Árvores Binárias de Busca
Adição e Busca

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

Baseado nos slides do Prof. Rudinei Goularte

Conteúdo

- Conceitos Introdutórios
- Operações
  - Inserção
  - Pesquisa
  - Remoção
- Conceitos Adicionais

Definições

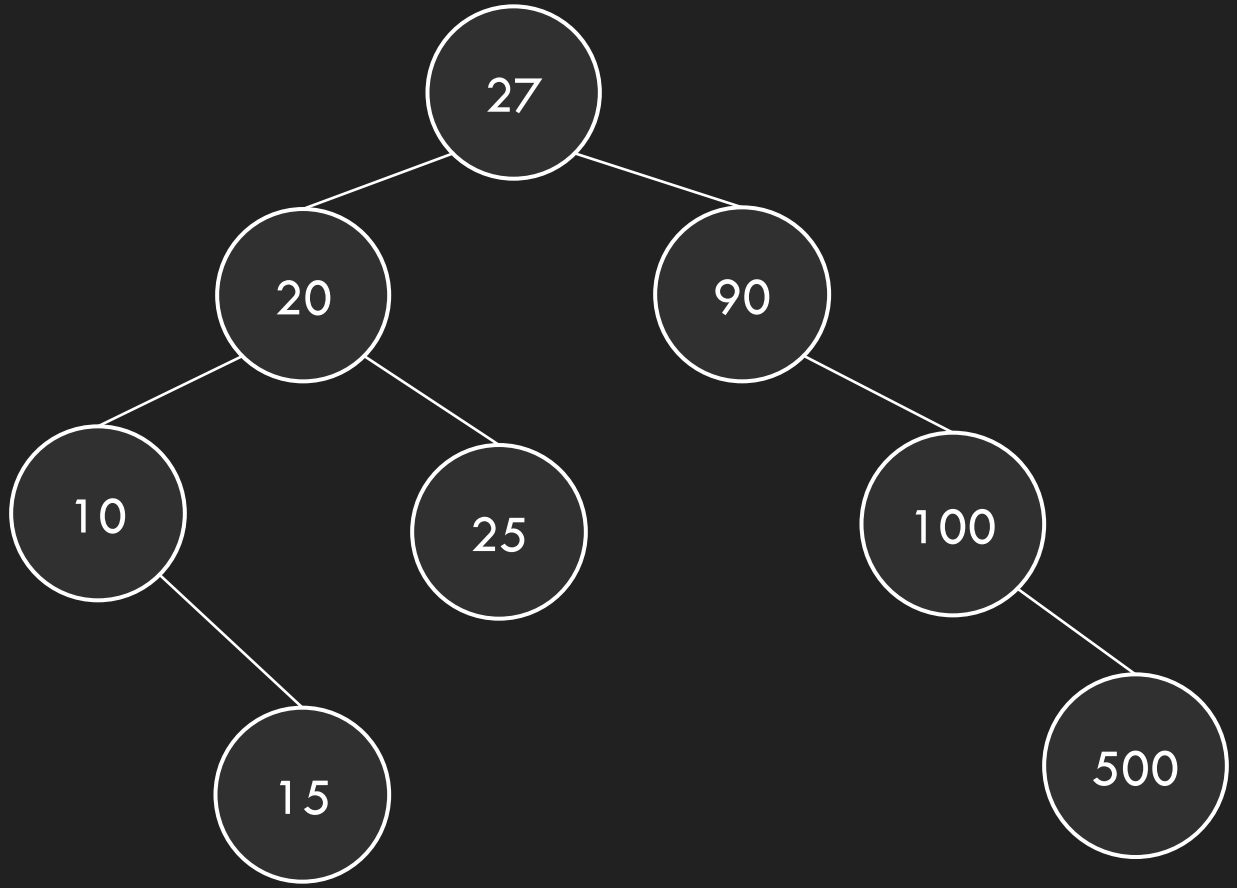
- Uma Árvore Binária de Busca (ABB) possui as seguintes propriedades
  - Seja $S = \{S_1, ..., S_n\}$ o conjunto de chaves dos nós da árvore T
    - Esse conjunto satisfaz $S_1 < ... < S_n$
    - A cada nó $v_j \in T$ está associada uma chave distinta $S_j \in S$, que pode ser consultada por $r(v_j) = S_j$

