inteli

Algoritmos e Percurso em Árvores

Prof. Rafael Will M. de Araujo

Operações em uma árvore binária de busca

Manipular dados em uma AAB balanceada é muito eficiente:

- **Buscar** um nó: O(log n)
- **Inserir** um novo nó: O(log n)
- **Remover** um nó: O(log n)

Operações em uma árvore binária de busca

Manipular dados em uma AAB balanceada é muito eficiente:

- **Buscar** um nó: O(log n)
- **Inserir** um novo nó: O(log n)
- **Remover** um nó: O(log n)

Entretanto, na prática não há garantia que a AAB estará sempre balanceada, portanto:

	Caso médio	Pior caso
Buscar	O(log n)	O(n)
Inserir	O(log n)	O(n)
Remover	O(log n)	O(n)

inteli

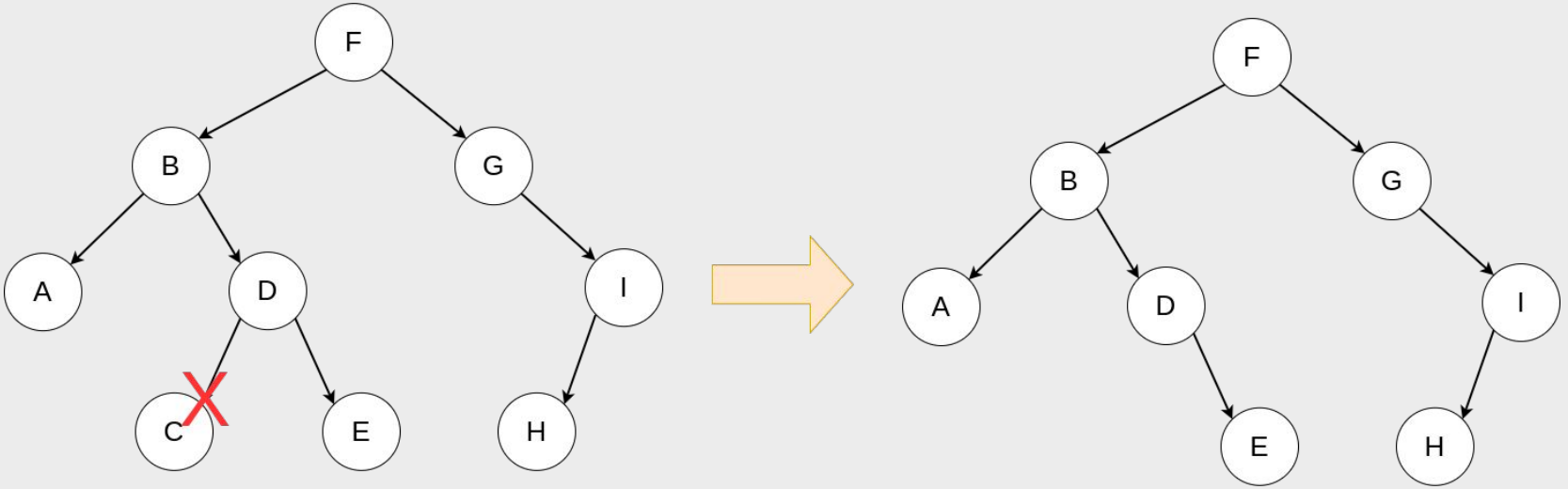
Remover um nó de uma AAB

Antes de remover um nó de uma AAB, temos que analisar os seguintes casos:

- Se o **nó é uma folha** (caso fácil)
- Se o **nó possui um único filho** (caso menos fácil)
- Se o **nó possui dois filhos** (caso difícil)

