Lista Árvore

Gabriel de Paula Gaspar Pinto

1 Introdução

Nesse trabalho, foi realizada os exercícios da "Lista árvore", disponível no Classroom, da matéria de Estrutura de Dados. Para realizar esta lista, foi utilizado o aplicativo Samsung Notes, em um tablet Android, para facilitar a visualização das árvores feitas manualmente e não precisar usar papel. Para confecção dos códigos, foi utilizado o Visual Studio Code, dentro do ambiente virtual do Windows Subsystem for Linux (WSL), além de auxílio da inteligência artifical generativa, ChatGPT.

Lista de Exercícios

Questão 1:

(a) A árvore com meu nome completo, tem altura 8, considerando que a raiz está no nível 1.

(b) Em um percurso pós-fixado(pós-ordem), a ordem dos nós fica da seguinte maneira: A A A D E A E B A O N P L P I P G L I R T S U R G.

(c) O sucessor do nó de maior altura é a letra A.

(d) O quarto nó inserido na árvore, é a letra R (G A B **R** I E L), então o seu predecessor é o nó raiz da árvore, com a letra G.

(e) Na figura 2 está o resultado da árvore quando se remove o nó raiz (G), resultado do maior nó dentro da subárvore da esquerda.

Abaixo estão as imagens da árvore binária de busca completa e a árvore com a raiz removida.

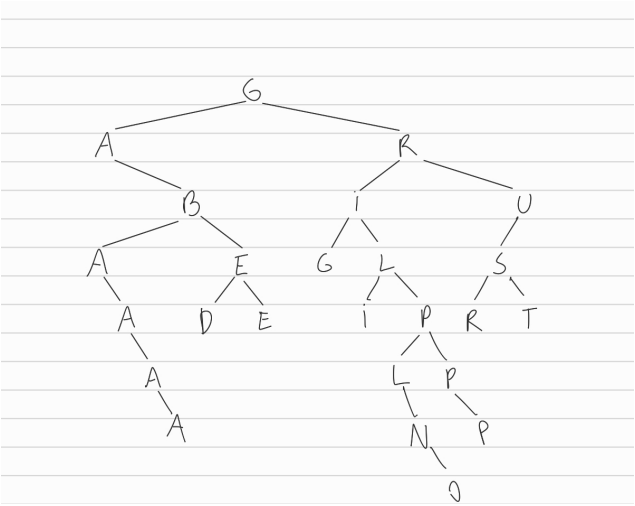

Figura 1: Árvore Binária de Busca completa

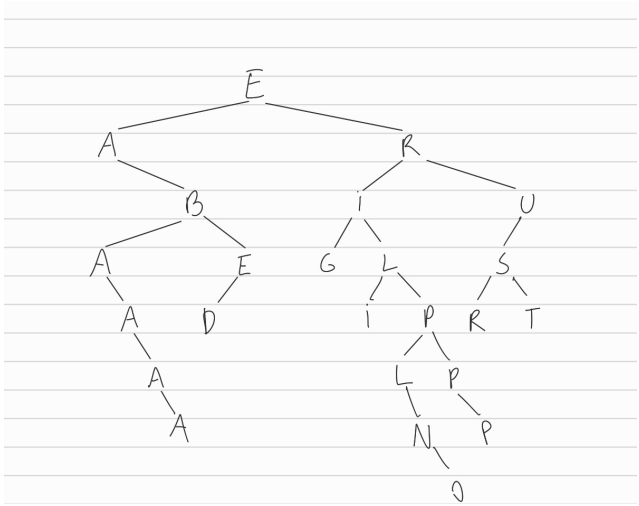

Figura 2: Árvore Binária de Busca após remover a raiz

Questão 2: Ao montar a árvore de Huffmann, com meu nome completo e sem espaços, fica do jeito que mostra na imagem a seguir

