**TRABALHOS PARA DISCIPLINA DE ESTRUTURAS DE DADOS**

**(UNIDADES II: Árvores AVL, Árvores B e Grafos)**

**JOSÉ DE SOUSA MAGALHÃES**

**1. Utilizando o exemplo ArvoresAVL.cpp e crie um exemplo com as seguintes opções:**

1. **Inserir elemento,**
2. **Mostrar arvore em pre-ordem**
3. **Mostrar arvore em pos-ordem**
4. **Mostrar arvore em em-ordem**
5. **Altura da arvore**
6. **Buscar nó**
7. **Excluir nó**
8. **Sair**

Relatório de implementação: O código base utilizado foi o código ArvoreAVL.cpp que sofreu alguns ajustes para atender o solicitado na questão.

* 1. **–** Inclusão da função void exibir\_Preordem( arvore \*aux) para mostrar a arvore em pre-ordem, conforme indicação do MENU de opções da aplicação, utilizando uma cópia da função void exibir\_ordenado( arvore \*aux) com mudança de ordem da seguinte linha: printf("%d(%d) ", aux->info, aux->fb);
  2. – Inclusão da função void exibir\_Posordem (arvore \*aux) para mostrar a arvore em pos-ordem, conforme indicação do MENU de opções da aplicação, utilizando uma cópia da função void exibir\_ordenado( arvore \*aux) com mudança de ordem da seguinte linha: printf("%d(%d) ", aux->info, aux->fb);
  3. – Criação da variável altura\_arvore para receber a função int altura (arvore \*aux), possibilitando a exibição da altura da arvore na opção 5.
  4. **–** Criação deum MENU de opções utilizando SWITCH/CASE dentro de DO/WHILE com nove opções. Foi acrescentado a opção Limpar Arvore e inclusão de mensagem “Opção Invalida!” retornada quando o usuário digita uma opção diferente das apresentadas no MENU. Segue cópia do código com as alterações realizadas.

Código em Anexo.

**2. Monte uma árvore AVL no link: https://www.cs.usfca.edu/~galles/visualization/AVLtree.html e explique como a inserção acontece para cada elemento inserido.**

Resposta: Os elementos foram inseridos na arvore na seguinte sequência:

10,30,50,9,7,43,54,60,63.

Passo 1: Inserção do número 10 – A aplicação coloca o valor 10 como raiz da arvore e repassa o valor 1 para a altura da arvore;

Passo 2: inserção do número 30 – A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é maior, o valor 30 é inserido a direita da raiz. A altura da arvore e do nó 30 é recalculada, atribuindo os valores 2 e 1 respectivamente.

Passo 3: Inserção do número 50: A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é maior, o valor 50 é inserido a direita da raiz. Após a inserção do elemento 50, a arvore fica desbalanceada pesando para a direita. Em seguida a aplicação faz uma rotação simples a esquerda, transformando o elemento 30 em raiz. A altura da arvore e dos nós é recalculada.

Passo 4: Inserção do número 9: A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é menor, o valor 9 é inserido a esquerda da raiz e a esquerda do elemento 10. A altura da arvore e dos nós é recalculada.

Passo 5: Inserção do número 7: A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é menor, o valor 7 é inserido a esquerda da raiz e a esquerda do elemento 9.Após a inserção do elemento 7, a arvore fica desbalanceada pesando para a esquerda. Em seguida a aplicação faz uma rotação simples a direita, transformando o elemento 9 em raiz da sub arvore esquerda.

Passo 6: Inserção do número 43: A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é maior, o valor 43 é inserido a direita da raiz e a esquerda do elemento 50. A altura da arvore e dos nós é recalculada.

Passo 7: Inserção do número 54: A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é maior, o valor 54 é inserido a direita da raiz e a direita do elemento 50. A altura da arvore e dos nós é recalculada.

Passo 8: Inserção do número 60: A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é maior, o valor 60 é inserido a direita da raiz e a direita do elemento 54. A altura da arvore e dos nós é recalculada.

Passo 9: Inserção do número 63: A aplicação verifica se o valor é maior ou menor que a raiz. Constatando que o valor é maior, o valor 63 é inserido a direita da raiz e a direita do elemento 60. Após a inserção do elemento 63, a arvore fica desbalanceada pesando para a direita. Em seguida a aplicação faz uma rotação simples a esquerda, trocando as posições dos elementos 60 e 54.O elemento 54 passa a ser um nó folha. A altura da arvore e dos nós é recalculada.

