

Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

КУРСОВАЯ РАБОТА

«Эйлеров обход дерева»

По дисциплине: Объектно-ориентированное программирование

Выполнил
студент гр. 3331506/70401

Д.С. Кученов

Преподаватель

М.С. Ананьевский

« ____ » _____ 2020 г.

Санкт-Петербург

2020

Оглавление

Введение.....	3
Описание алгоритма.....	4
Реализация на C++	6
Анализ алгоритма.....	9
Заключение	10
Список литературы	11

Введение

Привычные алгоритмы обхода деревьев: прямой, центрированный и обратный являются итеративными процессами. Каждый обход посещаются узлы дерева в определенном порядке, причем это происходит гарантированно и ровно по одному разу. Мы можем объединить алгоритмы обхода деревьев, приведенные выше, в единую структуру, при этом опустив требование о том, что каждый узел должен быть посещен ровно один раз. Полученный метод обхода называется Эйлеровым обходом дерева. Преимущество этого обхода заключается в том, что он позволяет более просто и обобщенно выразить остальные виды алгоритмов для обхода дерева. Метод был предложен в 1984 году Робертом Тарьяном.

В данной работе будет разобран принцип работы алгоритма, его реализации на C++, а также произведен анализ сложности и численный анализ алгоритма.

Описание алгоритма

Итак, Эйлеров обход по бинарному дереву можно неформально определить как «прогулку» вокруг дерева, т.е. мы начинаем идти от корня к его левому потомку, при этом рассматривая края как «стены», которые всегда должны быть слева от нас. Наглядно это отображено на рисунке 1 ниже

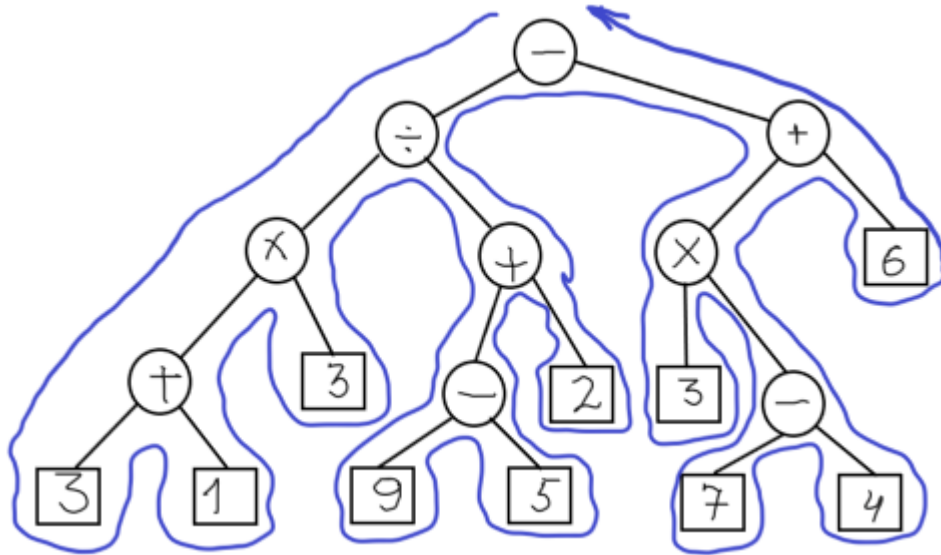


Рисунок 1 — Эйлеров обход дерева

Каждый узел встречается трижды во время обхода:

- «Слева»
- «Снизу»
- «Справа»

Если узел является внешним, то эти три «посещения» на самом деле все происходят одновременно

Если представить алгоритм на псевдокоде, то это будет выглядеть примерно так:

eulerTour(T,p)

Выполнить действие для посещения узла p слева,

Если p - внутренний узел, **то**

{

 Рекурсивный обход левого поддерева p путем вызова eulerTour (T, p.left ())

}

Выполнить действие для посещения узла p снизу,

Если p - внутренний узел, то

{

Рекурсивный обход правого поддерева p путем вызова `eulerTour (T, p.right ())`

}

Выполнить действие для посещения узла p справа

Прямой обход двоичного дерева эквивалентен обходу Эйлера, в котором каждый узел имеет связанное действие "посещения" только тогда, когда оно встречается слева. Подобным образом, центрированный и обратный обходы двоичного дерева являются эквивалентными Эйлерову обходу, где каждый узел имеет связанное действие "посещения" только тогда, когда оно встречается соответственно снизу или справа.

