Санкт-Петербургский государственный университет

Факультет Прикладной математики – процессов управления

Реализация и анализ алгоритмов Диница и Эдмондса-Карпа

Дисциплина: Теория конечных графов и её приложения

Работу выполнили: студенты 3 курса групп 21.Б11-пу, 21.Б12-пу

Шмелев Илья

Анатолий Шевцов

Верхоланцев Денис

Санкт-Петербург

2024 г.

Оглавление

Алгоритм Диница

Описание

1. Для каждого ребра (u, v) данной сети G зададим $f(u, v) = 0$.
2. Построим вспомогательную сеть G_Lиз дополняющей сети G_f данного графа G. Если $d[t] = \infty$, остановиться и вывести f.
3. Найдём блокирующий поток f' в G_L.
4. Дополним поток f найденным потоком f' и перейдём к шагу 2.

Корректность

Покажем, что если алгоритм завершается, то на выходе у него получается поток именно максимальной величины.

В самом деле, предположим, что в какой-то момент во вспомогательной сети, построенной для остаточной сети, не удалось найти блокирующий поток. Это означает, что сток вообще не достижим во вспомогательной сети из истока. Но поскольку она содержит в себе все кратчайшие пути из истока в остаточной сети, это в свою очередь означает, что в остаточной сети нет пути из истока в сток. Следовательно, применяя теорему Форда-Фалкерсона, получаем, что текущий поток в самом деле максимален.

Временная сложность

Можно показать, что каждый раз число в рёбер кратчайшем пути из источника в сток увеличивается хотя бы на единицу, поэтому в алгоритме не более $n - 1$ блокирующих потоков, где n — число вершин в сети. Вспомогательная сеть G_L может быть построена обходом в ширину за время $O(|V| + |E|)$, а блокирующий поток на каждом уровне графа может быть найден за время $O(|V||E|)$. Поэтому время работы алгоритма Диница есть $O(|V|) * \big(O(|V| + |E|) + O(|V||E|)\big) = O(|V|^2|E|)$.

Алгоритм Эдмондса-Карпа

Описание

1. Положим все потоки равными нулю. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим кратчайший путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (он называется увеличивающим путём или увеличивающей цепью) максимально возможный поток:
    1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью c_{min}.
    2. Для каждого ребра на найденном пути увеличиваем поток на c_{min}, а в противоположном ему — уменьшаем на c_{min}.
    3. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.
4. Возвращаемся на шаг 2

Корректность

На каждой итерации цикла $while$ поток в графе G увеличивается вдоль одного из кратчайших путей в G_f из истока s в сток t. Этот процесс повторяется до тех пор пока существует кратчайший $s \to t$ путь в G_f. Если в G_f не существует кратчайшего пути из s в t, значит, не существует вообще никакого $s \to t$ пути в G_f следовательно по теореме Форда-Фалкерсона найденный поток f максимальный.

Временная сложность

С учетом числа вершин V, числа ребер E, временная сложность алгоритма Эдмондса-Карпа составляет $O(|V| * |E|^2)$. Это означает, что Эдмондс-Карп не зависит от максимального потока, как Форд-Фалкерсон, а зависит от числа вершин и ребер.

Причина получения такой временной сложности для Эдмондса-Карпа заключается в том, что он использует BFS, которая имеет временную

сложность $O(|V| + |E|)$. Но если предположить худший сценарий для Эдмондса-Карпа, с графом, где количество ребер E намного больше вершин V, , временная сложность для BFS становится $O(E)$. BFS должен запуститься один раз для каждого увеличивающего пути, и на самом деле во время работы алгоритма Эдмондса-Карпа может быть найдено близко $V * E$ к увеличивающих путей. Таким образом, BFS с временной сложностью $O(E)$ может быть запущен близко к $V * E$ раз в худшем случае, что означает, что мы получаем общую временную сложность для Эдмондса-Карпа: $O(|V| * |E| * |E| = O(|V| * |E|^2)$.