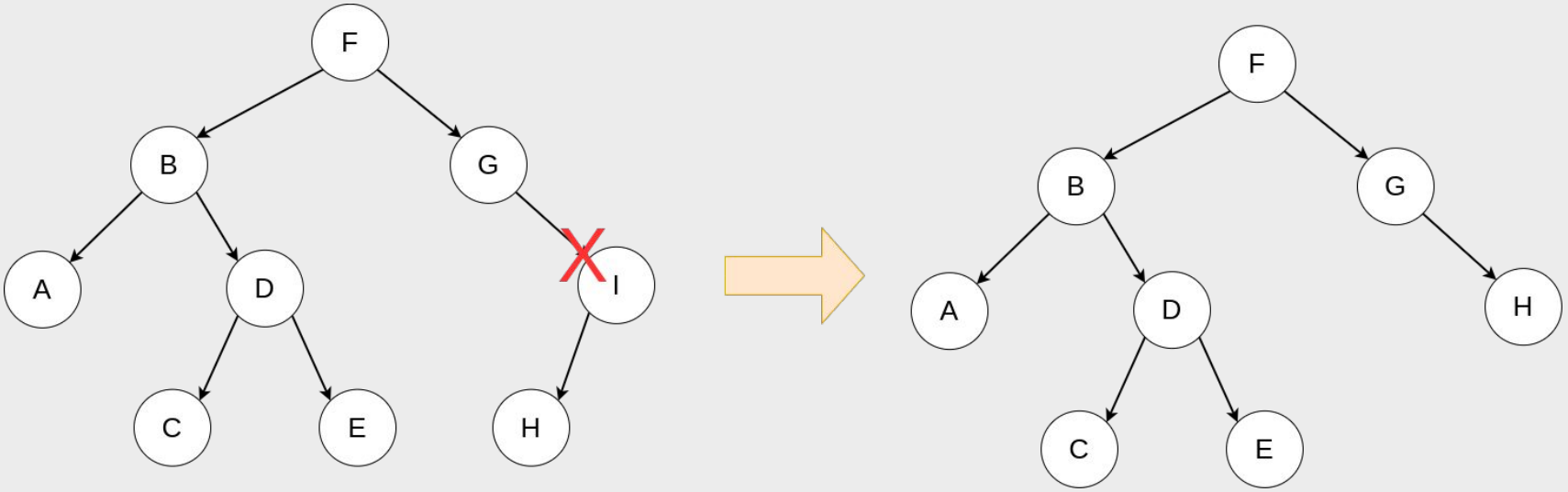
Remover nó folha

Caso mais simples: basta remover a referência para o nó (C) em seu nó pai (D). O garbage collector se encarregará de apagá-lo da memória.



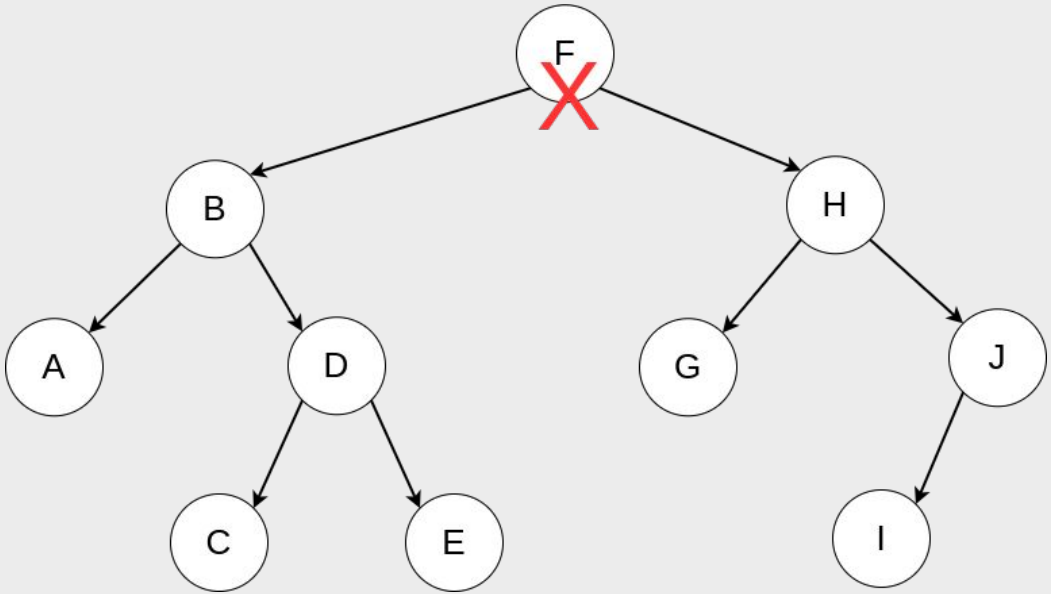
Remover nó com 1 filho

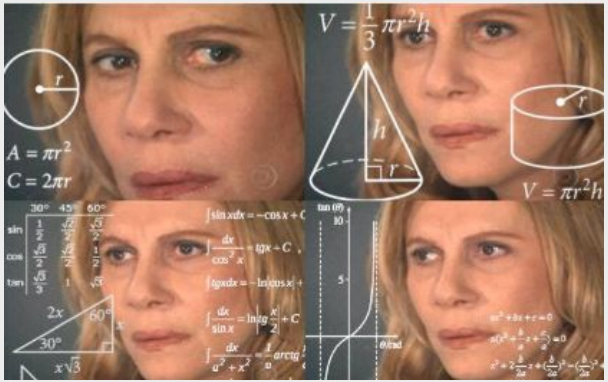
- Se o nó não possui filho direto, retorne o endereço do filho esquerdo;
- Se o nó não possui filho esquerdo, retorne o endereço do filho direito;
- O pai (G) do nó a ser removido (I) aponta para o endereço retornado (H) acima.



