**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР.**

отчет

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: «Двоичные деревья».

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1301 |  | Ахметзянов Д.А. |
| Преподаватель |  | Родионова Е. А. |

Санкт-Петербург

2022

Оглавление

[Формулировка задания. 3](#_Toc30565)

[Описание методов и оценка временной сложности. 3](#_Toc14052)

[Пример работы программы. 4](#_Toc31347)

[Текст программы. 6](#_Toc30347)

# Формулировка задания.

Реализовать структуру данных двоичное дерево поиска и следующие методы: нахождение минимума, нахождение максимума, прямой (preorder), центрированный (inorder) и обратный обход (postorder) по дереву, поиск элемента, нахождение следующего и предыдущего элемента, удаление элемента, обход в ширину.

Реализовать визуализацию дерева.

Указать теоретическую временную сложность для всех операций.

С помощью реализованной структуры данных написать программу, позволяющую преобразовать запись из префиксной/инфиксной/постфиксной нотации в префиксную/инфиксную/постфиксную нотацию.

# Описание методов и оценка временной сложности.

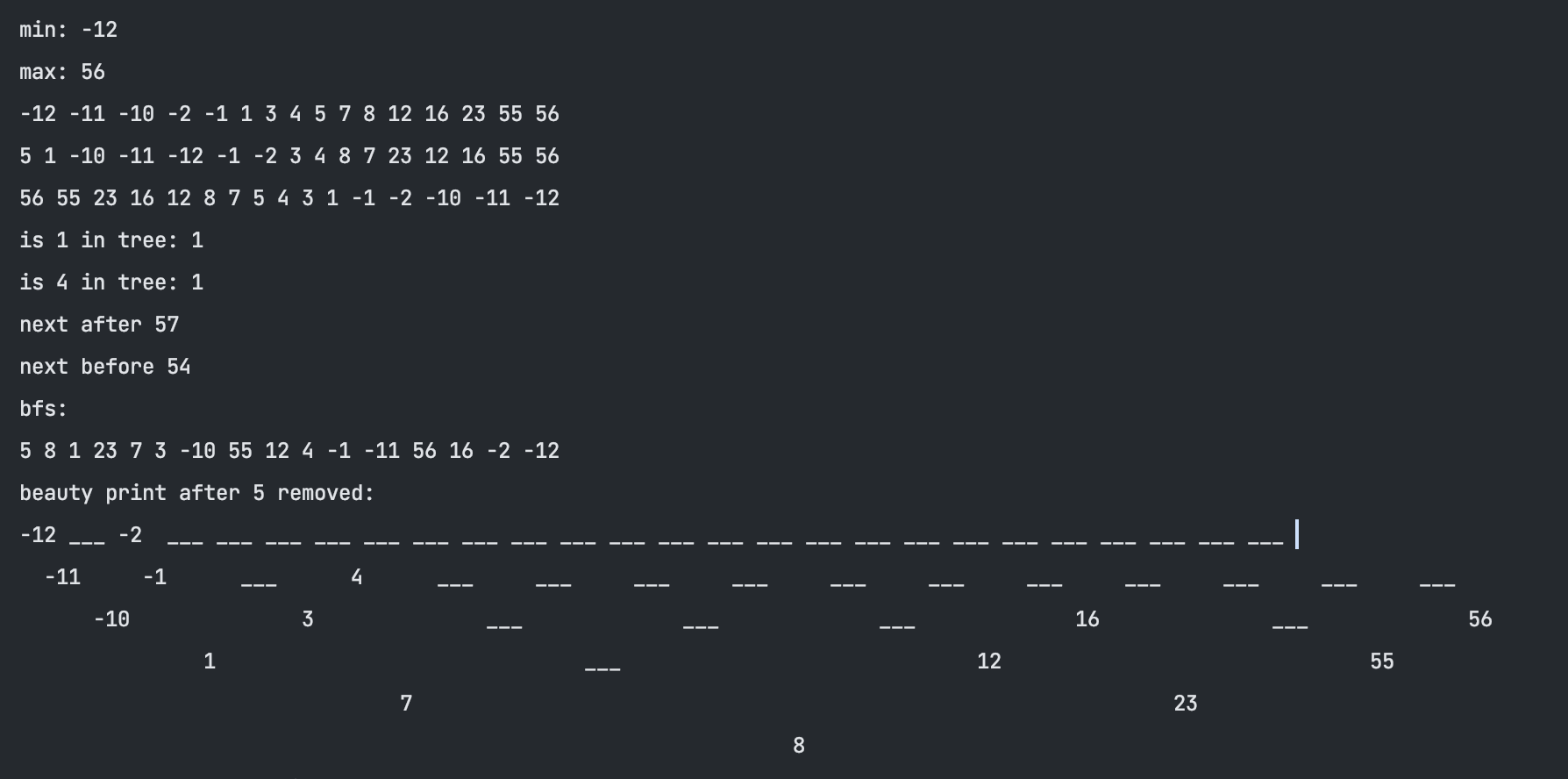
|  |  |  |  |
| --- | --- | --- | --- |
| Имя метода | Оценка сложности | | |
| Лучший случай | Средний случай | Худший |
| Нахождение минимума | O(1) | O(h), h - высота дерева | O(n) |
| Нахождение максимума | O(1) | O(h), h - высота дерева | O(n) |
| Прямой обход | O(n) | O(n) | O(n) |
| Центрированный обход | O(n) | O(n) | O(n) |
| Обратный обход | O(n) | O(n) | O(n) |
| Поиск элемента | O(1) | O(h) | O(h) |
| Нахождение следующего | O(1) | O(h) | O(h) |
| Нахождение предыдущего | O(1) | O(h) | O(h) |
| Удаление элемента | O(1) | O(h) | O(h) |
| Обход в ширину | O(n) | O(n) | O(n) |