**3. Usando a árvore do exercício anterior, faça uma busca de um elemento que pertence a árvore e depois faça a busca de um elemento que não pertence a árvore, explique como acontece.**

Resposta:

**Situação 1: Busca um de um elemento que pertence a arvore. O elemento 10.**

Passo 1: Compara o valor do elemento 10 com a raiz 30.O elemento é menor que a raiz, então segue para a esquerda;

Passo 2: Compara o valor do elemento 10 com a raiz da sub arvore a esquerda, o elemento 9. O elemento 10 é maior que a raiz, então segue para a direita;

Passo 3: Compara o valor do elemento 10 com o nó folha 10.O elemento 10 é igual ao nó folha 10. Então retorna a mensagem: O elemento 10 foi encontrado na arvore.

**Situação 2: Busca um de um elemento que não pertence a arvore. O elemento 40.** Passo 1: Compara o valor do elemento 40 com a raiz 30.O elemento é maior que a raiz, então segue para a direita;

Passo 2: Compara o valor do elemento 40 com a raiz da sub arvore a esquerda, o

elemento 50. O elemento 40 é menor que a raiz, então segue para a esquerda;

Passo 3: Compara o valor do elemento 40 com o nó folha 43.O elemento 40 é diferente do nó folha 43. Então retorna a mensagem: O elemento 40 não foi encontrado na arvore.

**4. Usando a mesma árvore do exercício 1, remova um elemento que não tem nenhum filho, depois um que tenha apenas um filho e outro que tem dois filhos e explique o que aconteça.**

Resposta: Os elementos foram inseridos na arvore na seguinte sequência: 10,30,50,9,7,43,54,60,63. Após a inserção e exibição da arvore **em ordem** a aplicação apresenta o seguinte resultado(elemento(fb)): 7(0) 9 (0) 10(0) 30(1) 43(0) 50(1) 54(0) 60(0) 63(0).

**Situação 1: Remover um elemento que não tem nenhum filho.**

Passo 1: O elemento escolhido foi o (7) que é um nó folha e não possui filhos;

Passo 2: Após a remoção e exibição da arvore **em ordem,** a aplicação apresenta o seguinte resultado: 9 (1) 10(0) 30(1) 43(0) 50(1) 54(0) 60(0) 63(0). Houve a exclusão do nó

(7) e alteração do fator de balanceamento(fb) do elemento (9).

**Situação 2: Remover um elemento que tem apenas um filho.**

Passo 1: O elemento escolhido foi o (9) que tem apenas um filho a direita, o elemento 10; Passo 2: Após a remoção e exibição da arvore **em ordem,** a aplicação apresenta o seguinte resultado: 10(0) 30(0) 43(0) 50(0) 54(0) 60(0) 63(0). Houve a exclusão do nó (9) e alteração do fator de balanceamento(fb) dos elementos (30) e (50). O elemento (50) agora é raiz da arvore.

**Situação 3: Remover um elemento que tem dois filhos.**

Passo 1: O elemento escolhido foi o (60) que é raiz da sub arvore a direita e tem dois filhos, o elemento (54) a esquerda e o elemento (63) a direita.

Passo 2: Após a remoção e exibição da arvore **em ordem,** a aplicação apresenta o seguinte resultado: 10(0) 30(0) 43(0) 50(0) 54(1) 63(0). Houve a exclusão do nó (60) e alteração do fator de balanceamento(fb) do elementos (54) que agora tem um filho a direita, o elemento (63).

**5. Monte uma árvore 2-3, onde cada nó possa ter nenhum, dois ou três filhos, usando o link:**

**https://www.cs.usfca.edu/~galles/visualization/BTree.html, e explique o que acontece a cada valor inserido.**

Resposta: Os elementos foram inseridos na arvore BTree na seguinte ordem:

50,15,30,5,10,40,45,20,25,65.

Passo 1: Inserir a chave (50) na arvore.

Passo 2: Inserir a chave (15) na arvore. Não houve a inclusão de um novo nó, ou seja, as chaves continuam no mesmo nó.

Passo 3: Inserir a chave (30) na arvore. O nó atingiu o seu limite, houve a inclusão de novos nós. Por ser a chave central, a chave 30 sobe, e o nó anterior é dividido em dois outros nós. Cada nó com um elemento.

Passo 4: Inserir a chave (5) na arvore. Não houve a inclusão de um novo nó. Agora as chaves (5) e (15) estão no mesmo nó.

Passo 5: Inserir a chave (10) na arvore. O nó atingiu o seu limite, houve a inclusão de novos nós. Por ser a chave central, a chave (10) sobe, e o nó da inclusão é dividido em dois outros nós, um nó com a chave (5) e outro com a chave (15).

Passo 6: Inserir a chave (40) na arvore. Não houve a inclusão de um novo nó. Agora as chaves (40) e (50) estão no mesmo nó.

Passo 7: Inserir a chave (45) na arvore. O nó atingiu o seu limite, houve a inclusão de novos nós. Por ser a chave central, a chave (45) sobe, e o nó da inclusão é dividido em dois outros nós, em seguida, o próximo nó também atinge o seu limite, obrigando a chave

(30) a subir por ser a chave central deste nó.

