Лабораторная работа №7 по дисциплине

“Типы и структуры данных”

Сорокин Антон ИУ7-32Б

Номер по списку – 24

Вариант 12

***Условие задачи***

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.

Задана система двусторонних дорог. Найти множество городов, расстояние от которых до выделенного города (столицы) больше, чем Т.

***Техническое задание***

*Исходные данные*

На вход программе подаётся текстовый файл с информацией о графе: первое число – количество вершин, затем идут тройки чисел, характеризующие рёбра, в формате “<вершина1> <вершина2> <вес>”; номер вершины, относительно которой вести расчёт («столица»), минимальное расстояние T.

*Результат*

Графическое изображение графа в формате png, на котором:

* красным цветом выделена «столица»;
* зелёным цветом выделены города искомого множества;
* белым цветом выделены остальные вершины.
* рёбра имеют подписи, означающие их веса.

*Описание задачи*

Программа считывает из файла информацию о графе, записывает его в структуру, определяет множество городов, кратчайшие расстояние от которых до выбранного пользователем города больше, чем введённое пользователем значение T.

*Аварийное завершение работы программы*

Программа завершается аварийно, если

* не подан файл в качестве аргумента на вход программе;
* файл невозможно открыть;
* файл пустой;
* в файле некорректные данные (символы, вещественные числа и т. д.):
  + символы;
  + вещественные числа;
  + отрицательные числа;
  + числа, большие количества вершин;
* случилась ошибка выделения памяти при чтении из файла в структуру.

***Обращение к программе***

Исполняемый файл app.exe создается путем автоматической сборки проекта с помощью файла makefile. Для выполнения работы следует запустить данный исполняемый файл с указанием имени файла с исходными данными в качестве аргумента. Для запуска модуля анализа работы функций по времени и объёму памяти собирается исполняемый файл test.exe. Для выполнения его работы дополнительных аргументов указывать не требуется.

***Алгоритм***

Для определения кратчайших расстояний от указанной вершины до остальных применяется алгоритм Дейкстра:

* Создать массив кратчайших расстояний по количеству вершин.
* Создать массив, который будет отвечать за то, пройдена ли вершина или нет.
* Проинициализировать все значения массива кратчайших расстояний как -1, кроме вершины, расстояния до которой высчитываются – для этой вершины проинициализировать как 0.
* В цикле до тех пор, как будут пройдены все вершины:
  + Определить вершину A с минимальным расстоянием до рассматриваемой вершины из числа не пройденных смежных ей.
  + Определить расстояние до рассматриваемой вершины через вершину A от всех остальных вершин.
  + Если текущее расстояние в массиве кратчайших до заданной больше найденного в предыдущем действии, записать найденное значение в массив.
  + Пометить вершину A как пройденную.

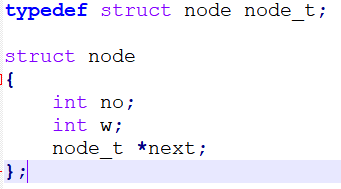
Данный алгоритм является более подходящим, в отличие от алгоритма Флойда - Уоршелла, поскольку в данной задаче отрицательные веса не рассматриваются и сложность алгоритма в худшем случае составит O(n2 + m), где n – количество вершин, m – количество рёбер (сложность алгогитма Флойда – Уоршелла составляет O(n3)).

***Тестовые данные***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Граф | Верш. | Расст. | Выход | Класс |
| 0 | - | - | Сообщение об ошибке | Некорректное количество вершин  (ноль) |
| -1 | - | - | Сообщение об ошибке | Некорректное количество вершин  (отрицательное) |
| a | - | - | Сообщение об ошибке | Некорректное количество вершин (символ) |
| 2  a 2 4 | - | - | Сообщение об ошибке | Некорректная вершина (символ) |
| 2  -1 2 3 | - | - | Сообщение об ошибке | Некорректная вершина (отрицательное) |
| 2  0 2 3 | - | - | Сообщение об ошибке | Некорректная вершина  (ноль) |
| 2  3 2 3 | - | - | Сообщение об ошибке | Некорректная вершина  (номер больше кол-ва) |
| 1  1 1 1 | 2 | - | Сообщение об ошибке | Несуществующая вершина (больше кол-ва) |
| 1  1 1 1 | 1 | 0 | Нет таких вершин! | Граф из одной вершины |
| 2  1 2 1 | 1 | 2 | Нет таких вершин! | Граф из двух вершин, расстояние больше |
| 4  1 2 4  1 3 5  2 3 4  2 4 5  3 4 1 | 1 | -1 | 2 3 4 | Расстояние неположительное, все вершины достижимы |
| 5  1 2 4  1 3 5  2 3 4  2 4 5  3 4 1 | 1 | -1 | 2 3 4 | Расстояние неположительное, есть недостижимые верщины |
| 5  1 2 4  1 3 5  2 3 4  2 4 5  3 4 2 | 1 | 1 | 2 3 4 | Расстояние положительное, но меньше всех весов графа |
| 5  1 2 4  1 3 5  2 3 4  2 4 5  3 4 1 | 1 | 5 | 4 | Расстояние положительно, одна вершина в решении |

***Внутренние структуры данных***

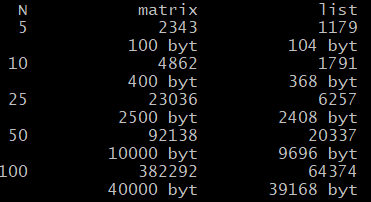
Для хранения графа используется массив списков смежности, элементы которого представляют собой структуру из трёх полей: номер вершины, с которым связана данная; вес ребра; указатель на следующий элемент в списке.



Удобнее использовать списки смежности, поскольку по алгоритму Дейкстра требуется определять минимальные расстояния до смежных вершин. В списках хранятся только информация по рёбрам со смежными вершинами, в то время как в матрице смежности имеются «пустые» элементы – значения которых говорят о том, что ребра нет. Из-за дополнительной проверки на существование ребра затрачивается лишнее время. Вдобавок ко всему матрица будет проигрывать по памяти при её разреженности: если количество рёбер значительно меньше количества вершин.

***Оценка эффективности***

*Сравнение времени и памяти использования матриц и списков смежности*



Исходя из полученных данных, можно сделать вывод о том, что для хранения графа для данной задачи эффективнее использовать списки смежности. По памяти списки незначительно выигрывают, но при этом они выигрывают по времени у матрицы в 4-6 раз.

***Вывод***

При реализации стека эффективнее использовать статический массив и по времени, и по памяти. Однако, если неизвестно, каким может быть максимально допустимое количество элементов, или максимальной границы нет как таковой для решения задачи, то лучше использовать список, т. к. при выделении памяти под динамический массив необходимо перезаписать полностью весь массив.

***Ответы на вопросы***

**1. Что такое граф?**

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

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

Графы представляются в памяти в виде матрицы и списков смежности, матрицы и списков инцидентности.

**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?**

Существует два способа обхода графов: поиск в глубину и поиск в ширину.

**5. Где используются графовые структуры?**

Чаще всего графы используют при решении различных задач о путях и расстояниях: построение маршрутов и коммуникационных путей между городами, например.

**6. Какие пути в графе Вы знаете?**

* Эйлеров путь – путь, проходящий через каждое ребро ровно один раз.
* Непростой путь – путь, проходящий по некоторым вершинам несколько раз.
* Гамильтонов путь – путь, проходящий через каждую вершину ровно один раз.

**7. Что такое каркасы графа?**

Каркас графа – это максимальный подграф графа, являющееся деревом.