Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Пояснительная записка к курсовой работе по теме «Графы»**

**по дисциплине**

**«Алгоритмы и структуры данных»**

**Вариант 34**

|  |  |
| --- | --- |
| Выполнил студент гр.9308: | Дементьев Д. |
| Проверил: | Колинько П.Г. |

Санкт-Петербург, 2020 г.

Оглавление

[​ Введение 3](#__RefHeading___Toc453_1055385205)

[​ 1. Задание 3](#__RefHeading___Toc455_1055385205)

[​ 2. Формализация задания 3](#__RefHeading___Toc457_1055385205)

[​ 3. Обоснование выбора способа представления графа в памяти ЭВМ 4](#__RefHeading___Toc618_2492656717)

[​ 4. Временная сложность функций 5](#__RefHeading___Toc461_1055385205)

[​ 5. Контрольные примеры 6](#__RefHeading___Toc463_1055385205)

[​ 6. Тестирование программы 8](#__RefHeading___Toc465_1055385205)

[​ Вывод 11](#__RefHeading___Toc467_1055385205)

[​ Список используемых источников 12](#__RefHeading___Toc469_1055385205)

[​ Приложение 1 (Исходный текст программы) 13](#__RefHeading___Toc471_1055385205)

## Введение

Исследование алгоритмов, реализуемых с помощью графов.

## 1. Задание

Проверка на изоморфизм произвольных корневых деревьев.

## 2. Формализация задания

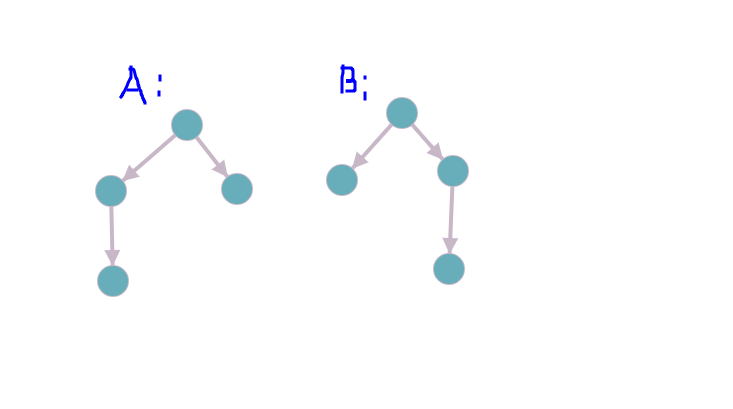
Дерево называется корневым, если оно ориентированно, и из какой-то вершины (называемой корнем) можно попасть во все остальные.

*Изоморфизм* графов G1 и G2 — это биекция (взаимно-однозначное соответствие) f из V(G1) в V(G2) такая, что ребро (uv) в графе G1 существует тогда и только тогда, когда в графе G2 существует ребро (f(u)f(v)). Иными словами, изоморфизм — это способ перенумеровать вершины графа G1 так, чтобы получился в точности граф G2. Соответственно, два графа называются изоморфными, если существует изоморфизм.

В задании рассматривается частный случай, когда оба графа являются деревьями. Понятно, что если у них разный размер, то они неизоморфны. Поэтому будем рассматривать корневые деревья с одинаковым количеством вершин .

Два дерева называются изоморфными, если они имеют одну структуру.

К примеру, эти два дерева будут изоморфны друг другу.

Рисунок 1: Пример изоморфных деревьев

Пусть каждое дерево имеет свой код, который создается следующим образом: если вершина является листом - ей присваивается код «10», а если нет, то ее кодом будет являться конкатенация кодов ее детей и приписанные в начало «1» и «0» в конец. Таким образом, пусть у вершины k есть два ребенка с кодами str1 и str2 соответственно, тогда код вершины k будет выглядеть следующим образом: k = 1 + str1 + str2 + 0. Получается код всего дерева будет содержаться в его корне. Можно заметить, что у двух изоморфных деревьев будет одинаковый код, если его отсортировать. Отсортированы кодом назовем код, полученный поэтапной сортировкой кодов, расположенных от листов к корню. Т.е. к примеру, мы имеем 3 кода: 1100, 10 и 110100, тогда в код текущей вершины сначала запишется 10, затем 1100 и уже после 110100 и мы получим: code = 1 + 10 + 1100 + 110100 + 0.

Для обхода деревьев и получения их кодов будем использовать поиск в глубину. Как только мы приходим в лист, присваиваем ему код «10», и возвращаем этот код как результат функции. Тогда для k — ой вершины мы получим, после обхода n ее детей — n кодов, которые нужно прост отсортировать и приписать «1» в начало и «0» в конец. Таким образом мы получили коды всех деревьев, которые осталось только сравнить.

