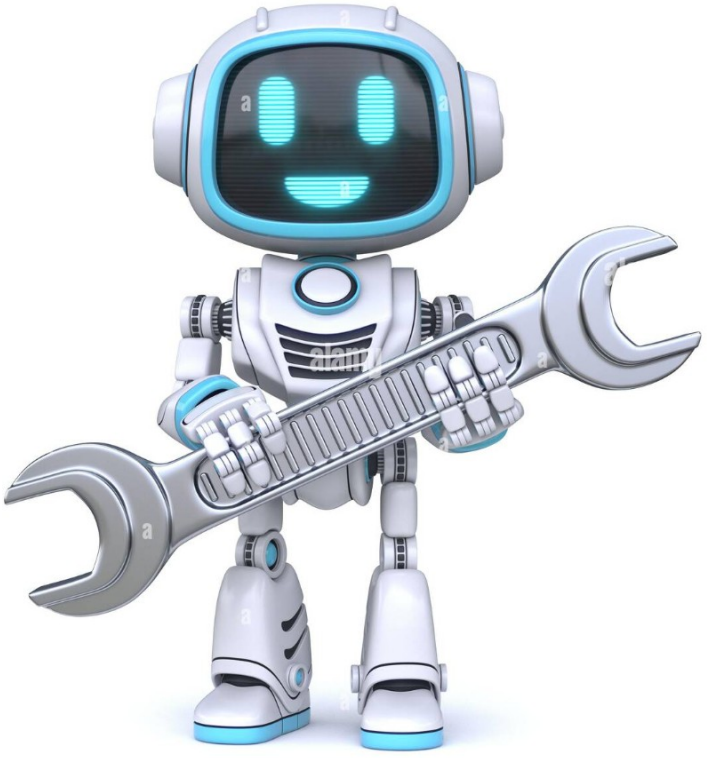
O vetor é um heap?

A = (45, 27, 30, 25, 23, 22, 21, 11, 20, 12, 18, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6)



