Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Programa de Pós-Graduação em Ciência da Computação

Árvores Balanceadas

Autor: Nelson Cruz Sampaio Neto

Carlos Gustavo Resque dos Santos
gustavoresqueufpa@gmail.com

10 de abril de 2019

Introdução

- A árvore de busca (ou pesquisa) é uma estrutura de dados muito eficiente para armazenar informação.
- Estrutura adequada quando existe necessidade de considerar todos ou alguma combinação de requisitos, tais como:

  i. acessos direto e sequencial eficientes;

  ii. facilidade de inserção e retirada de registros;

  iii. boa taxa de utilização de memória; e

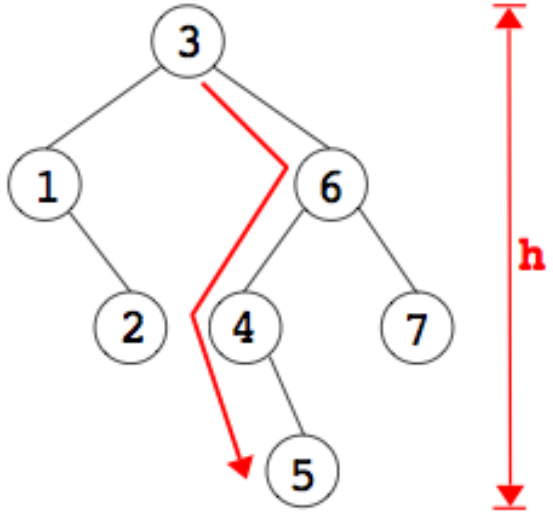
  iv. utilização de memória primária e secundária.

Introdução

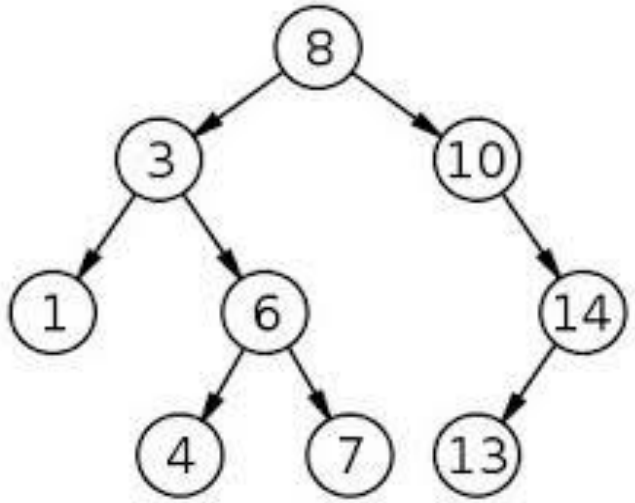
- As árvores de busca binária são estruturas de dados encadeadas em que cada nó é um objeto.

- Toda árvore de busca binária deve satisfazer a propriedade:

  - Seja x um nó qualquer da árvore
  - Seja y um nó da sub-árvore da esquerda, então chave[y] < chave[x]
  - Seja y um nó da sub-árvore da direita, então chave[y] > chave[x]