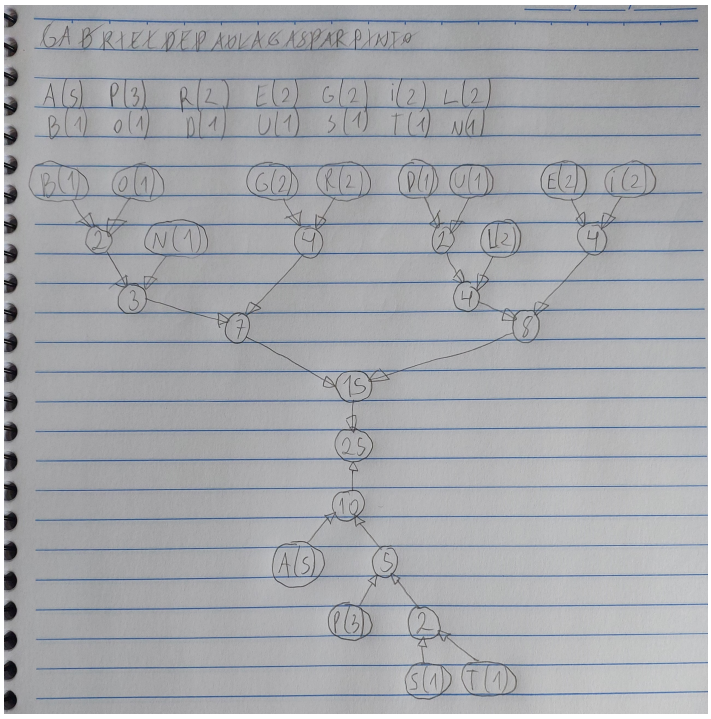

Figura 3: Árvore de Huffmann, com meu nome completo

Questão 3: A seguir está o meu código para calcular a altura de uma árvore binária de pesquisa. Para poder testar o código, eu inseri a árvore binária de pesquisa do meu nome, presente na questão 1. Neste caso, quando rodar o programa, o resultado impresso na tela é 8.

```cpp
#include <iostream>
using namespace std;

//no = nó
struct No {
    int dado;
    No* esquerda;
    No* direita;

    No(int valor) {
        dado = valor;
        esquerda = nullptr;
        direita = nullptr;
    }
};

int alturaArvore(No* raiz) {
    if (raiz == nullptr)
        return 0;

    int alturaEsquerda = alturaArvore(raiz->esquerda);
    int alturaDireita = alturaArvore(raiz->direita);
    return max(alturaEsquerda, alturaDireita) + 1;
}

int main() {
    //usando a árvore com meu nome, trocando as letras por números, A=1, B=2, ... , Z=26
    No* raiz = new No(7);
    raiz->esquerda = new No(1);
    raiz->esquerda->direita = new No(2);
    raiz->esquerda->direita->esquerda = new No(1);
    raiz->esquerda->direita->direita = new No(5);
    raiz->esquerda->direita->esquerda->direita = new No(1);
    raiz->esquerda->direita->esquerda->direita->direita = new No(1);
    raiz->esquerda->direita->esquerda->direita->direita->direita = new No(1);
```

```
    raiz->esquerda->direita->direita->direita = new No(5);
    raiz->esquerda->direita->direita->esquerda = new No(4);
    raiz->direita = new No(18);
    raiz->direita->esquerda = new No(9);
    raiz->direita->esquerda->esquerda = new No(7);
    raiz->direita->esquerda->direita = new No(12);
    raiz->direita->esquerda->direita->esquerda = new No(9);
    raiz->direita->esquerda->direita->direita = new No(16);
    raiz->direita->esquerda->direita->direita->direita = new No(16);
    raiz->direita->esquerda->direita->direita->direita->direita = new No(16);
    raiz->direita->esquerda->direita->direita->esquerda = new No(12);
    raiz->direita->esquerda->direita->direita->esquerda->direita = new No(14);
    raiz->direita->esquerda->direita->direita->esquerda->direita->direita = new No(15);
    raiz->direita->direita = new No(21);
    raiz->direita->direita->esquerda = new No(19);
    raiz->direita->direita->esquerda->esquerda = new No(18);
    raiz->direita->direita->esquerda->direita = new No(20);

    cout << "Altura da árvore: " << alturaArvore(raiz) << endl;

    return 0;
}
```

