Programação

IFMG CODAAUT

Prof. Marco Antonio M. Carvalho

UFOP
Universidade Federal de Ouro Preto

PROEX
Pró-reitoria de extensão da UFOP

decom
departamento de computação

INSTITUTO FEDERAL
MINAS GERAIS

Lembretes

- **Lista de discussão**
  - Endereço:
    - programaacao@googlegroups.com
  - Solicitem acesso:
    - http://groups.google.com/group/programaacao
- **Página com material dos treinamentos**
  - http://www.decom.ufop.br/marco/extensao/obi/
- **Repositório online de problemas das edições passadas da OBI**
  - http://br.spoj.com/problems/obi/sort=-7
- **Moodle**
  - http://programaacao.net.br/login/index.php

Avisos

- Enviar o id do URI Online Judge;
- Aceitar o convite do grupo criado.

Na aula de hoje

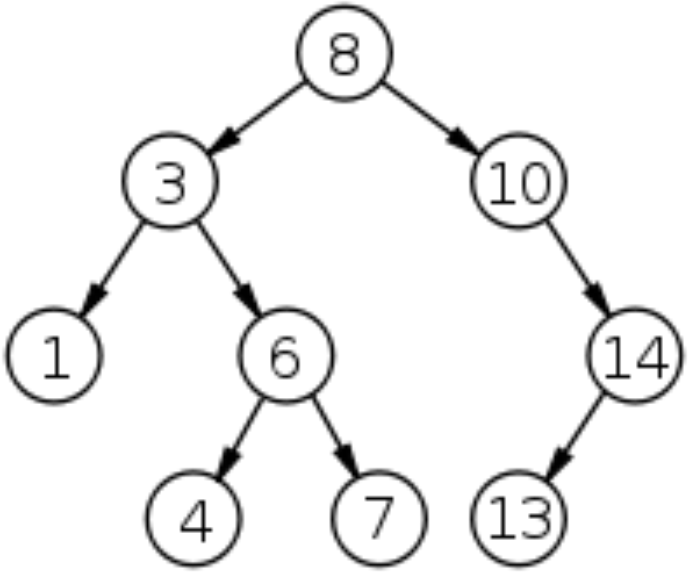
- Árvores Binárias de Busca
- *set*
- *multiset*
- Problemas Selecionados
- Um Problema de Lógica

Árvores Binárias de Busca

Árvore de Busca Binária

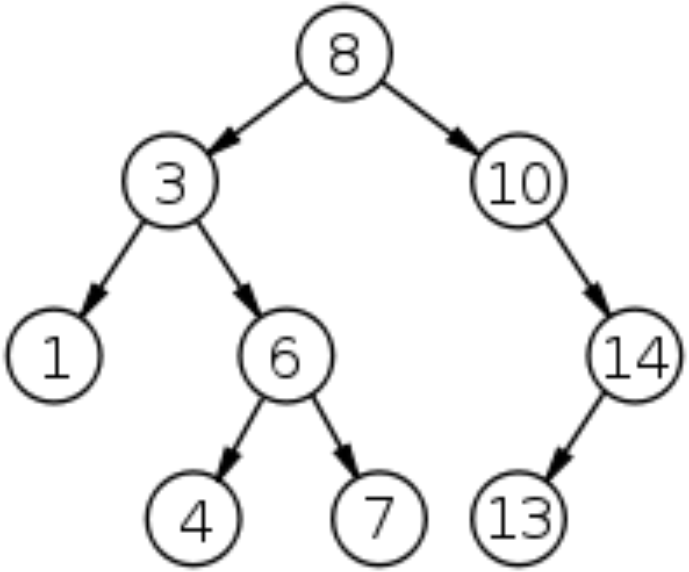
- Como o próprio nome diz, uma **árvore binária de busca** é uma estrutura de dados do tipo **árvore**
  - Armazena os dados de forma hierárquica;
  - **Nós** são todos os elementos guardados na árvore;
  - **Raiz** é o nó do topo da árvore;
  - **Filhos** são os nós que vem depois dos outros nós;
  - **Pais** são os nós que vem antes dos outros nós;
  - **Folhas** são os nós que não têm filhos; são os últimos nós da árvore.

