Домашнее задание 2.

Алгоритмы на графах

Общее описание задания

Задание направленно на проверку знаний структур данных для представления графов, а также приобретение навыков реализации различных алгоритмов на графах. Задание состоит из следующих этапов:

1. написание ряда алгоритмов для работы с графами (см. описание ниже);
2. проведение тестирования написанной программы с точки зрения корректности работы программы;
3. теоретический анализ времени работы реализованных алгоритмов и замеры производительности полученных реализаций указанных алгоритмов;
4. написание отчета, описывающего результаты всех указанных этапов.

Для выполнения задания предполагается использовать систему контроля версий Git. Каждый студент должен создать отдельный проект на сайте http://mks1.cmc.msu.ru/ и добавить меня в проект для осуществления выполнения задания. Проект должен иметь следующее название: «[SurnameNS] – Autumn 2014 – HW2», где [SurnameNS] – это ваша фамилия и инициалы на английском языке (например, IvanovSI). Задания, выполненные без использования системы Git и соответствующего сайта проверяться не будут. Любые вопросы (в том числе, если у Вас есть возражения, связанные с использованием Dropbox) по заданию присылать по электронной почте на следующий адрес: mikle.shupletsov@gmail.com. Тема письма имеет следующий формат: [318] [Фамилия Имя] [Вопрос].

Для ускорения проверки задания, Вы может в системе GitLab сделать соответствующий ticket на мое имя, таким образом, я буду понимать, что Вы закончили выполнение задания и его можно проверять.

Срок выполнения задания: до конца семестра.

Формат входных данных.

Предполагается, что на вход программы подается ориентированный разреженный связанный граф с неотрицательными весами на ребрах. Программа должна обеспечивать загрузку описани графа из файла. Описание графа имеет следующий формат.

Считается, что все вершины графа занумерованы (Ваша программа не должна зависеть от того, каким образом занумерованы вершины). Первая строка файла содержит целое число N – число вершин в графе, вторая строка число K – число ребер в графе. Далее, файл содержит K строк, каждая из которых задает описание одного из ребер графа в следующем формате: сначала указывается номер вершины, из которой ребро исходит, потом номер вершины, в которую ребро приходит, а в конце указывается вес ребра(вещественное число). Все элементы разделены пробелами.

Пример корректного входного описания графа:

```
3
0 1 0.2
1 2 10.34
2 3 5.4
```

Реализуемые алгоритмы

Предполагается, что Ваша программа реализует следующий набор алгоритмов для работы с графами:

1. поиск и подсчет числа сильно связанных компонент графа;
2. подсчет числа ориентированных путей между заданной парой вершин графа;
3. поиск минимального остовного дерева (предполагается, что в графе существует единственное минимальное остовное дерево).

При каждом запуске программы должен выполняться только один из представленных алгоритмов. При этом выбор алгоритма для обработки графа должен быть реализован при помощи параметров командной строки. Формат параметров командной строки будет описан ниже.

Формат выходных данных

Результат работы программы передается либо в стандартный поток вывода (cout), либо в файл. В зависимости от выбранного алгоритма обработки графа программа выдает разную информацию. Далее, описан формат выходных данных для каждого алгоритма.

Поиск и подсчет числа сильно связанных компонент графа

Первая строка содержит целое число M – число сильно связанных компонент графа. Остальные M строк содержат списки номеров вершин, которые попадают в заданную сильно связанную компоненту. При этом в каждой строке номера вершин упорядочены по возрастанию и разделены пробелами, а сами строки упорядочены по возрастанию минимального номера вершины. Например, для графа, который содержит следующие сильно связанные компоненты {1,3,5}{2,6,7}{4,8,9,10} программа должна выдать следующее:

```
3
1 3 5
2 6 7
4 8 9 10
```

Подсчет числа путей между заданной парой вершин

На выходе одна строка, которая содержит три числа, разделенных пробелом. Сначала указывается первая вершина и вторая вершина, между которыми производится поиск числа путей, а потом указывается само число путей. Например, если между вершиной с номером 10 и вершиной с номером 35 в граве существует 150 путей, то программа должна выдать следующее:

```
10 35 150
```