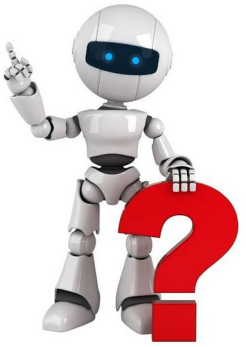
Corrige Subindo

Inserção - Heap

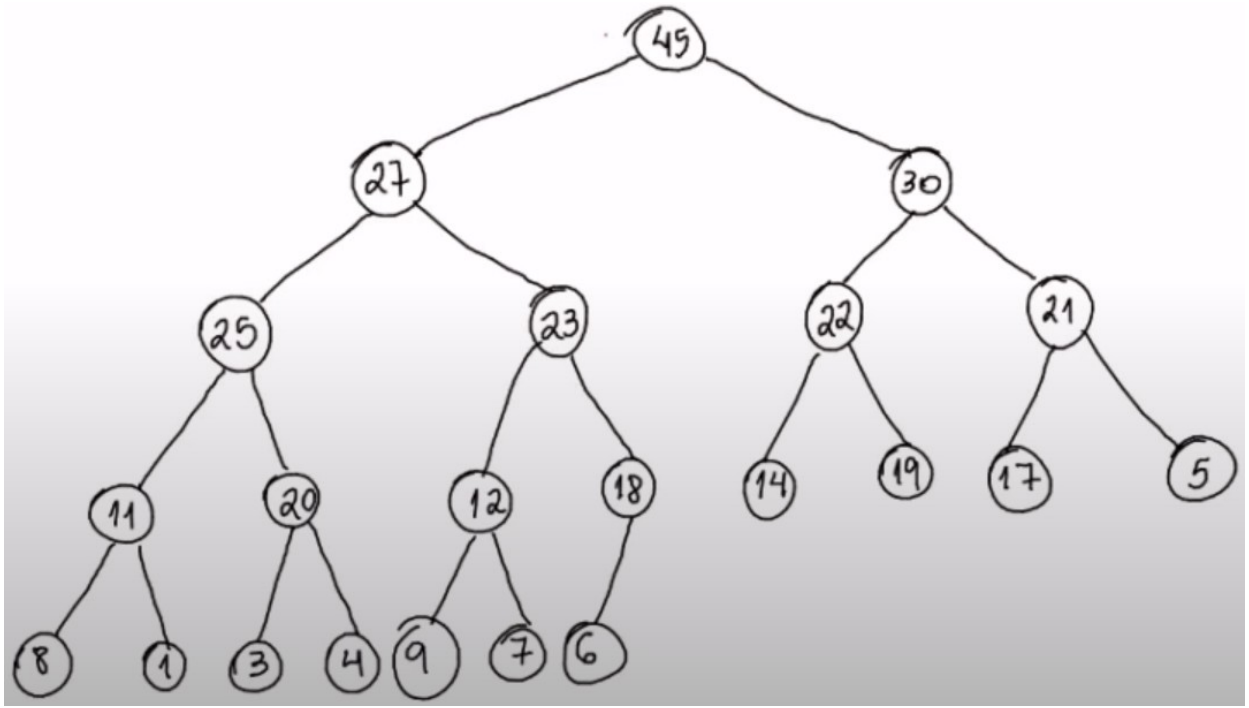


Prof. Felipe Oliveira ICET / UFAM

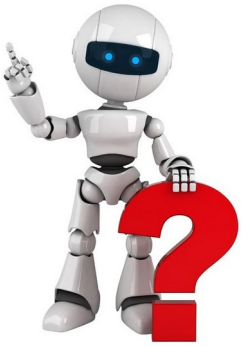
Inserção Heap



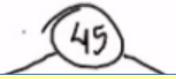
A = (45, 27, 30, 25, 23, 22, 21, 11, 20, 12, 18, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6)



Inserção Heap

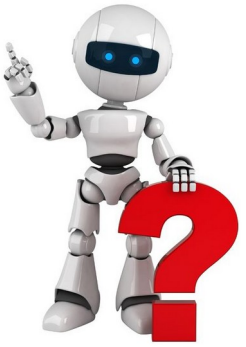


A = (45, 27, 30, 25, 23, 22, 21, 11, 20, 12, 18, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6)