Программы

https://github.com/ShmelJUJ/Graph-Theory-Project

Результаты тестирования

```
test dinic on datasets.py test 1.txt,  time: 0.00011887700384249911, N = 7, M = 7, maxflow = 935, maxflow nx = 935
.test 2.txt,  time: 0.000418285999330692, N = 16, M = 16, maxflow = 2789, maxflow nx = 2789
.test 3.txt,  time: 2.5367000489495695e-05, N = 4, M = 4, maxflow = 2000000, maxflow nx = 2000000
.test 4.txt,  time: 5.8206001995131373e-05, N = 6, M = 6, maxflow = 23, maxflow nx = 23
.test 5.txt,  time: 0.00020016900089103729, N = 50, M = 50, maxflow = 256, maxflow nx = 256
.test 6.txt,  time: 0.0003796480013988912, N = 100, M = 100, maxflow = 523, maxflow nx = 523
.test d1.txt,  time: 0.0301795650011627, N = 100, M = 100, maxflow = 171, maxflow nx = 171
.test d2.txt,  time: 0.0313832590036327, N = 100, M = 100, maxflow = 8023, maxflow nx = 8023
.test d3.txt,  time: 0.34008939700288465, N = 500, M = 500, maxflow = 9078, maxflow nx = 9078
.test d4.txt,  time: 1.3162212600000203, N = 1000, M = 1000, maxflow = 9072, maxflow nx = 9072
.test d5.txt,  time: 9.336996299680322e-06, N = 2, M = 2, maxflow = 3278, maxflow nx = 3278
.test rd01.txt,  time: 0.00012891199730802327, N = 17, M = 17, maxflow = 24080, maxflow nx = 24080
.test rd02.txt,  time: 0.00045812899770680815, N = 47, M = 47, maxflow = 37897, maxflow nx = 37897
.test rd03.txt,  time: 0.0014093110003159381, N = 93, M = 93, maxflow = 68537, maxflow nx = 68537
.test rd04.txt,  time: 0.0030879769983584993, N = 100, M = 100, maxflow = 62599, maxflow nx = 62599
.test rd05.txt,  time: 0.032466654003656004, N = 498, M = 498, maxflow = 153728, maxflow nx = 153728
.test rd06.txt,  time: 0.08893363899551332, N = 970, M = 970, maxflow = 224507, maxflow nx = 224507
.test rd07.txt,  time: 3.4388561739979195, N = 10013, M = 10013, maxflow = 730313, maxflow nx = 730313
.test rl01.txt,  time: 0.00011680700117722154, N = 17, M = 17, maxflow = 15966, maxflow nx = 15966
.test rl02.txt,  time: 0.0008398220015806146, N = 47, M = 47, maxflow = 36865, maxflow nx = 36865
.test rl03.txt,  time: 0.0011591590009629726, N = 93, M = 93, maxflow = 63681, maxflow nx = 63681
.test rl04.txt,  time: 0.0010874850049731322, N = 100, M = 100, maxflow = 55434, maxflow nx = 55434
.test rl05.txt,  time: 0.020357020999654196, N = 498, M = 498, maxflow = 138688, maxflow nx = 138688
.test rl06.txt,  time: 0.013385191996349022, N = 498, M = 498, maxflow = 141131, maxflow nx = 141131
.test rl07.txt,  time: 0.03891051900427556, N = 970, M = 970, maxflow = 204082, maxflow nx = 204082
.test rl08.txt,  time: 0.04262271200423129, N = 970, M = 970, maxflow = 211102, maxflow nx = 211102
.test rl09.txt,  time: 0.03188807999686105, N = 970, M = 970, maxflow = 218990, maxflow nx = 218990
.test rl10.txt,  time: 0.39177784400089877, N = 4952, M = 4952, maxflow = 482613, maxflow nx = 482613
```

