

*Отчет по дисциплине:*

**Типы и структуры данных**

*Лабораторная работа:*

**“ГРАФЫ”**

Выполнил студент: Сироткина П.Ю.

Группа: ИУ7-36Б

Преподаватель: Силантьева А.В.

**Москва, 2020 г.**

1. **Условие задачи**

Заданы две системы двухсторонних дорог с одним и тем же множеством городов (железные и шоссейные дороги). Найти минимальный по длине путь из города A в город B, который может проходить как по железной, так и по шоссейной дорогам, и места пересадок с одного вида транспорта на другой на этом пути.

1. **Описание ТЗ**

*А) Входные и выходные данные:*

Входные данные:

Целое число от 1 до 4, представляющее собой пункт меню программы.

В зависимости от выбранного пункта может потребоваться ввод дополнительных данных, информация о которых будет сообщена пользователю (например, указание имени файла или название городов назначения/отправки).

Примечание:

Организация входных данных в текстовом файле: первой строкой вводится количество городов, дальше вводится название города, каждое - на новой строке. Затем вводятся тройки: пункт назначения (на отдельной строке), пункт назначения (на отдельной строке), длина пути через пробел 0, если пути соединяет шоссейная дорога, и 1 иначе).

Выходные данные:

Командно-зависимо:

* Сообщение об успешности ввода данных
* Сообщение об успешности выполнения алгоритма поиска кратчайшего пути и вывод соответствующей информации с пояснениями на экран
* Вывод информации о программе на экран

*Б) Задачи, реализуемые программой:*

Программа способна решать следующие задачи, которые запускаются посредством выбора соответствующего пункта меню:

**1) Ввести данные в систему из текстового файла -** ввод имени файла для указания системе, с каким файлом необходимо работать.

**2) Найти минимальный путь из одного города в другой -** пользователю необходимо ввести пункты отправки и назначения, затем выводится соответствующая информация и указание, как получить изображение кратчайшего пути. (Примечание: красным цветом на изображении помечены вершины, через которые необходимо пройти. Зелеными ребрами помечены Ж/Д дороги, синим - шоссейные. На каждом ребре отмечена длина пути).

**3) Справка о программе -** вывод справки о программе на экран

**4) Выход -** завершение работы программы.

*В) Способы обращения к программе:*

Пользователь работает с программой посредством меню, команды которого описаны выше. Информация о необходимых дополнительных действиях будет своевременно и подробно выведена на экран (например, о том, как получить изображение графа в формате PNG).

1. **Описание внутренних структур данных**

Выбирая между связным линейным списком и матрицей смежности был сделан выбор в пользу второго варианта представления графа. С одной стороны, может показаться, что хранение матриц смежности занимает больше памяти, чем связный список, особенно на больших графах (т.к. он дороги двусторонние -> граф неориентированный -> матрицы смежности симметрична относительно главной диагонали -> неэффективность), однако это полностью нивелируется тем фактом, что алгоритм поиска кратчайшего пути для матриц будет работать существенно быстрее, т.к. здесь доступна прямая индексация. Это особенно актуально для восстановления пути по конечному пункту и для поиска “соседей вершины”.

Также стоит отметить тот факт, что все названия можно поместить в массив, и, таким образом, поставить в соответствие каждому названию целое число (индекс массива), тогда преимущества использования матриц только возрастают, т.к. можно оперировать не словами, а целыми числами.

Специальных типов данных не заводилось - использовалось 2 обычных матрицы смежности для каждого типа дороги, а также строковый массив для названий городов.

1. **Описание алгоритма**

Для реализации поиска кратчайшего пути между двумя вершинами графа был использован алгоритм Дейкстры, т.к. алгоритм работает только с графом, вес каждого ребра которого является неотрицательным, а это как раз соответствует задаче - длина пути не может быть отрицательной.

Алгоритмы, в которых учитываются и отрицательные веса, лишь усложнили бы поиск, поэтому алгоритм Дейкстры - оптимальный.

