Отчёт

1. **Дисциплина: Типы и структуры данных**
2. **Лабораторная работа №7**
3. **по теме: «Графы»**
4. Работу выполнил:
5. студент группы ИУ7-32Б
6. Иванов Всеволод

# Описание условия задачи

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

# Описание технического задания

Программа для обработки информации о городах, и соединяющих их дорогах, хранящейся в текстовом файле в виде рёбер графа. Программа должна построить граф из исходных данных и найти путь, наименьший по сумме расстояния и стоимости, между двумя указанными пользователем городами.

Область применения – текстовые файлы с информацией о дорогах

Сроки выполнения – 2 недели

Основой для разработки программы послужила потребность в реализации методов поиска путей в графах. Выполняемой задачей был поставлен поиск пути между двумя городами, с минимальной величиной S+P, где S – сумма длин дорог пути, P – сумма пошлин проезжаемых дорог.

Программа должна:

* Содержать указание формата и диапазона данных при вводе
* Содержать указание операций, производимых программой
* Контролировать правильность ввода данных
* Иметь пояснения при выводе результата
* Обеспечить вывод графа в графическом виде

# Этап постановки

Исходными данными является информация о дорогах, хранящаяся в текстовом файле в формате:

Город Город Длина дороги Стоимость пути.

Результатом является граф в графическом формате с отмеченным искомым путём.

Обращение к программе происходит путём выбора входного файла и городов через консоль

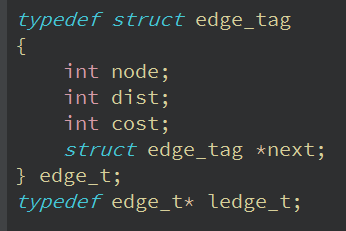
Возможными ошибками являются:

* Ввод несуществующего города
* Отсутствие входного файла
* Пустой файл или некорректный формат входного файла:
  + Символы в полях расстояния и стоимости
  + Неполная строка
  + Повторяющиеся рёбра

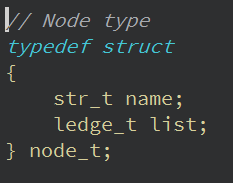
Особых требований для технических средств, аппаратной совместимости, программных средств не имеется.

# Описание СД

Хранение графа реализовано при помощи списков смежных вершин

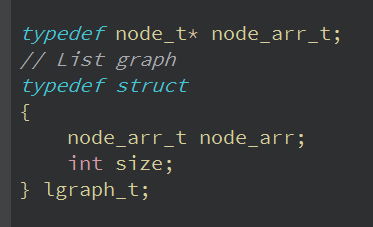
Для хранения смежного ребра была создана структура edge\_t:

* node (int) – номер смежного ребра в графе
* dist (int) – расстояние до города
* cost (int) – стоимость проезда до города
* next (указатель на edge\_t) – следующее ребро в списке смежности

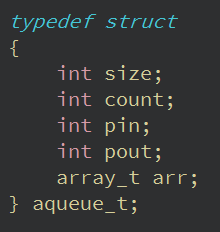
Для хранения вершин была создана структура node\_t:

* name (str) – имя вершины
* list (указатель на edge\_t) – список ребер для данной вершины

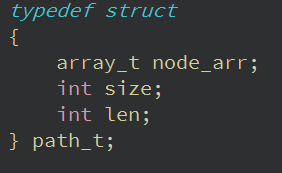
Для хранения графа была создана структура lgraph\_t:

* node\_arr (указатель на node\_t) – динамический массив вершин графа
* size(int) – количество рёбер

Также была использована очередь, представленная структурой aqueue\_t:

* size (int) – размер выделенной памяти под очередь
* pin и pout (int) – индекс начала и конца очереди в массиве
* count (int) – счётчик количества элементов
* arr (указатель на int) – массив элементов очереди (вершин)

Для хранения найденного пути использована структура path\_t:

* node\_arr (указатель на int) – массив вершин, состоящих в пути
* len (int) – количество вершин в пути
* size (int) – размер массива

# Описание алгоритма

# Сначала из файла считываются рёбра, из которых составляется массив вершин графа. После, в повторном считывании, к каждой вершине добавляются смежные ей рёбра