Обход Эйлера расширяет прямой, центрированный и обратный обходы, но с помощью него можно выполнять и другие виды обходов или использовать его для других задач. Например, нам необходимо вычислить число потомков каждого узла p в двоичном дереве T с n -количеством узлов. Мы начинаем обход Эйлера с инициализации счетчика нулем, а затем увеличиваем счетчик каждый раз, когда мы посещаем узел слева. Чтобы определить количество потомков в узле p , мы вычисляем разницу между значениями счетчика, когда p был посещен слева и когда посещен справа, и добавляем 1. Это дает нам число потомков p , потому что каждый узел в поддереве с корнем p подсчитывается между посещением p слева и посещением p справа.

Реализация на C++

Реализация Эйлера обход для дерева, приведенного на рисунке 2. На нем же показано визуально пошаговое посещение каждого узла.

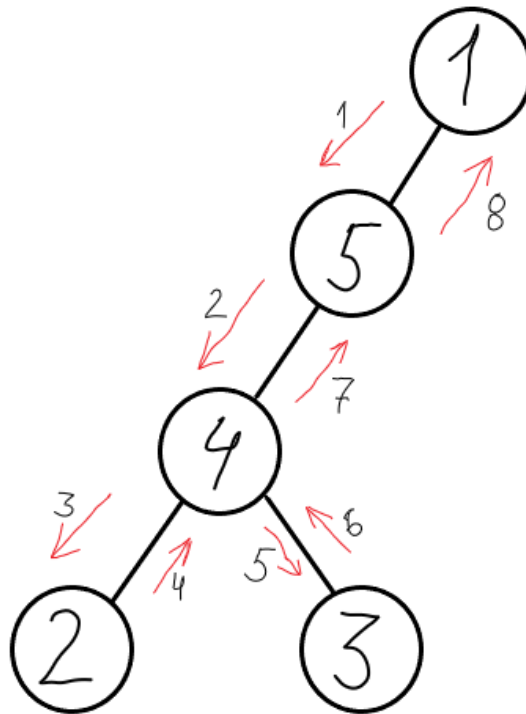


Рисунок 2 — Визуализация программы Эйлера обхода дерева

Ниже приведен листинг программы на языке C++

```
#include <iostream>
#include <vector>
using namespace std;
#define MAX 1000
// Представление дерева
vector<int> adj[MAX];
// Посещенный массив для отслеживания посещенных узлов за обход
int vis[MAX];
// Массив для хранения обхода
int Euler[2 * MAX];
// Функция для добавления ребер к дереву
void add_edge(int u, int v)
{
```

```

    adj[u].push_back(v);
    adj[v].push_back(u);
}

// Функция для хранения Эйлера обхода дерева
void eulerTree(int u, int& indx)
{
    vis[u] = 1;
    Euler[indx++] = u;
    for (auto it : adj[u]) {
        if (!vis[it]) {
            eulerTree(it, indx);
            Euler[indx++] = u;
        }
    }
}

// Функция для вывода Эйлера обхода дерева
void printEulerTour(int root, int N)
{
    int index = 0;
    eulerTree(root, index);
    for (int i = 0; i < (2 * N - 1); i++)
        cout << Euler[i] << " ";
}

int main()
{
    int N = 5;
    add_edge(1, 5);
    add_edge(5, 4);
    add_edge(4, 2);

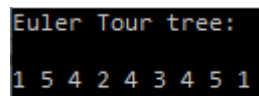
```

```

add_edge(4, 3);
add_edge(5, 1);
cout << "Euler Tour tree:";
cout << endl;
cout << endl;
// Здесь единица будет корнем
printEulerTour(1, N); printEulerTour(1, N);
cout << endl;
cout << endl;
cout << endl;
return 0;
}

