



Programação Avançada

2b

Árvores Binárias de Pesquisa | Algoritmos

Bruno Silva, Patrícia Macedo

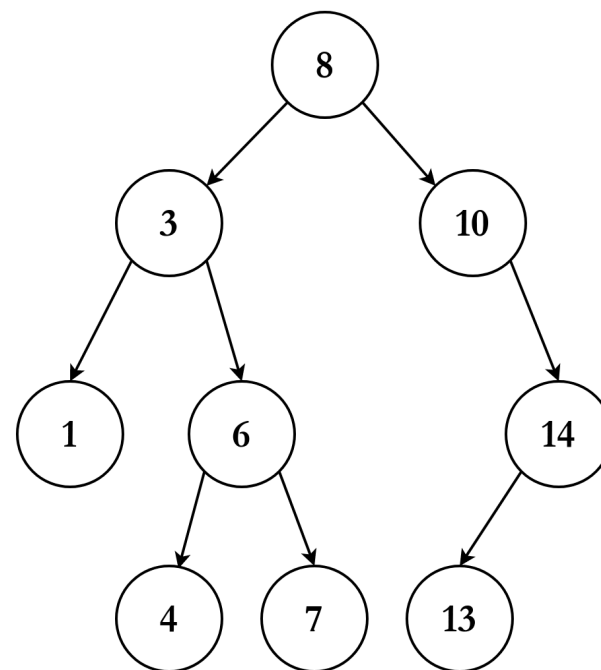
Sumário



- Caracterização
- Motivação de uso
- Algoritmos
 - Pesquisa
 - Inserção
 - Remoção
- Exercícios

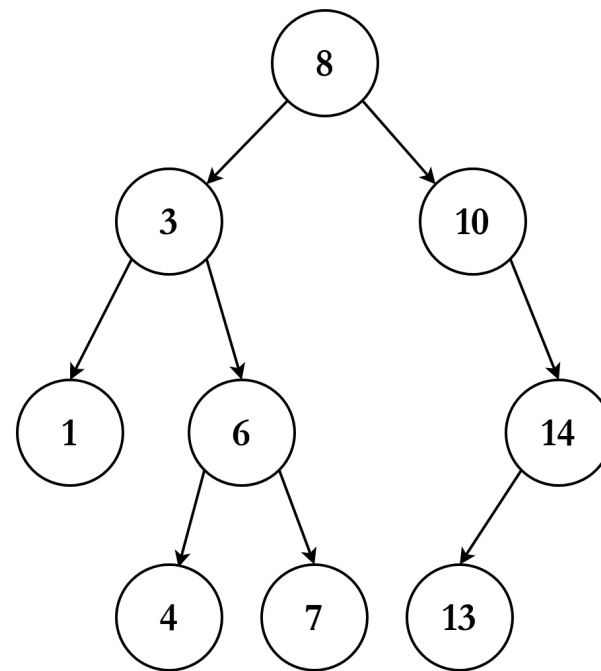
Características

- Uma árvore binária de pesquisa (em inglês, *binary search tree*) é uma estrutura de dados hierárquica especializada;
 - Consiste numa árvore binária "ordenada".
- É **caracterizada** pelo seguinte:
 - um **nó** contém um elemento/chave maior que os da **sub-árvore esquerda**;
 - um **nó** contém um elemento/chave menor que os da **sub-árvore direita**;



Características

- Deverá ser óbvio que é necessário um **critério de comparação** para os elementos/chaves numa árvore binária de pesquisa;
- **Não são permitidos** elementos/chaves **repetidos**;
- Vamos ilustrar com números (comparação natural), mas são válidas para quaisquer outros dados "comparáveis".



Motivação

Como o nome sugere, permitem acelerar a pesquisa de elementos.