Поиск минимального остовного дерева

Первая строка содержит целое число M – число ребер в минимальном остовном дереве, вторая строка содержит вещественное число V – суммарный вес всех ребер минимального остовного дерева. Остальные M строк содержат пары номеров вершин, задающие ребра минимального остовного дерева. При этом сначала указывается номер вершины, из которой ребро исходит, а потом номер вершины, куда приходит ребро. Номера вершин разделены пробелом. Кроме того, список ребер упорядочен лексикографически. Например, для графа, который содержит следующее минимальное остовное дерево веса 10.23 и состоящее из ребер (0,1), (1,2), (0,3), (3,4) и (4,5) программа должна выдать следующее:

```
5
10.23
0 1
0 3
1 2
3 4
4 5
```

Параметры командной строки.

Программа должна поддерживать следующие параметры командной строки:

1. `--help, -h` – при передаче этого параметра программа должна вывести краткую справку о работе с программой, которая включает краткое описание программы, а также все параметры командной строки и их назначение;
2. `--input_file "filename.txt", -I "filename.txt"` – если указан этот параметр, то программа считывает вход из файла с именем `"filename.txt"`, если параметр не указан, то считывание происходит со стандартного потока ввода (cin);
3. `--output_file "filename.txt", -O "filename.txt", -O -` если указан этот параметр, то программа записывает результат работы в файл с именем `"filename.txt"`, если параметр не указан, то запись происходит в стандартный поток вывода (cout);
4. `--algorithm (S|P|T), -a (S|P|T)` – параметр определяет тип исполняемого алгоритма (в скобках указан тип исполняемого алгоритма: `S` – поиск и подсчет числа сильно связанных компонент графа, `P` – поиск числа путей между заданной парой вершин, `T` – поиск минимального остовного дерева).
5. `--input_vertex "vertex_index", -i "vertex_index"` – параметр определяет номер вершины, из которой исходят пути при реализации алгоритма подсчета числа путей (для других алгоритмов этот параметр игнорируется);
6. `--output_vertex "vertex_index", -o "vertex_index"` – параметр определяет номер вершины, из которой исходят пути при реализации алгоритма подсчета числа путей (для других алгоритмов этот параметр игнорируется);

Тестирование программы и замеры производительности.

Программа должна быть протестирована на предмет корректного решения поставленной задачи, корректного считывания входного и генерации выходного форматов представления лабиринта и минимального пути в нем.

Должно быть проведено тестирование программы на случайно сгенерированных лабиринтах разного размера. Для генерации случайных графов предполагается написать соответствующую программу. Установить максимальные значения параметров лабиринта, при которых программа работает за приемлемое время (меньше одной минуты) и не превышает заданных ограничений по памяти (например, память Вашего компьютера) для разных форматов описания лабиринта. Сравнить полученные данные с теоретической оценкой времени работы реализованного Вами алгоритма поиска пути.

Полученные каждым студентом тесты, будут собраны в общую библиотеку тестов, которая будет использоваться для тестирования всех программ. Тесты, которые позволили найти ошибки в чужих программах, будут давать студентам дополнительные баллы.

Требования к отчету

Отчет по заданию должен содержать следующие основные разделы:

1. Введение.
2. Описание алгоритма.
3. Результаты тестирования и замеры производительности.
4. Список литературы.

Раздел «Введение» содержит постановку задачи и краткое описание полученных результатов.

Раздел «Описание алгоритма» содержит описание реализованных алгоритмов для работы с графами и теоретическую оценку времени их работы.

Раздел «Результаты тестирования и замеры производительности» содержит описания того, как написанная программа тестировалась с точки зрения корректности решения задачи, и результаты сравнения скорости работы программы на различных входных данных с теоретической оценкой времени работы алгоритма. Для наглядности требуется привести не только таблицы, но и графики. Кроме того, в этом разделе должен быть приведен анализ полученных замеров производительности.

Раздел «Список литературы» содержит ссылки на статьи и электронные ресурсы, если таковые были упомянуты в тексте отчета.

Критерии оценки

Результаты выполнения домашнего задания оцениваются по следующим основным критериям:

1. корректность и качество реализации алгоритмов;
2. правильность считывания входных форматов и генерации выходных форматов; качество оформления кода (styleguide);
3. тестирование программы и анализ производительности;
4. структура и содержание отчета;
5. использование системы контроля версий.