Questão 4: A seguir está o meu código para calcular o nível (altura) de um nó em uma árvore. Para poder testar o código, eu também inseri a árvore binária de pesquisa do meu nome, presente na questão 1. Os resultados são das letras A, L, P, U, O e da raiz (letra G), respectivamente. Neste caso, os resultados que o programa deve mostrar são, respectivamente: 2, 4, 5, 3, 8 e 1. Para alterar os valores procurados, é necessário somente mudar o valor do meio quando as funções são chamadas, dentro do cout, considerando que cada letra equivale a um número, sendo A = 1, B = 2 e assim por diante.

```cpp
#include <iostream>
using namespace std;

//no = nó
struct No {
    int dado;
    No* esquerda;
    No* direita;

    No(int valor) {
        dado = valor;
        esquerda = nullptr;
        direita = nullptr;
    }
};

int nivelNo(No* raiz, int valor, int nivelAtual) {
    if (raiz == nullptr)
        return -1;

    if (raiz->dado == valor)
        return nivelAtual;

    int nivelEsquerda = nivelNo(raiz->esquerda, valor, nivelAtual + 1);
    if (nivelEsquerda != -1)
        return nivelEsquerda;

    return nivelNo(raiz->direita, valor, nivelAtual + 1);
}

int main() {
    No* raiz = new No(7);
    raiz->esquerda = new No(1);
    raiz->esquerda->direita = new No(2);
    raiz->esquerda->direita->esquerda = new No(1);
    raiz->esquerda->direita->direita = new No(5);
    raiz->esquerda->direita->esquerda->direita = new No(1);
    raiz->esquerda->direita->esquerda->direita->direita = new No(1);
    raiz->esquerda->direita->esquerda->direita->direita->direita = new No(1);
    raiz->esquerda->direita->direita->direita = new No(5);
    raiz->esquerda->direita->direita->esquerda = new No(4);
    raiz->direita = new No(18);
    raiz->direita->esquerda = new No(9);
    raiz->direita->esquerda->esquerda = new No(7);
    raiz->direita->esquerda->direita = new No(12);
    raiz->direita->esquerda->direita->esquerda = new No(9);
    raiz->direita->esquerda->direita->direita = new No(16);
    raiz->direita->esquerda->direita->direita->direita = new No(16);
    raiz->direita->esquerda->direita->direita->direita->direita = new No(16);
    raiz->direita->esquerda->direita->direita->esquerda = new No(12);
    raiz->direita->esquerda->direita->direita->esquerda->direita = new No(14);
    raiz->direita->esquerda->direita->direita->esquerda->direita->direita = new No(15);
    raiz->direita->direita = new No(21);
    raiz->direita->direita->esquerda = new No(19);
    raiz->direita->direita->esquerda->esquerda = new No(18);
    raiz->direita->direita->esquerda->direita = new No(20);

    cout << "Nível do nó A: " << nivelNo(raiz, 1, 1) << endl;
    cout << "Nível do nó L: " << nivelNo(raiz, 12, 1) << endl;
    cout << "Nível do nó P: " << nivelNo(raiz, 16, 1) << endl;
    cout << "Nível do nó U: " << nivelNo(raiz, 21, 1) << endl;
    cout << "Nível do nó O: " << nivelNo(raiz, 15, 1) << endl;
    cout << "Nível da raiz: " << nivelNo(raiz, 7, 1) << endl;

    return 0;
}
```

2 Nota

Como este arquivo LaTeX utiliza arquivos externos (para imagens e códigos-fonte), este trabalho está disponivel em um repositório pessoal do GitHub, junto com todos os outros trabalhos que eu utilizei LaTeX.

GitHub - LaTeX