Algoritmos

```
classe no()
    funcao _init_() then
        valor = 0
        esq = Nulo
        dir = Nulo
        prox = Nulo
        ant = Nulo
```

Resumo: Aqui é criado uma classe do tipo nó. Dentro dessa classe tem um método do tipo construtor ou init com os respectivos atributos dessa classe.

```
classe Pilha()
    funcao __init__() then
        topo = Nulo
    funcao empilhar(chave) then
        novo = no()
        novo.valor = chave
        novo.prox = topo
        topo = novo
        return topo

    funcao desempilhar() then
        if topo != Nulo then
            valor_desempilhado = topo.valor
            aux = topo
            topo = topo.prox
            dell aux
        else
            print("Pilha vazia")
```

return valor_desempilhado

Resumo: Aqui é criado uma classe do tipo Pilha. Dentro dessa classe tem um método do tipo construtor ou init com os respectivos atributos dessa classe, no caso o topo da pilha. Além disso, tem um método em empilhar, que cria um novo nó na pilha encadeada e atribui ao topo; e o método desempilhar que deleta um nó do topo e retorna o valor que ali estava.

```
funcao Busca(pont, chave, pai=Nulo) then
    if pont.valor == chave then
        f = 1
    else:
        if chave<pont.valor then
            if pont.esq == Nulo:
                f = 2
            else:
                pai = pont
                pont = pont.esq
                pont,f, pai = Busca(pont, chave,pai)
        else:
            if pont.dir == Nulo then
                f = 3
            else:
                pai = pont
                pont = pont.dir
                pont,f, pai = Busca(pont,chave,pai)
    return pont, f, pai
```

Resumo: Aqui é uma função de busca para árvore binária de busca, que recebe como parâmetros pont, chave e o pai. Quando encontrar o elemento na árvore, retornará f =1. O pai de pont e pont também são retornados, pois irão ser úteis nos próximos algoritmos.

```
classe ArvoreBinariaBusca()
    funcao _init_()
        raiz = Nulo
    funcao Inclusao( chave) then
        if raiz == Nulo then
            novo = no()
            novo.valor = chave
            novo.esq = Nulo
            novo.dir = Nulo
            raiz = novo
        else
            pont = raiz
            pont, f,pai = Busca(pont,chave)
            if f==1 then
                print("Elemento existe")
            else
                novo = no()
                novo.valor = chave
                novo.esq = Nulo
                novo.dir = Nulo
                if f==2 then
                    pont.esq = novo
                else
                    pont.dir = novo
        return raiz
    funcao Exclusao( chave) then
        pont, f, pai = Busca(raiz,chave)
        if f==1 then
            if pont.esq == Nulo then
                if pont == raiz then
```

```
            raiz = raiz.dir
        else
            if pont == pai.esq then
                pai.esq = pont.dir
            else
                pai.dir = pont.dir
    else
        if pont.dir == Nulo then
            if pont == raiz then
                raiz = raiz.esq
            else then
                if pont == pai.esq then
                    pai.esq = pont.esq
                else
                    pai.dir = pont.esq
        else
            pont2 = pont.dir
            pai2 = pont
            while pont2.esq!=Nulo then
                pai2 = pont2
                pont2 = pont2.esq
            if pai2 != pont then
                pai2.esq = pont2.dir
                pont2.dir = pont.dir

            pont2.esq = pont.esq
            if pont == raiz then
                raiz = pont2
            else
                if pai.esq == pont then
```

```
                pai.esq = pont2
            else
                pai.dir = pont2
    dell pont
funcao Pre_Ordem() then
    pont = raiz
    if pont.valor == 0 then
        print("Árvore vazia")
    else
        pilha = Pilha()
        pilha.empilhar(pont)
        while pilha.topo != Nulo then
            pont = pilha.desempilhar()
            print(pont.valor)
            if pont.dir != Nulo then
                pilha.empilhar(pont.dir)
            if pont.esq != Nulo then
                pilha.empilhar(pont.esq)


funcao imprimir_arvore(no, nivel=0, lado=Nulo)
    if no != Nulo then
        imprimir_arvore(no.esq, nivel + 1, 'E')
        print(' ' * 7 * nivel + f'{no.valor} ({lado})')
        imprimir_arvore(no.dir, nivel + 1, 'D')
```

Resumo: Aqui é criado uma classe do tipo ArvoreBinariaBusca. Dentro dessa classe tem um método do tipo construtor ou init com os respectivos atributos dessa classe, no caso a raiz da arvore.

Além disso, tem um método em inclusão, que verifica se a raiz é igual a nulo. Se caso não for, ao invés de atribuir o novo elemento à raiz, verifica e ele já existe. Caso não exista, inclui o novo elemento na árvore.

Outro método que há dentro da classe ArvoreBinariaBusca é o de exclusão. Esse método recebe uma chave como parâmetro, logo depois faz uma busca para saber a posição desse elemento. Caso o

encontre, fará o processo de exclusão para os casos de a esquerda ou direita do nó a ser excluído ser nulo. Caso não forem, tratará do caso onde ambos forem diferentes de nulo.

Para além disso, existe também o método de pré ordem. Esse método é recursivo. Primeiro verifica se árvore está vazia, e se caso não tiver iniciará o caminhamento. O método inicia colocando a raiz na pilha, e, enquanto a pilha não estiver vazia, será desempilhada. Dentro desse loop, caso a direita ou esquerda, nessa ordem, do elemento for diferente de nulo, o respectivo elemento da esquerda ou direita será empilhado, possibilitando assim o caminhamento de pré ordem ao desempilhar a pilha.

Por fim, há o algoritmo de imprimir árvore, que é recursivo. A função dele é percorrer cada nó da árvore, de forma recursiva, verificando se é diferente de nulo. Cada nó é impresso em sua posição com o seu valor e se é filho direito (D) ou esquerdo (E) de seu pai.