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.

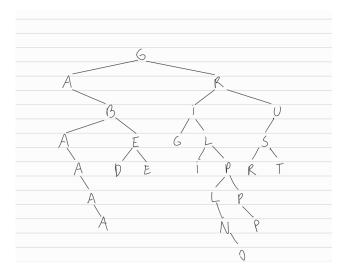


Figura 1: Árvore Binária de Busca completa

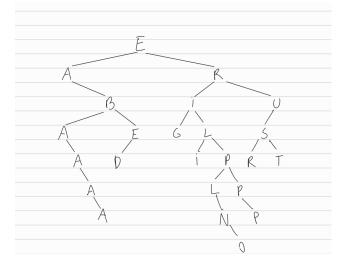


Figura 2: Árvore Binária de Busca após remover a raiz

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

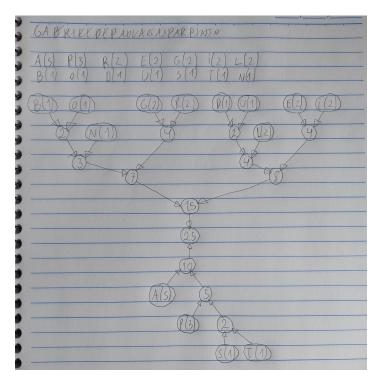


Figura 3: Árvore de Huffmann, com meu nome completo

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.

```
#include <iostream>
using namespace std;
  //no = nó
  struct No {
      int dado;
      No* esquerda;
      No* direita;
      No(int valor) {
          dado = valor;
          esquerda = nullptr;
          direita = nullptr;
14
15 };
16
  int alturaArvore(No* raiz) {
17
      if (raiz == nullptr)
18
19
          return 0;
20
      int alturaEsquerda = alturaArvore(raiz->esquerda);
21
22
      int alturaDireita = alturaArvore(raiz->direita);
      return max(alturaEsquerda, alturaDireita) + 1;
23
24
  }
25
  int main() {
      //usando a árvore com meu nome, trocando as letras por números, A=1, B=2, ..., Z=26
27
      No* raiz = new No(7);
28
29
      raiz->esquerda = new No(1);
      raiz->esquerda->direita = new No(2);
30
31
      raiz->esquerda->direita->esquerda = new No(1);
      raiz->esquerda->direita->direita = new No(5);
32
      raiz->esquerda->direita->esquerda->direita = new No(1);
33
      raiz->esquerda->direita->esquerda->direita->direita = new No(1);
34
      raiz->esquerda->direita->esquerda->direita->direita->direita = new No(1);
35
```

```
raiz->esquerda->direita->direita->direita = new No(5);
36
                 raiz->esquerda->direita->direita->esquerda = new No(4);
37
                 raiz->direita = new No(18);
38
39
                 raiz->direita->esquerda = new No(9);
                raiz->direita->esquerda->esquerda = new No(7);
40
                raiz->direita->esquerda->direita = new No(12);
41
42
                raiz->direita->esquerda->direita->esquerda = new No(9);
                raiz->direita->esquerda->direita->direita = new No(16);
43
44
                  raiz->direita->esquerda->direita->direita = new No(16);
                raiz->direita->esquerda->direita->direita->direita->direita = new No(16);
45
                raiz->direita->esquerda->direita->esquerda = new No(12);
                raiz->direita->esquerda->direita->direita->esquerda->direita = new No(14);
47
                raiz->direita->esquerda->direita->esquerda->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->direita->dire
48
                 raiz->direita->direita = new No(21);
49
                raiz->direita->direita->esquerda = new No(19);
50
                raiz->direita->direita->esquerda->esquerda = new No(18);
51
                raiz->direita->direita->esquerda->direita = new No(20);
52
53
                  cout << "Altura da árvore: " << alturaArvore(raiz) << endl;</pre>
54
55
                 return 0;
57 }
```

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.

```
#include <iostream>
using namespace std;
4 //no = nó
5 struct No {
      int dado;
6
      No* esquerda;
      No* direita;
8
9
      No(int valor) {
10
          dado = valor;
11
12
           esquerda = nullptr;
           direita = nullptr;
13
14
15 };
16
int nivelNo(No* raiz, int valor, int nivelAtual) {
      if (raiz == nullptr)
18
19
           return -1;
20
21
      if (raiz->dado == valor)
          return nivelAtual:
22
23
      int nivelEsquerda = nivelNo(raiz->esquerda, valor, nivelAtual + 1);
24
      if (nivelEsquerda != -1)
25
          return nivelEsquerda;
26
27
      return nivelNo(raiz->direita, valor, nivelAtual + 1);
28
29 }
30
31 int main() {
      No* raiz = new No(7);
32
      raiz->esquerda = new No(1);
33
      raiz->esquerda->direita = new No(2);
34
      raiz->esquerda->direita->esquerda = new No(1);
35
36
      raiz->esquerda->direita->direita = new No(5);
      raiz->esquerda->direita->esquerda->direita = new No(1);
37
38
      raiz->esquerda->direita->esquerda->direita = new No(1);
      raiz->esquerda->direita->esquerda->direita->direita->direita = new No(1);
39
      raiz->esquerda->direita->direita = new No(5);
40
      raiz->esquerda->direita->direita->esquerda = new No(4);
41
42
      raiz->direita = new No(18);
      raiz->direita->esquerda = new No(9);
43
      raiz->direita->esquerda->esquerda = new No(7);
44
      raiz->direita->esquerda->direita = new No(12);
45
      raiz->direita->esquerda->direita->esquerda = new No(9);
46
      raiz->direita->esquerda->direita->direita = new No(16);
47
48
      raiz->direita->esquerda->direita->direita = new No(16);
      raiz->direita->esquerda->direita->direita->direita = new No(16);
49
      raiz->direita->esquerda->direita->esquerda = new No(12);
50
      \label{lem:condition} {\tt raiz->direita->esquerda->direita->esquerda->direita = {\tt new} \ \ {\tt No} \ (14) \ ;
51
      raiz->direita->esquerda->direita->esquerda->direita = new No(15);
52
      raiz->direita->direita = new No(21);
53
      raiz->direita->direita->esquerda = new No(19);
54
      raiz->direita->esquerda->esquerda = new No(18);
      raiz \rightarrow direita \rightarrow direita \rightarrow esquerda \rightarrow direita = new No(20);
56
57
      cout << "Nível do nó A: " << nivelNo(raiz, 1, 1) << endl;</pre>
58
      cout << "Nível do nó L: " << nivelNo(raiz, 12, 1) << endl;</pre>
59
      cout << "Nível do nó P: " << nivelNo(raiz, 16, 1) << endl;
60
      cout << "Nível do nó U: " << nivelNo(raiz, 21, 1) << endl;
61
       cout << "Nível do nó 0: " << nivelNo(raiz, 15, 1) << endl;</pre>
62
      cout << "Nível da raiz: " << nivelNo(raiz, 7, 1) << endl;</pre>
63
64
65
      return 0;
66 }
```

2 Nota

Como este arquivo IATEX utiliza arquvios 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 IATEX.

 $\operatorname{GitHub} - \operatorname{\LaTeX}$