Aluno: Cyril Zakhia.

Avaliação 02 – Arvores Binarias

**Descrição do código:**

**Arvore Binaria:**

1. Push (inserção):

A computer screen shot of a black screen

Description automatically generated

O método push aqui vai inserir os elementos dentro da arvore . Ele vai passar e ver onde o elemento desejado deveria ser inserido de acordo com as propriedades da BST (se o elemento e maior ou igual ao nó ,colocar na direita se não na esquerda) . Eu usei a recursão aqui para chegar até as nos folhas para pode implementar o novo elemento.

2-Delete\_node (remoção):

A screen shot of a computer program

Description automatically generatedA computer screen with text and images

Description automatically generated

Nesse método , eu estou procurando o elemento desejado para fazer remoção usando também a recursão. Quando achar o elemento ele vai ver se ele e repetido ou não , se sim ele vai continuar a buscar o elemento com frequência de 1 , se não ele vai verificar se ele não tem filhos. Se sim ele só vai retornar null. Assim o no parente dele não sera mais ligado com o node que queremos remover. Se ele tem um filho , ele vai retornar o filho e se ele tem dois filhos (esquerda e direita ) vai buscar o elemento menores dos maiores ( então na subarvore direita acha o menor elemento – ver foto em baixo) para trocar com o elemento excluído.

A computer screen shot of a code

Description automatically generated

1. Search(Buscar):

A screen shot of a computer program

Description automatically generated

Esse método vai permitir achar o node com o valor “elemento” na arvore.

E retornar esse node.

**Arvore Binaria - AVL:**

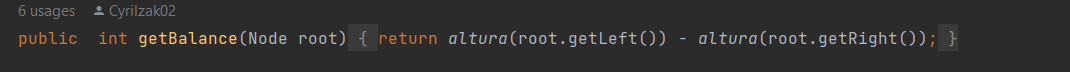
1. Insert(inserção):

E a mesma coisa para inserção na arvore binaria normal mas tem um trecho para verificar o balanceamento de cada no usando esses métodos:

A computer screen shot of a blue screen

Description automatically generated

-Altura que vai calcular a altura do no desejado.



-getBalance que vai calcular o balanceamento de um no

A screen shot of a computer code

Description automatically generated

- left\_rotate que vai fazer a rotação para esquerda quando o fator de balanceamento de um no ficar >1 para poder balancear de novo a arvore

A screen shot of a computer program

Description automatically generated

- right\_rotate que vai fazer a rotação para direita quando o fator de balanceamento de um no ficar < -1.

A screen shot of a computer program

Description automatically generated

-trecho adicional para inserção numa arvore avl :

Esse trecho vai verificar se o no esta desbalanceado e precisa de qual operação para balancear ele de novo.

1. Se o o bf <-1 e no.getData >= key(elemento que foi inserido) :

Fazer left\_rotate

A diagram of the right and left rotation

Description automatically generated

2-Se o bf <-1 e no.getData < key:

.Fazer rightrotate(no.getRight)

.Fazer leftrotate(no)

A diagram of a left and right case

Description automatically generated with medium confidence

3- Se o bf>1 e key < no.getLeft

Fazer right\_rotate

A diagram of the left and right rotation

Description automatically generated

4- Se o bf > 1 e key >= no.getLeft

Fazer left\_rotate(no.getLeft)

Fazer right\_rotate(no)

A diagram of a triangle with text

Description automatically generated

2- Delete (remover):

A primeira parte da remoção e igual do que a arvore binaria normal mais o segundo trecho para verificar o balanceamento e e fazer a rotações necessária para poder manter o balanceamento da arvore esta aqui:

A screenshot of a computer program

Description automatically generated

1. Se o o bf <-1 e no.getRight.getBalance <=0 :

Fazer left\_rotate

A diagram of the right and left rotation

Description automatically generated

2-Se o bf <-1 e no.getRight.getBalance > 0:

.Fazer rightrotate(no.getRight)

.Fazer leftrotate(no)

A diagram of a left and right case

Description automatically generated with medium confidence

3- Se o bf>1 e no.getLeft.getBalance >= 0

Fazer right\_rotate

A diagram of the left and right rotation

Description automatically generated

4- Se o bf>1 e no.getLeft.getBalance < 0

Fazer left\_rotate(no.getLeft)

Fazer right\_rotate(no)

A diagram of a triangle with text

Description automatically generated

3- Search :

A computer screen shot of a blue screen

Description automatically generated

Mesmo método do que a arvore binaria normal.

