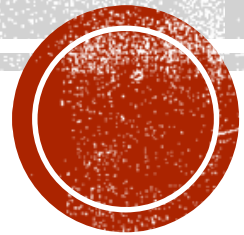
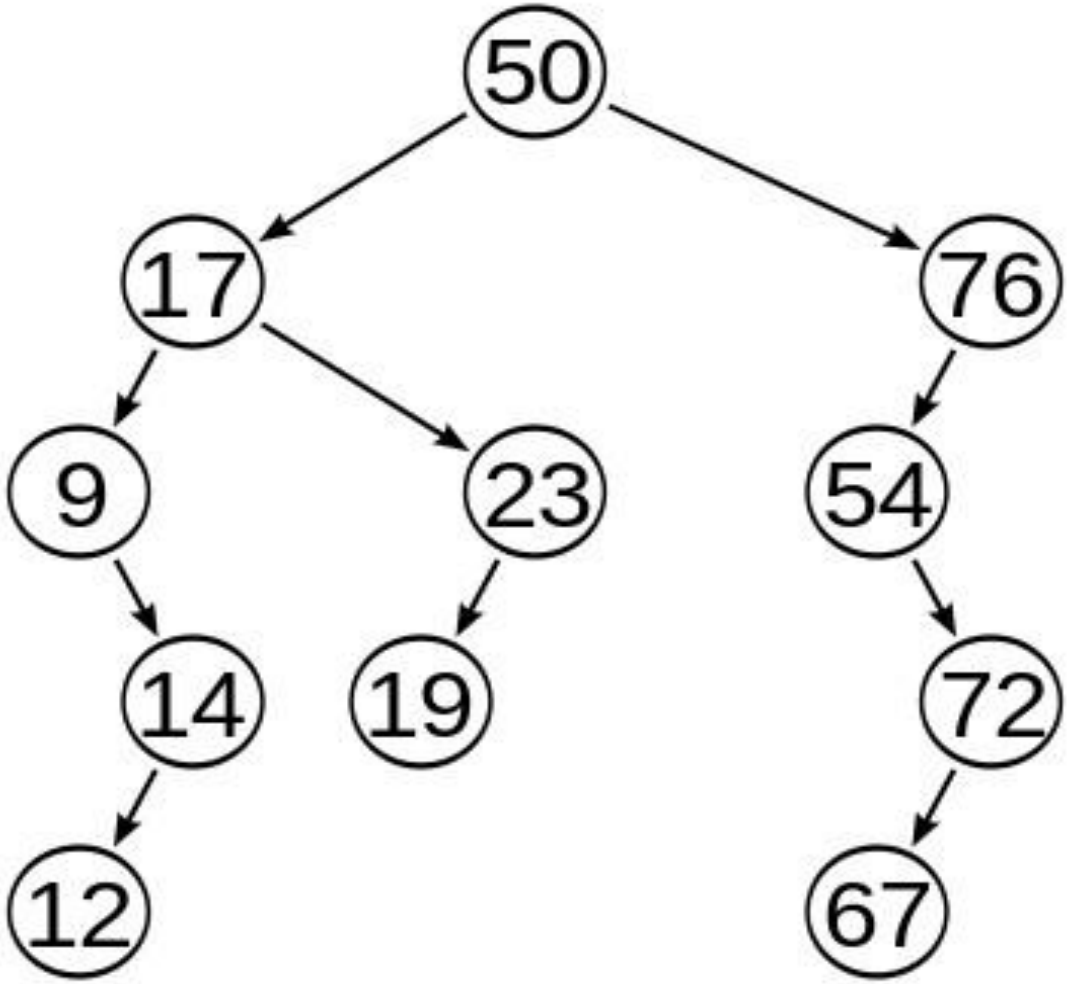
ALGORITMOS AVANÇADOS

Comparação de Complexidade

Aluno: Elbert Ribeiro





Árvore Binária

Tendo a árvore binária ao lado, termos a seguinte tarefa; desenvolver um algoritmo que possa fazer ordenação em: Pré-ordem, In-ordem, Pós-ordem e Buca em ABB, além de verificar quantas instruções devem ser executadas em cada um dos algoritmos para encontrar a chave 67.

Para o projeto em questão, foi utilizado o Python para desenvolver o algoritmo.

EM PRÉ-ORDEM

- A busca em pré-ordem consiste em ao visitar cada nó, primeiro visitar o número nele contido e depois a sub-árvore da esquerda e por último a da direita, recursivamente. No caso de haver mais do que dois filhos, eles devem ser visitados da esquerda para a direita.

- Para essa algoritimo em pré-ordem encontar a folha “67”, ele executa 11 instruções.

```
def preOrder(self, atual):
    if atual != None:
        print(atual.item,end=" ")
        self.preOrder(atual.esq)
        self.preOrder(atual.dir)
```

IN-ORDEM

- Busca em-ordem consiste em, ao visitar cada nó, primeiro visitar o filho esquerdo, depois o próprio nó e depois o filho direito.

- Para essa algoritimo em in-ordem encontar a folha “67”, ele executa 9 instruções.

```
def inOrder(self, atual):
    if atual != None:
        self.inOrder(atual.esq)
        print(atual.item,end=" ")
        self.inOrder(atual.dir)
```

- Busca em pós-ordem consiste em, visitar o nó só após os filhos serem visitados.

- Para essa algoritimo em pós-ordem encontar a folha “67”, ele executa 7 instruções.

```
def posOrder(self, atual):
    if atual != None:
        self.posOrder(atual.esq)
        self.posOrder(atual.dir)
        print(atual.item,end=" ")
```

PÓS-ORDEM

ÁRVORE BINÁRIA DE BUSCA

```
def buscar(self, chave):
    if self.root == None:
        return None
    atual = self.root
    while atual.item != chave:
        if chave < atual.item:
            atual = atual.esq
        else:
            atual = atual.dir
        if atual == None:
            return None
    return atual
```

- A busca começa examinando o nó raiz. Se a árvore está vazia, o valor procurado não pode existir na árvore. Caso contrário, se o valor é igual a raiz, a busca foi bem sucedida. Se o valor é menor do que a raiz, a busca segue pela sub-árvore esquerda. Similarmente, se o valor é maior do que a raiz, a busca segue pela sub-árvore direita. Esse processo é repetido até o valor ser encontrado ou a sub-árvore ser nula (vazia). Se o valor não for encontrado até a busca chegar na sub-árvore nula, então o valor não deve estar presente na árvore.

- Para essa algoritimo encontar a folha "67", ele executa 5 instrições.

A analise a execução das instruções de cada algoritmo e diga (de acordo com o número de instruções executadas) qual deles seria o de menor complexidade e por quê?

LEVANDO EM CONSIDERAÇÃO O CONCEITO DE COMPLEXIDADE CONSTANTE, ONDE QUANTO MENOR O NUMERO DE PASSOS, MENOR A COMPLEXIDADE. CHEGAMOS A CONCLUSÃO DE QUE A MELHOR ESCOLHA SERIA O ALGORITIMO DE BUSCA-BINÁRIA, POR TER APENAS 5 INSTRUÇÕES ATÉ A FOLHA “67” SENDO ASSIM, DE MENOR COMPLEXIDADE.