Definições

- Uma Árvore Binária de Busca (ABB) possui as seguintes propriedades
  - Dado um nó v de T
  - Se v_i pertence à sub-árvore esquerda de v, então $r(v_i) < r(v_j)$
  - Se v_i pertence à sub-árvore direita de v, então $r(v_i) > r(v_j)$

Definições

- Os nós pertencentes à sub-árvore esquerda possuem valores menores do que o valor associado ao nó-raiz r
- Os nós pertencentes à sub-árvore direita possuem valores maiores do que o valor associado ao nó-raiz r



Exemplo de ABB

Definições

- Um **percurso em-ordem** em uma ABB resulta na sequência de valores em **ordem crescente**
- Se **invertêssemos as propriedades** descritas na definição anterior, de maneira que a sub-árvore esquerda de um nó contivesse valores maiores e a sub-árvore direita valores menores, o percurso em-ordem resultaria nos valores em **ordem decrescente**

Definições

- Uma **ABB** criada a partir de um conjunto de valores **não é única:** o resultado depende da sequência de inserção dos dados
- A grande utilidade da árvore binária de busca é armazenar dados contra os quais outros dados são frequentemente verificados (busca!)

Definições

➔ Uma árvore binária de busca é dinâmica e pode sofrer alterações (inserções e remoções de nós) após ter sido criada

Lista vs ABB

Lista vs ABB

- O tempo de busca é estimado pelo número de comparações entre chaves.
- Em listas de n elementos, temos:
  - Sequenciais (Array): O(n) se não ordenadas; ou O(log2n), se ordenadas
  - Encadeadas (Dinâmicas): O(n)

Lista vs ABB

➔ As ABB constituem a alternativa que combina as vantagens de ambos: são encadeadas e permitem a busca binária O(log2n)!!!

Operações em ABB

Operações em ABB

- Inserção
- Pesquisa/Busca
- Remoção

Inserção em ABB

Inserção em ABB

- Passos do algoritmo de inserção
  - Procure um “local” para inserir o novo nó, começando a procura a partir do nó-raiz
  - Para cada nó-raiz de uma sub-árvore, compare:
    - Se o novo nó possui um valor menor do que o valor no nó-raiz (vai para sub-árvore esquerda), ou
    - Se o valor é maior que o valor no nó-raiz (vai para sub-árvore direita)

Inserção em ABB

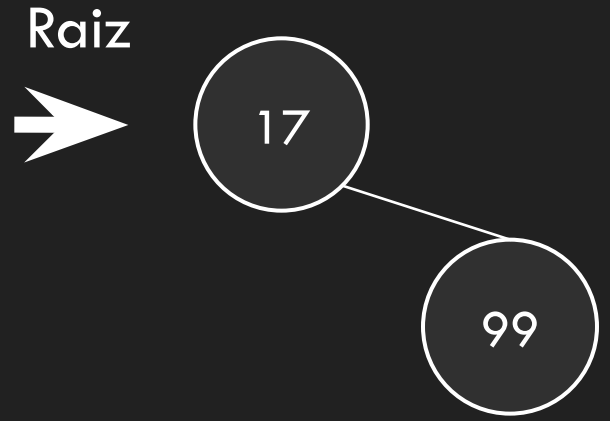
- Passos do algoritmo de inserção
  - Se um ponteiro (filho esquerdo/direito de um nó-raiz) nulo é atingido, coloque o novo nó como sendo filho do nó-raiz

Inserção em ABB

- Para entender o algoritmo considere a inserção do conjunto de números, na sequência
  - 17, 99, 13, 1, 3, 100, 400
- No início a ABB está vazia!