Comparacao entre o BST e AVL .

Usei o JConsole para ter as métricas do cpu , heap memory e classes. Para medir o tempo de execução de cada operação usei :

A screen shot of a computer code

Description automatically generated

-Insercao 100 elementos

Avl: A screenshot of a computer

Description automatically generated Tempo execução : 4 ms

BST: A screenshot of a computer

Description automatically generated Tempo execução : 3ms

Insercao 500 elementos:

AVL:

A screenshot of a computer

Description automatically generated



BST:

A screenshot of a computer screen

Description automatically generated



Inserção 1000 elementos :

Avl: A screenshot of a computer

Description automatically generated

Tempo execução : 12ms

BST A screenshot of a computer screen

Description automatically generated

Tempo de execução : 6ms

Insercao 10000 elementos:

AVL: A screenshot of a computer

Description automatically generated



BST: A screenshot of a computer screen

Description automatically generated



Insercao 20000 elementos:

AVL:A screenshot of a computer

Description automatically generated

Tempo execução : 3327 ms

BST: A screenshot of a computer

Description automatically generated

Tempo execução : 271ms

Remoção numa arvore com 100 elementos : (Aqui vou escolher o elemento folha para poder comparar a velocidade)

AVL

A screenshot of a computer

Description automatically generated

Tempo execução : 39000 ns

BST

A screenshot of a computer screen

Description automatically generated

Tempo execucacao : 43300 ns

Remoção numa arvore com 500 elementos:

AVL:

A screenshot of a computer screen

Description automatically generated

Valor a remover : 496



BST:

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated



Remoção numa arvore com 1000 elementos :

AVL:

A screenshot of a computer screen

Description automatically generated

Tempo de execução: 76000 ns

BST

A screenshot of a computer

Description automatically generated Tempo execução : 43900 ns

Remoção numa arvore com 20000 elementos :

AVL :

A screenshot of a computer

Description automatically generated

Tempo de execução: 378200 ns

BST: A screenshot of a computer screen

Description automatically generated Tempo execucao : 64700 ns

-Search com arvore com 100 elementos

Avl:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

BST:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

-Search numa arvore com 500 elementos:

AVL:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

BST:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

NB: aqui eu acho que a busca foi mais rapida com a arvore bst porque a profundidade do elemento na bst era menor do que a profundidade na avl

Remocao numa arvore com 10000 elementos:

Avl:

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated



BST:

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated



-Search com arvore com 1000 elementos

Avl:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

BST:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

Search com 10000 elementos:

AVL: A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

BST:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

-Search com arvore com 20000 elementos:

AVL :

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

BST:

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

**Resultado :**

Fazendo os testes de performance e comparando os dois, tirei a conclusão que as operações de remoção e inserção numa arvore avl leva mais tempo por causa das operações de balanceamento que não são presentes num BST normal , mais o BST pega mais espaço (heap memory). Mas o BST e muito mais lento na operação de busca por causa do desbalanceamento e os elementos podem ser muito mais profundos do que a arvore avl o que resulta num tempo maior para achar o elemento desejado.

Enfim eu acho que quando tiver muitos elementos (>10000) elementos e melhor usar a estrutura AVL pois ela deixa os arvores balanceados e a operacao de busca fica mais agil. Mas quando tiver menos elementos (<=500) e melhor usar o BST pois a estrutura e mais simples a implementar.