



**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

Графы

Вариант 1

Студент Абдуллаев Ш. В.

Группа ИУ7-34Б

Название предприятия НУК ИУ МГТУ им. Н. Э. Баумана

Студент _____ Абдуллаев Ш. В.

Преподаватель _____ Силантьева А. В.

2024

Описание условия задачи

Задана система двусторонних дорог, где для любой пары городов есть соединяющий их путь. Найти город с минимальной суммой расстояний до остальных городов.

Описание входных данных

Программа принимает на вход текстовый файл, содержащий целочисленные значения. На основе этих данных создается граф в виде матрицы. Эта структура может служить входными данными для выполнения различных операций. Также программа принимает на вход целочисленное значение для добавления в граф.

Описание исходных данных

Программа представляет собой консольное приложение для графа. Основные возможности работы со структурами включают:

- Ввести связи между городами и дорогами из файла.
- Ввести связи между городами и дорогами вручную полностью.
- Ввести связи между городами и дорогами вручную координатно.
- Вывести связи между городами и дорогами.
- Найти город с минимальной суммой расстояний до остальных городов.

Имеются следующие ограничения:

- Ожидаемый формат данных: только целые числа. Ввод некорректных данных (например, букв или символов) завершает программу с сообщением об ошибке.
- Работа с пустыми структурами: некоторые операции (например, поиск) недоступны, если структуры пустые.

Допустимый ввод	Недопустимый ввод
Выберите команду: 1	Выберите команду: q
Введите количество ребер: 42	Введите количество ребер: abc

Введите количество городов: 15	Введите количество городов: 15abc
Введите вес ребра: 9	Введите вес ребра: 7x

Таблица 1. Примеры ввода

Описание результатов

1. Выйти из программы: Завершение работы программы с освобождением всех выделенных ресурсов. После этого программа завершает выполнение.
2. Ввести связи между городами и дорогами из файла: Эта команда позволяет загружать данные о связях между городами и дорогами из заранее подготовленного файла. Файл должен содержать информацию о городах и пропусках между ними.
3. Ввести связи между городами и дорогами вручную полностью: С помощью этой команды пользователь может вводить данные о связях между всеми городами и дорогами вручную. Это означает, что потребуется ввести каждую связь отдельно.
4. Ввести связи между городами и дорогами вручную координатно: Эта команда позволяет вводить связи между городами и дорогами, указывая точечные связи между городами.
5. Вывести связи между городами и дорогами: Данная команда позволяет визуализировать связи между городами и дорогами.
6. Найти город с минимальной суммой расстояний до остальных городов: Команда осуществляет поиск города, который имеет наименьшую сумму расстояний до всех остальных городов в сети. Это может быть полезно для определения наиболее центрального города в маршруте.

Описание задачи, реализуемой программой

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

Способ обращения к программе

Обращения к программе пользователем происходит с помощью вызова исполняемого файла (app.exe).

Описание возможных аварийных ситуаций и ошибок пользователя

- NO_DATA_ERROR: ошибка отсутствия файла данных. Возникает, если в программу не передан путь к файлу при запуске.
- INPUT_COMMAND_ERR: ошибка ввода команды. Возникает, если пользователь вводит некорректное значение при выборе команды из меню.

Описание внутренних структур данных

Программа содержит структуру, которая используется для хранения графа, представленную в Листинге 1.

```
#define MAX_CITIES 100  
typedef int graph_t[MAX_CITIES][MAX_CITIES];
```

Листинг 1. Структура graph_t

Рассмотрим структуру:

- MAX_CITIES: Константа, определяющая максимальное количество городов, которые могут быть представлены в графе. Это значение используется для задания размера массива.
- graph_t: Тип данных, представляющий граф в виде матрицы смежности. Это двумерный массив целых чисел размером MAX_CITIES x MAX_CITIES.
 - Каждый элемент массива graph_t[i][j] хранит стоимость (вес) пути из города i в город j.
 - Если путь между городами отсутствует, то значение равно INF.

Описание алгоритма

1. Инициализация системы и выделение памяти.
2. Основной цикл программы начинается. Пользователю предоставляется меню с различными опциями, и программа ожидает ввода выбора пользователя.
3. В зависимости от выбора пользователя, программа выполняет следующие действия:

- Ввести связи между городами и дорогами из файла.
- Ввести связи между городами и дорогами вручную полностью.
- Ввести связи между городами и дорогами вручную координатно.
- Вывести связи между городами и дорогами.
- Найти город с минимальной суммой расстояний до остальных городов.

Алгоритм находит город с минимальной суммой расстояний до всех остальных городов в графе, представленном в виде матрицы смежности. Сначала применяется алгоритм *Флойда-Уоршелла, чтобы вычислить кратчайшие расстояния между всеми парами городов, обновляя матрицу расстояний `dist`. Затем, для каждого города вычисляется сумма расстояний до всех других городов (игнорируя недостижимые узлы с расстоянием `INF`). Город с минимальной суммой расстояний считается оптимальным.

