GGI026 - Árvore binária

Marcelo K. Albertini

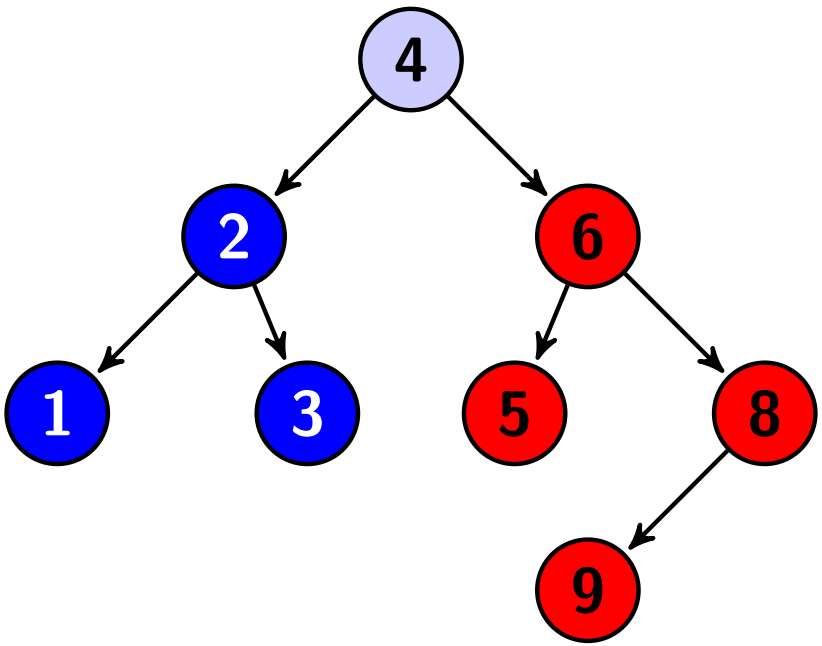
7 de Agosto de 2013

Aula de hoje

Nesta aula veremos

- Percurso em árvore

Uma árvore de busca binária



- toda sub-árvore à esquerda tem id menor que o id do nó-pai
- toda sub-árvore à direita tem id maior que o id do nó-pai

Estrutura de dados e operações de uma ABB

- criação de uma árvore
- inserção de um novo nó
- exclusão de nó
- busca por nó
- caminhos em árvores
- determinar altura da árvore

```
1  class Tree {
2      Node raiz;
3
4      public Tree() { ... } // criacao
5      public void insercao(Node n) { ... }
6      public boolean exclusao(int id) { ... }
7      public Node busca(int id) { ... }
8      public Node[] caminhoEmOrdem() { ... }
9      public Node[] caminhoEmPreOrdem() { ... }
10     public Node[] caminhoEmPosOrdem() { ... }
11     public int altura() { ... }
12 }
```

Exclusão de nó em árvore

Dois casos:

- remove raiz
- remove nó central da árvore (inclusive folha)

Exclusão: caso raiz

```
1  public void exclusao(int idr) {
2    if ("idr é a raiz") {
3      if ("só existe ramo à direita") {
4        raiz = raiz.dir;
5      } else if ("só existe ramo à esquerda") {
6        raiz = raiz.esq;
7      } else {
8        No tmp_dir = raiz.dir;
9        raiz = raiz.esq;
10       raiz.add(tmp_dir);
11     }
12 } else {
13     // remove nó central
14 }
```

Exclusão: caso central

```
1  removeCentral(No pai, int idr) {
2    if ("este nó tem id == idr") {
3      if ("só tem ramo à esquerda") {
4        pai.esq = esq;
5      } else if ("só tem ramo à direita")
6        pai.dir = dir;
7      else if ("tem os dois ramos") {
8        tmp_ramo = dir;
9        pai.esq = esq;
10       pai.add(tmp_ramo);
11     }
12 } else if ("idr esta à direita") {
13     esq.removeCentral(this, idr);
14   } else {
15     dir.removeCentral(this, idr);
16   }
17 }
```