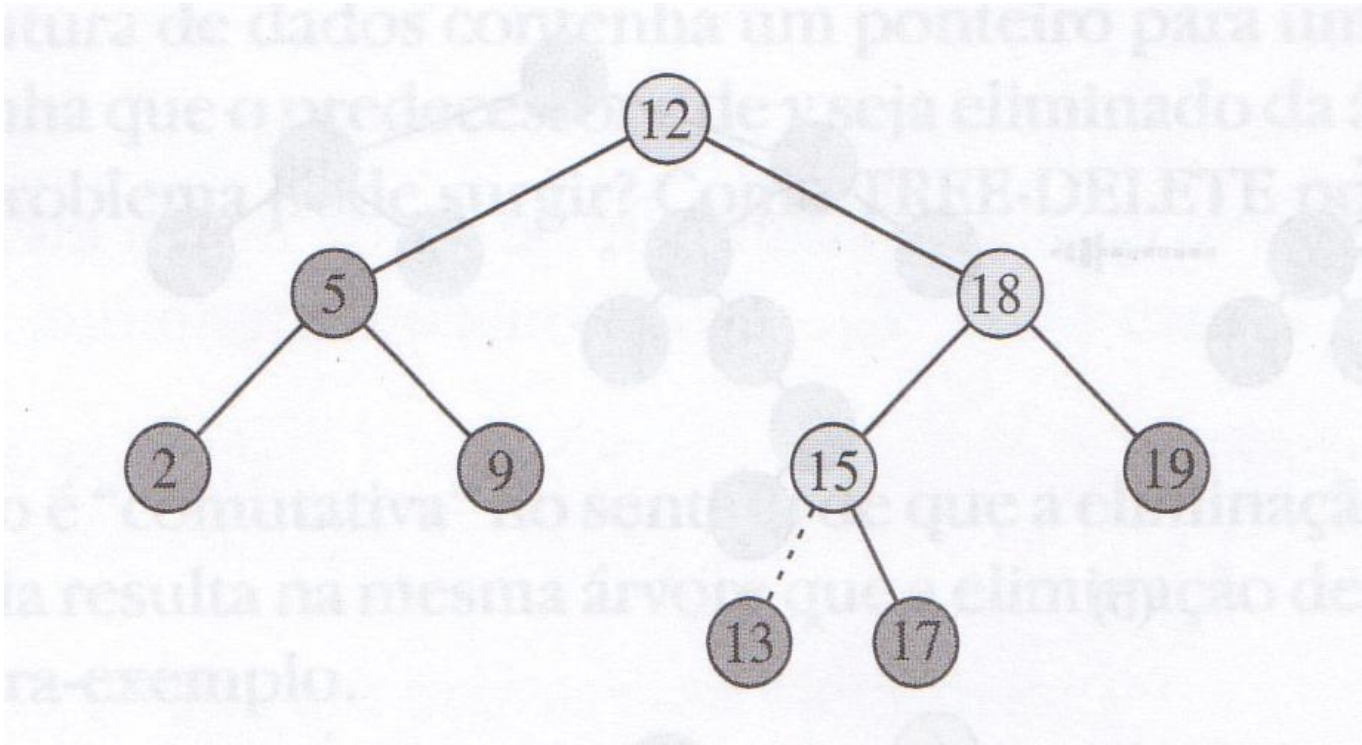
Introdução



- Busca pela chave 4 na árvore binária acima: 8 – 3 – 6 – 4.
- A chave mínima é 1, seguindo os ponteiros a esquerda a partir da raiz.
- A chave máxima é 14, seguindo os ponteiros a direita a partir da raiz.

Introdução

- Inserção do nó com chave 13. Os nós levemente sombreados indicam o caminho desde a raiz até a posição em que o item é inserido.



Introdução

- A operação de remoção de um nó *z* em uma árvore binária de busca consiste de 3 casos:

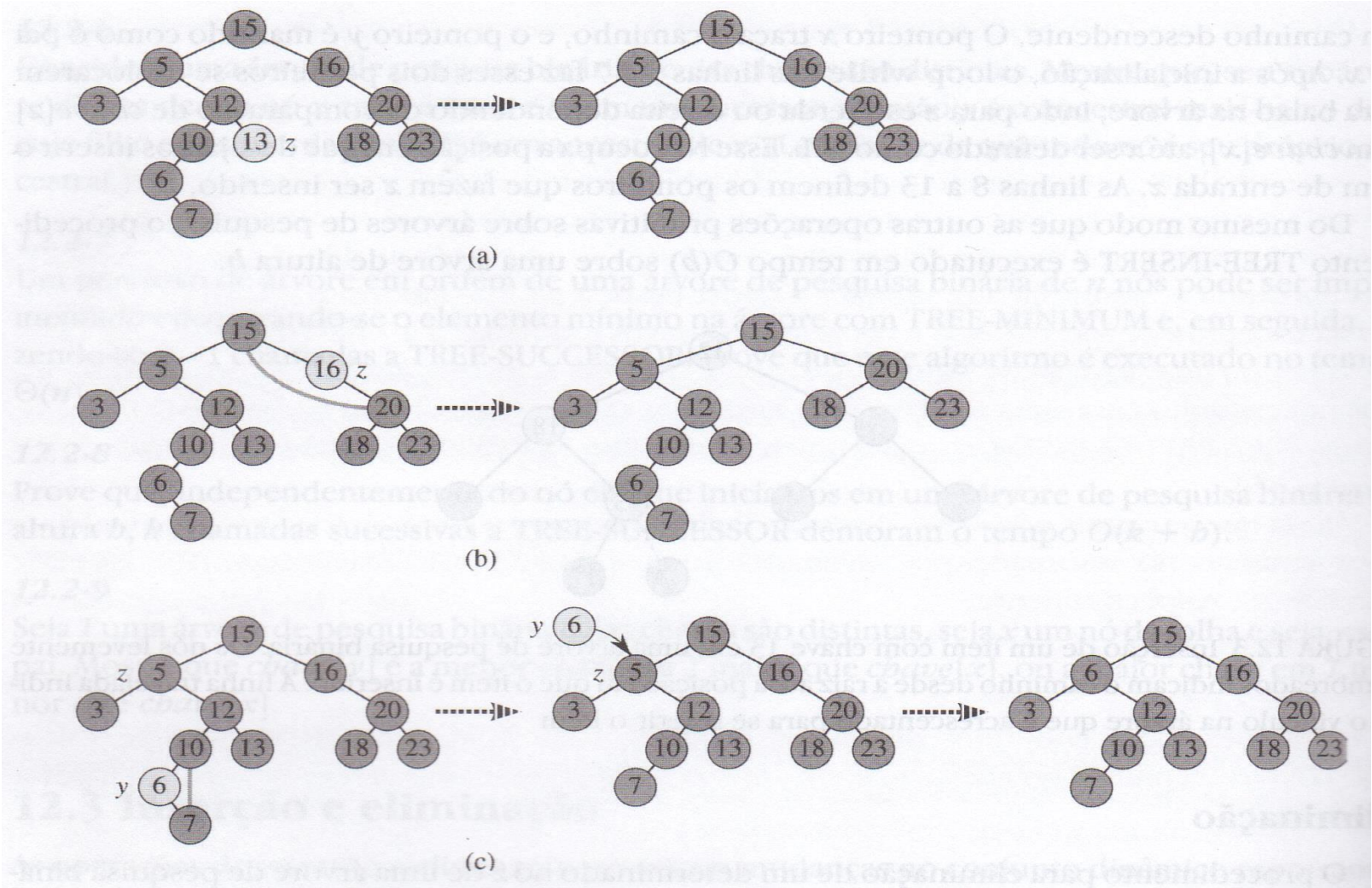
- a) Se *z* não tem filhos, modificar o pai de *z* para que este aponte para NIL.