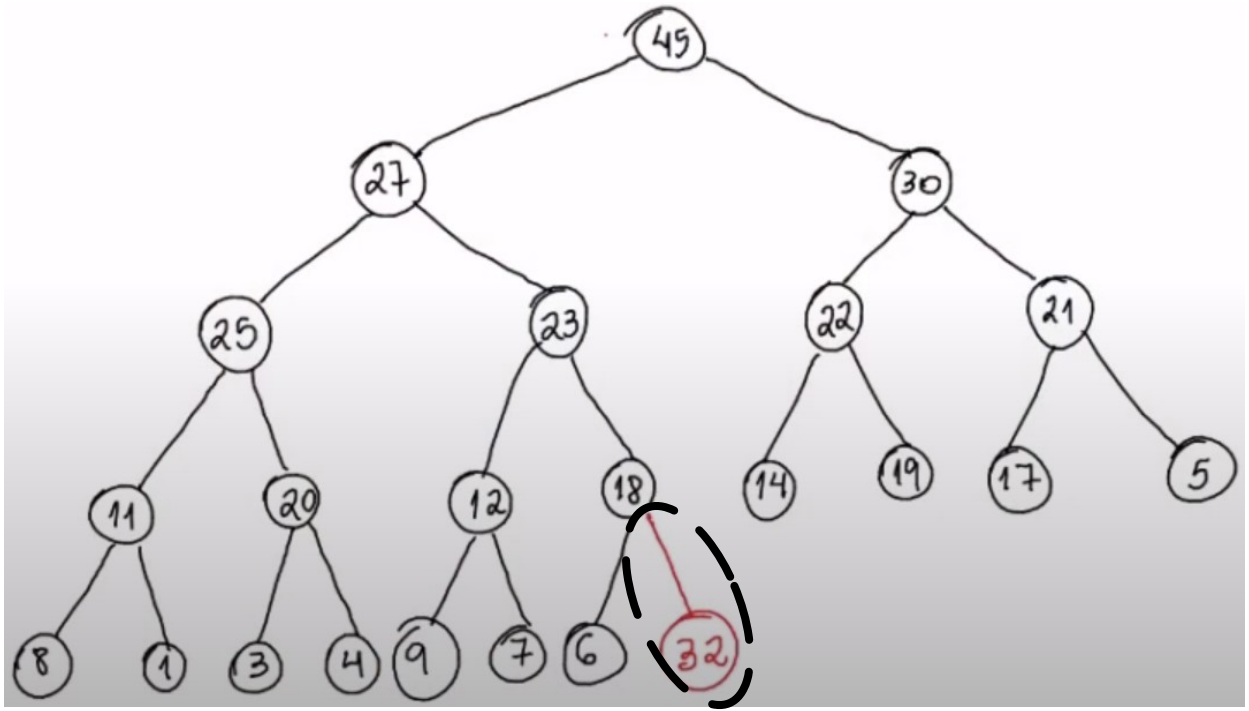


Inserir o valor 32

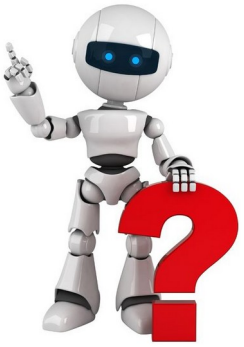
Inserção Heap



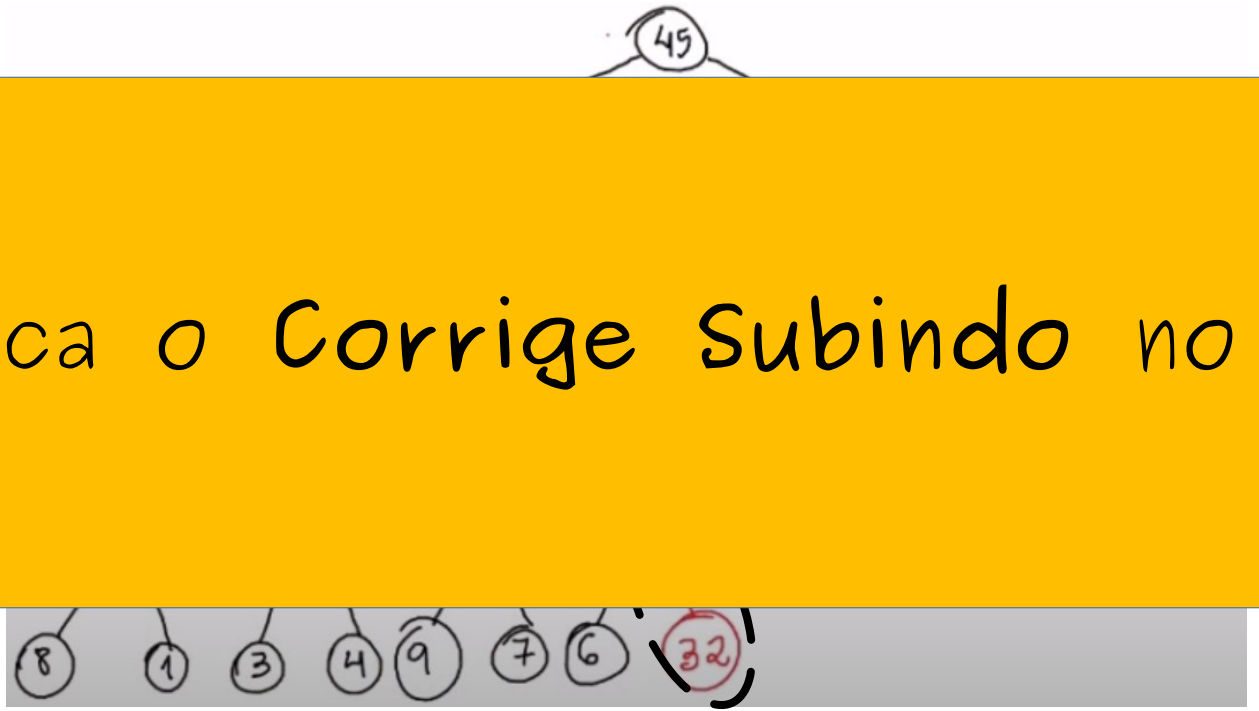
A = (45, 27, 30, 25, 23, 22, 21, 11, 20, 12, 18, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6, **32**)



Inserção Heap

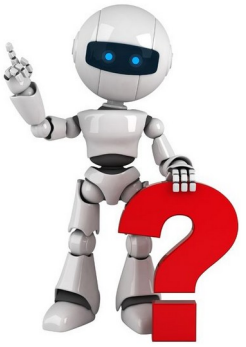


A = (45, 27, 30, 25, 23, 22, 21, 11, 20, 12, 18, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6, **32**)

