Лабораторная работа № 9 по курсу дискретного анализа: графы

Выполнил студент группы М8О-307Б-21 Меркулов Фёдор.

Условие

Задан взвешенный ориентированный граф, состоящий из *n* вершин и m ребер. Вершины пронумерованы целыми числами от *1* до *n*. Необходимо найти величину максимального потока в графе. Истоком является вершина с номером *1*, стоком – вершина с номером *n*. Вес ребра равен его пропускной способности. Граф не содержит петель и кратных ребер.

Вариант алгоритма: E. 7 Поиск максимального потока алгоритмом Форда-Фалкерсона

Для достижения приемлемой производительности в алгоритме рекомендуется использовать поиск в ширину, а не в глубину.

Формат ввода

В первой строке заданы $1 \leq n \leq 2000$ и $1 \leq m \leq 10000$. В следующих m строках записаны ребра. Каждая строка содержит три числа – номера вершин, соединенных ребром, и вес данного ребра. Вес ребра – целое число от *0* до 10^9.

Формат вывода

Необходимо вывести одно число – искомую величину максимального потока. Если пути из истока в сток не существует, данная величина равна нулю.

Метод решения

Задача состоит в том, чтобы отправлять как можно больше груза из истока в сток, не превышая пропускной способности рёбер. Необходимо найти величину потока каждого ребра, которые удовлетворяют набору линейных ограничений и максимизируют величину потока. Задача о максимальном потоке сводится к задаче линейного программирования.

Идея алгоритма заключается в следующем. Изначально величина потока присваивается значение 0: $f(u, v) = 0$ для всех u, v из V. Затем величина потока итеративно увеличивается посредством поиска увеличивающего пути (путь от источника s к стоку t, вдоль которого можно послать ненулевой поток). Шаг алгоритма состоит в выборе пути из истока в сток в остаточной сети с увеличением потока вдоль его насколько возможно (на этом пути нас ограничивает ребро с наименьшей производной способностью в остаточной сети). Процесс повторяется, пока можно найти увеличивающий путь.

Описание программы

Изначально обнуляем все потоки так, что остаточная сеть совпадает с исходной сетью. В остаточной сети находим кратчайший путь из источника в сток. Если такого пути нет, останавливаемся. Пускаем через пройденный путь максимально возможный новый поток: ищем в нем ребро с минимальной пропускной способностью, для каждого ребра на найденном пути увеличиваем поток на это число, а в противоположном ему уменьшаем. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им ребер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.

Дневник отладки

Неправильное прочтение условий привело к $WA4$

Тест производительности

Тест производительности представляет собой сравнение с тем же алгоритмом Форда-Фалкерсона, в котором используется поиск в глубину вместо поиска в ширину.

Генерирую 4 файла, содержащие полные графы из 10 (input10), 20 (input20), 30 (input30) вершин.

papik@papik-VirtualBox:~/DA9$ cat input10 | ./da9

Ford-Fulkerson with BFS time: 1.6e-05 sec.

papik@papik-VirtualBox:~/DA9$ cat input10 | ./dfs

Ford-Fulkerson with DFS time: 0.000208 sec.

papik@papik-VirtualBox:~/DA9$ cat input20 | ./da9

Ford-Fulkerson with BFS time: 0.000415 sec.

papik@papik-VirtualBox:~/DA9$ cat input20 | ./dfs

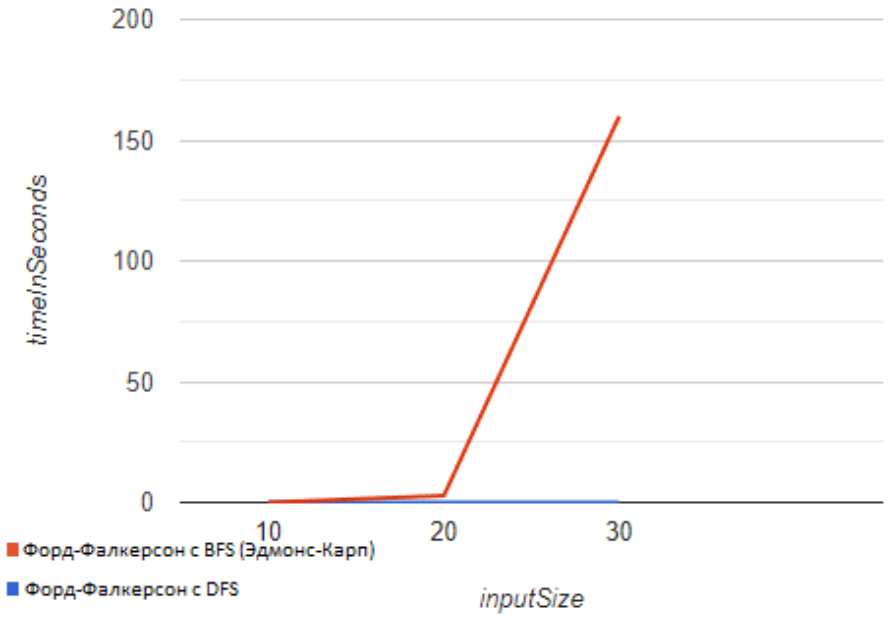
Ford-Fulkerson with DFS time: 2.79952 sec.

papik@papik-VirtualBox:~/DA9$ cat input30 | ./da9

Ford-Fulkerson with BFS time: 0.000686 sec.

papik@papik-VirtualBox:~/DA9$ cat input30 | ./dfs

Ford-Fulkerson with DFS time: 159.879 sec.



	10	20	30
Форд-Фалкерсон с BFS (Эдмонс-Карп)	0.000016 с	0.000415 с	0.000686 с
Форд-Фалкерсон с DFS	0.000208 с	2.79952 с	159.879 с

Как видим, разница между алгоритмами ощутима. Дело в том, что при поиске в глубину путь из источника в сток выбирается неоптимально, и количество итераций по ребрам становится слишком большим, в то время как поиск в ширину ищет путь с наименьшим числом ребер. Сложность алгоритма Форда-Фалкерсона с DFS $O(m * maxflow)$, где maxflow – величина максимального потока. Сложность алгоритма Форда-Фалкерсона с BFS (Эдмонса-Карпа) $O(nm^2)$. Существует еще более быстрый алгоритм решения этой задачи, использующий проталкивание предпотока – алгоритм Диница с асимптотической сложностью $O(n^2m)$.

Выводы

Выполнив девятую лабораторную работу по курсу «Дискретный анализ», я познакомился с алгоритмом Форда-Фалкерсона, с использованием BFS, (алгоритм Эдмонса-Карпа) то есть с алгоритмом нахождения максимального потока в транспортной сети, узнал, насколько стабильнее является реализация данного алгоритма с использованием BFS, по отношению к реализации с использованием DFS.