Рисунок 1 - алгоритм Диница на датасетах

```
.
test dinic on random graph.py
 test 1: average time = 3.0032999347895384e-05  ,    max time = 0.00013823400513501838 ,    N = 10 , M = 20 , U = 30
.test 2: average time = 0.00013014600030146538  ,    max time = 0.0005674870044458658 ,    N = 10 , M = 40 , U = 30
.test 3: average time = 0.00022104341973317788  ,    max time = 0.0006057260034140199 ,    N = 10 , M = 80 , U = 30
.test 4: average time = 0.00014670470030978322  ,    max time = 0.0007138199944165535 ,    N = 50 , M = 100 , U = 50
.test 5: average time = 0.000520136440609349  ,    max time = 0.0014502899939543568 ,    N = 50 , M = 200 , U = 50
.test 6: average time = 0.0007043603002966848  ,    max time = 0.0019078840050497092 ,    N = 50 , M = 400 , U = 50
.test 7: average time = 0.0001765420992160216  ,    max time = 0.0006467139974120073 ,    N = 100 , M = 200 , U = 100
.test 8: average time = 0.0006457444607804064  ,    max time = 0.001462123000237625 ,    N = 100 , M = 400 , U = 100
.test 9: average time = 0.0012470060399209614  ,    max time = 0.0032459799986099824 ,    N = 100 , M = 800 , U = 100
.test 10: average time = 0.00017327844063402153  ,    max time = 0.0004079200007254258 ,    N = 10 , M = 90 , U = 30
.test 11: average time = 0.00023850887970183976  ,    max time = 0.0006674909964203835 ,    N = 20 , M = 90 , U = 30
.test 12: average time = 0.00015370781999081375  ,    max time = 0.0007446109957527369 ,    N = 40 , M = 90 , U = 30
.test 13: average time = 0.002723883340659086  ,    max time = 0.004438169002241921 ,    N = 50 , M = 2450 , U = 100
.test 14: average time = 0.002715407280193176  ,    max time = 0.005329088002326898 ,    N = 100 , M = 2450 , U = 100
.test 15: average time = 0.0032474498201918323  ,    max time = 0.00660610700288089 ,    N = 200 , M = 2450 , U = 100
.test 16: average time = 0.010200781040039146  ,    max time = 0.022498632999948629 ,    N = 100 , M = 9900 , U = 100
.test 17: average time = 0.0099774006397638  ,    max time = 0.02124050899874419 ,    N = 200 , M = 9900 , U = 100
.test 18: average time = 0.012228911259880989  ,    max time = 0.02323669099860126 ,    N = 400 , M = 9900 , U = 100
.
```

Рисунок 2 - алгоритм Диница на случайно сгенерированных тестах

```
test edmonds karp on datasets.py
test 1.txt,  time: 5.627200152957812e-05, N = 7, M = 38, maxflow = 935, maxflow nx = 935
.
test 2.txt,  time: 0.00015134300338104367, N = 16, M = 226, maxflow = 2789, maxflow nx = 2789
.
test 3.txt,  time: 1.5899997379165143e-05, N = 4, M = 5, maxflow = 2000000, maxflow nx = 2000000
.
test 4.txt,  time: 2.3353000869974494e-05, N = 6, M = 10, maxflow = 23, maxflow nx = 23
.
test 5.txt,  time: 3.880199801642448e-05, N = 50, M = 97, maxflow = 256, maxflow nx = 256
.
test 6.txt,  time: 7.923899829620495e-05, N = 100, M = 99, maxflow = 523, maxflow nx = 523
.
test d1.txt,  time: 0.005705879004381131, N = 100, M = 197, maxflow = 171, maxflow nx = 171
.
test d2.txt,  time: 0.006380904997058678, N = 100, M = 197, maxflow = 8023, maxflow nx = 8023
.
test d3.txt,  time: 0.11606375500559807, N = 500, M = 997, maxflow = 9078, maxflow nx = 9078
.
test d4.txt,  time: 0.42540186099358834, N = 1000, M = 1997, maxflow = 9072, maxflow nx = 9072
.
test d5.txt,  time: 9.779003448784351e-06, N = 2, M = 1, maxflow = 3278, maxflow nx = 3278
.
test rd01.txt,  time: 0.00021336000645533204, N = 17, M = 54, maxflow = 24080, maxflow nx = 24080
.
test rd02.txt,  time: 0.0008924420035327785, N = 47, M = 186, maxflow = 37897, maxflow nx = 37897
.
test rd03.txt,  time: 0.004966123997292016, N = 93, M = 398, maxflow = 68537, maxflow nx = 68537
.
test rd04.txt,  time: 0.00838498400116805, N = 100, M = 429, maxflow = 62599, maxflow nx = 62599
.
test rd05.txt,  time: 0.16201947400259087, N = 498, M = 2342, maxflow = 153728, maxflow nx = 153728
.
test rd06.txt,  time: 0.6138401009957306, N = 970, M = 4644, maxflow = 224507, maxflow nx = 224507
.
test rd07.txt,  time: 80.22378965700045, N = 10013, M = 49422, maxflow = 730313, maxflow nx = 730313
.
test rl01.txt,  time: 9.859800047706813e-05, N = 17, M = 32, maxflow = 15966, maxflow nx = 15966
.
test rl02.txt,  time: 0.0009318209995399229, N = 47, M = 127, maxflow = 36865, maxflow nx = 36865
.
test rl03.txt,  time: 0.005034047004301101, N = 93, M = 357, maxflow = 63681, maxflow nx = 63681
.
test rl04.txt,  time: 0.005040063006163109, N = 100, M = 402, maxflow = 55434, maxflow nx = 55434
.
test rl05.txt,  time: 0.20405678700626595, N = 498, M = 2954, maxflow = 138688, maxflow nx = 138688
.
test rl06.txt,  time: 0.19069063899951288, N = 498, M = 2773, maxflow = 141131, maxflow nx = 141131
.
test rl07.txt,  time: 0.7355636810025317, N = 970, M = 5655, maxflow = 204082, maxflow nx = 204082
.
test rl08.txt,  time: 0.9348183330002939, N = 970, M = 7756, maxflow = 211102, maxflow nx = 211102
.
test rl09.txt,  time: 1.0548257019981975, N = 970, M = 9840, maxflow = 218990, maxflow nx = 218990
.
test rl10.txt,  time: 36.754468419996556, N = 4952, M = 48838, maxflow = 482613, maxflow nx = 482613
.
```