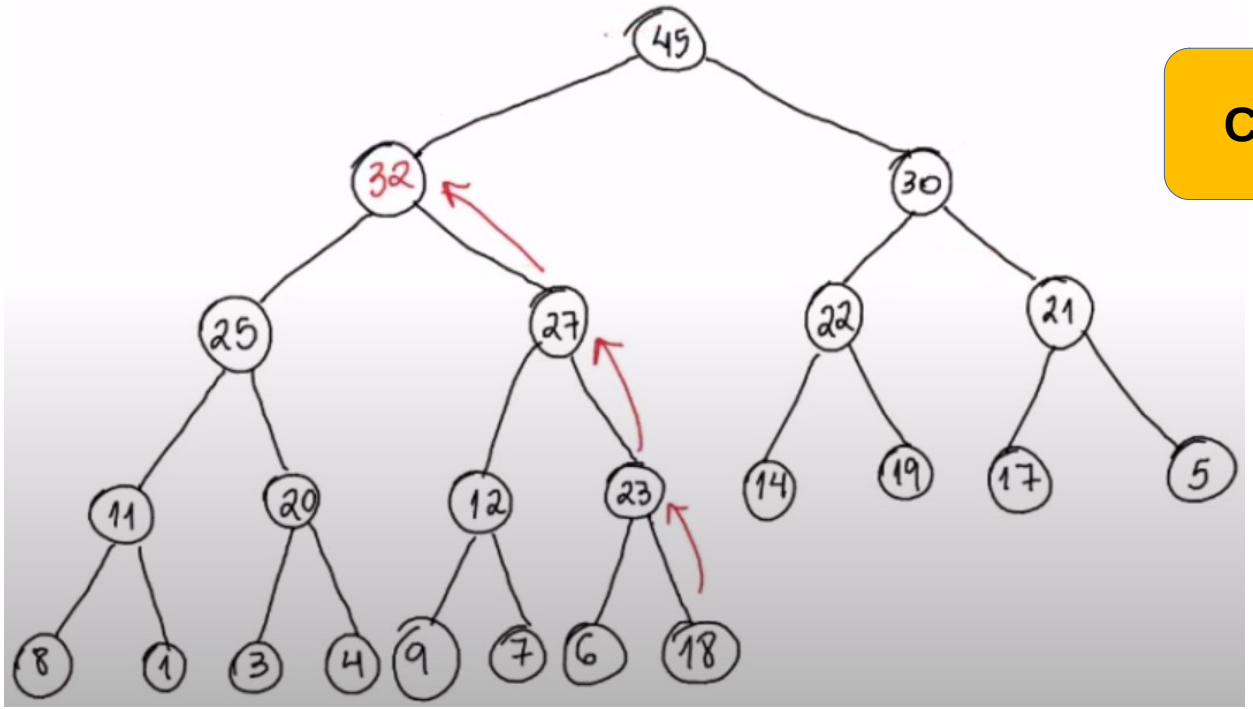


Aplica o **Corrige Subindo** no **32**

Inserção Heap

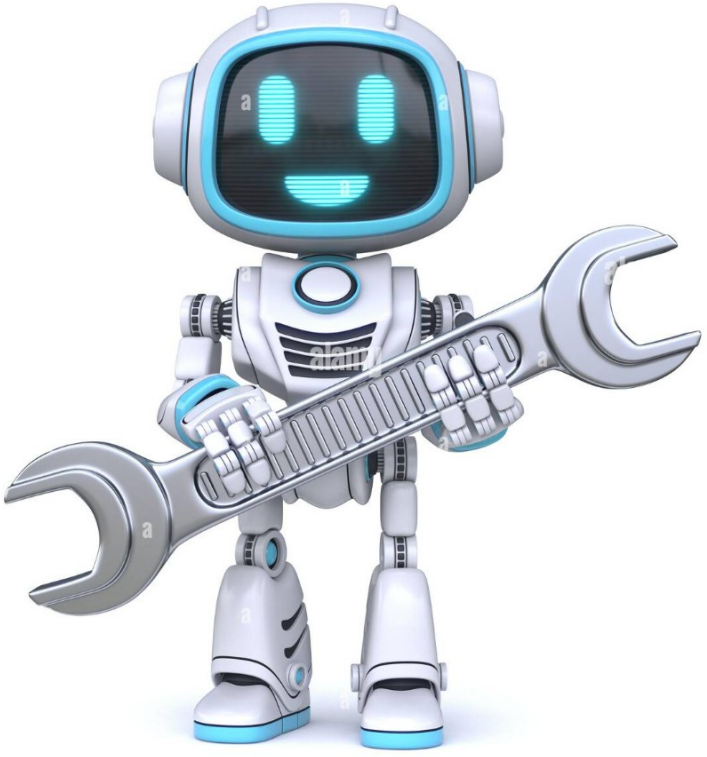


A = (45, **32**, 30, 25, 27, 22, 21, 11, 20, 12, 23, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6, 18)



