Национальный исследовательский университет “Высшая школа экономики”.

Факультет компьютерных наук. Программная инженерия.

Построение и анализ алгоритмов.

Домашнее задание №3 студента группы БПИ213 Абрамова Александра Сергеевича.

СОДЕРЖАНИЕ

Разработанные материалы

При выполнении задания были разработаны следующие элементы:

1. Папка $algorithms$ содержит программы на языке C++, реализующие 6 алгоритмов поиска кратчайшего пути между двумя вершинами неориентированного графа и производящие измерение эффективности их работы. При запуске программы происходит считывание количества вершин, количества рёбер, списка рёбер, каждое из которых задаётся вершиной начала, вершиной конца и весом, а также стартовой и конечной вершин из текстового файла, указываемого первым аргументом командной строки, решение поставленной задачи, а также вывод результата работы и времени выполнения в стандартный поток вывода.
2. Папка $report$ содержит следующие файлы:
   2.1. $data.json$ с результатами всех произведённых замеров
   2.2. $graphs - algorithms.html$, производящий построение графиков зависимости времени работы от размера графа с группировкой по алгоритму.
   2.3. $graphs - types.html$, производящий построение графиков зависимости времени работы от размера графа с группировкой по типу графа (полный, средний, разреженный).
   2.4. $graphs - types - no - floyd.html$, производящий построение графиков зависимости времени работы от размера графа с группировкой по типу графа (полный, средний, разреженный) без алгорита Флойда-Уоршелла (см. далее).
3. Папка $scripts$ содержит вспомогательные программы для генерации тестовых данных, сборки и запуска программ, а также для обработки выходных данных.
   3.1. Файл $config.js$ содержит настройки экспериментов:
      1) $vertexes$ - список количества вершин в графах от 10 до 1010 с шагом 50
      2) $algorithms$ - список кодовых имён анализируемых алгоритмов:
         a) dijkstra-set - реализация алгоритма Дейкстры с помощью set
         b) dijkstra-set-optimized - реализация алгоритма Дейкстры с помощью set с хранением только номеров вершин
         c) dijkstra-queue - реализация алгоритма Дейкстры с помощью $priority\_queue$
         d) dijkstra-queue-delete - реализация алгоритма Дейкстры с помощью приоритетной очереди с возможностью удаления
         e) ford-bellman - алгоритм Форда-Беллмана
         f) floyd-warshall - алгоритм Флойда-Уоршелла

3) $iterations$ - количество запусков программы на каждом наборе тестовых данных при измерении времени выполнения для получения более точного результата.
4) MIN_EDGE_WEIGHT - минимальный допустимый вес ребра графа
5) MAX_EDGE_WEIGHT - максимальный допустимый вес ребра графа

3.2. Файл $gen.js$ содержит исходный код программы, которая генерирует наборы тестовых данных и сохраняет их в директории $tests$. Для каждого количества вершин генерируется три графа: полный граф, разреженный граф (дерево) и средний граф, получаемый из разреженного путём добавления рёбер, пока коэффициент плотности не превысит 0.45

3.3. Файл $run.js$ содержит исходный код программы, производящей сборку и запуск алгоритмов:

3.3.1. При компиляции программ используется утилита g++ со следующими параметрами:

3.3.1.1. `-O2` позволяет компилятору производить оптимизации программы для более эффективной её работы

3.3.1.2. `-std=c++20` указывает версию спецификации языка C++, которую необходимо использовать при компиляции - `C++20`

3.3.1.3. `-Werror` для прекращения процесса сборки программы с ошибкой в случае обнаружения любого предупреждения компилятора

3.3.1.4. `-Wall` и `-Wextra` для включения всех предупреждений компилятора.

3.3.2. При запуске программ производится обработка выходных данных и проверка ответа: выходные данные всех алгоритмов должны совпадать. Это позволяет обеспечить достаточную уверенность в правильности их работы.

3.3.3. Результатом работы скрипта является файл data.json директории report, содержащий результаты всех запусков анализируемых алгоритмов.

4. Файл $package.json$ содержит ряд “скриптов” для запуска соответствующих вспомогательных программ.
5. Файл $SystemInformation.txt$ содержит описание окружения, в котором производилось измерение эффективности работы программ.

Эксперименты и анализ

Для получения результатов скрипт run был запущен на системе под управлением $WSL\ Debian$. Результаты сохранены в директории $report$. Файл $SystemInformation.txt$ содержит описание окружения при проведении эксперимента.

Для построения графиков следует открыть файлы с расширением $html$ директории $report$ в любом браузере и загрузить файл $data.json$ эксперимента. При этом для графиков зависимости времени выполнения от количества рёбер по оси абсцисс использована логарифмическая шкала.

По графикам $graphs - algorithms.html$ можно отметить следующее:

1. Все вариации алгоритма Дейкстры работают наиболее медленно на полном графе и наиболее быстро на разреженном графе при одинаковом количестве вершин. Тем не менее при одинаковом количестве рёбер ситуация обратная: алгоритмы работают наиболее быстро на полных графах.
2. Алгоритм Форда-Беллмана ведёт себя аналогично алгоритму Дейкстры: при фиксированном количестве вершин время работы увеличивается с повышением количества рёбер, а при фиксированном количестве рёбер - возрастает с увеличением количества вершин.
3. Все линии на графике зависимости времени работы алгоритма Флойда-Уоршелла от количества вершин проходят одинаково. Следовательно, алгоритм Флойда-Уоршелла не зависит от количества рёбер в графе при фиксированном количестве вершин. Тем не менее на графике зависимости времени работы от числа рёбер ситуация другая: алгоритм наименее эффективен на разреженных графах и наиболее эффективен на полных графах, что подтверждает сильную зависимость времени работы алгоритма от количества вершин.
4. Нетрудно заметить, что все линии на графиках достаточно хорошо приближаются полиномиальными линиями тренда степени 3: коэффициент детерминации почти во всех случаях превышает 0.99. Таким образом. можно сделать вывод, что все алгоритмы имеют асимптотическую сложность в зависимости от числа вершин и ребёр не более, чем $O(n^3)$. Более того, для всех алгоритмов, кроме алгоритма Флойда-Уоршелла, коэффициент при старшей степени уравнений линий тренда пренебрежимо мал, по чему можно предположить, что эти алгоритмы имеют асимптотическую сложность $O(n^2)$ или $O(n \cdot log(n))$

