

UFPI - CCN - DC
Ciência da Computação
Estrutura de Dados

Árvore Binária



Prof. Raimundo Moura
rsm@ufpi.edu.br

106

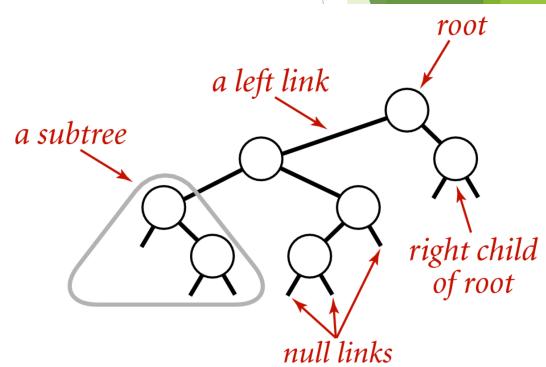
Árvores Binárias de Busca (BSTs)

- ▶ BSTs servem para implementar **TSs ordenadas**, ou seja, TSs cujas chaves são comparáveis.
- ▶ BSTs combinam as vantagens das implementações elementares **SequentialSearchST** e **BinarySearchST**: elas podem ser vistas como uma maneira de implementar busca binária em uma lista ligada.

107

O que é uma Árvore Binária?

- ▶ BT = binary tree = árvore binária.
- ▶ Cada nó tem no máximo dois filhos:
um *esquerdo* e um *direito*.



Anatomy of a binary tree

108

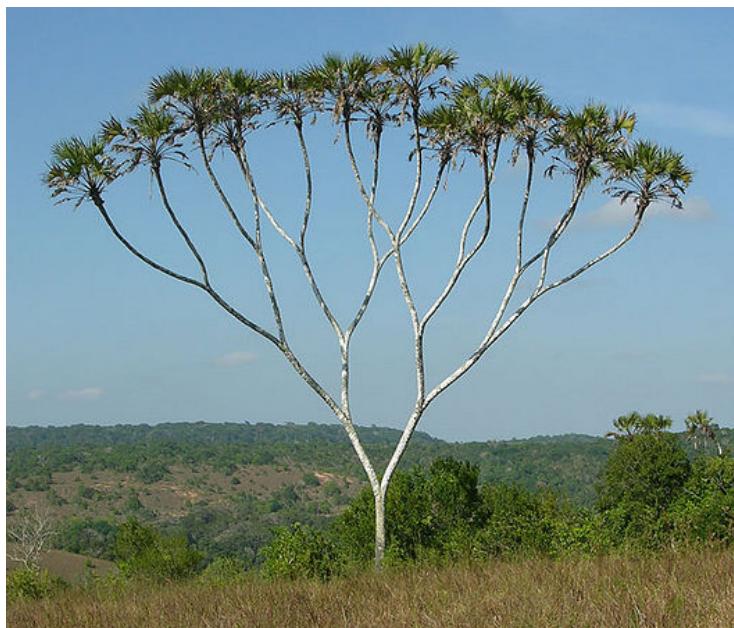
Árvore Binária na Natureza



© Shlomit Pinter

109

Árvore Binária na Natureza



110

Árvore Binária

► Definição:

```
private class Node {  
    private Node left, right;  
}
```

► A raiz (root) é o único nó que não é filho de outro. A árvore é vazia se root == null.

```
private Node root;
```

► BTs são estruturas *recursivas*: cada nó da BT é raiz de uma sub-BT

111

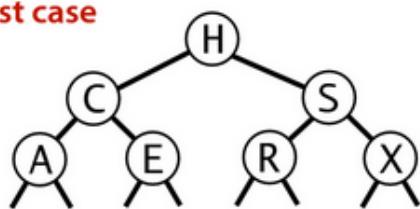
Árvore Binária

- ▶ A *profundidade (depth)* de um nó de uma BT é o número de links no caminho que vai da raiz até o nó.
- ▶ A *altura (height)* de uma BT é o máximo das profundidades dos nós, ou seja, a profundidade do nó mais profundo.
- ▶ Uma BT com N nós, tem altura no máximo $N-1$ e no mínimo $\lfloor \lg N \rfloor$. Se a altura estiver perto de $\lg N$, a BT é *balanceada*.

