
UNIVERSIDADE
FEDERAL DO CEARÁ

**Discente:** Antonio Emerson Gomes Camelo e Pablo Kauan Martins Timbó
**Matrícula:** 555535 e 556012

**Docente:** Lisieux Marie
**Disciplina:** Estrutura de Dados Avançada

# Árvore B

**Cratéus - CE**
**2024**

# Sumário

# 1. Introdução

Inicialmente, visando a proposta de escolher a linguagem que menos tornaria complexa a criação da árvore, decidimos que iríamos utilizar o python.

Sendo definida a linguagem, decidimos que estruturarmos cada parte do algoritmo separa, gerando assim um arquivo “No.py” e “ArvoreB.py”, além controlador “Main.py” que executara o passo a passo solicitado no trabalho, usando-se das ferramentas criadas nos outros dois arquivos.

O trabalho consiste em receber um arquivo “entrada.txt” com números um abaixo do outro, realizar a leitura e inserção na árvore mantendo suas propriedades e após gerar um arquivo “saida.txt”, listando a Árvore, ambos no diretório raiz, nas seções abaixo explicaremos o funcionamento do algoritmo criado.

# 2. Arquivo Main.py

O arquivo “Main.py” como mencionado ficou responsável por fazer a gerência e utilização dos objetos e atividades que estes tais, devem realizar para que o intuito do trabalho seja alcançado.

### 2.1.1. Funcionalidade

Sua funcionalidade é utilizar os outros arquivos para realizar a demanda do trabalho. Inicialmente ele cria uma variável elementos (para receber os elementos do arquivo) e uma variável árvore que irá servir de local para nossa árvore. Após isso com ajuda da função with e readlines do Python, ele executa a leitura do arquivo e transforma a variável elementos em um array de strings para cada linha do arquivo lido, após isso ele executa um for para percorrer os elementos e para a primeira linha ou elemento ele define a criação da árvore usando o valor da leitura como ordem e os demais, inseri os mesmos como nos usando uma função designada na árvore. E por final realiza o uso da função escrever presente na árvore para transcrever a mesma em um arquivo “saida.txt”.

# 3. Arquivo No.py

## 3.1. Funcionalidade

Tem por finalidade manter e estabelecer a estrutura a ser utilizada para a construção dos Nós/Paginas da árvore junto de suas respectivas funções associadas.

## 3.2. Funções

### 3.2.1. __init__

O método __init__ é um método em Python que é chamado quando um novo objeto da classe é criado. Ele serve para inicializar os atributos do objeto, que estão definidos. Os atributos que definimos para a Classe No foram os seguintes: um inteiro para guardar a ordem da árvore B e um parâmetro folha booleano que indica se o nó é uma folha ou não. Além de uma lista chamada chaves que armazenam as chaves do nó e uma lista chamada filhos para armazenar os filhos do nó.

### 3.2.2. inserir_incompleto

Ela tem o papel de inserir os elementos em nós não cheios, o processo de inserção inicia-se na raiz da árvore. A função checa se o nó não é uma folha, caso não seja: A chave a ser inserida é comparada com as chaves presentes no nó atual para determinar qual filho deve ser seguido.

Essa busca continua recursivamente até que se alcance uma folha na qual ele possa ser inserido. Se a folha ainda tiver espaço disponível para a nova chave, a inserção é realizada diretamente, adicionando-se um espaço vazio no final e passando os nos maiores para frente até que seja encontrado um no menor que a chave a ser inserida, que após vai ser inserida na frente deste no menor, mantendo a ordem das chaves.

Porém, se a folha estiver cheia, é necessário realizar uma divisão. Nesse caso, será realizado o mesmo processo porém com utilização da função dividir_pagina, para eleger um nó a subir na hierarquia e surgir duas novas folhas para manter o balanceamento da árvore.

### 3.2.3. dividi_pagina

Sobre a função dividi_pagina, usa-se quando um nó da árvore B atinge sua capacidade máxima de chaves, essa função é acionada para dividir o nó em dois. A chave do meio desse nó é promovida para o nível superior da árvore, enquanto as chaves menores permanecem no nó original e as

maiores são movidas para o novo nó, mais especificamente. A linha “z.chaves=y.chaves[ordem:(2*ordem-1)]” copia a metade direita das chaves do nó y para o novo nó z. Essas chaves serão as maiores chaves do novo nó. A linha y.chaves=y.chaves[0:(ordem-1)] atualiza o nó original y mantendo apenas a metade esquerda das chaves, excluindo a chave mediana que já foi promovida para o nó pai, realizando esse processo de forma recursiva até atingir o balanceamento da árvore.

# 4. Arquivo ArvoreB.py

## 4.1. Funcionalidade

Tem por finalidade manter e estabelecer a estrutura a ser utilizada para a construção e edição da Árvore B e suas respectivas funções associadas.