- Pesquisa sequencial num array - complexidade $O(n)$
 - 10.000 elementos ➡ 10.000 comparações no pior caso 😞
- Pesquisa numa árvore binária de pesquisa - complexidade $O(\log n)$
 - 10.000 elementos ➡ ≈ 13 comparações no pior caso 😊

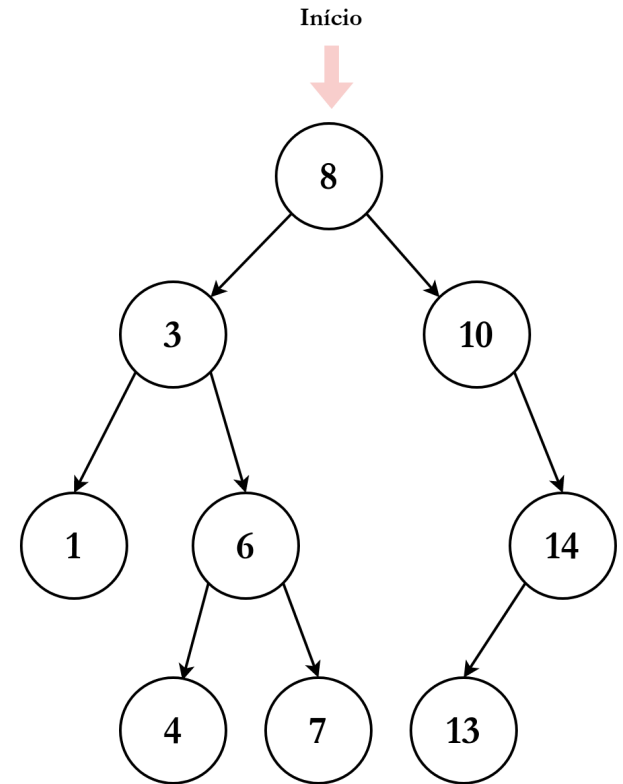
Algoritmos

- Os principais algoritmos são:
 - 🔍 Pesquisa (elemento/chave, mínimo e máximo);
 - + Inserção de um novo nó (⚠️)
 - — Remoção de um nó (⚠️)
- Os algoritmos que alteram uma árvore binária de pesquisa (⚠️) têm de garantir/manter as suas características.
- Os algoritmos serão apresentados em *linguagem natural*.

Pesquisa

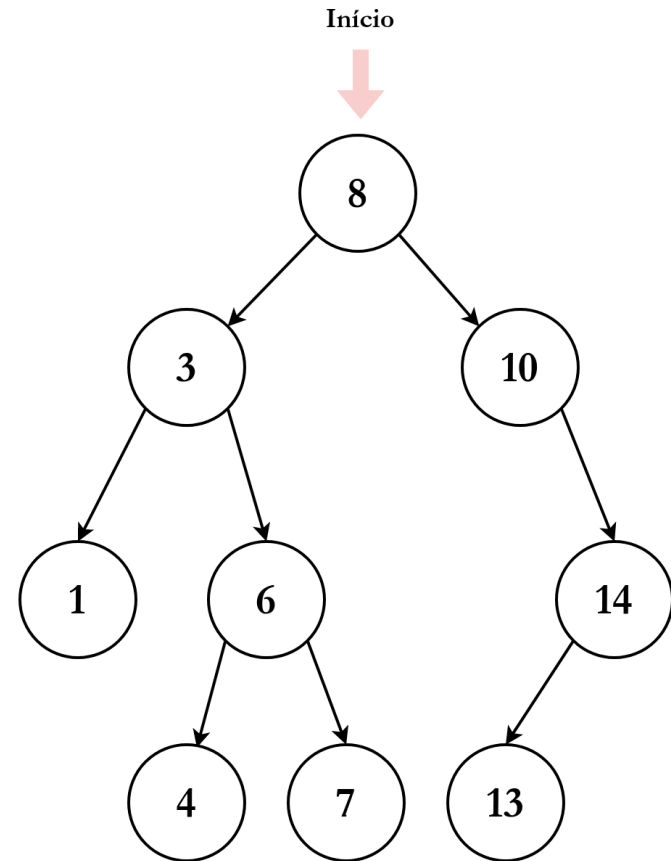
Para pesquisar um elemento/chave *target*:

