Aula 06-05-2024 - Árvore Binária

DEFINIÇÕES E CONCEITOS

- Inserir,Pesquisar e Remover com custo de **pode** ser Θ de lg(n)
- Altura é a maior distancia entre um nó e a raíz (O ideal é que todas as folhas tenham a mesma altura)

Árvore Binária de Pesquisa

- Cada nó é **maior** que todos seus vizinhos a **esquerda** e **menor** que todos a **direita**

Árvore Binária Completa

- Cada nó é uma folha **OR** possui exatamente dois filhos
- O número de nós internos é 2^h - 1
- O número de nós folhas é 2^h
- O número total de nós é (2^h - 1) + (2^h) = 2^(h+1) - 1

> 💡 A partir desse ponto, neste material, considera-se que todas as árvores binárias serão de pesquisa

Classe nó em java

```
class No{
    int elemento;
    No esq;
    No dir;
    No(int elemento){
        this(elemento,null,null);
    }
    No(int elemento, No esq, No dir){
        this.elemento = elemento;
        this.esq = esq;
        this.dir = dir;
    }
}
```

Classe Arvore Binaria em Java

```
class ArvoreBinaria{
    No raiz;
    ArvoreBinaria(){ raiz = null }
    void inserir(int x){}
    void inserirPai(int x){}
    boolean pesquisar(int x){}
    void caminharCentral(){}
    void caminharPre(){}
    void caminharPos(){}
    void remover(int x){}
}
```

Árvore Binária - Inserção

- Se a raiz estiver vazia, inserir o elemento nela
- Se o novo elemento for **menor** que a raiz, caminhar para a **esquerda**
- Se o novo elemento for **maior** que a raíz, caminhar para a **direita**
- Se o novo elemento for **igual** ao da raíz, não inserir um elemento repetido

Inserção em Java com retorno de referência

```
void inserir(int x) throws Exception{
    raiz = inserir(x,raiz);
}

No inserir(int x , No i)    throws Exception{
    if(i == null){
        i = new No(x);
    } else if(x< i.elemento){
        i.esq = inserir(x,i.esq);
    } else if(x>i.elemento){
        i.dir = inserir(i.dir);
    } else{
            throw new Exception("Erro!");
    }
    return i;
}
```

Inserção em Java com Passagem de Pai

```
void inserirPai(int x) throws Exception{
    if(raiz == null){
        raiz = new No(x);
    }else if(x<raiz.elemento){
        inserirPai(x, raiz.esq, raiz){
    }else if(x>raiz.elemento){
        inserirPai(x,raiz.dir,raiz);
```

```
    } else{
        throw new Exception("Erro!");
    }
    }

}

void inserirPai(int x, No i, No pai) throws Exception{
    if(i == null){
        if(x<pai.elemento){
            pai.esq = new No(x);
        } else{
            pai.dir = new No(x);
        }
    } else if(x< i.elemento) {
        inserirPai(x,i.esq,i);
    } else if(x>i.elemento){
        inserirPai(x,i.dir,i);
    } else{
        throw new Exception("Erro!");
    }
}
```

💡 Não mudamos o valor do ponteiro pai e sim dos seus filhos

Análise de Complexidade Inserção

- **Melhor caso :** teta de 1
- **Pior caso:** teta de n (Quando inserimos os elementos na ordem crescente ou decrescente
- **Caso médio:** teta de log(n) (Quando está balanceado)