С учётом описанного, можно составить таблицу зависимости времени работы алгоритмов от количества вершин и рёбер, а также выдвинуть ряд предположений об их асимптотической сложности:

<table>
<tr><td>Алгоритм</td><td>Зависимость от количества вершин</td><td>Зависимость от количества рёбер</td></tr>
<tr><td rowspan="2">Алгоритм Форда-Беллмана</td><td>$O(V^2)$, но для графов с большим числом рёбер коэффициент при старшей степени больше. Следовательно, зависимость от количества вершин, скорее всего, линейная: $O(V)$, и присутствует зависимость от числа рёбер вида $O(V \cdot f(E))$, где f - некоторая функция от количества рёбер</td><td>$O(E)$ с достаточно большим коэффициентом детерминации</td></tr>
<tr><td colspan="2">Следовательно, предположительная сложность - $O(VE)$</td></tr>
<tr><td>Алгоритм Флойда-Уоршелла</td><td>$O(V^3)$ - коэффициент детерминации буквально равен единице</td><td>Не зависит</td></tr>
<tr><td rowspan="2">Алгоритм Дейкстры независимо от реализации</td><td>Коэффициенты при второй степени количества вершин велики, хотя и не достаточны для того, чтобы утверждать о квадратичной сложности алгоритма. Более того, коэффициенты не одинаковы для разных типов графов. Следовательно, зависимость от количества вершин, скорее всего, чуть больше, чем линейная. С учётом знания устройства алгоритма имеет смысл предположение о сложности $O(V \cdot logV)$</td><td>$O(E)$ с достаточно большим коэффициентом детерминации. Коэффициенты при квадрате количества рёбер заметно отличаются для разных типов графов. Следовательно, общая сложность - $O(E \cdot f(V))$, где f - некоторая функция от количества вершин</td></tr>
<tr><td colspan="2">Следовательно, с учётом теоретических знаний, предположительная сложность этого алгоритма - $O((V + E) \cdot logV)$</td></tr>
</table>

По графикам $graphs - types.html$ можно отметить лишь одно: алгоритм Флойда-Уоршелла работает значительно хуже остальных, из-за чего на графиках без приближения фактически виден результат только для этого алгоритма. Это коррелирует с выдвинутыми гипотезами - алгоритма Флойда-Уоршелла единственный имеет кубическую сложность.

Для получения более интересных результатов файл $graphs - types - no - floyd.html$ строит аналогичные графики, но без алгоритма Флойда-Уоршелла, что позволяет сделать ряд выводов:

1. На полном графе все алгоритмы показывают себя приблизительно одинаково. Тем не менее явно заметно, что алгоритм Дейкстры, реализованный с помощью приоритетной очереди, показывает себя немного лучше остальных как в зависимости от числа вершин, так и в зависимости от числа рёбер.

2. На разреженном графе алгоритм Форда-Беллмана заметно оказывается наиболее эффективным, что подтверждает предположение о его сложности, а реализации алгоритма Дейкстры с использованием приоритетной очереди оказываются заметно эффективнее реализаций с использованием set. Интересно, что добавление возможности удаления из приоритетной очереди не повышает эффективность алгоритма на тестовых данных этого типа. Затраты на организацию возможности удаления оказываются больше выигрыша в связи с отсутствием дубликатов в очереди.
3. Тем не менее на среднем графе реализация алгоритма Дейкстры с помощью приоритетной очереди с возможностью удаления заметно оказывается наиболее эффективной, явно превосходя даже реализацию с использованием обычной приоритетной очереди, которая показывает себя приблизительно как алгоритм Форда-Беллмана.

Интересно, что алгоритм Форда-Беллмана во всех случаях показывает себя не хуже, а иногда даже лучше алгоритма Дейкстры, хотя предположение об асимптотической сложности этих алгоритмов говорит об обратном. Это, скорее всего, связано со слишком маленьким объёмом тестовой выборки, из-за чего неасимптотические оптимизации влияют на время работы слишком сильно.

Вывод

Таким образом, наиболее эффективным алгоритмом в общем случае оказывается алгоритм Дейкстры, реализованный с помощью приоритетной очереди с возможностью удаления. Этот алгоритм лишь незначительно проигрывает своим аналогам на полном и разреженном графах, но показывает себя очень хорошо на среднем случайном графе.

Тем не менее если в качестве входных данных ожидается разреженный граф, имеет смысл использовать алгоритм Форда-Беллмана, который оказывается наиболее эффективным в этом случае.

Алгоритм Флойда-Уоршелла же совершенно не подходит для решения задачи поиска пути между двумя вершинами графа. Действительно, этот алгоритм предназначен для поиска путей между всеми парами вершин, из-за чего и показал худшие результаты в проведённом эксперименте.

Следовательно, анализ подтверждает выдвинутые предположения об асимптотической сложности алгоритмов:

1. Алгоритм Флойда-Уоршелла: $O(V^3)$
2. Алгоритм Форда-Беллмана: $O(VE)$
3. Алгоритм Дейкстры: $O((V + E) \cdot logV)$