UNIPAC
Barbacena

Centro Universitário Presidente Antônio Carlos
Teoria de Grafos

Impressão e Implementação Árvores

Felipe Roncalli de Paula Carneiro
felipecarneiro@unipac.br

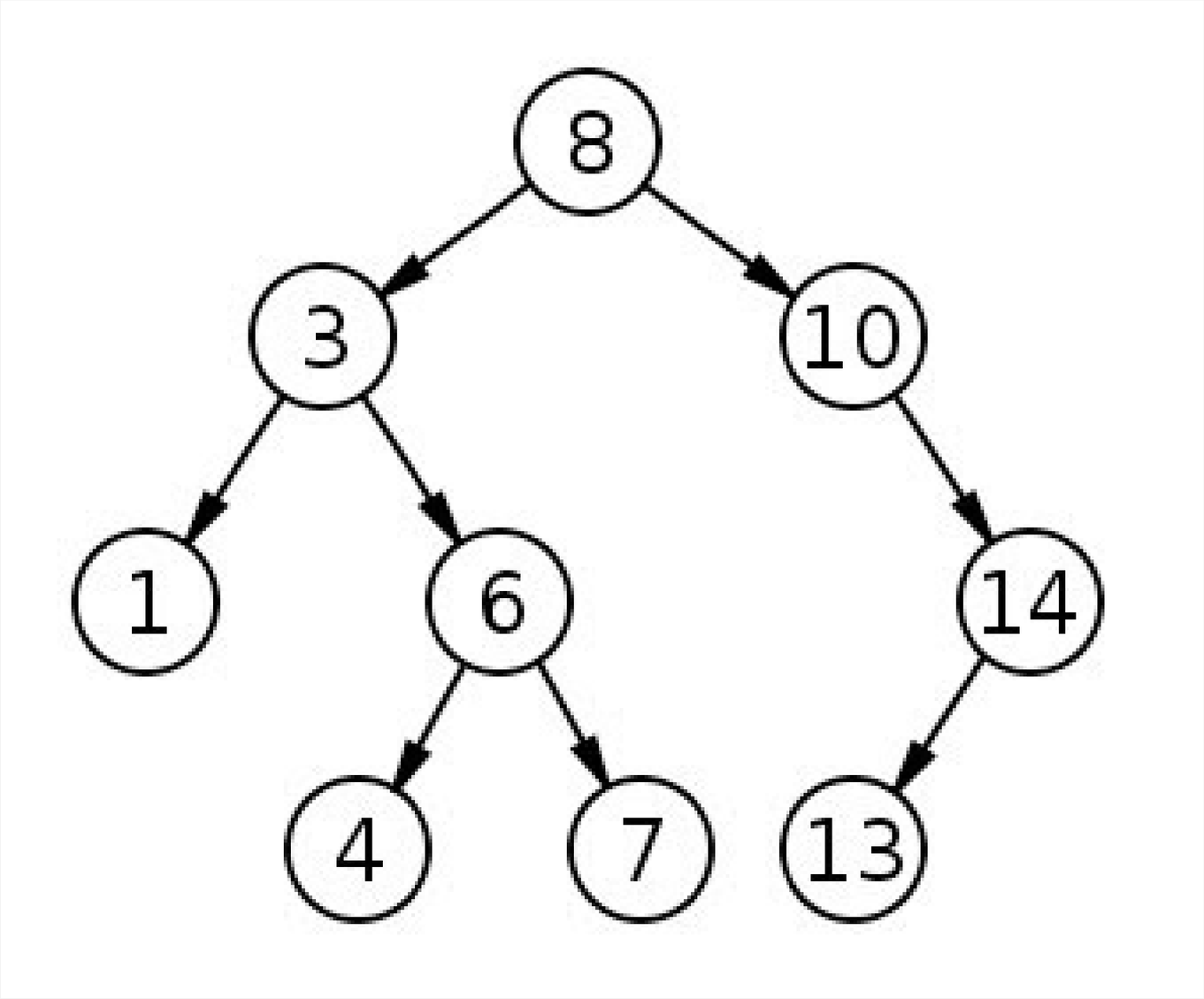
O que vamos aprender nessa aula

- Recursividade;
- Implementação de uma Árvore Binária;
- Pré-Fixado ou Pré-Ordem
- Pós-Fixado ou Pós-Ordem
- Em Ordem

Recursividade

É o mecanismo de programação no qual uma definição de função ou de outro objeto refere-se ao próprio objeto sendo definido. Assim função recursiva é uma função que é definida em termos de si mesma.

Árvore Binária



Árvore Binária

No exemplo acima tem-se uma árvore binária onde a raiz é o elemento 8, o filho da esquerda do elemento 8 é o elemento 3, o filho da direita é o elemento número 10. Nota-se que todos elementos da árvore binária possuem no máximo dois filhos, sendo o da esquerda sempre menor e o da direita sempre maior que o elemento pai.