*Описание:*

Сначала выбирается начальная вершина (применительно к решаемой задаче - пункт отправки). В соответствии с алгоритмом, будут рассчитаны кратчайшие пути от этой вершины до всех остальных, затем необходимо лишь выбрать нужную вершину (пункт назначения).

Шаг алгоритма:

Если все вершины посещены или граф несвязен, алгоритм завершается.

Иначе, относительно текущей вершины (назовем ее ***u***) выбирается ее “сосед”, расстояние до которого является минимальным (рассматриваются все возможные пути, в которых эта вершина является предпоследней).

Для каждого соседа вершины ***u***, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки ***u*** и длины ребра, соединяющего ***u*** с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину ***u*** как посещённую и повторим [шаг алгоритма](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B#%D0%A8%D0%B0%D0%B3).

1. **Тестирование программы**

|  |  |
| --- | --- |
| Действие | Результат |
| Выбор пункта 1  Ввод empty.txt (Пустой файл)  Выбор пункта 2 | Сообщение “Произошла ошибка во время считывания информации о городах.” |
| Выбор пункта 1  Выбор wrong.txt (Количество введенных городов не соответствует указанному)  Выбор пункта 2 | Сообщение “Произошла ошибка во время считывания информации о городах.” |
| Выбор пункта 1  Выбор wrong.txt (Ошибка в записи связей) | Сообщение: “Произошла ошибка во время считывания информации о путях.  ” |
| Выбор пункта 1  Выбор корректного файла  Выбор 2  Выбор несуществующего города | Сообщение:  “Город с таким названием не найден в системе.  ” |
| Выбор пункта 1  Выбор корректного файла (граф несвязен)  Выбор 2 | Сообщение: “Пути между этими городами не существует.” |
| Выбор пункта 1  Выбор корректного файла  Выбор 2  Выбор 1 6 | Красным помечен оптимальный путь. Зеленые ребра - Ж/Д, синие - шоссе. Как видим, кратчайший путь из 1 в 6 действительно проходит через вершины 1, 3, 6 и составляет 11 единиц. |
| Выбор пункта 1  Выбор корректного файла  Выбор 2  Выбор 1 5 | Действительно, оптимальный путь из вершины 1 в вершину 5 составляет 18 единиц и проходит через вершины 1, 2, 4, 5 с 2 пересадками. |

1. **Ответы на контрольные вопросы**

*1. Что такое граф?*

Граф – это конечное множество вершин и ребер, соединяющих их, т. е.:

G = < V,E >,

где V – конечное непустое множество вершин; Е – множество ребер (пар вершин).

*2. Как представляются графы в памяти?*

С помощью матрицы смежности либо связного линейного списка.

*3. Какие операции возможны над графами?*

* поиск кратчайшего пути от одной вершины к другой (если он есть);
* поиск кратчайшего пути от одной вершины ко всем другим;
* поиск кратчайших путей между всеми вершинами;
* поиск эйлерова пути (если он есть);
* поиск гамильтонова пути (если он есть).

*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?*

Графы нашли применение практически во всех отраслях научных знаний: физике, биологии, химии, математике, истории, лингвистике, социальных науках, технике и т.п. Наибольшей популярностью теоретико-графовые модели используются при исследовании коммуникационных сетей, систем информатики, химических и генетических структур, электрических цепей и других систем сетевой структуры.

Прежде всего, графовые структуры используются в задачах, в которых необходимо установить некоторые связи, необязательно иерархические.

*6. Какие пути в графе Вы знаете?*

Эйлеров путь, Гамильтонов путь, простой путь, замкнутый путь (контур или цикл), сложный путь.

*7. Что такое каркасы графа?*

Каркасом, или остовным деревом для графа называется связный подграф этого графа, содержащий все вершины графа и не имеющий циклов. Количество ребер в каркасе связного графа всегда на единицу меньше количества вершин графа.