1. Se a árvore estiver vazia, não existe;
2. Se *target* é igual ao elemento na raiz da árvore, sucesso!;
3. Senão:
 - Se *target* é menor que o elemento na raiz, retornar o resultado da pesquisa na sub-árvore esquerda;
 - Se *target* é maior que o elemento na raiz, retornar o resultado da pesquisa na subárvore direita;



Pesquisa

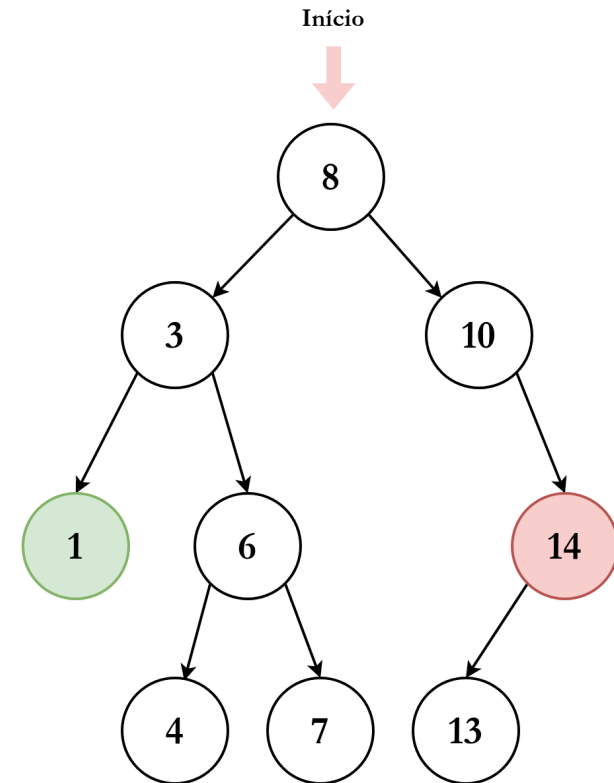
- Quais os passos para *target* =
 - 5?
 - 8?
 - 13?
 - 15?



Mínimo & Máximo

- Elemento/chave **mínimo** esta contido no nó mais à esquerda da árvore;
- Elemento/chave **máximo** esta contido no nó mais à direita da árvore;

? Exemplifique com um algoritmo.

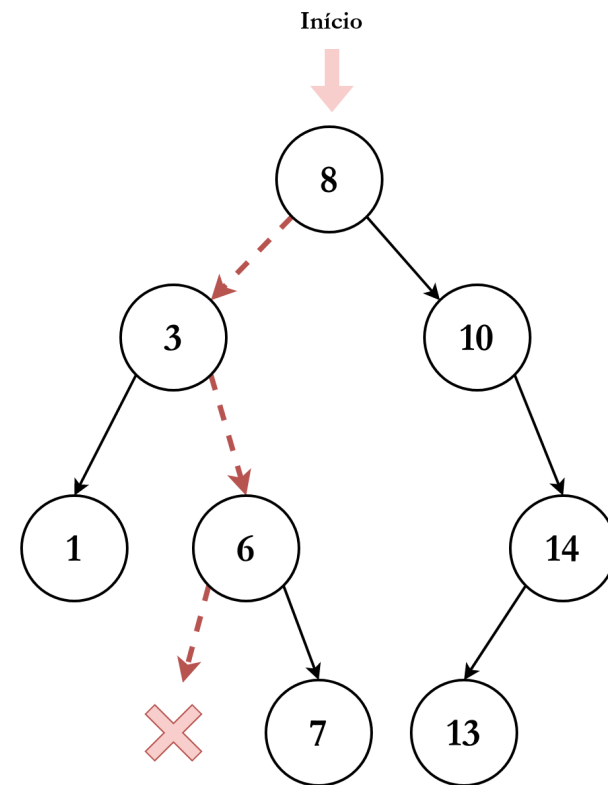


+ Inserção (⚠)

- A inserção de um elemento deverá garantir que são mantidas as características de uma árvore binária de pesquisa:
 - Sem repetição de elementos/chaves;
 - Sub-árvores esquerdas contêm elementos menores que a raiz;
 - Sub-árvores direitas contêm elementos maiores que a raiz;