Рисунок 3 - алгоритм Эдмонда-Карпа на датасетах

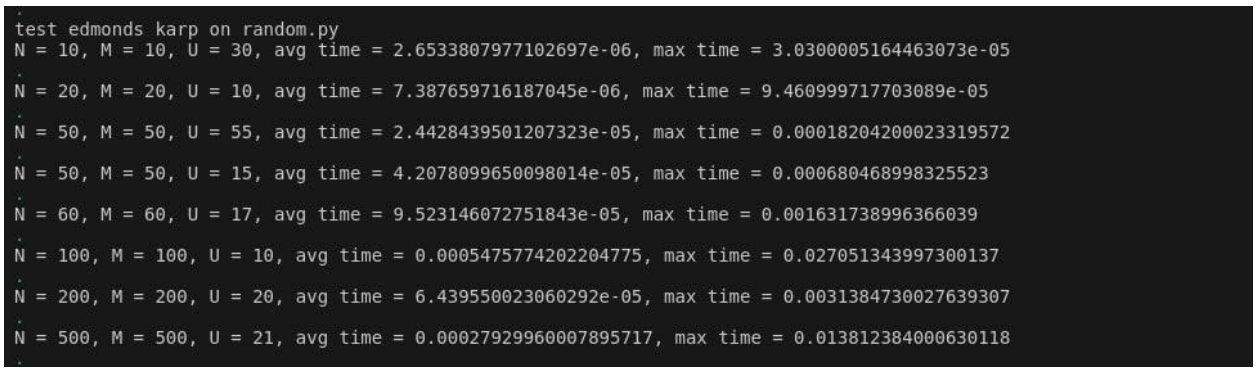

Рисунок 4 - алгоритм Эдмонда-Карпа на случайно сгенерированных тестах

Как можно заметить, на графах, где количество рёбер значительно меньше, чем количество вершин, алгоритм Диница обычно показывает более высокую производительность. Это связано с его способностью быстро находить увеличивающие пути и использовать метод уровней. Однако, если граф плотный, то есть количество рёбер близко к квадрату количества вершин, то алгоритм Эдмонса-Карпа может оказаться более эффективным. Это происходит потому, что в этом случае BFS, используемый в Эдмонсе-Карпе, имеет меньшую сложность по сравнению с методом уровней, применяемым в Динице.

Источники

- https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%AD%D0%B4%D0%BC%D0%BE%D0%BD%D0%B4%D1%81%D0%B0-%D0%9A%D0%B0%D1%80%D0%BF%D0%B0#cite_note-1
- https://www.w3schools.com/dsa/dsa_algo_graphs_edmondskarp.php
- https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A4%D0%BE%D1%80%D0%B4%D0%B0-%D0%A4%D0%B0%D0%BB%D0%BA%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0_%D0%B4%D0%BB%D1%8F_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0_%D0%BC%D0%B0%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D0%B0%D1%80%D0%BE%D1%81%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D1%8F
- https://neerc.ifmo.ru/wiki/index.php?title=%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B0_%D0%94%D0%B8%D0%BD%D0%B8%D1%86%D0%B0

Характеристики вычислительной среды и оборудования

ОС: Win 11

Среда: VS Code, Ubuntu

Процессор: intel i5 2.10 GHz

Оперативная память: 8 Гб