Árvore de Busca Binária



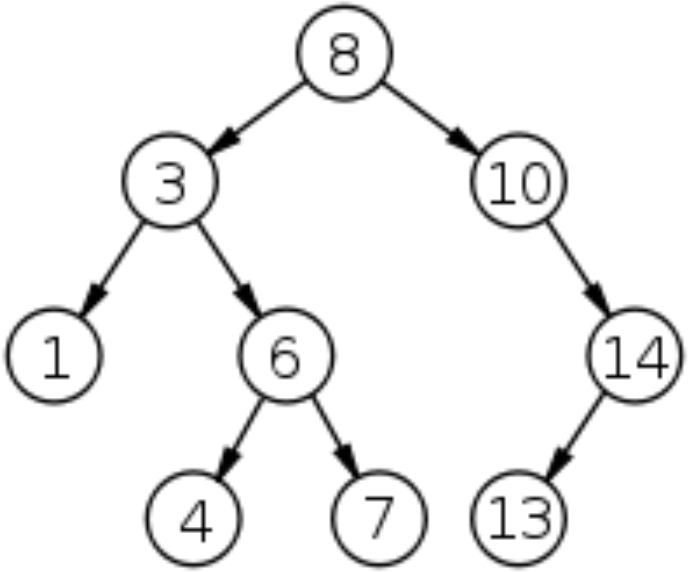
- **Tamanho:** 9;
- **Raiz**: vértice 8;
- **Folhas**: vértices 1, 4, 7 e 13

Árvore de Busca Binária



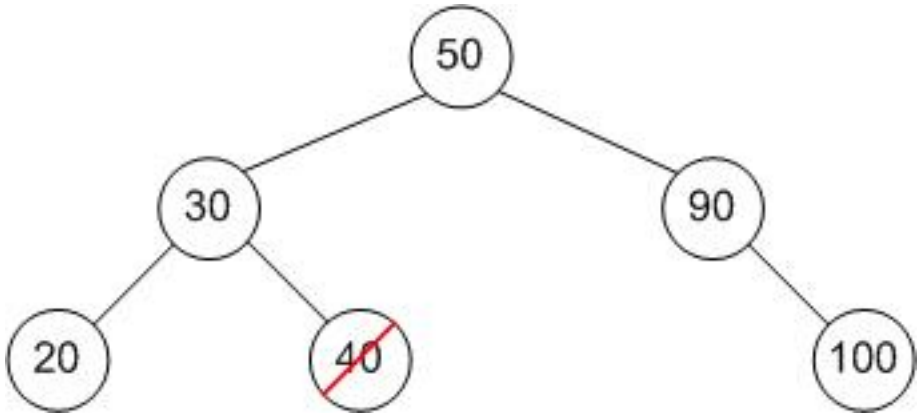
- **Características:**
  - A **subárvore** esquerda de um nó **X** possui apenas valores menores do que **X**;
  - A **subárvore** direita de um nó **X** possui apenas valores maiores do que **X**;
  - Toda subárvore também deve ser uma árvore binária de busca;
  - Não há valores duplicados.