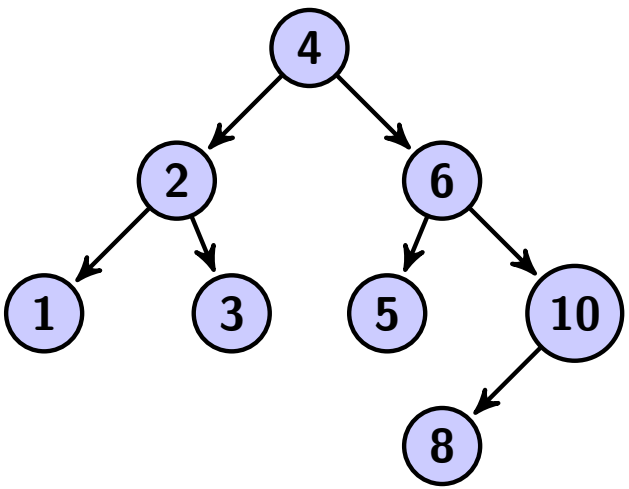
Propriedades de um nó de árvore

Um nó de árvore binária tem:

- filho-esquerda e filho-direita
- chave identificadora – `id`

```
1 class Node {
2     Node esq; // filho-esquerda
3     Node dir; // filho-direita
4     int id;   // chave identificadora
5 }
```

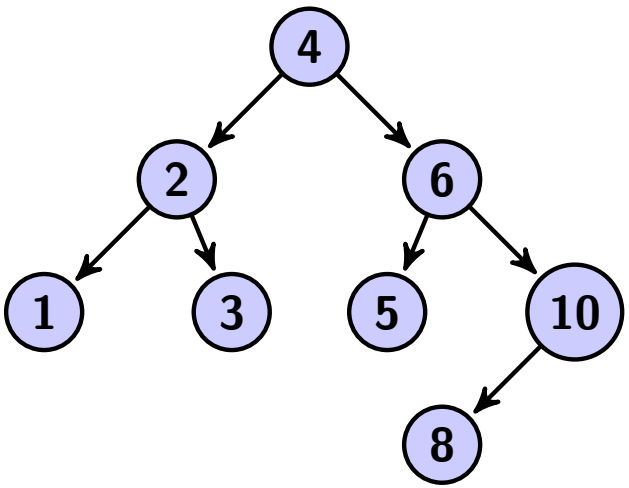
Percurso em pré-ordem




```
1 void caminhoEmPreOrdem(
      Vector<No> lista) {
2   // ``visita'' este nó
3   lista.add(this);
4
5   if (esq != null)
6   esq.caminhoEmOrdem(lista);
7   if (dir != null)
8   dir.caminhoEmOrdem(lista);
9 }
```

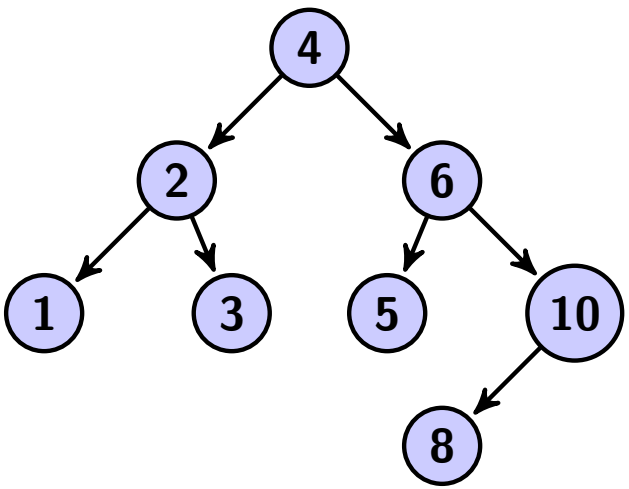
Útil para clonar árvores.

Percurso em ordem




```
1  void caminhoEmOrdem(Vector<
       No> lista) {
2
3    if (esq != null)
4    esq.caminhoEmOrdem(lista);
5
6    // ``visita'' este nó
7    lista.add(this);
8
9    if (dir != null)
10   dir.caminhoEmOrdem(lista);
11 }
```

Percurso em pós-ordem




```
1  void caminhoEmOrdem(Vector<
       No> lista) {
2
3    if (esq != null)
4    esq.caminhoEmOrdem(lista);
5
6    if (dir != null)
7    dir.caminhoEmOrdem(lista);
8
9    // ``visita'' este nó
10   lista.add(this);
11
12 }
```

Usado para operação de destruir uma árvore.