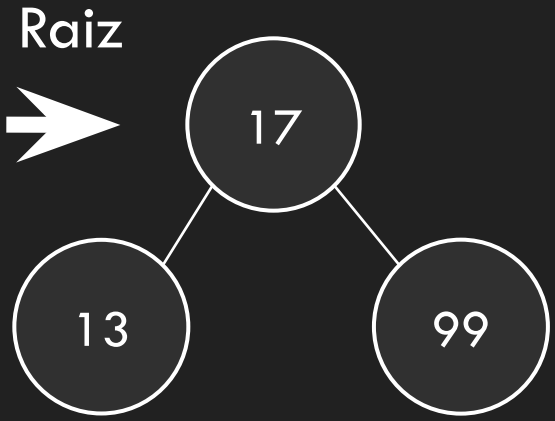
Inserção em ABB

- O número 17 será inserido tornando-se o nó raiz
- A inserção do 99 inicia-se na raiz. Compara-se 99 com 17
- Como 99 > 17, 99 deve ser colocado na sub-árvore direita do nó contendo 17 (subárvore direita, inicialmente, nula)



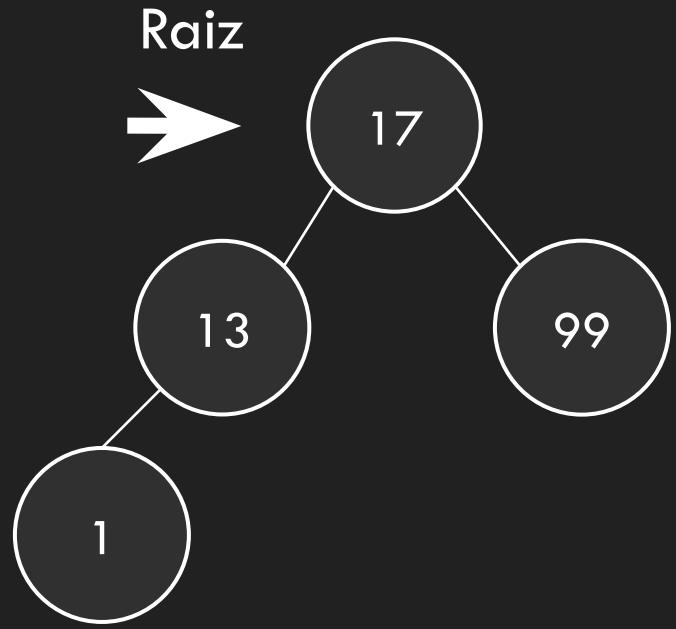
Inserção em ABB

- A inserção do 13 inicia-se na raiz
- Compara-se 13 com 17. Como 13 < 17, 13 deve ser colocado na sub-árvore esquerda do nó contendo 17
- Já que o nó 17 não possui descendente esquerdo, 13 é inserido na árvore nessa posição



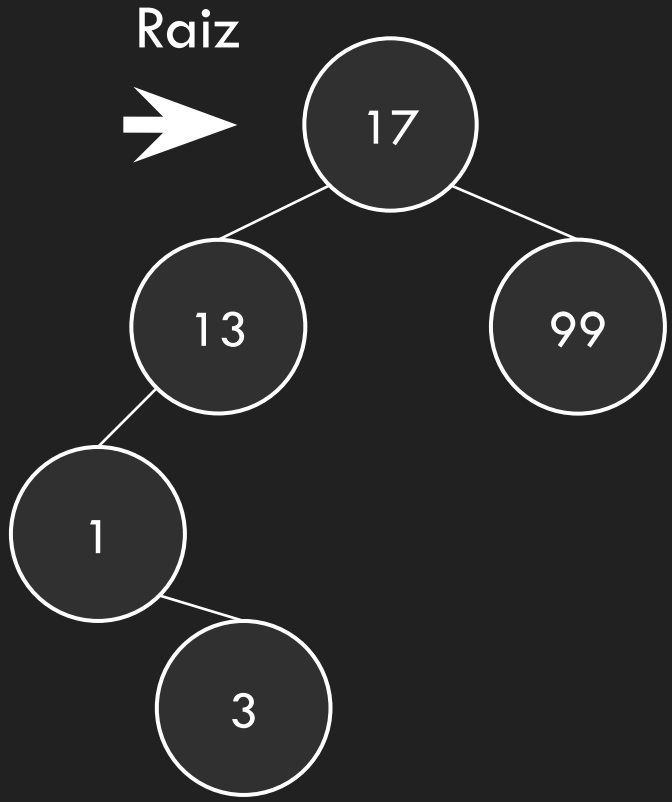
Inserção em ABB

- Repete-se o procedimento para inserir o valor 1
- 1 < 17, então será inserido na sub-árvore esquerda
- Chegando nela, encontra-se o nó 13, 1 < 13 então ele será inserido na sub-árvore esquerda de 13