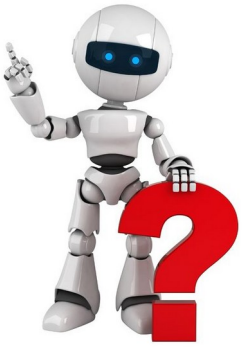
Corrige Subindo

Remoção - Heap



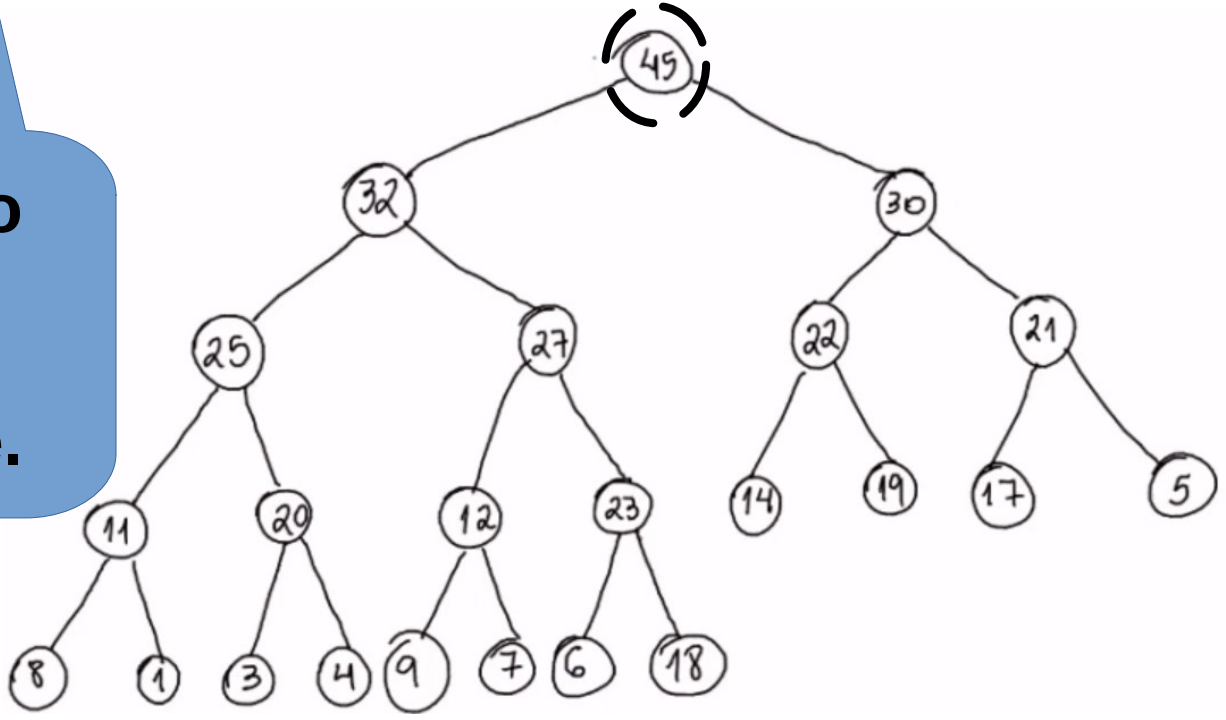
Prof. Felipe Oliveira ICET / UFAM

Remoção Heap

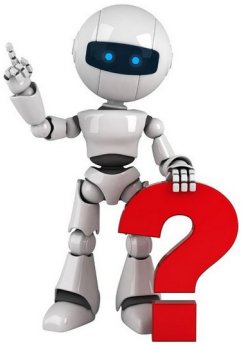


A = (**45**, 32, 30, 25, 27, 22, 21, 11, 20, 12, 23, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6, 18)

Remover o elemento de maior prioridade.



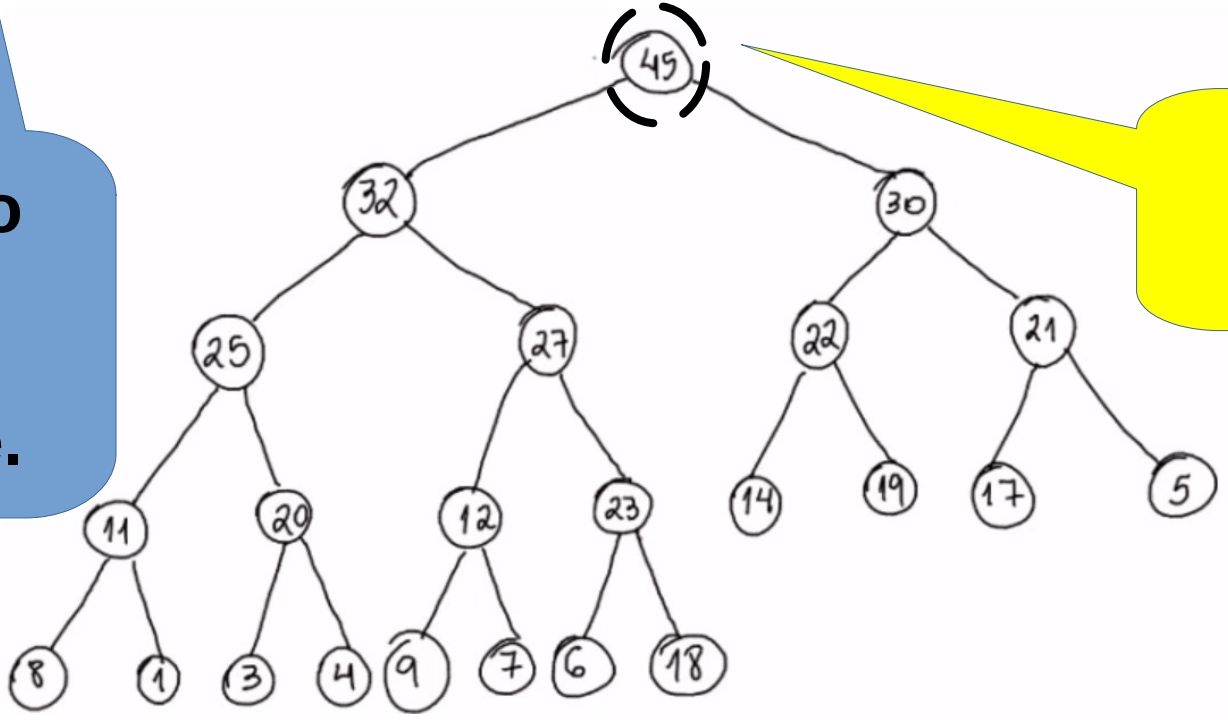
Remoção Heap



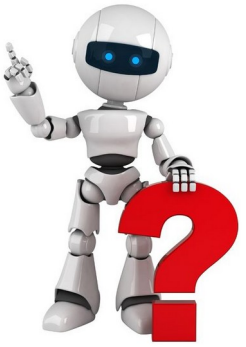
A = (**45**, 32, 30, 25, 27, 22, 21, 11, 20, 12, 23, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6, 18)

Remover o elemento de maior prioridade.