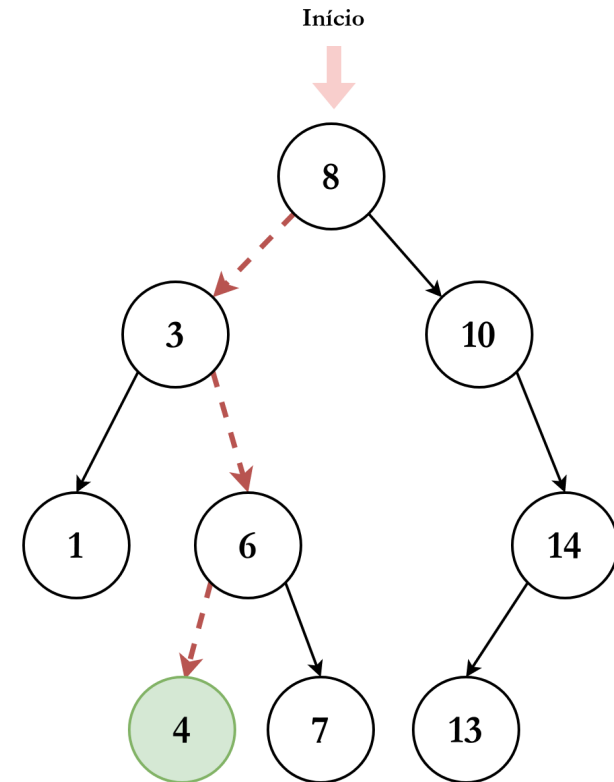
+ Inserção

- O algoritmo de inserção procede de forma análoga ao de pesquisa até que a sub-árvore que teria de conter o elemento esteja vazia.
- A figura ➡ ilustra esta situação para o elemento 4.



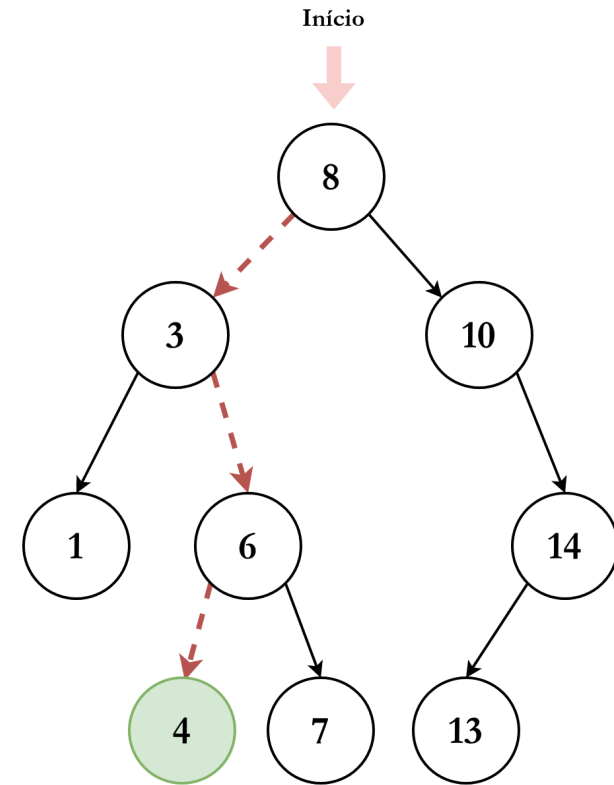
+ Inserção

- Nesta situação é adicionado um novo nó contendo o elemento (correspondendo a uma nova sub-árvore com o elemento como raiz).



+ Inserção

- Se o elemento a inserir (e.g., o elemento 4) já existir então o algoritmo nada faz.