Passo 8: Inserir a chave (20) na arvore. Não houve a inclusão de um novo nó. Agora as chaves (15) e (20) estão no mesmo nó.

Passo 9: Inserir a chave (25) na arvore. O nó atingiu o seu limite, houve a inclusão de novos nós. Por ser a chave central, a chave (20) sobe, e o nó da inclusão é dividido em dois outros nós, um nó com a chave (15) e outro com a chave (25).

Passo 10: Inserir a chave (65) na arvore. Não houve a inclusão de um novo nó. Agora as chaves (50) e (65) estão no mesmo nó. Fim do teste.

**6. Usando a árvore do exercício anterior, faça busca de um elemento que existe e de um que não existe na árvore, descreva o que acontece.**

Resposta: A busca em uma Árvore-BTree é um procedimento semelhante a busca em árvore de busca binária, exceto que devemos decidir entre vários caminhos. Como as chaves estão ordenadas, basta realizarmos uma busca binária nos elementos de cada nó, se o elemento não for encontrado naquele nó vamos para o filho apropriado e realizamos a busca binária.

**Situação 1: Busca de um elemento que pertence a arvore:**

Passo 1: Busca da chave (25) – A busca começa pelo nó raiz que possui apenas a chave (30). A chave procurada não foi encontrada, então a busca segue.

Passo 2: A busca segue pelo nó filho que possui apenas as chaves (10) e (20). A chave procurada não foi encontrada, então segue a busca.

Passo 3: A busca segue para próximo nó filho que possui apenas a chave (25). A chave procurada foi encontrada, então encerra a busca.

**Situação 2: Busca de um elemento que não pertence a arvore:**

Passo 1: Busca da chave (52) – A busca começa pelo nó raiz que possui apenas a chave (30). A chave procurada não foi encontrada, então a busca segue.

Passo 2: A busca segue pelo nó filho que possui apenas a chave (45). A chave procurada não foi encontrada, então segue a busca.

Passo 3: A busca segue para próximo nó filho que possui apenas a chave (50) e (65). A chave procurada também não foi encontrada. Por não existir mais nós filhos neste caminho, então encerra a busca.

**7. Usando a mesma árvore do exercício 4, remova um elemento que não tem nenhum filho, depois um que tenha apenas um filho e outro que tem dois filhos e explique o que aconteça.**

Resposta: Os elementos foram inseridos na arvore na seguinte sequência: 10,30,50,9,7,43,54,60,63. Após a inserção e exibição da arvore **em ordem** a aplicação apresenta o seguinte resultado(elemento(fb)): 7(0) 9 (0) 10(0) 30(1) 43(0) 50(1) 54(0) 60(0) 63(0).

**Situação 1: Remover um elemento que não tem nenhum filho.**

Passo 1: O elemento escolhido foi o (7) que é um nó folha e não possui filhos;

Passo 2: Após a remoção e exibição da arvore **em ordem,** a aplicação apresenta o seguinte resultado: 9 (1) 10(0) 30(1) 43(0) 50(1) 54(0) 60(0) 63(0). Houve a exclusão do nó

(7) e alteração do fator de balanceamento(fb) do elemento (9).

**Situação 2: Remover um elemento que tem apenas um filho.**

Passo 1: O elemento escolhido foi o (9) que tem apenas um filho a direita, o elemento 10; Passo 2: Após a remoção e exibição da arvore **em ordem,** a aplicação apresenta o seguinte resultado: 10(0) 30(0) 43(0) 50(0) 54(0) 60(0) 63(0). Houve a exclusão do nó (9) e alteração do fator de balanceamento(fb) dos elementos (30) e (50). O elemento (50) agora é raiz da arvore.

**Situação 3: Remover um elemento que tem dois filhos.**

Passo 1: O elemento escolhido foi o (60) que é raiz da sub arvore a direita e tem dois filhos, o elemento (54) a esquerda e o elemento (63) a direita.

Passo 2: Após a remoção e exibição da arvore **em ordem,** a aplicação apresenta o seguinte resultado: 10(0) 30(0) 43(0) 50(0) 54(1) 63(0). Houve a exclusão do nó (60) e alteração do fator de balanceamento(fb) do elementos (54) que agora tem um filho a direita, o elemento (63).