# Пример работы программы.

Код:

vector<int> v **{**5,1,8,3,-10,23,55, 12, 7, -1, -2, 4, 16, 56, -11, -12**}**;  
  
Tree<int> tree;  
  
for (int i : v) {  
 tree.Add(i);  
}  
  
cout<<"min: "<<tree.GetMin()<<"\nmax: "<<tree.GetMax()<<"\n";  
  
tree.inorderPrint();  
tree.preorderPrint();  
tree.postorderPrint();  
  
cout << "is 1 in tree: " << bool(tree.search(1)) <<"\n";  
cout << "is 4 in tree: " << bool(tree.search(4))<<"\n";  
  
cout<<"next after 5"<<tree.GetNext(5)<<"\n";  
cout<<"next before 5"<<tree.GetPrev(5)<<"\n";  
  
cout<<"bfs:\n";  
tree.bfsPrint();  
tree.RemoveNode(5);  
cout<<"\nbeauty print after 5 removed:\n";  
tree.beautyPrint();

Результат работы:

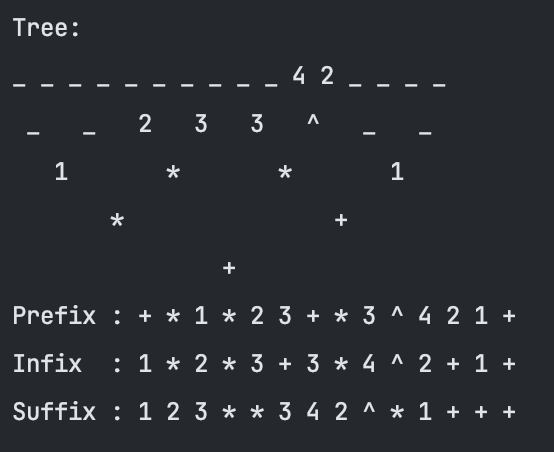


**Изменение нотации математического выражения.**

Код:

Tree<string> expressionTree;  
  
expressionTree.fromExpression("1 \* 2 \* 3 + 3 \* 4 ^ 2 + 1", Infix);  
  
cout << "Tree: " << endl;  
expressionTree.beautyPrint();  
  
cout << "Prefix : " << expressionTree.toExpression(Prefix) << endl;  
cout << "Infix : " << expressionTree.toExpression(Infix) << endl;  
cout << "Suffix : " << expressionTree.toExpression(Suffix) << endl;

Результат работы:



# Текст программы.

[Ссылка на github](https://github.com/DamirAhm/AADS/tree/master/Binary%20tree)