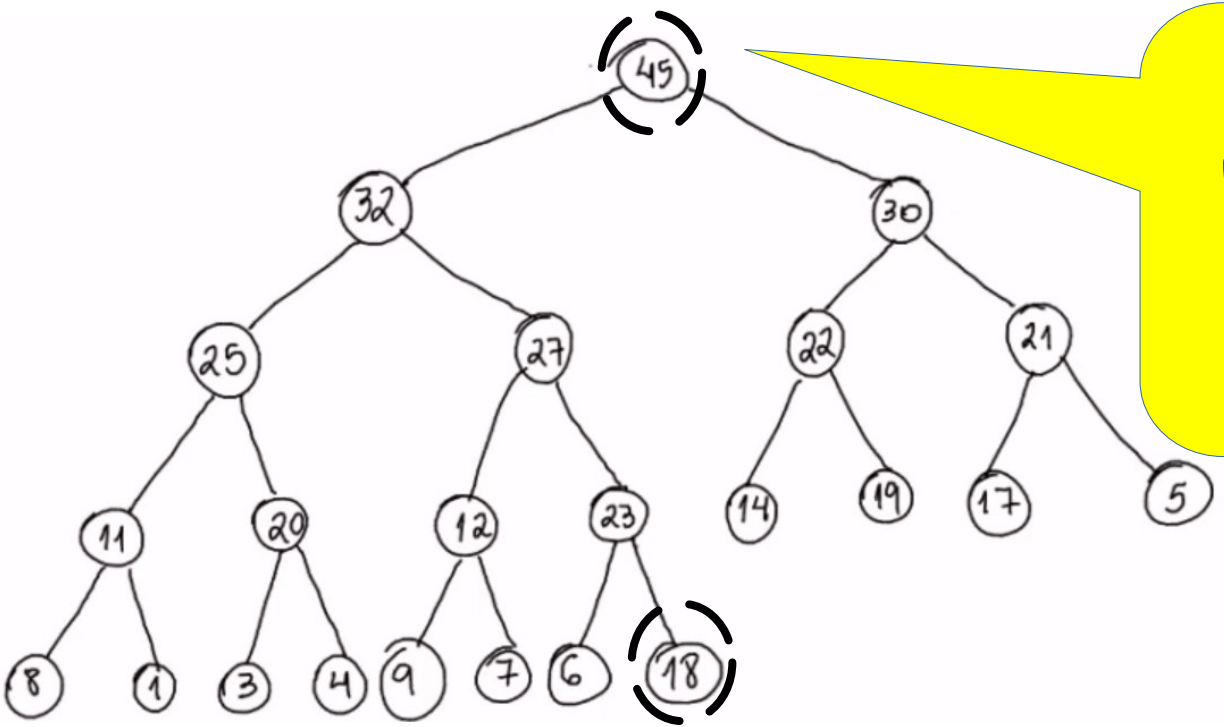
Como ?????



Remoção Heap

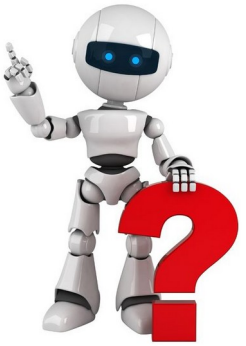


A = (**45**, 32, 30, 25, 27, 22, 21, 11, 20, 12, 23, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6, 18)