Árvore de Busca Binária

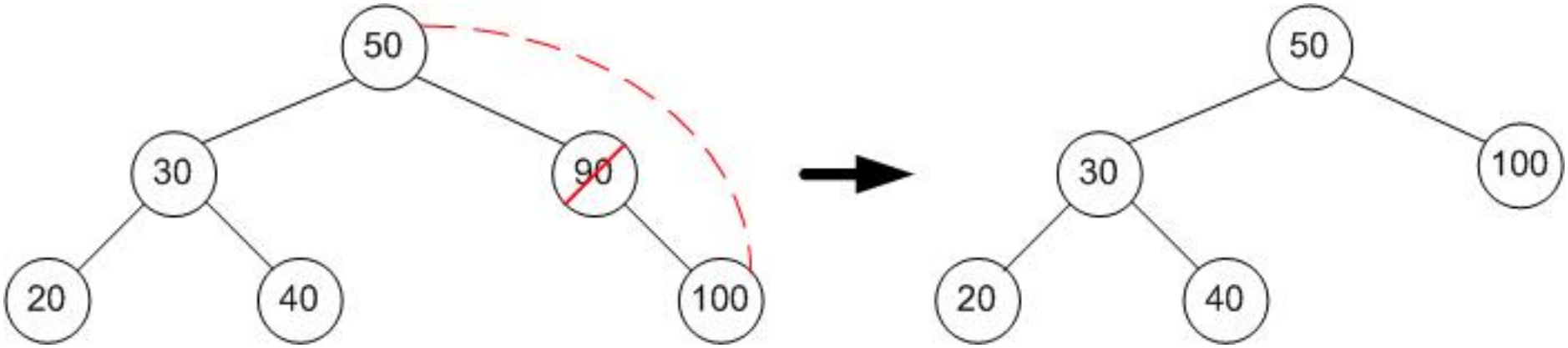


Operação	Média	Pior Caso
Memória	O(*n*)	O(*n*)
Busca	O(log *n*)	O(*n*)
Inserção	O(log *n*)	O(*n*)
Remoção	O(log *n*)	O(*n*)

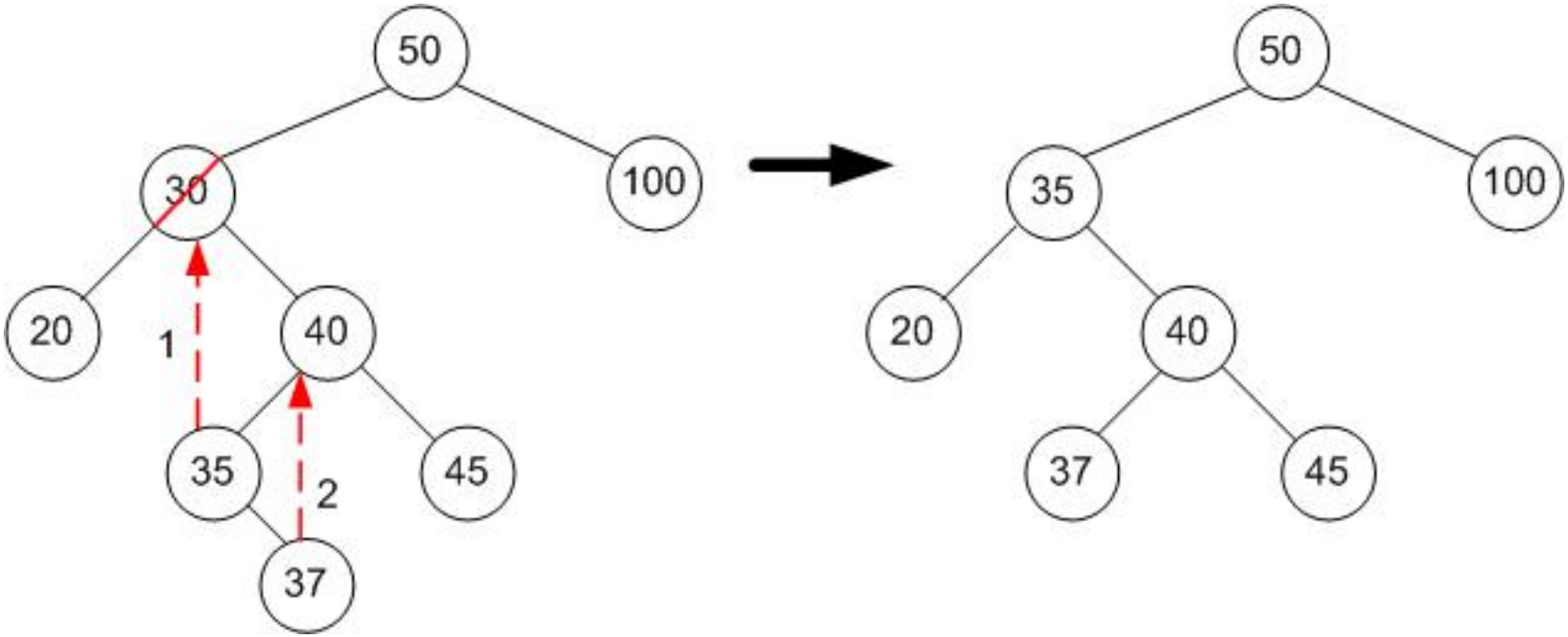
Árvore de Busca Binária - Remoção de um nó folha