Árvore Binária Como percorrer?

Uma árvore binária pode ser percorrida utilizando caminhamento prefixado, pós-fixado e em ordem.

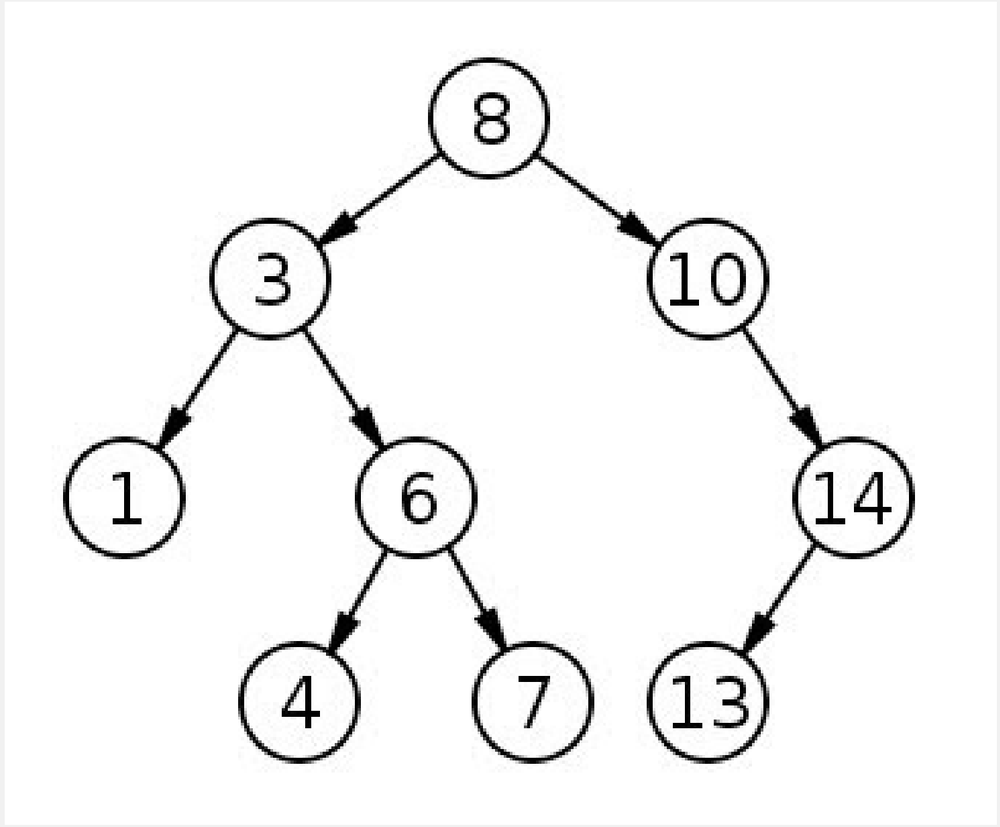
Árvore Binária - Pré-Ordem

Imprime o valor do nó (vértice) que está sendo visitado;
Recursivamente percorre-se a subárvore esquerda
Recursivamente percorre-se a subárvore direita

```java
public void prefixado(No no) {
    if(no != null){
        System.out.print(no.valor + " ");
        prefixado(no.esquerda);
        prefixado(no.direita);
    }
}
```

Exercício

1) Imprime em Pré-Ordem o seguinte Grafo abaixo.