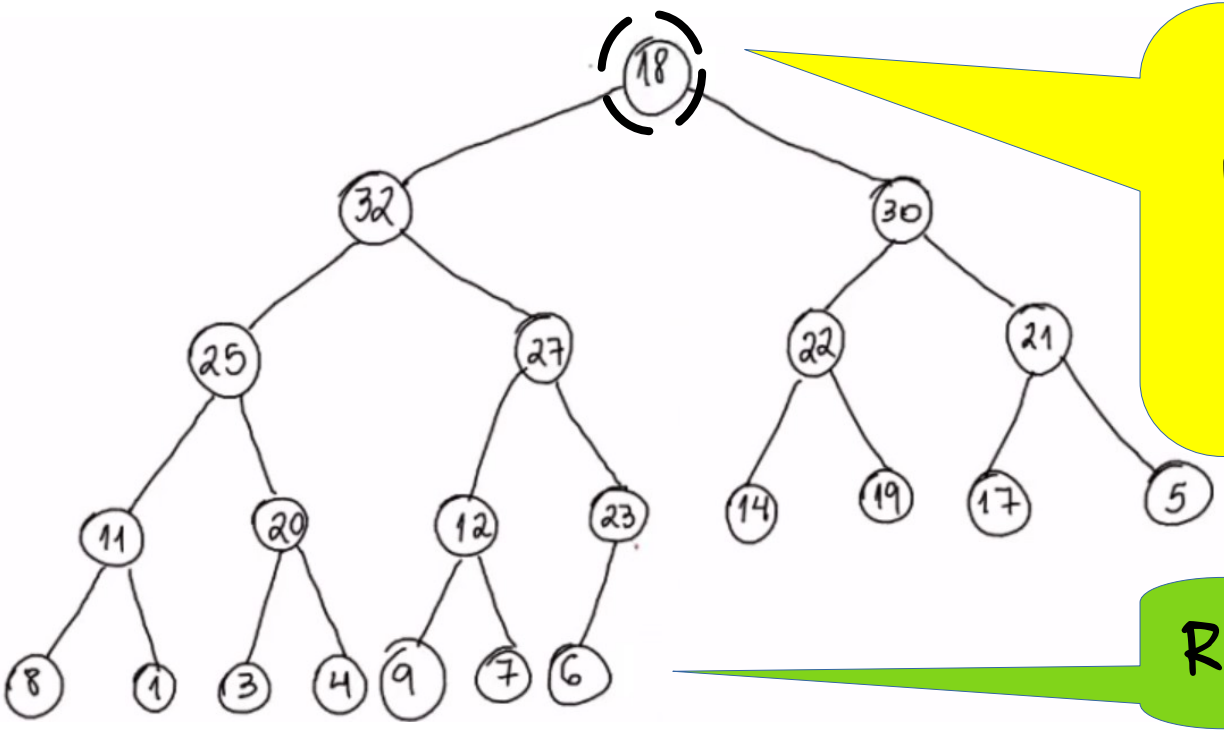


Troca a raiz com menor a direita

Remoção Heap



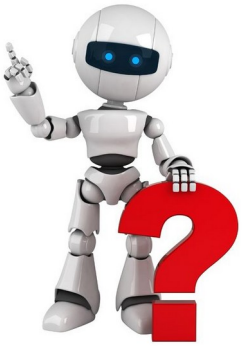
A = (**18**, 32, 30, 25, 27, 22, 21, 11, 20, 12, 23, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6)



