МГТУ им. Н. Э. Баумана

Отчет по лабораторной работе №7 по предмету «Типы и структуры данных»
«Графы»

Выполнила Параскун София,
ИУ7-34Б
Проверила
Никульшина Т. А.

Москва, 2020 г.

1. Описание условия задачи

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

Задана система двусторонних дорог. Для каждой пары городов найти длину кратчайшего пути между ними.

2. Описание ТЗ, включающего внешнюю спецификацию

a) описание исходных данных и результатов

Основная программа получает на вход выбранный пользователем пункт меню (целое число). В соответствии с ним производятся операции моделирования процесса обслуживания очереди или вывода состояния.

Программа получает от пользователя информацию о ребрах графа в формате «вершина1 вершина2 вес_ребра», либо использует заранее заготовленный граф для дальнейшей обработки.

В зависимости от выбранного режима результатом являются либо матрица кратчайших путей между городами, либо визуальное представление введенного графа в формате png.

b) Способ обращения к программе

Программа «app.exe» запускается через консоль, после чего можно увидеть меню программы, состоящее из 5 пунктов:
1 — Ввести граф,
2 — Выбрать заготовленный граф,
3 — Показать граф,
4 — Найти кратчашие пути,
0 — Выход из программы.
Все вводы и выводы также осуществляются в консоли.

c) Описание возможных аварийных ситуаций и ошибок пользователя

№ ситуации	Аварийная ситуация	Реакция программы
1	Ввод некорректного пункта меню	Сообщение «Its end of working the program!», завершение

		программы
2	Ввод некорректного кол-ва вершин графа	Сообщение «Wrong size!», возврат в главное меню
3	Ввод некорретного ребра (в т.ч. номер вершины 0 и меньше, больше кол-ва вершин, вес ребра меньше или равен 0)	Сообщение «Wrong input! Try again.», возврат в главное меню

3. Описание внутренних СД

```
struct matrix_graph
{
    int size;
    int *adjacence_matrix;
};
```

Единственной используемой структурой является matrix_graph, в которой size - количество вершин графа, adjacenece_matrix – матрица смежности.

4. Описание алгоритма

Поиск кратчайшего пути между каждой парой вершин графа был реализован алгоритмам Дейкстры. Было совершено n проходов, где n – количество вершин графа.

Метка текущей вершины приравнивается к 0, остальные к наибольшему числу — INT_MAX. Изначально все вершины графа считаются непосещенными. Как только мы обойдем все вершины, обработка графа завершается.

Из еще непосещенных вершин выбираем ту, которая имеет наименьшую метку, после чего рассматриваем сумму метки данной вершины и ее соседей. В случае, если сумма меньше значения метки соседа, метку заменяем на соответствующее значение, текущую вершину отмечаем посещенной и переходим к выбранному соседу.

5. Описание используемых функций

В используемых ниже функциях matrix_graph или graph – структура хранения графа.

*Void input_graph(struct matrix_graph *graph)* — осуществляет ввод ребер графа пользователем.

*Void deafult_graph(struct matrix_graph *graph)* – осуществляет заполнение графа заранее предусмотренными значениями.

Void output_graph(struct matrix_graph graph) – вывод визуально формы графа в формате png.

Void find_min(struct matrix_graph matrix_graph) – собирательная функция для find_matrix и out_res.

Struct matrix_graph *find_matrix(struct matrix_graph matrix_graph) – осуществляет поиск кратчайших путей и заполнение результирующей матрицы.

*Void out_res(struct matrix_graph *res)* – вывод матрицы кратчайших путей между городами.

6. Выводы по проделанной работе

В ходе этой работы удалось реализовать хранение графа в форме матрицы смежностей. Была выбрана именно эта структура так как в алгоритме обработки графа было удобнее проводить индексацию, а не каждый раз проходить по всем элементам списка смежностей. В таком же формате мы хранили и матрицы-результат для полученной задачи.

Поставленная задача — поиск кратчайшего пути между любой парой двух вершин графа — была реализована алгоритмом Дейкстры, который значительно эффективнее, чем поиск всех возможных путей, по количеству обходов.

Помимо всего этого была осуществлена работа с библиотекой GraphViz для вывода результата в графической форме.

Ответы на контрольные вопросы

1. Что такое граф?

Граф — конечное множество вершин и ребер соединяющих их, т. е. G = <V, E>, где G – граф, V – конечное непустое множество вершин, E – множество ребер (пар вершин).

2. Как представляются графы в памяти?

Один из видов представления графов — матрица смежности B(n*n).В этой матрице элемент b[i,j] = 1, если ребро, связывающее вершины Vi и Vj существуют и b[i,j] = 0, если ребра нет. У неориентированных графов матрица смежности всегда симметрична.
Еще одним способом представления в памяти является список смежности. Он содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с этой вершиной. Каждый элемент списка является записью, содержащей данную вершину и указатель на следующую запись в списке. Входы в списки смежностей для каждой вершины графа хранятся в массиве.

3. Какие операции возможны над графами?

Основные операции:
-добавление/удаление вершин;
-добавление/удаление ребер;
-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?

Граф можно обойти двумя способами — в глубину и в ширину.
Поиск в глубину (depth first search, DFS), при котором, начиная с произвольной вершины v0, ищется ближайшая смежная вершина v, для которой, в свою очередь, осуществляется поиск в глубину (т.е. снова ищется ближайшая, смежная с ней вершина) до тех пор, пока не встретится ранее просмотренная вершина, или не закончится список смежности вершины v (то есть вершина полностью обработана). Если нет новых вершин, смежных с v, то вершина v считается использованной, идет возврат в вершину, из которой попали в вершину v, и

процесс продолжается до тех пор, пока не получим v = v0. Иными словами, поиск в глубину из вершины v основан на поиске в глубину из всех новых вершин, смежных с вершиной v.
Путь, полученный методом поиска в глубину, в общем случае не является кратчайшим путем из вершины v в вершину u. Это общий недостаток поиска в глубину.
Указанного недостатка лишен другой метод обхода графа – поиск в ширину (breadth first search, BFS). Обработка вершины v осуществляется путем просмотра сразу всех новых соседей этой вершины. При этом полученный путь является кратчайшим путем из одной вершины в другую

5. Где используются графовые структуры?

Самой распространенной областью применения графовых структур является построение коммуникационных линий между городами.

6. Какие пути в графе вы знаете?

Эйлеров путь — произвольный путь графа, проходящий через каждое ребро точно один раз.
Непростой путь - произвольный путь графа, проходящий через каждую вершину неоднократно.
Гамильтонов путь — путь в графе, проходящий в точности один раз через каждую вершину графа (а не каждое ребро).

7. Что такое каркасы графа?

Каркас графа — дерево, содержащее все вершины графа. Стоимостью этого дерева является сумма всех стоимостей ребер.