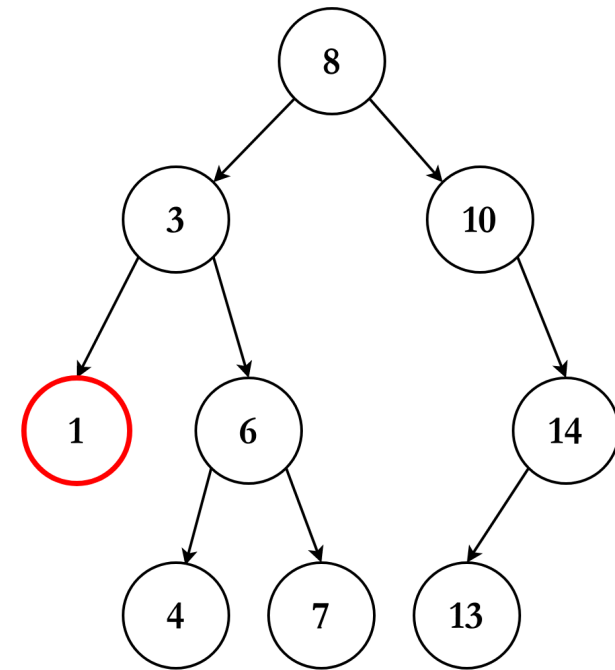


— Remoção (⚠️)

- Existem três situações possíveis a contemplar para o nó que contém o elemento a remover:
 - [1] Não possui sub-árvores 😊
 - [2] Apenas possui uma sub-árvore 😞
 - [3] Possui duas sub-árvores 😫
- Os algoritmos de remoção dos elementos também têm de manter as características de uma árvore binária de pesquisa.

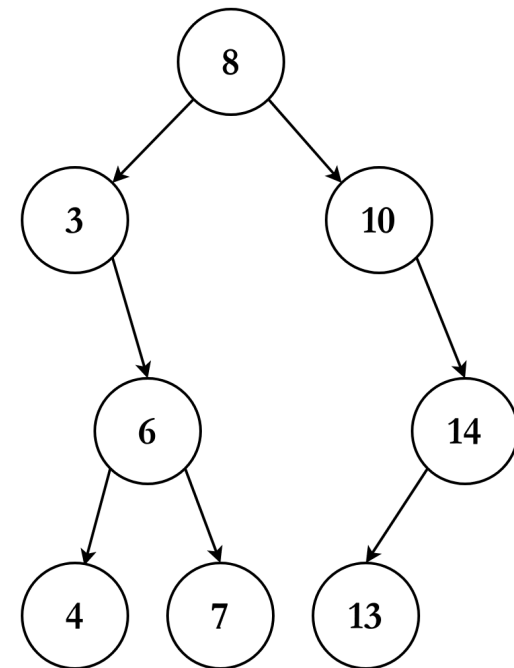
— Remoção

- [1] Não possui sub-árvores:
 - Figura ➡ com exemplo para o elemento 1.



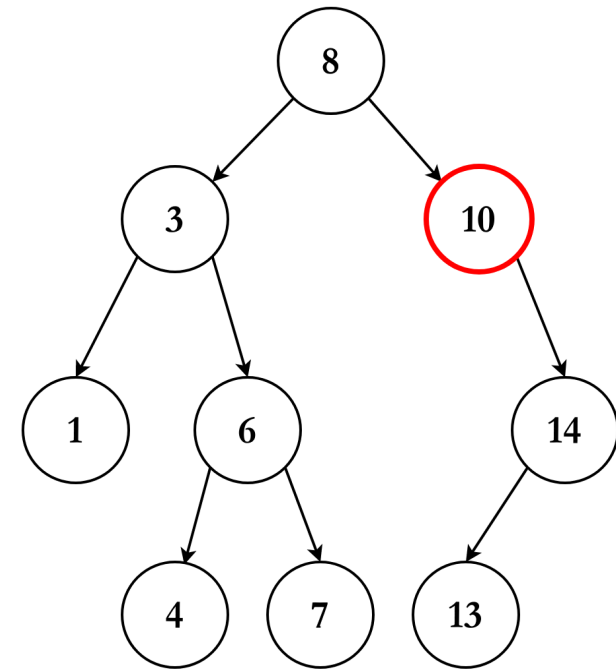
— Remoção

- [1] Não possui sub-árvores:
 - Figura ➡ com exemplo para o elemento 1.
 - **Remove-se simplesmente o nó.**