Troca a raiz com menor a direita

Remove 45

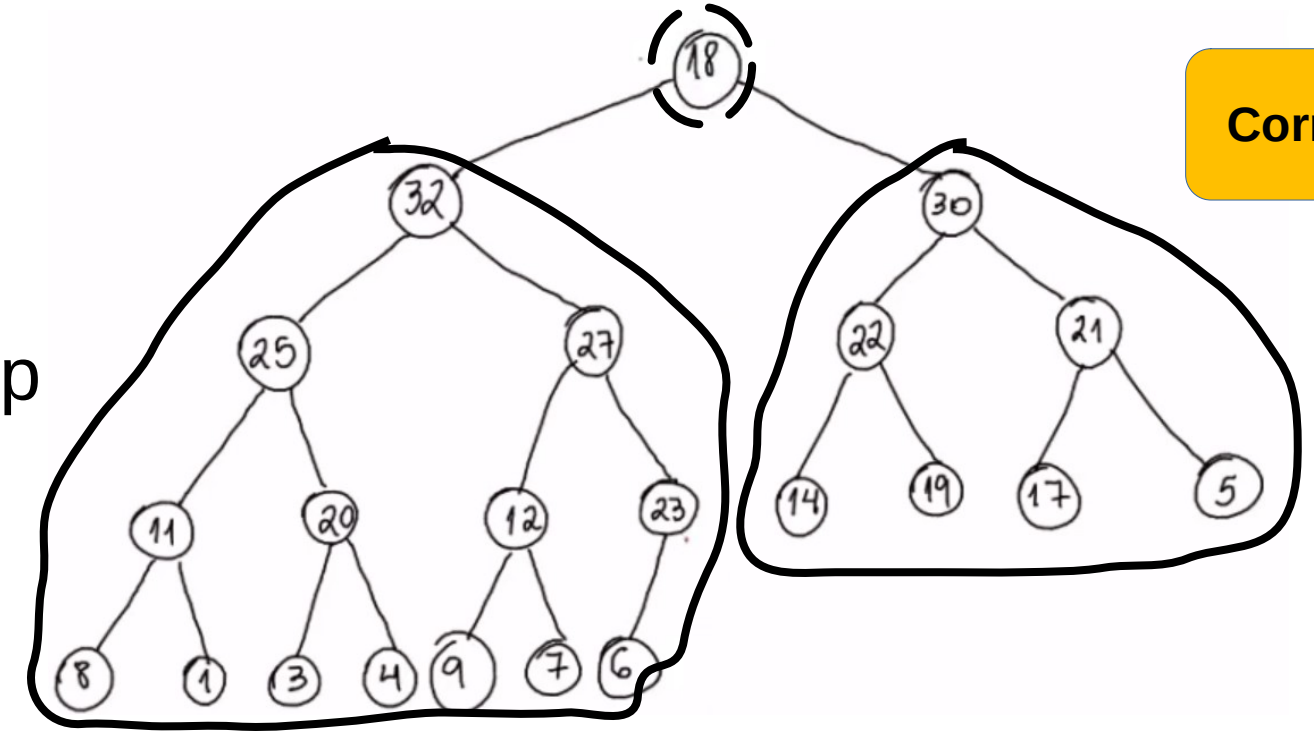
Remoção Heap



A = (**18**, 32, 30, 25, 27, 22, 21, 11, 20, 12, 23, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6)

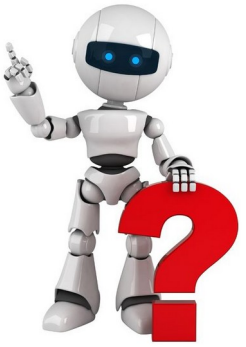
Corrige Descendo

Heap

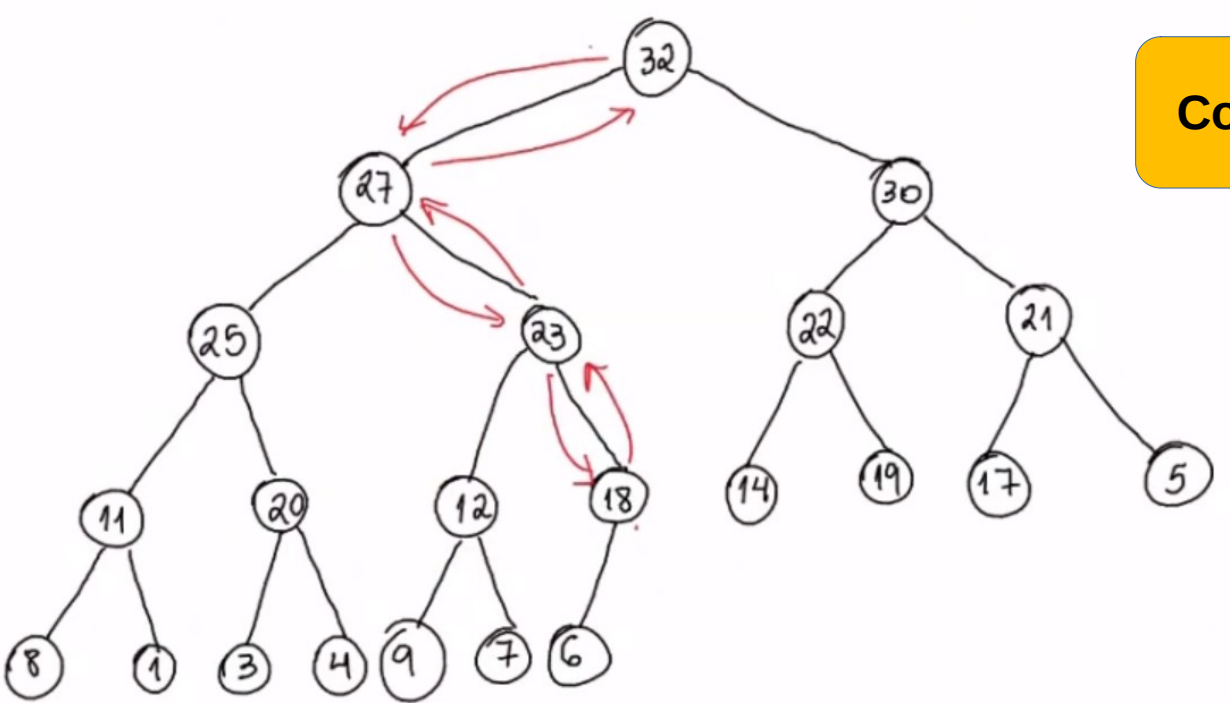


Heap

Remoção Heap



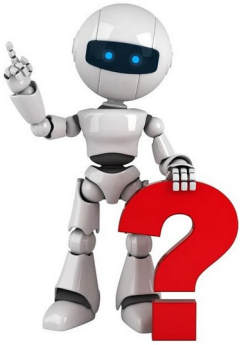
A = (**32**, 27, 30, 25, 23, 22, 21, 11, 20, 12, 18, 14, 19, 17, 5, 8, 1, 3, 4, 9, 7, 6)



Corrige Descendo

Dúvidas

Atividade Prática



- Resolva o 1º Trabalho Prático.

Algoritmos e Estruturas de Dados III

Prof. Dr. Felipe Oliveira
felipeoliveira@ufam.edu.br