Árvore Binária - Pré-Ordem

- imprime(8) //inicialmente executa-se a impressão do valor
- preordem(8.esquerda) //chama prefixado(no.esquerda);
- Imprime(3) //ao chamar novamente o método imprime o 3 que é o nó corrente
- preordem(3.esquerda) //chama prefixado(no.esquerda);
- Imprime(1) //ao chamar novamente o método imprime o 1 que é o nó corrente
- preordem(1.esquerda) //chama prefixado(no.esquerda);
- preordem(1.dir) //como prefixado(no.esquerda) retornou null, executa-se agora o prefixado(no.direita)
- preordem(3.dir) //como prefixado(no.direita) é null, volta-se para o elemento anterior que é o 3 (antes do 1). Como o 1 também já executou o prefixado(no.esquerda) executa-se agora o prefixado(no.direita)
- Imprime(6) //ao chamar novamente o método imprime o 6 que é o nó corrente
- preordem(6.esquerda) //chama prefixado(no.esquerda) que ainda não foi chamado para este elemento.
- Imprime(4) //ao chamar novamente o método imprime o 4 que é o nó corrente
- preordem(4.esquerda) //chama prefixado(no.esquerda) que ainda não foi chamado para este elemento.
- preordem(4.direita) //como prefixado(no.esquerda) retornou null, executa-se agora o prefixado(no.direita).
- preordem(6.direita) //como prefixado(no.direita) é null, volta-se para o elemento anterior que é o 6 (antes do 4). Como o 6 também já executou o prefixado(no.esquerda) executa-se agora o prefixado(no.direita)
- Imprime(7) //ao chamar novamente o método imprime o 7 que é o nó corrente.
- preordem(7.esquerda) //chama prefixado(no.esquerda) que ainda não foi chamado para este elemento.
- preordem(7.direita) //como prefixado(no.esquerda) retornou null, executa-se agora o prefixado(no.direita).
- preordem(8.direita) //como prefixado(no.direita) é null, volta-se para o elemento anterior que é o 8. Como o 8 também já executou o prefixado(no.esquerda) executa-se agora o prefixado(no.direita)
- Imprime(10) //ao chamar novamente o método, imprime-se o 10 que é o nó corrente
- preordem(10.esquerda) //chama prefixado(no.esquerda) que ainda não foi chamado para este elemento.
- preordem(10.direita) //como prefixado(no.esquerda) retornou null, executa-se agora o prefixado(no.direita).
- Imprime(14) //ao chamar novamente o método, imprime-se o 14 que é o nó corrente
- preordem(14.esquerda) //chama prefixado(no.esquerda) que ainda não foi chamado para este elemento.
- Imprime(13) //ao chamar novamente o método, imprime-se o 13 que é o nó corrente
- preordem(13.esquerda) //chama prefixado(no.esquerda) que ainda não foi chamado para este elemento.
- preordem(13.direita) //como prefixado(no.esquerda) retornou null, executa-se agora o prefixado(no.direita).
- preordem(14.direita) //como prefixado(no.direita) é null, volta-se para o elemento anterior que é o 14. Como o 14 também já executou o prefixado(no.esquerda) executa-se agora o prefixado(no.direita) que também é null. Agora nota-se que não há mais nada na pilha, todos os elementos e seus filhos esquerdos e direitos já foram executados, portanto está finalizado o algoritmo.

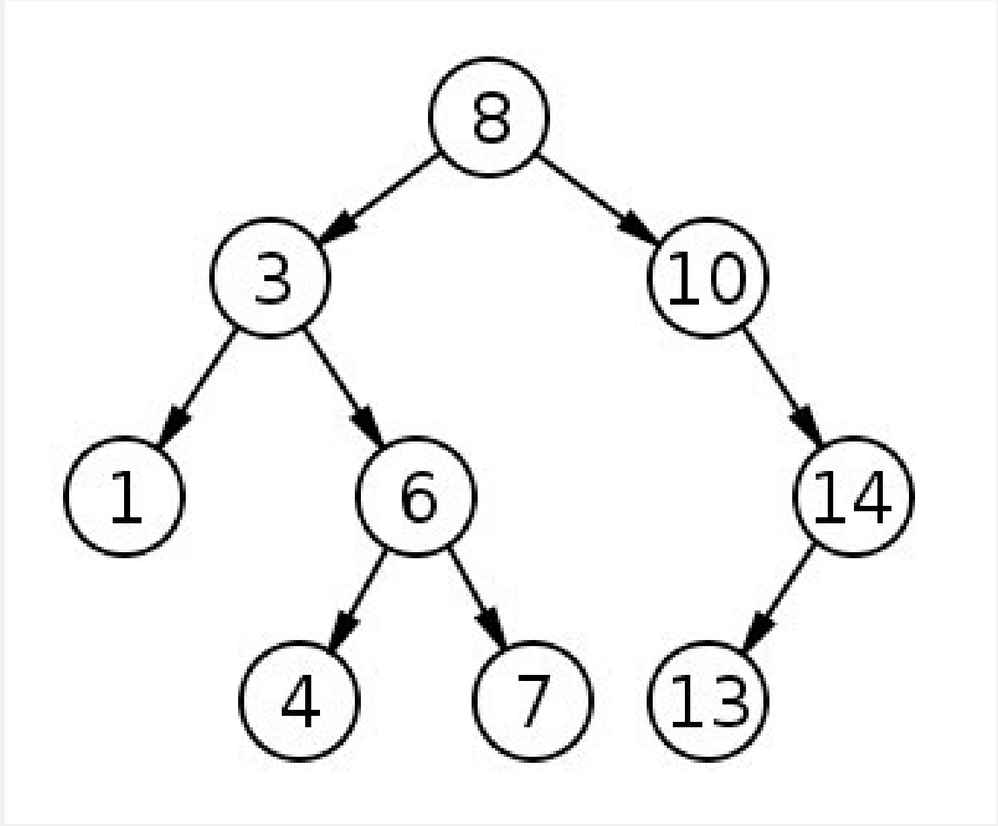
Árvore Binária - Pós-Ordem

Recursivamente percorre-se a subárvore esquerda;
Recursivamente percorre-se a subárvore direita;
Imprime o valor do nó (vértice) que está sendo visitado;

```java
public void posfixado(No no) {
    if(no != null){
      posfixado(no.esquerda);
      posfixado(no.direita);
        System.out.print(no.valor + " ");
    }
}
```

Exercício

1) Imprime em Pós-Ordem o seguinte Grafo abaixo.