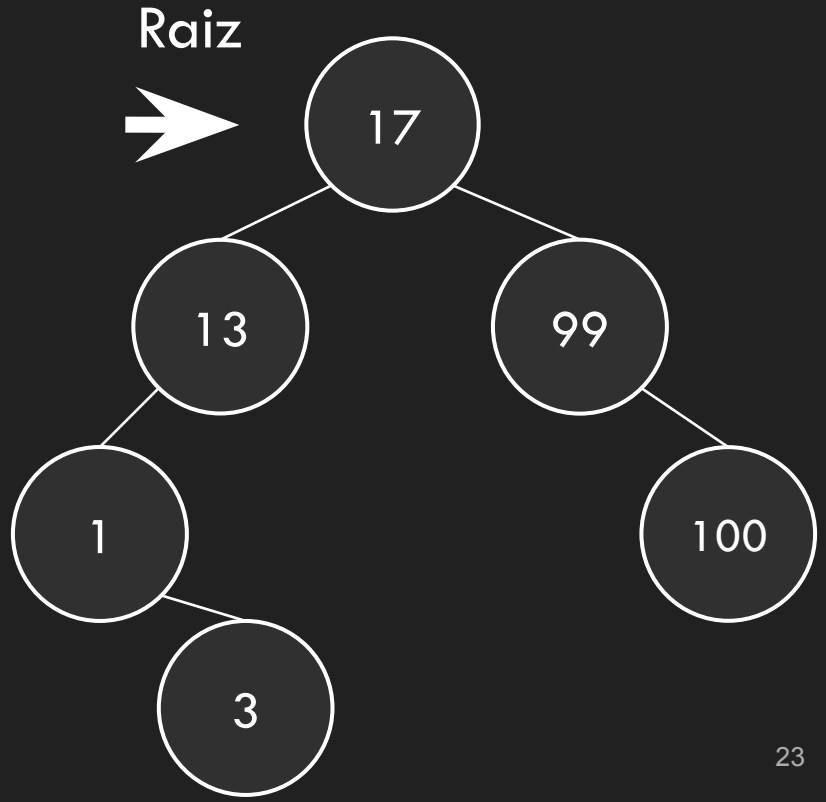
Inserção em ABB

- Repete-se o procedimento para inserir o elemento 3
- 3 < 17
- 3 < 13
- 3 > 1



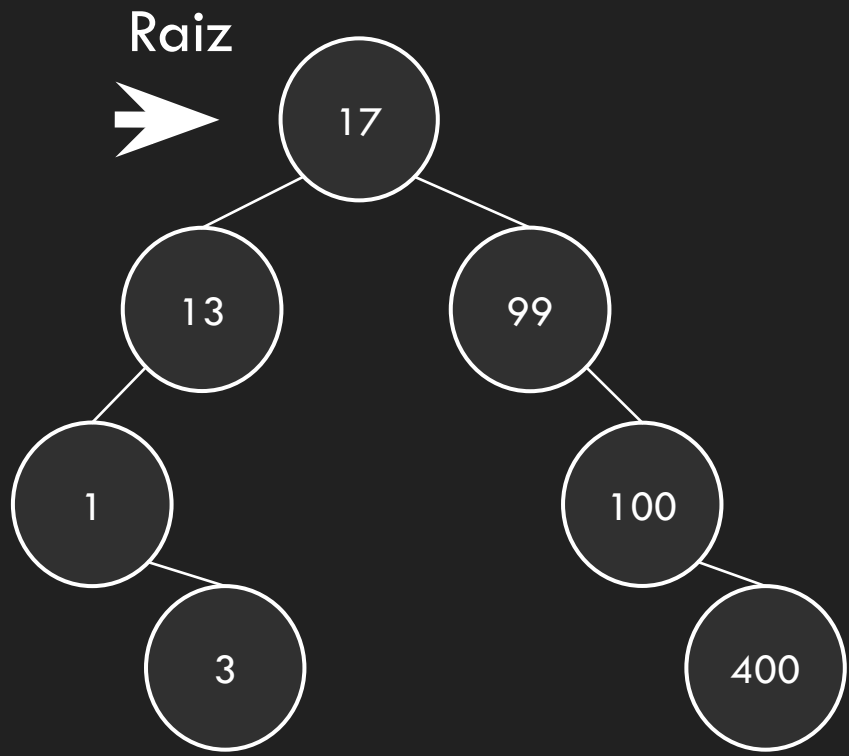
Inserção em ABB

- Repete-se o procedimento para inserir o elemento 100
- 100 > 17
- 100 > 99



Inserção em ABB

- Repete-se o procedimento para inserir o elemento 400
- 400 > 17
- 400 > 99
- 400 > 100