## 4.2. Funções

### 4.2.1. __init__

O método __init__ é um método em Python que é chamado quando um novo objeto da classe é criado. Ele serve para inicializar os atributos do objeto, que estão definidos. Os atributos que definimos para a Classe ArvoreB foram os seguintes: um objeto No chamado raiz para guardar a raiz da árvore B e um inteiro para guardar a ordem da árvore.

### 4.2.2. inserir

A função, como próprio nome induz, serve para inserir uma chave na árvore, seguindo os devidos passos, primeiramente ela instancia uma variável raiz para receber de referência a raiz de nossa árvore.

Logo após ele checa se a raiz está cheia, se sim cria um novo nó raiz e inseri a chave na raiz cheia, faz o processo de quebra subindo a mediana para a nova raiz e após o balanceamento, ela atribui a variável raiz da árvore a nova variável raiz. E caso não esteja cheia ela apenas induz a realizar o método inserir_incompleto.

### 4.2.3. procurar

A função tem como objetivo localizar uma chave específica dentro de uma árvore B. Ao ser instanciado recebe como parâmetro a chave que se deseja encontrar na árvore, depois chama  a função procurar_auxiliar, pois a

primeira função procurar é chamada apenas para conseguirmos instanciar uma função recursiva a partir do nó raiz.

A função procurar_auxiliar é chamada recursivamente para realizar a busca propriamente dita, a função procurar_auxiliar recebe como parâmetros a raiz da árvore e a chave a ser procurada. A função irá começar a checar nó por nó, enquanto o nó for menor que nossa chave, até ser encontrado e retorna ou até chegar na última folha possível que ele podia estar, e não sendo encontrado retorna none.

### 4.2.4. quantidade_de_niveis

Essa função trata-se de um aproveitamento, ela serve para descobrir quantos níveis nossa árvore tem, ela utiliza-se da função ultimo_nivel que retorna qual o nível mais profundo da árvore, e a função quantidade_de_niveis irá acrescer de um esse retorno para contabilizar o nivel zero, assim retornando o número exato de niveis da arvore, incluindo a raiz.

### 4.2.5. ultimo_nivel

O objetivo dela é  determinar a altura de uma árvore com raiz. Primeiro, o método ultimo_nivel recebe como parâmetro o nó a partir do qual se deseja calcular a altura, depois, a condição verifica se o nó atual é uma folha. Se for, significa que não há níveis abaixo dele, o nível do nó é 0 e a função retorna 0, indicando que a altura da árvore com raiz nesse nó é zero. Se o nó não for uma folha, a função  faz chamadas recursivas para cada filho do nó atual, a função ultimo_nivel é chamada recursivamente, calculando a altura da subárvore com raiz nesse filho.

Usando da função MAX para encontrar o maior valor retornado pelas chamadas recursivas. Esse maior valor representa a maior altura entre todas as subárvores dos filhos do nó atual, sendo assim necessita adicionar um ao maior valor encontrado para contabilizar o nível do nó atual, retornando assim o número que representa o último nível.

### 4.2.6. escrever

A função tem como objetivo transcrever a árvore em um arquivo, primeiramente ela cria um novo arquivo chamado "saida.txt" e o deixa limpo sem nenhum caractere, caso já exista esse arquivo, ela apenas reseta o arquivo, o deixando limpo novamente.

Com o arquivo já aberto e limpo, ela checa se a raiz da árvore está vazia, se sim, significa que a árvore está vazia e uma mensagem informando este fato é escrita no arquivo. Caso a árvore não seja vazia, a função transcreve a ordem da árvore (máximo número de chaves por nó) e a altura da árvore (número de níveis) são escritas na primeira linha do arquivo, sendo a altura calculada usando a função quantidade_de_niveis.

Para a transcrição da árvore é usada uma lista, para armazenar os nós da árvore, inicialmente, a raiz da árvore é adicionada à fila com o nível 0 e logo em seguida a função estabelece uma variável nivel_atual que é utilizada para acompanhar o nível atual sendo processado. A lista nos_do_nivel, mencionada anteriormente armazena as chaves dos nós do nível atual, enquanto a fila não se der pro vazia, ocorre-se o seguinte: O primeiro nó da fila é removido e suas informações são extraídas para a lista, caso o nível do nó atual for maior que o nível atual, as chaves dos nós do nível anterior são escritas no arquivo e a lista nos_do_nivel é reinicializada para o novo nível.

As chaves do nó atual são adicionadas à lista nos_do_nivel. Os filhos do nó atual são adicionados à fila com o nível incrementado. Após o término do laço principal, verifica-se se a lista nos_do_nivel contém algum nó. Se sim, as chaves desses nós são escritas no arquivo, representando o último nível da árvore.

## 5. Conclusão

Este foi o seguinte trabalho, realizado por nós, diante do desafio impostos tivemos alguns empecilhos/desafios de implementação que acabaram que sendo sanados com o desenvolvimento do trabalho e muito tempo dispostos a pensar em soluções lógicas paras os problemas, acreditamos que o que buscamos desde o início tenha sido alcançado, tanto o objetivo do trabalho, como o intuito de fixarmos uma estrutura bastante complexa em nosso aprendizado.