Árvore de Busca Binária - Remoção de um nó com um filho



Árvore de Busca Binária- Remoção de um nó com dois filhos



Árvore de Busca Binária

- Operações em uma árvore binária requerem comparações entre nós
  - Essas comparações são feitas com chamadas a um **comparador**, que é uma *subrotina* que calcula a ordem linear em dois valores quaisquer;
  - Esse comparador pode ser explícita ou implicitamente definido, dependendo da linguagem em que a árvore binária de busca está implementada.

set

set

- Conjuntos são estruturas de dados associativas que armazenam elementos sem repetição, seguindo uma ordenação específica;
- Em um conjunto, o valor do elemento também é sua própria identificação
  - Por isto deve ser único.
- Uma vez armazenado, um elemento não pode ter seu valor alterado
  - Porém, o elemento pode ser removido e inserido com outro valor.

set

- Internamente, os elementos estão sempre ordenados de acordo com o **comparador** fornecido;
- Os conjuntos da STL são implementados como **árvores binárias de busca** ou **árvores vermelho-e-preto**
  - Vantagem da busca binária;
  - Elementos sempre ordenados.
- Conjuntos não possuem acesso direto aos elementos.

set

```
#include <iostream>
#include <set>
using namespace std;
int main ()
{
  set<int> first; //conjunto vazio de inteiros
  //insere o elemento
  first.insert(10);
  first.insert(20);
  first.insert(40);
```

set

```
//remove o elemento usando o valor
first.erase(40);
//localiza o elemento
 set<int>::iterator it = first.find(20);
 if(it == first.end())
        cout<<"não encontrou";
 else
        //remove o elemento usando um iterador
        first.erase(it);
```

set

```
//imprime todos os elementos
  for (it=first.begin(); it!=first.end(); it++)
    cout << " " << *it;
  cout << endl;
 //esvazia o conjunto
  first.clear()
  return 0;
}
```

multiset

multiset

- Um multiconjunto possui todas as características de um conjunto
  - Porém, permite elementos repetidos.

multiset

```
#include <iostream>
#include <set>
using namespace std;

int main ()
{
  multiset<int> first;  // multiset de inteiros vazio

 //insere no conjunto
  first.insert(15);
```

multiset

```
  //conta quantos elementos 15 existem no conjunto
  cout<<first.count(15)<<endl;

   //encontra a ocorrência do valor 15
  multiset<int>::iterator result = first.find(15);

//retorna um iterador para o final se não achar
  if(result == first.end())
        cout<<"não encontrou";
  return 0;
}
```

Problemas Selecionados

Problemas Selecionados

- Andy's First Dictionary

http://www.urionlinejudge.com.br/judge/en/problems/view/1215

Um Problema de Lógica

Um Problema de Lógica

- Se um tijolo pesa 1kg + meio tijolo, quanto pesa um tijolo e meio?

Desafio

Desafio

- Descobrir qual função da STL ajuda a resolver o seguinte problema

http://www.urionlinejudge.com.br/judge/en/problems/view/1259

- Pesquise no cpluplus.com

Perguntas?