Данный алгоритм носит название AHU, в честь трёх ученых Aho, Hopcroft и Ullman. Работает за время O(n\*logn), так как обход в глубину O(n), и быстрая сортировка O(n\*logn).

## **3. Обоснование выбора способа представления графа в памяти ЭВМ**

Граф в памяти представлен в виде списка смежности. На это есть свои причины:

Во-первых, алгоритм основывается на обходе графа. Для такого представления обход графа имеет линейную сложность O(n). Если же использовать матрицу смежности, то сложность будет O(n^2).

Во-вторых, данный способ представления удобен для графов, в которых количество ребер не очень велико. Максимальное количество вершин дерева, которое может быть в программе — 26, количество букв латинского алфавита. Тогда максимальное количество ребер может будет m = n-1 = 25. Получается, что для хранения такого дерева понадобится n+m памяти 26+25 = 51, что меньше, чем при использовании матрицы смежности, для которой необходимо n^2 памяти 26^2 = 676.

## 4. Временная сложность функций

1. **Ввод графа.**

Сложность: O(n+m), так как заполняем списоки смежности

1. **Генерация случайного графа.**

Сложность: O(n^2), так как для генерации графа матрица смежности заполняется случайными 1 и 0, после чего она преобразуется в списки смежности.

1. **Построение кода:**

Сложность: O(n\*logn), так как обход в глубину O(n), и быстрая сортировка O(n\*logn).

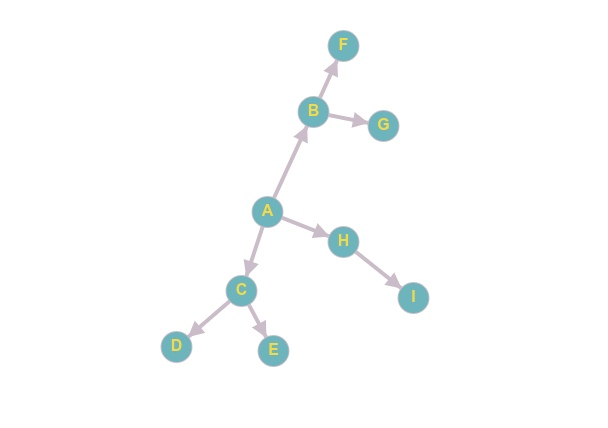
## 5. Контрольные примеры

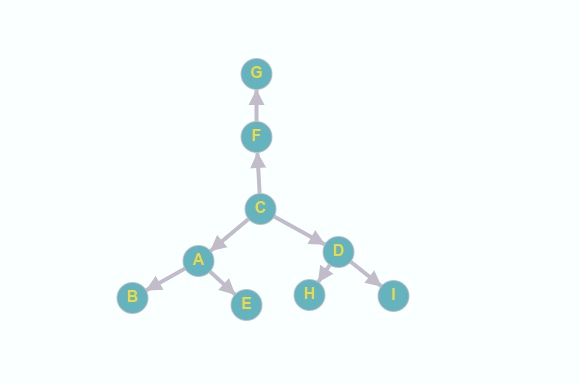
В таблице 1 представлены контрольные примеры, которые использовались для тестирования программы.

Таблица 1: Контрольные примеры

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Входные данные | | | | | Результат |
| Кол-во вершин | Корень | Списки смежности | Корень | Списки смежности |
| 1 | 9 | a | a: -bc----h-  b: -----fg--  c: ---de----  d: ---------  e: ---------  f: ---------  g: ---------  h: --------i  i: --------- | c | a: -b--e----  b: ---------  c: a--d-f---  d: -------hi  e: ---------  f: ------g--  g: ---------  h: ---------  i: --------- | Деревья изоморфны |
| 2 | 9 | f | a: --c----h-  b: a-----g--  c: ---de----  d: ---------  e: ---------  f: -b-------  g: ---------  h: --------i  i: --------- | g | a: -b--e----  b: ---------  c: a--d-----  d: -------hi  e: ---------  f: --c------  g: -----f---  h: ---------  i: --------- | Деревья не изоморфны |
| 3 | 4 | a | a: -bc-  b: ---d  c: ----  d: ---- | a | a: -bc-  b: ----  c: ---d  d: ---- | Деревья изоморфны |
| 4 | 3 | a | a: -bc  b: ---  c: --- | a | a: -b-  b: --c  c: --- | Деревья не изоморфны |
| 5 | 6 | a | a: -b-d--  b: --c-e-  c: ------  d: -----f  e: ------  f: ------ | a | a: -b-de-  b: --c--f  c: ------  d: ------  e: ------  f: ------ | Деревья не изоморфны |