1. **Em relação a grafos explique o que é a matriz de incidência, link importante:** [**http://pt.wikipedia.org/wiki/Matriz\_de\_incidência.**](http://pt.wikipedia.org/wiki/Matriz_de_incid%EAncia) **Explicar em teoria e não o código.**

Resposta:

A matriz de incidência B = [b ij] de um grafo G = (V,E) , com V = (v 1, v 2, . . . , v n) e E = (e 1, e 2, . . . , e m), é definida da seguinte forma:

1 se vi € ej

B( i, j) =

0 caso contrario

Uma **matriz de incidência** representa computacionalmente um [grafo](https://pt.wikipedia.org/wiki/Grafo) através de uma matriz bidimensional, onde uma das dimensões são [vértices](https://pt.wikipedia.org/wiki/V%C3%A9rtice) e a outra dimensão são [arestas.](https://pt.wikipedia.org/wiki/Aresta)

Dado um grafo G com *n* vértices e *m* arestas, podemos representá-lo em uma [matriz](https://pt.wikipedia.org/wiki/Matriz_(matem%C3%A1tica)) *n* x *m* M. A definição precisa das entradas da matriz varia de acordo com as propriedades do grafo que se deseja representar, porém de forma geral guarda informações sobre como os vértices se relacionam com cada aresta (isto é, informações sobre a *incidência* de uma aresta em um vértice).

1. **Em relação a grafos explique o que é a matriz de adjacência, link importante:** [**http://pt.wikipedia.org/wiki/Matriz\_de\_adjacência.**](http://pt.wikipedia.org/wiki/Matriz_de_adjac%EAncia) **Explicar em teoria e não o código.**

Resposta: Trata-se de uma representação que favorece a recuperação mais rápida de informação nos grafos, basicamente por não possuir informações de não adjacência (os zeros na matriz de adjacência).

Uma **matriz de adjacência** é uma das formas de se representar um [grafo.](https://pt.wikipedia.org/wiki/Teoria_dos_grafos)

Dado um grafo *G* com *n* [vértices,](https://pt.wikipedia.org/wiki/V%C3%A9rtice) podemos representá-lo em uma [matriz](https://pt.wikipedia.org/wiki/Matriz_(matem%C3%A1tica)) *n* x *n* *A(G)=*[*aij*] (ou simplesmente *A*). A definição precisa das entradas da matriz varia de acordo com as propriedades do grafo que se deseja representar, porém de forma geral o valor *aij* guarda informações sobre como os vértices *vi* e *vj* estão relacionados (isto é, informações sobre a adjacência de *vi* e *vj*).

1. **Escolha e explique um dos seguintes algoritmos: Algoritmo de Dijkstra e Algoritmo de Kruskal. Explicar em teoria e não o código Links importantes:**

[**http://pt.wikipedia.org/wiki/Algoritmo\_de\_Dijkstra**](http://pt.wikipedia.org/wiki/Algoritmo_de_Dijkstra)  **http://pt.wikipedia.org/wiki/Algoritmo\_de\_Kruskal**

Resposta:

* + O **algoritmo de Dijkstra**, concebido pelo [cientista da computação](https://pt.wikipedia.org/wiki/Ci%C3%AAncia_da_computa%C3%A7%C3%A3o) holandês [Edsger](https://pt.wikipedia.org/wiki/Edsger_Dijkstra)

[Dijkstra](https://pt.wikipedia.org/wiki/Edsger_Dijkstra) em 1956 e publicado em 1959,

soluciona o [problema do caminho mais curto](https://pt.wikipedia.org/wiki/Problema_do_caminho_mais_curto) num [grafo dirigido](https://pt.wikipedia.org/wiki/Grafo_dirigido) ou não dirigido com arestas de peso não negativo, em tempo computacional O([m+n]log n) onde m é o número de arestas e n é o número de vértices.

* + O **algoritmo de Kruskal** é um [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo) em [teoria dos grafos](https://pt.wikipedia.org/wiki/Teoria_dos_grafos) que busca uma [árvore geradora mínima](https://pt.wikipedia.org/wiki/%C3%81rvore_geradora_m%C3%ADnima) para um [grafo](https://pt.wikipedia.org/wiki/Grafo) [conexo](https://pt.wikipedia.org/wiki/Conexidade) com [pesos.](https://pt.wikipedia.org/wiki/Grafo_valorado) Isto significa que ele encontra um [subconjunto](https://pt.wikipedia.org/wiki/Subconjunto) das [arestas](https://pt.wikipedia.org/wiki/Arestas) que forma uma árvore que inclui todos os [vértices,](https://pt.wikipedia.org/wiki/V%C3%A9rtices) onde o peso total, dado pela soma dos pesos das arestas da árvore, é minimizado. Se o grafo não for conexo, então ele encontra uma *floresta geradora mínima* (uma árvore geradora mínima para cada [componente conexo](https://pt.wikipedia.org/wiki/Componente_conexo) do grafo). O algoritmo de Kruskal é um exemplo de um [algoritmo guloso](https://pt.wikipedia.org/wiki/Algoritmo_guloso) (também conhecido como ganancioso ou *greedy*).

**Entregar: Código Fonte e Relatório de implementação, contendo: Descrição de cada questão explicando cada função implementada de forma objetiva.**