```

На рисунке 3 приведен результат выполнения программы.



```

Euler Tour tree:
1 5 4 2 4 3 4 5 1

```

Рисунок 3 — результат выполнения программы

Анализ алгоритма

Определить сложность алгоритма достаточно легко.

Посещение узла занимает время $O(1)$. А именно, в каждом обходе мы тратим постоянное количество времени на каждом узле дерева во время прохождения, таким образом для n узлов время будет равно **$O(n)$** . Таким образом, время выполнения программы должно расти линейно в зависимости от количества узлов в дереве.

На рисунке 4 показан график зависимости времени выполнения программы (ось y) от количества узлов n (ось x) в дереве.

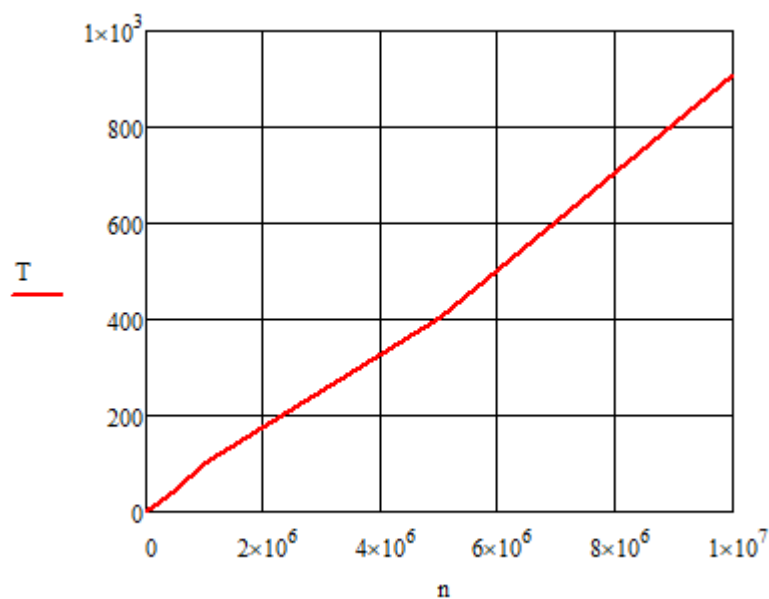


Рисунок 4 — Время выполнения программы

Заключение

В ходе выполнения работы был рассмотрен принцип работы алгоритма Эйлера обхода дерева, приведена его реализация на языке C++, проведен анализ сложности и времени выполнения алгоритма в зависимости от количества узлов в дереве.

Эйлеров обход дерева очень часто применяется в теории графов для решения многих задач. Благодаря этому алгоритму решаются такие задачи, как поиск элемента в глубину и в ширину, определение наименьшего предка двух узлов, подсчет количества ребер и т.д.

Список литературы

1. Michael T. Goodrich, Roberto Tamassia, David M. Mount. Data Structures and Algorithms in C++ 2nd Edition, 2011.

2. Викиконспекты (Источник из интернета)

https://neerc.ifmo.ru/wiki/index.php?title=Деревья_Эйлера_обхода#.D0.97.D0.B0.D0.B4.D0.B0.D1.87.D0.B0_.D0.BE_.D0.B4.D0.B8.D0.BD.D0.B0.D0.BC.D0.B8.D1.87.D0.B5.D1.81.D0.BA.D0.BE.D0.B9_.D1.81.D0.B2.D1.8F.D0.B7.D0.BD.D0.BE.D1.81.D1.82.D0.B8