Código para Inserção em ABBs

Inserção em ABB

```
NO *abb_inserir_no(NO *raiz, ITEM *item){
    if (raiz == NULL)
        raiz = abb_cria_no(item);
    else if(item_get_chave(item) > item_get_chave(raiz->item))
        raiz->dir = abb_inserir_no(raiz->dir,item);
    else if(item_get_chave(item) < item_get_chave(raiz->item))
        raiz->esq = abb_inserir_no(raiz->esq,item);
    return(raiz);
}
boolean abb_inserir (ABB *T, ITEM *item){
    return((T->raiz = abb_inserir_no(T->raiz, item)) != NULL);
}
```

Custo da Inserção em ABB

- ➔ A inserção requer uma busca pelo lugar da chave, portanto, com custo de uma busca qualquer (tempo proporcional à altura da árvore).
- ➔ O custo da inserção, após a localização do lugar, é constante; não depende do número de nós.
- ➔ Logo, tem complexidade análoga à da busca.

Exercício

Criar um método iterativo para inserção em ABB

Busca em ABB

Busca em ABB

- Passos do algoritmo de busca
  - Comece a busca a partir do nó-raiz
  - Para cada nó-raiz de uma sub-árvore compare:
    - Se o valor procurado é menor que o valor no nó-raiz (continua pela sub-árvore esquerda)
    - Se o valor é maior que o valor no nó-raiz (sub-árvore direita)

Busca em ABB

- Passos do algoritmo de busca
  - Caso o nó contendo a chave pesquisada seja encontrado
    - Retorne o “item” do nó pesquisado
  - Caso contrário
    - Retorne nulo

Busca em ABB

```
ITEM *abb_busca2(NO *raiz, int chave){
     if(raiz == NULL)
          return NULL;
     if(chave == item_get_chave(raiz->item))
          return (raiz->item);
     if(chave < item_get_chave(raiz->item))
          return (abb_busca2(raiz->esq, chave));
     else
          return (abb_busca2(raiz->dir, chave));
}
ITEM *abb_busca(ABB *T, int chave){
     return(abb_busca2(T->raiz, chave));
}
```

Custo da Busca em ABB

- Pior caso
  - Número de passos é determinado pela altura da árvore
  - Árvore degenerada possui altura igual a n
- Altura da árvore depende da sequência de inserção das chaves
  - O quê acontece se uma sequência ordenada de chaves é inserida?

Custo da Busca em ABB

- ➔ Busca é eficiente se árvore está razoavelmente balanceada
  - ◆ $O(Log_2 n)$

ABB Aleatória

ABB Aleatória

- Nós externos:
  - Descendentes dos nós folhas
    - (não estão, de fato, na árvore)
- Uma árvore A com n nós possui n+1 nós externos
- Uma inserção em A é considerada “aleatória” se ela tem probabilidade igual de acontecer em qualquer um dos n+1 nós externos

ABB Aleatória

- Uma ABB aleatória com n nós é uma árvore resultante de n inserções aleatórias sucessivas em uma árvore inicialmente vazia
- É possível demonstrar que para uma ABB “aleatória” o número esperado de comparações para recuperar um registro qualquer é cerca de $1{,}39 * \log_2(n)$
  - 39% pior do que o custo de acesso em uma árvore balanceada

ABB Aleatória

➔ Pode ser necessário garantir um melhor balanceamento da ABB para melhor desempenho na busca

Referências

- Material baseado no originais produzidos pelo professor Rudinei Gularte
- SZWARCFITER, J. L.; MARKENZON, L. Estruturas de Dados e seus Algoritmos, Livros Técnicos e Científicos, 1994.
- TENEMBAUM, A.M., e outros Data Structures Using C, Prentice-Hall, 1990.
- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.