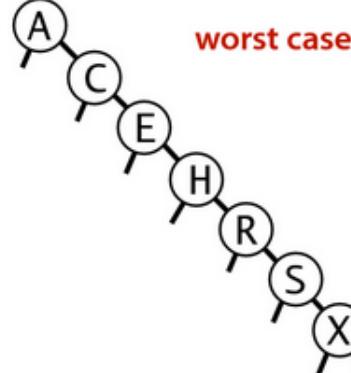
112

Árvore Binária: Exemplos

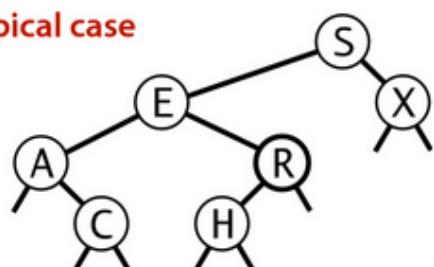
best case



worst case



typical case



113

Árvore Binária

- ▶ O *comprimento interno* (internal path length) de uma BT é a soma das profundidades dos seus nós, ou seja, a soma dos comprimentos de todos os caminhos que levam da raiz até um nó.

(Esse conceito é usado para estimar o desempenho esperado de TSs implementadas com BSTs)

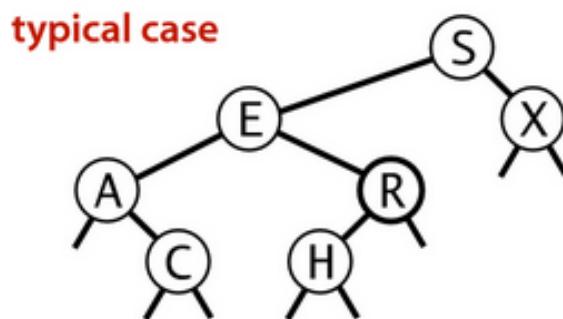
114

Árvore Binária: Travessias

- ▶ Para visitar os nós de uma árvore existem três tipos de travessias ou percursos:
 - **Em-ordem:** visitar a subárvore da esquerda; visitar a raiz; e visitar a subárvore da direita. (*Esq-Raiz-Dir*).
 - **Pre-ordem:** visitar a raiz; visitar a subárvore da esquerda; e visitar a subárvore da direita. (*Raiz-Esq-Dir*).
 - **Pos-ordem:** visitar a subárvore da esquerda; visitar a subárvore da direita; e visitar a raiz. (*Esq-Dir-Raiz*).