Remover nó com 2 filhos

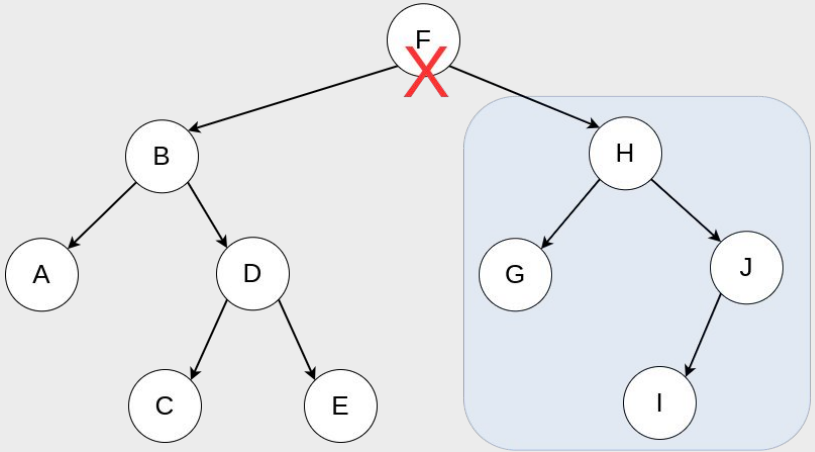
Caso mais complicado! E agora? 😳

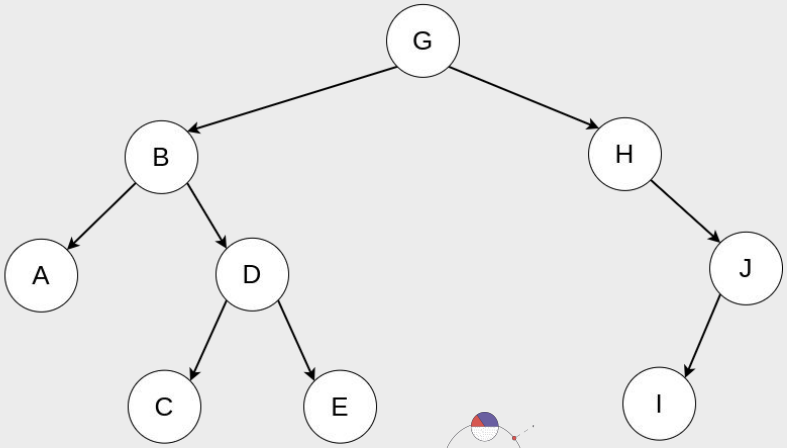




Remover nó com 2 filhos

- Percorra a subárvore direita do nó a ser removido, procurando pelo menor elemento contido nela (algoritmo **min()** do Coursera);
- Uma vez encontrado, este elemento ficará no lugar do nó a ser removido!
- Mas ele, por sua vez, precisa ser removido também de sua posição original...
  - Algoritmo **deleteMin()** do Coursera!





inteli

Percursos em árvores binárias

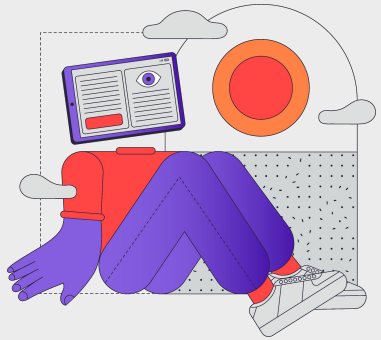
Percursos em árvores binárias podem ser implementados de maneira simples através do uso da recursão.

Tipos de percurso:

- **Pré-ordem**: (dica: R-E-D)
  - Imprime/armazena a raiz
  - Visite a subárvore da esquerda
  - Visite a subárvore da direita

- **Em-ordem (ordem simétrica)**: (dica: E-R-D)
  - Visite a subárvore da esquerda
  - Imprime/armazena a raiz
  - Visite a subárvore da direita