* Алгоритм Флойда-Уоршелла используется для нахождения кратчайших путей между всеми парами вершин взвешенного графа. Исходная матрица смежности копируется в матрицу расстояний `dist`, где `dist[i][j]` представляет длину пути из вершины `i` в вершину `j`. Алгоритм итеративно проверяет, можно ли улучшить путь между двумя вершинами `i` и `j` через промежуточную вершину `k`. Если путь через `k` короче текущего, значение `dist[i][j]` обновляется. Таким образом, по завершении трех вложенных циклов матрица `dist` содержит длины кратчайших путей между всеми парами вершин.

4. После выполнения каждой операции программа возвращает пользователя в главное меню, где он может выбрать следующее действие.

Набор тестов

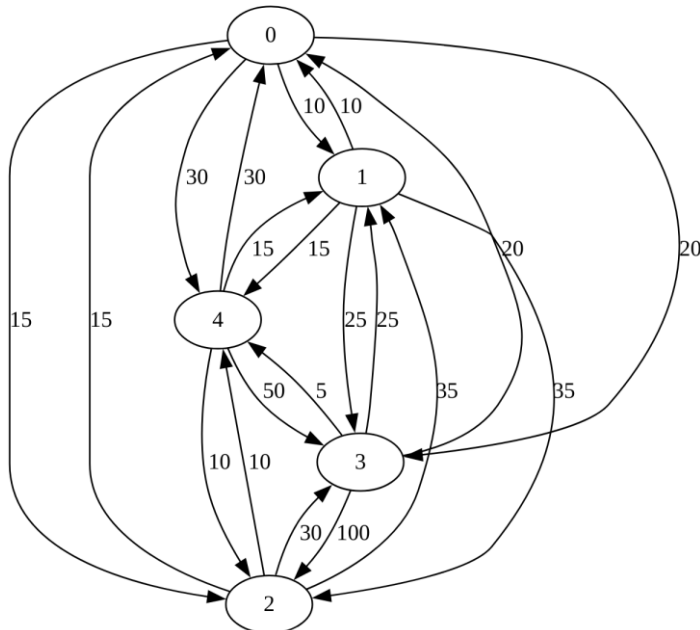
Описание	Результат
Ввести связи между городами и дорогами из файла	Успешно считаны города
Ввести связи между городами и дорогами вручную полностью 5 0 10 15 20 30 10 0 100 25 15 15 35 0 30 10 20 25 30 0 5 30 15 10 50 0	Успешно считаны города
Вывести связи между городами и дорогами	<p>Граф экспортирован в файл graph.dot</p> 
Найти город с минимальной суммой расстояний до остальных городов	<p>Город с минимальной суммой расстояний:</p> <p>4</p>

Таблица 2. Набор позитивных тестов

Описание	Результат
Попытка не передавать датасет	Возврат ошибки NO_DATA_ERROR
Попытка ввести символы вместо команды	Возврат ошибки INPUT_COMMAND_ERR

Таблица 3. Набор негативных тестов

Контрольные вопросы

1) Что такое граф?

Граф — это структура данных, состоящая из множества вершин (узлов) и рёбер (связей между вершинами).

2) Как представляются графы в памяти?

- Список смежности: Для каждой вершины хранится список соседних вершин.
- Матрица смежности: Двумерный массив, где элемент $[i][j]$ равен весу ребра между вершинами i и j или 1 – если связь есть, иначе – 0.
- Матрица инцидентности: двумерная матрица, где строки — вершины, а столбцы — рёбра, 1 – если вершина инцидентна ребру, иначе – 0.

3) Какие операции возможны над графами?

- Добавление/удаление вершин.
- Добавление/удаление рёбер.
- Поиск пути между вершинами.
- Проверка связности графа.
- Поиск кратчайшего пути.
- Поиск минимального остовного дерева (каркаса).
- Обход графа.

4) Какие способы обхода графов существуют?

- Поиск в глубину (DFS, Depth-First Search)
- Поиск в ширину (BFS, Breadth-First Search)

5) Где используются графовые структуры?

В случаях, когда требуется представить попарные связи между объектами, например:

- Компьютерные сети
- Навигация (построение путей)
- Алгоритмы поиска
- Базы данных (зависимости между записями) и т.д.

6) Какие пути в графе Вы знаете?

- Простой путь – путь, в котором все вершины уникальны.
- Эйлеров путь – проходит через каждое ребро графа ровно один раз.
- Эйлеров цикл – Эйлеров путь, который начинается и заканчивается в одной вершине.
- Гамильтонов путь – проходит через каждую вершину ровно один раз.
- Гамильтонов цикл – Гамильтонов путь с возвращением в начальную вершину.
- Кратчайший путь – путь с минимальной суммарной стоимостью рёбер.

7) Что такое каркасы графа?

Каркас графа (минимальное остовное дерево) — это подграф, соединяющий все вершины исходного графа минимальным числом рёбер без образования циклов.