Отчет по 7 лабораторной работе

По дисциплине «Типы и структуры данных»

Подготовил Жабин Дмитрий

Группа ИУ7-34Б

Вариант 2

**Цель работы:** реализовать алгоритмы обработки графовых структур: поиск различных путей, проверка связности, построение остовых деревьев минимальной стоимости.

**Задание**

Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных – на усмотрение программиста. Результат выдать в графической форме.

Определить, является ли связным заданный граф.

**Внешняя спецификация программы.**

**Входные данные:** имя текстового файла (строка); файл содержит количество вершин графа (натуральное число), а затем пары номеров смежных вершин графа (целые числа от 0 до n-1, где n - введенное количество вершин), введенные построчно.

**Выходные данные:** изображение графа, результат проверки его связности.

**Задача программы:**

Проверка связности графа, вывод графа.

**Способ обращения к программе:** Запуск программы в консоли с аргументом в виде имени файла с данными (./app.exe filename.txt).

**Аварийные ситуации**

1) Некорректный ввод имени файла.

2) Пустой файл в качестве аргумента программы.

3) Некорректный ввод количества/номеров вершин графа.

Во всех указанных случаях программа завершает работу с соответствующим сообщением.

**Структуры данных.**

Для представления графа в программе использовалась матрица смежности его вершин — динамический двумерный массив:

int \*\*matrix;

При обходе вершин понадобился массив с информацией о том, была посещена вершина или нет:

int visited[n];

**Функции**

int allocate\_matrix(int n, int m, int \*\*\*data); - функция выделения памяти под матрицу

void free\_data(int \*\*data, int n); - функция освобождения памяти из-под матрицы

int input\_graph(char \*name, int \*n, int \*\*\*matrix); - функция ввода графа

void init\_arr(int \*a, int len); - функция инициализации массива

int dfs(int u, int \*\*matrix, int n, int \*visited); - функция поиска в глубину

**Алгоритм**

Для проверки связности графа достаточно запустить поиск в глубину от любой из его вершин и сравнить количество посещенных вершин с общим числом вершин графа: если они совпадают, то граф связен, в противном случае - нет.

Поиск в глубину происходит следующим образом:

Текущая вершина помечается как посещенная. Перебираются все вершины графа, смежные с данной, если очередная вершина ещё не была посещена - для неё запускается поиск в глубину.

**Тесты**

|  |  |  |
| --- | --- | --- |
| Тест | Ввод пользователя | Результат |
| Некорректный ввод имени файла | wrong.txt (файл не существует) | Неверное имя файла! |
| Пустой файл | empty.txt (файл пуст) | Файл пуст! |
| Некорректный ввод количества вершин графа | 0/-10 | Некорректное количество вершин графа! |
| Некорректный ввод номеров вершин графа | 4  0 1  0 2  3 4 | Некорректный ввод ребер графа! |
| Корректный ввод всех характеристик | 4  0 1  0 2  3 2 | Заданный граф связен |
| Корректный ввод всех характеристик | 5  0 1  0 2  3 2 | Заданный граф не является связным |

**Оценка эффективности**

|  |  |  |
| --- | --- | --- |
| Количество вершин | Время работы  (в тактах процессора) | Объем занимаемой памяти (в байтах) |
| 5 | 852 | 140 |
| 10 | 2526 | 480 |
| 15 | 3223 | 1020 |

**Вывод**

В данной реализации был выбран алгоритм поиска в глубину, так как он отлично подходит для проверки связности графа (обхода всех вершин). Если V – количество вершин графа, а E – количество ребер, то алгоритм имеет асимптотическую сложность O(V+E). Для представления графа была использована матрица смежности его вершин, так как при поиске в глубину это очень удобно.

Разработанную программу можно использовать, например, для того чтобы определить, из каждого ли населенного пункта из определенного набора можно добраться в любой другой.

**Контрольные вопросы**

**1. Что такое граф?**

Граф – конечное множество вершин и соединяющих их ребер; G = <V, E>. Если пары Е (ребра) имеют направление, то граф называется ориентированным; если ребро имеет вес, то граф называется взвешенным.

**2. Как представляются графы в памяти?**

С помощью матрицы смежности или списков смежности.

**3. Какие операции возможны над графами?**

Обход вершин, поиск различных путей, исключение и включение вершин.

**4. Какие способы обхода графов существуют?**

Обход в ширину (BFS – Breadth First Search), обход в глубину (DFS – Depth First Search).

**5. Где используются графовые структуры?**

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

**6. Какие пути в графе Вы знаете?**

Простой путь, эйлеров путь, гамильтонов путь.

**7. Что такое каркасы графа?**

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (необязательно все) его рёбра.