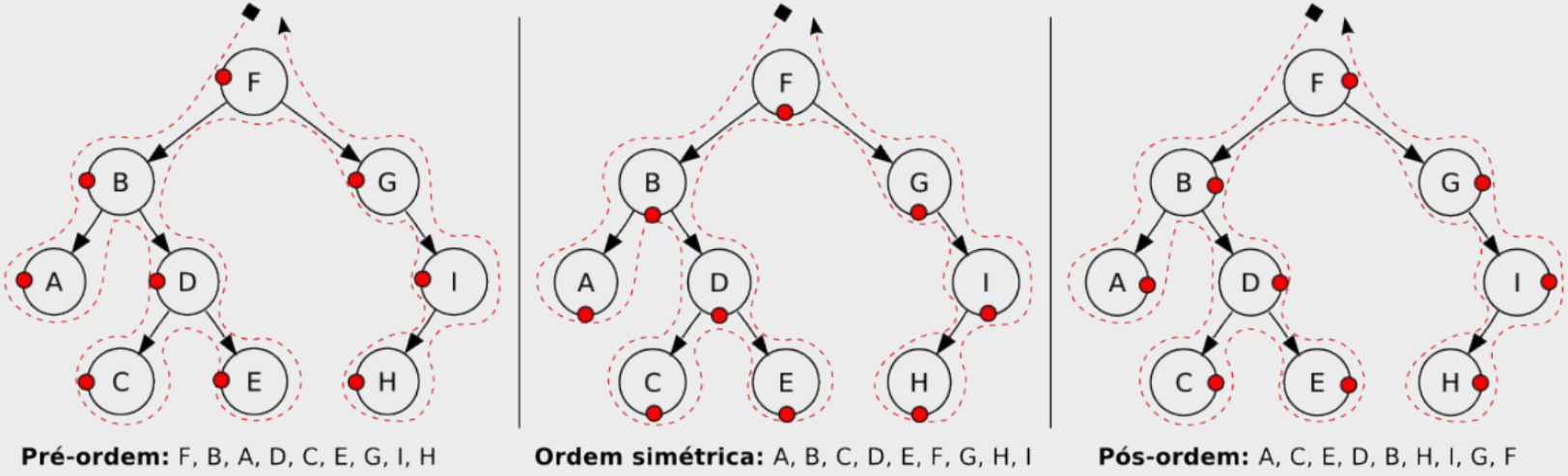
- **Pós-ordem**: (dica: E-D-R)
  - Visite a subárvore da esquerda
  - Visite a subárvore da direita
  - Imprime/armazena a raiz



inteli

Percursos em árvores binárias

- **Pré-ordem** (R-E-D)
- **Em-ordem** (E-R-D)
- **Pós-ordem** (E-D-R)

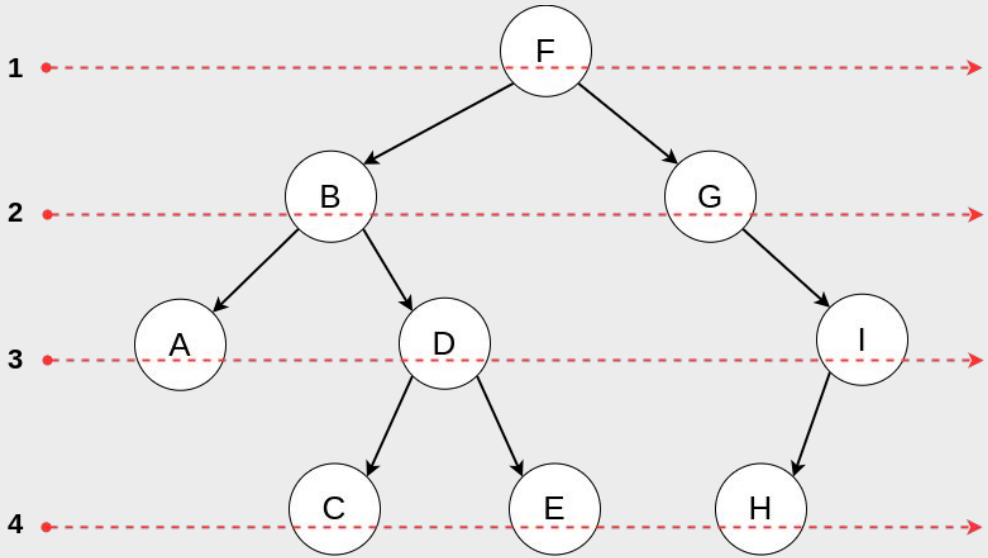


Pense nas visitas às subárvores de maneira recursiva: para cada nó, o restante das tarefas (**ex**: imprimir a raiz, visitar a subárvore da direita, etc) ficam pendentes para serem resolvidas mais à frente!

inteli

Percurso em Largura (Breadth-First)

Também é possível percorrer os elementos de uma árvore “em largura” com auxílio de uma fila!



Percurso em largura: F, B, G, A, D, I, C, E, H

inteli


inteli