# После выбора двух городов, производится поиск пути между ними. Поиск основан на алгоритме Дейкстры:

# Создаётся массив расстояний всех вершин, изначально заполненный 0 (кроме стартовой вершины)

* Создаётся массив вершин, через которые к i-ой вершине пролегает наикратчайший путь

# Стартовая вершина добавляется в очередь. После чего до момента опустошения очереди выполняется следующее:

* + Подбирается очередной узел из очереди, исследуются все смежные ему вершины. В случае, если путь через данный узел короче уже найденного, происходит добавление смежной вершины в очередь и обновление расстояния до него и вершины, через которого оно достигается
* Происходит составление пути от конечной вершины к начальной при помощи массива вершин.

# Набор тестов

Негативные тесты

|  |  |  |
| --- | --- | --- |
| Ввод | Проверяемый случай | Вывод |
| da.exe | Некорректный или несуществующий входной файл | Ошибка: файл нечитаемый |
| data.txt:  Mos Kiev 1a2 2 | Некорректное содержание файла | Ошибка: неправильный формат файла |
| data.txt:  Mos Kiev | Некорректное содержание файла | Ошибка: неправильный формат файла |
| data.txt:  Mos Kiev 12 2  Mos Kiev 2 21 | Повторяющиеся дороги в файле | Ошибка: 2 и более дорог между городами |
| data.txt:  Mos Kiev 12 2  \*консоль\*  Moscow  Kiev | Нахождение пути от несуществующего города | Ошибка: города нет в графе |

Тепличные тесты

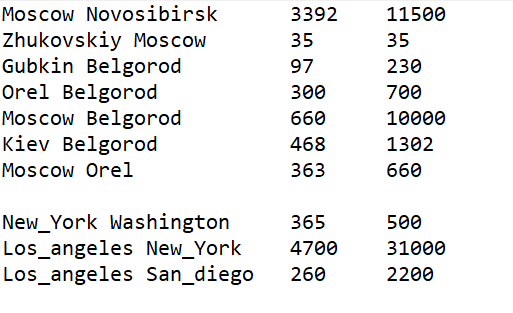
|  |  |  |
| --- | --- | --- |
| Ввод | Проверяемый случай | Вывод |
| data.txt:  Moscow Kiev 12 2  \*консоль\*  Moscow  Kiev | Поиск пути между городами | Вывод графа с отмеченным путём |

Граничные тесты

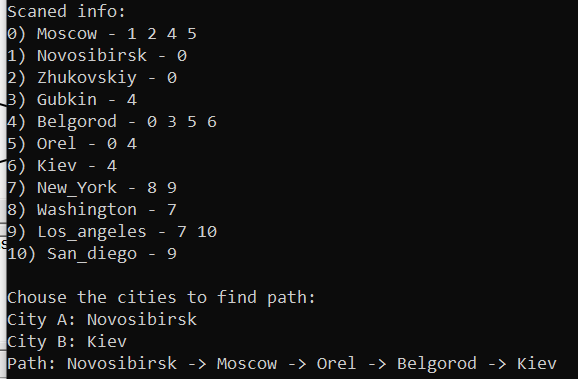
|  |  |  |
| --- | --- | --- |
| Ввод | Проверяемый случай | Вывод |
| data.txt:  Mos Kiev 12 2  \*консоль\*  Kiev  Kiev | Нулевой путь | Граф без отмеченных дорог пути |