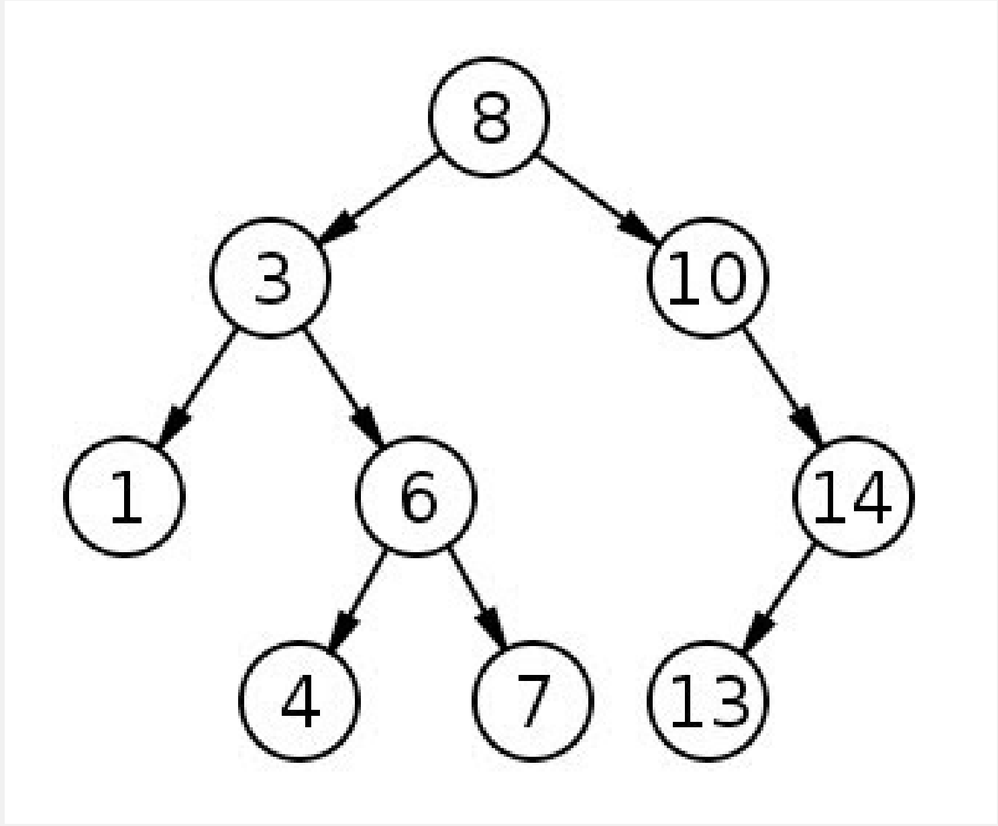
Árvore Binária - Em Ordem

Recursivamente percorre-se a subárvore esquerda;
Imprime o valor do nó (vértice) que está sendo visitado;
Recursivamente percorre-se a subárvore direita;

```java
public void emordem(No no) {
    if(no != null){
      emordem(no.esquerda);
System.out.print(no.valor + " ");
      emordem(no.direita);
    }
}
```

Exercício

1) Imprime em Em Ordem o seguinte Grafo abaixo.



Árvore Binária - Implementação - Inserir

```java
public void inserir(No node, int valor) {
  //verifica se a árvore já foi criada
   if (node != null) {
    //Verifica se o valor a ser inserido é menor que o nodo corrente da árvore, se sim vai para
    if (valor < node.valor) {
        //Se tiver elemento no nodo esquerdo continua a busca
        if (node.esquerda != null) {
            inserir(node.esquerda, valor);
        } else {
            //Se nodo esquerdo vazio insere o novo nodo aqui
            System.out.println("  Inserindo " + valor + " a esquerda de " + node.valor);
            node.esquerda = new No(valor);
        }
    //Verifica se o valor a ser inserido é maior que o nodo corrente da árvore, se sim vai para
    } else if (valor > node.valor) {
        //Se tiver elemento no nodo direito continua a busca
        if (node.direita != null) {
            inserir(node.direita, valor);
        } else {
            //Se nodo direito vazio insere o novo nodo aqui
            System.out.println("  Inserindo " + valor + " a direita de " + node.valor);
            node.direita = new No(valor);
        }
    }
  }
}
```

Dúvidas??