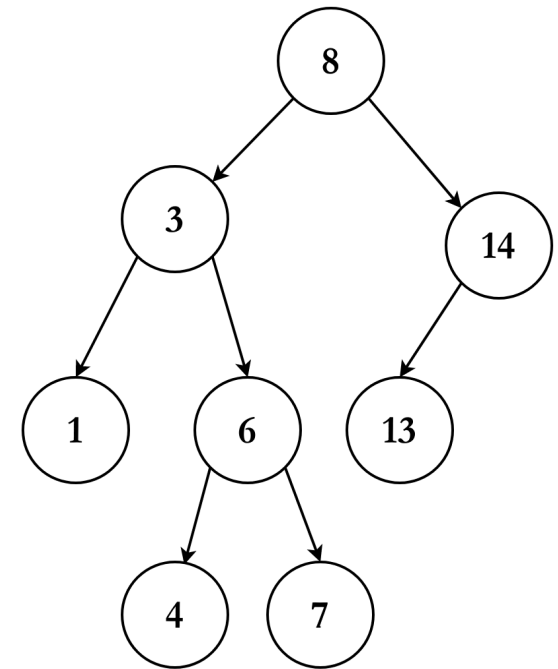
— Remoção

- [2] Apenas possui uma sub-
árvore:
 - Figura ➡ com exemplo para
o elemento 10.



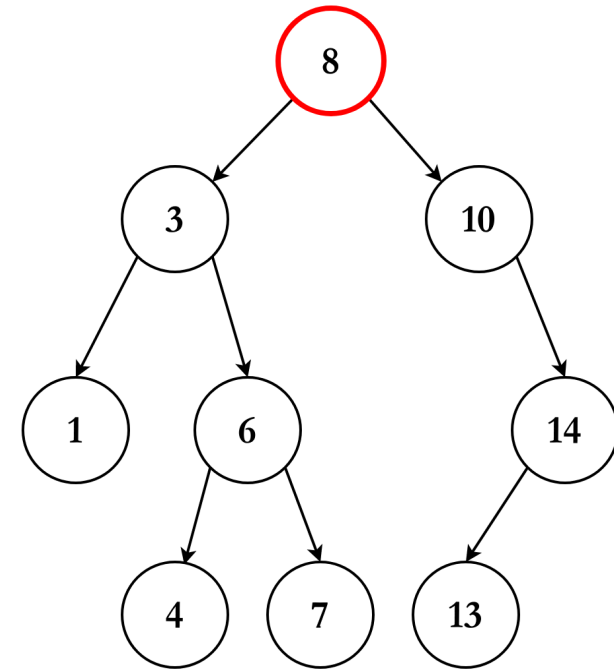
— Remoção

- [2] Apenas possui uma sub-
árvore:
 - Figura ➡ com exemplo para o elemento 10.
 - **Substitui-se o nó pela sua sub-árvore.**



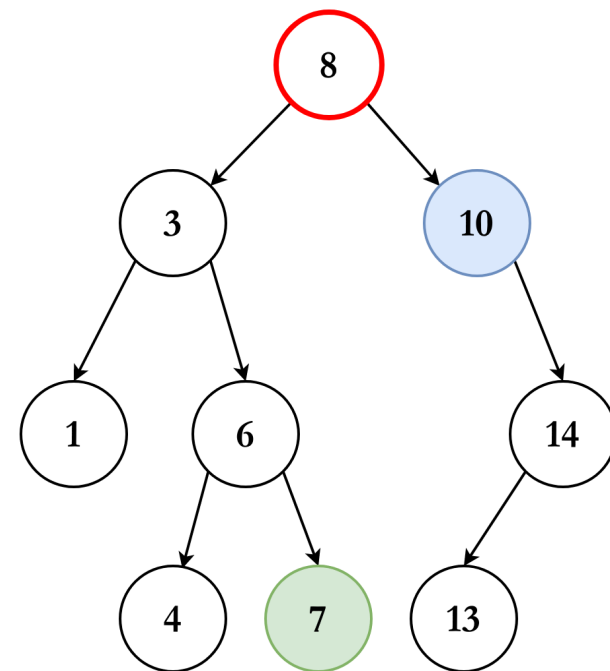
— Remoção

- [3] Possui duas sub-árvores:
 - Figura ➡ com exemplo para o elemento 8.