115

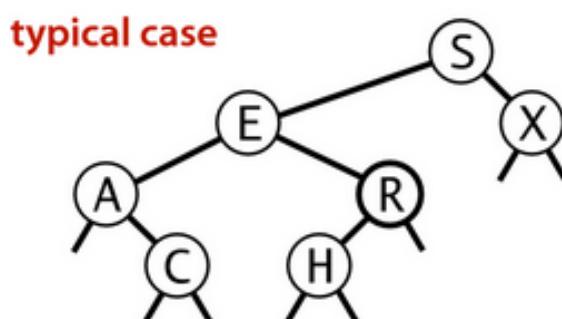
BST: Execução: S, X, E, R, H, A, C



- ▶ Altura: 3
- ▶ Comprimento interno: 12
- ▶ levelOrdem: S, E, X, A, R, C, H

116

Árvore Binária: Travessias

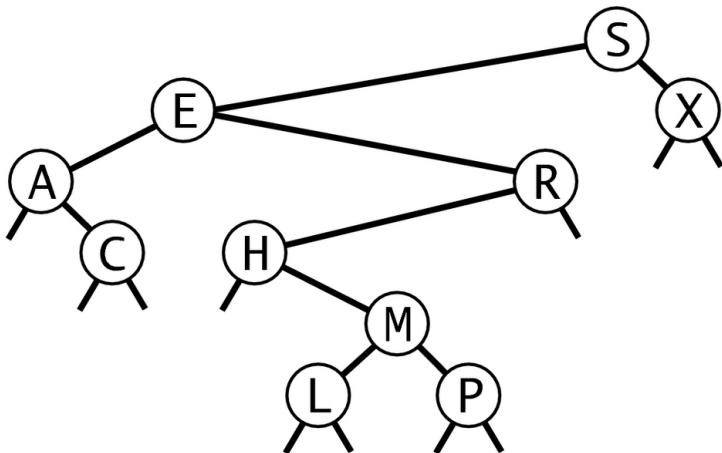


- ▶ Em-Ordem: A, C, E, H, R, S, X
- ▶ Pre-Ordem: S, E, A, C, R, H, X
- ▶ Pos-Ordem: C, A, H, R, E, X, S

117

Árvore Binária: Exercícios

Considere a Árvore Binária da Figura



118

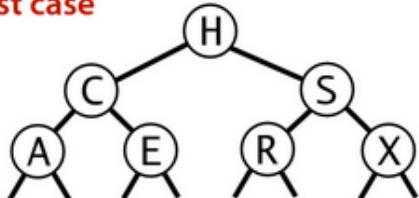
Árvore Binária: Exercícios

1. Para a BT que aparece na figura do slide anterior, escolha um nó no meio da árvore e diga qual a profundidade do nó.
2. Qual a altura da árvore da figura? Qual a profundidade do nó M? Qual a altura da sub-árvore cuja raiz é M?
3. Dê a altura e o comprimento interno de cada uma das BTs que aparecem nas figuras da próxima página.
4. **Prove que a profundidade de um nó em uma BT com N nós é no máximo $N-1$ e no mínimo 0.**
5. **Prove que a altura de toda BT com N nós é no máximo $N-1$ e no mínimo $\lfloor \lg N \rfloor$.**

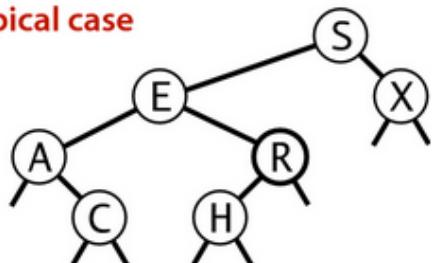
119

Árvore Binária: Exemplos

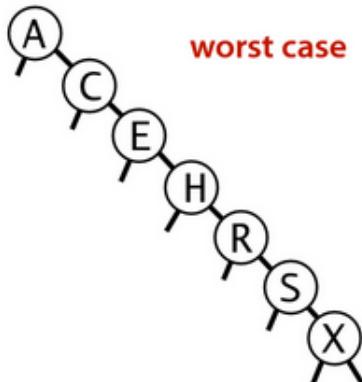
best case



typical case



worst case



120

Árvore Binária: Implementações

1. **Tamanho.** Escreva um método recursivo que devolva o número de nós de uma árvore binária.
2. **Altura.** Escreva um método (recursivo) que calcule a altura de uma árvore binária. Sua implementação deve ser do tipo preguicosa. (Não é preciso calcular as profundidades antes)
3. Acrescente a Node um campo **depth** para armazenar a profundidade do nó. Escreva um método **setDepthField()** que defina o valor do campo **depth** de todos os nós.
4. Escreva um método **internalPathLength** que calcule o comprimento interno de uma BT (árvore binária).
5. **Percursos (traversals).** Escreva um método que imprima as chaves de uma BT em *in-ordem* (ou seja, na ordem esquerda-raiz-direita); use recursão. Repita para *pós-ordem* (ordem esquerda-direita-raiz). Repita para *pré-ordem* (ordem raiz-esquerda-direita). **SUGESTÃO:** Use uma fila para armazenar as chaves antes de imprimir.

121