- b) Se *z* possui apenas um filho, fazemos o pai de *z* apontar para o filho de z.

- c) Se *z* possui dois filhos, colocamos no lugar de *z* o seu sucessor *y*, que com certeza não possui filho à esquerda.

- O sucessor de um nó *x* é o nó *y* com a menor chave que seja maior do que a chave de *x*.
-

Introdução



Introdução

- Como visto, árvores de pesquisa binária admitem operações de conjuntos dinâmicos: seleção, inserção, remoção, entre outras.

- As operações levam um tempo proporcional à **altura da árvore**.
  Complexidade temporal no pior caso: $O(h)$.

- Para uma árvore completa de n nós, tais operações levam um tempo proporcional a $O(\log n)$. Muito bom!

- Mas se a árvore for uma lista de nós (ou seja, degenerada), as operações podem levar $O(n)$. Ruim!

Introdução

- Seja *T* uma **árvore binária completa** com *n* nós e altura *h*. Então,
  - $\mathbf{2^h \leq n \leq 2^{h+1} - 1}$

- O valor $n = 2^{h+1} - 1$ ocorre quando a árvore é cheia. Nesse caso, basta observar que o número de nós em um dado nível é igual ao dobro do anterior.

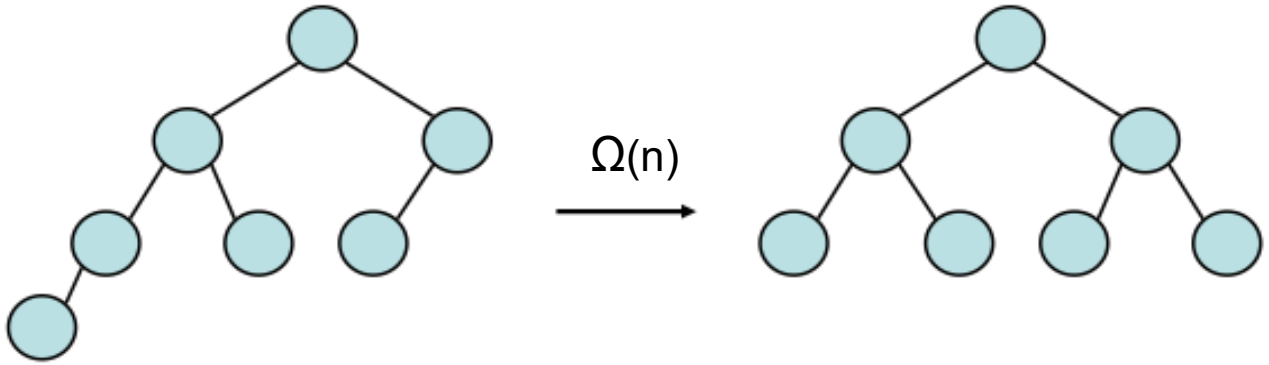
- O valor $n = 2^h$ corresponde ao caso em que há apenas um nó no último nível de *T*.

- Com isso, conclui-se que $h \leq log_2 n$, ou seja, $h = O(log\ n)$.

Introdução

- Embora uma árvore completa possua altura proporcional a *log n*, após uma operação de inserção, p.e., a árvore pode perder essa característica (figura(a)).

- Uma solução seria aplicar um algoritmo que tornasse a árvore novamente completa (figura (b)), porém, o custo dessa operação seria no mínimo proporcional a *n*, ou seja, Ω(*n*). Ruim!



(a) (b)

Introdução

- **Objetivo**: Manter o custo das operações na mesma ordem de grandeza da altura de uma árvore completa, ou seja, $O(\log n)$, onde n é o número de nós da árvore. Em outras palavras, ter sempre uma **árvore balanceada**.

- Mas para uma árvore ser balanceada (ou seja, $h = O(\log n)$) ela precisa ser completa? Não!

- Há implementações de árvores de pesquisa binária balanceadas que garantem altura $O(\log n)$, tais como árvores vermelho-preto e árvores AVL, mas que não são obrigatoriamente completas.

- As árvores B também são consideradas balanceadas, mas não são binárias e possuem como característica o armazenamento de mais de uma chave por nó.