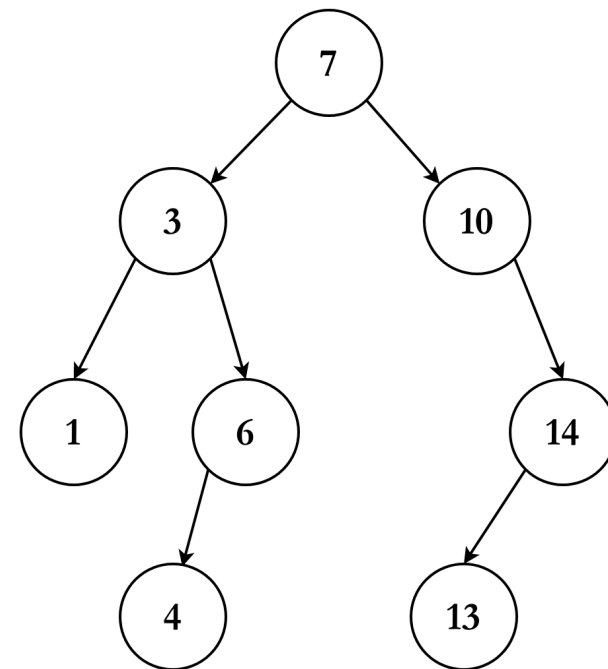
— Remoção

- [3] Possui duas sub-árvores:
 - Figura ➡ com exemplo para o elemento 8.
 - Duas hipóteses (substituição e algoritmo de remoção):
 - Elemento máximo da sub-árvore esquerda.
 - Elemento mínimo da sub-árvore direita.



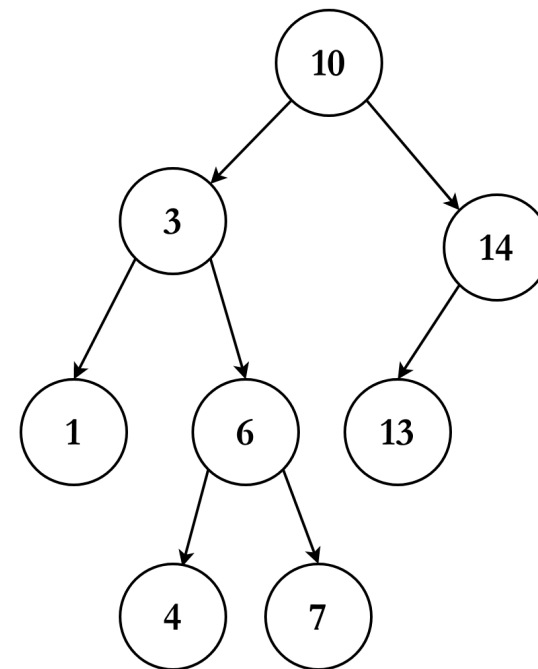
— Remoção

- [3] Possui duas sub-árvores:
 - Figura ➡ com exemplo para o elemento 8.
 - 1ª Hipótese (substituição e algoritmo de remoção):
 - Elemento máximo da sub-árvore esquerda.



— Remoção

- [3] Possui duas sub-árvores:
 - Figura ➡ com exemplo para o elemento 8.
 - Hipótese (substituição e algoritmo de remoção):
 - Elemento mínimo da sub-árvore direita.



Exercícios

1. Elabore a ficha de atividades disponível no Moodle:

- `2b_FichaAtividades.pdf`

Exercícios

2. Aceda ao seguinte link que permite visualizar a manipulação de uma árvore binária de pesquisa:

<https://www.cs.usfca.edu/~galles/visualization/BST.html>

3. Tente uma ordem de inserção de elementos por forma a obter a árvore ilustrada no slide #22.

4. Qual a árvore resultante da inserção sequencial de $\{1, 4, 7, 9, 10, 18\}$?

- Vê algum problema na árvore resultante?
- https://en.wikipedia.org/wiki/AVL_tree