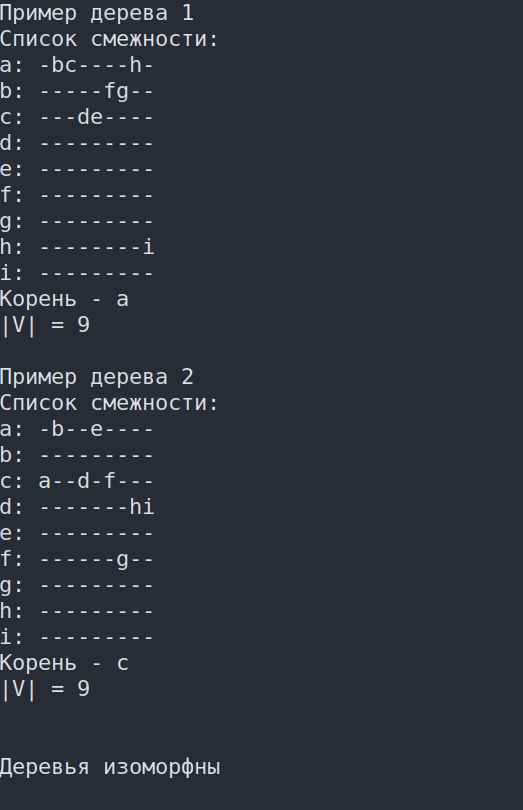
На рисунках 2 и 3 представлено графическое представление исходного графа и результата для примера 1. Данный пример используется в программе, как демонстрационный.

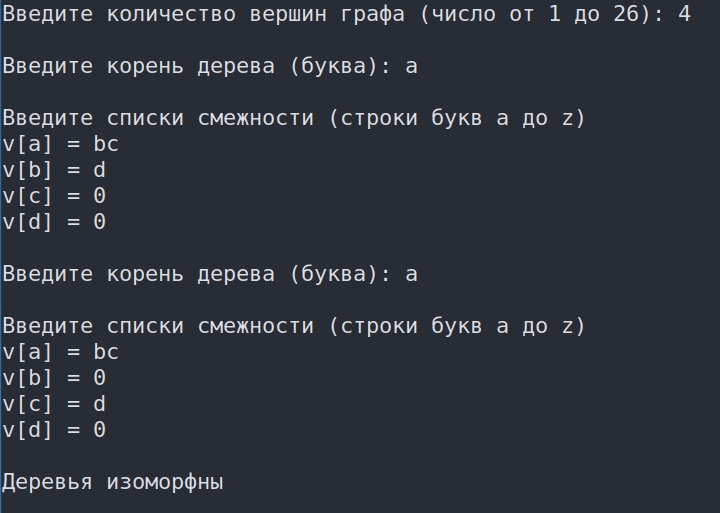
Рисунок 2: Визуализация первого дерева из примера 1

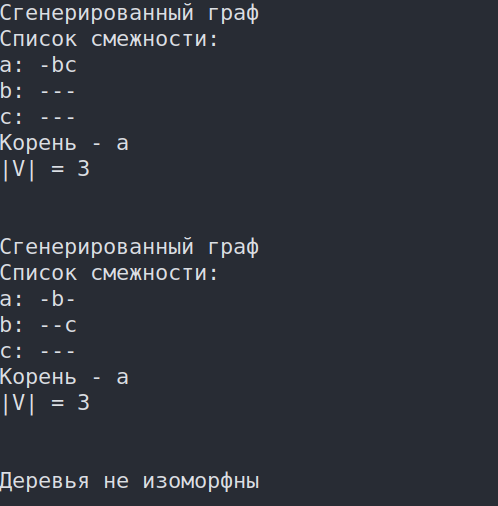
Рисунок 3: Визуализация второго дерева из примера 1

## 6. Тестирование программы

На скриншотах ниже представлены результаты прогона программы для различных тестов из таблицы 1.

Рисунок 4: Пример работы

Рисунок 5: Пример работы

Рисунок 6: Пример работы

## Вывод

* + - * 1. В ходе выполнения работы был исследован алгоритм проверки изоморфизма корневых деревьев. При тестировании программы с различными данными ошибки не были обнаружены.

На практике необходимость проверки изоморфизма деревьев возникает при решении задач хемоинформатики, математической или компьютерной химии, автоматизации проектирования электронных схем (верификация различных представлений электронной схемы) и оптимизации программ (выделение общих подвыражений).

## Список используемых источников

1. Колинько П.Г. Пользовательские структуры данных / Методические указания по дисциплине «Алгоритмы и структуры данных» - Санкт-Петербург: СПбГЭТУ «ЛЭТИ», 2020.

## Приложение 1 (Исходный текст программы)