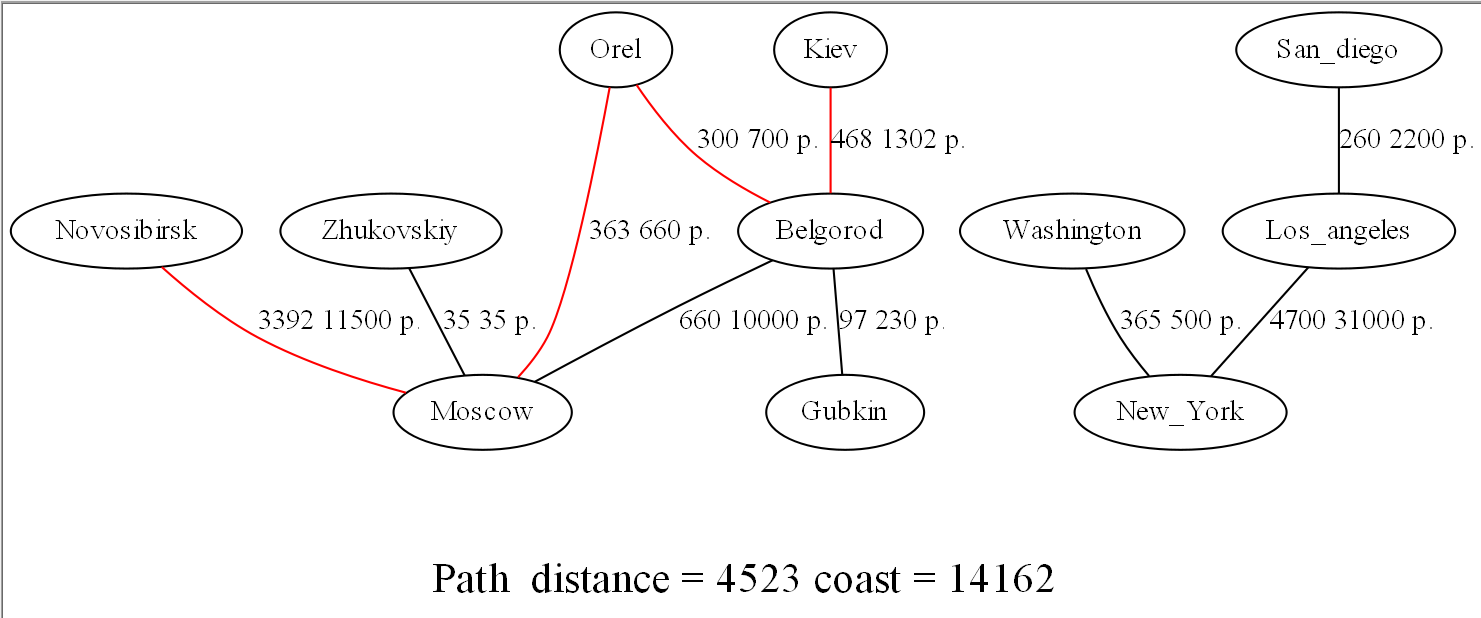
# Пример работы программы

Для входного файла data.txt:



И выбранного пути строится граф и путь:





# Ответы на вопросы

1. Что такое граф?

Граф – это конечное множество вершин и рёбер, соединяющих эти вершины

2. Как представляются графы в памяти?

Граф из n вершин может быть представлен в памяти матрицей смежностей A n\*n (в которой A[i][j] = 1, если ребро i – j существует, и 0 иначе) или при помощи списка смежности, содержащего все вершины, с которыми данная вершина связана рёбрами

3. Какие операции возможны над графами?

Для графов возможны следующие операции:

* 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 и гамильтонова пути

4. Какие способы обхода графов существуют?

Для графов существует обход в глубину и в ширину. Оба обхода осуществляются за счёт исследования рёбер от уже открытых вершин для нахождения ещё не посещённых. Обход в глубину исследует новые вершины от открытых в порядке LIFO, обход в ширину в порядке FIFO

5. Где используются графовые структуры?

Графы могут применяться при построении маршрутов на картах, составлении различных логистических сетей в городах

6. Какие пути в графе Вы знаете?

Эйлеров путь – путь, проходящий через все рёбра графа ровно один раз

Гамильтонов путь – путь, проходящий через все вершины графа ровно один раз

7. Что такое каркасы графа?

Каркасом графа называется его остовный подграф, который не имеет циклов. Следовательно, каркас связанного графа будет являться деревом

# Выводы по проделанной работе

В процессе работы было определено, что в случае, когда вершина связана с большинством других вершин графа, предпочтительнее использовать хранение графа при помощи матрицы смежности. В случае же, когда вершины в графе имеют меньшее количество рёбер, лучше себя проявляет хранение в форме списка смежных вершин