Флойд. Форд-Беллман

Сапожников Денис

Contents

1 Алгоритм Флойда

1.1 Алгоритм

Дан граф ориентированный взвешенный граф G. Необходимо найти кратчайшее расстояние между всеми парами вершин, если нет отрицательных циклов.

Алгоритм Флойда — это типичная динамика. Пусть $d_{i,j,k}$ — кратчайший путь из i в j, проходящий по вершинам из множества $\{1, 2, \ldots, k\} \cup \{i\} \cup \{j\}$. Легко понять, что переходы в такой динамике следующие:

$$dp_{i,j,k} = \min(dp_{i,j,k-1}, dp_{i,k,k-1} + dp_{k,j,k-1})$$

А сам алгоритм пишется в 4 строчки:

```
// d[i][j] = INF, if edge does not exists
for (int k = 0; k < n; ++k)
  for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
      d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
```

1.2 Восстанавливаем ответ

Но вот есть у нас Флойд, а можно как-то восстановить кратчайшие пути между каждой парой вершин?

Да, конечно же, можно. Нужно для каждой пары вершин поддерживать информацию о том, через какую вершину мы обновили минимум.

1.3 Отрицательные циклы

Но вот бывает такое, что в графе есть отрицательный цикл, как определить, что он есть?

Если он есть, то на какой-то диагональной клетке появится отрицательное число. Более того, они появятся во всех $d_{i,i}$ таких, что i содержится в отрицательном цикле, но могут появиться ещё и в вершинах, достижимых из такого отрицательного цикла.

Прошлый пункт помогает даже восстановить какой-нибудь отрицательный цикл.

2 Алгоритм Форда-Беллмана

2.1 Алгоритм

Дан граф ориентированный взвешенный граф G. Необходимо найти кратчайшее расстояние от s до всех остальных вершин при условии, что в графе нет отрицательных циклов.

Обратите внимание, тут разрешаются отрицательные рёбра, в отличии от Дейкстры.

Пусть $d_{v,i}$ – это кратчайшее расстояние от s до v за не более чем i шагов. Изначально $d_s = 0, d_{v \neq s} = +\infty$. Переходы очень простые:

$$d_{v,i} = \min \begin{cases} \min\limits_{(v,u) \in E} d_{u,i-1} + w(v,u) \\ d_{v,i-1} \end{cases}$$

Такая динамика считается за $O(nm)$ и требует $O(n)$ памяти, так как мы каждый раз обращаемся только к предыдущему слою, при этом нам достаточно посчитать $n-1$ слой (самый длинный по количеству вершин простой путь — состоит из n вершин и $n-1$ ребра)

Реализация:

```
vector<int> d(n, INF);
d[s] = 0;
for (int iter = 0; iter < n - 1; ++iter)
  for (int v = 0; v < n; ++v)
    for (auto [u, w] : gr[v])
      d[v] = min(d[v], d[u] + w);
```

2.2 Восстановление ответа

Ничего не мешает нам обновлять информацию о том, откуда мы обновили минимум — p_v. Так можно легко восстановить путь.

2.3 Отрицательные циклы

А давайте сделаем n итераций алгоритма. Тогда если на n-й что-то поменяется в массиве d, то это значит, что есть отрицательный цикл.

2.4 Больше, чем отрицательные циклы

А что если мы хотим решить следующую задачу классификации вершин:

1. Вершина лежит в достижимом отрицательном цикле
2. Вершина достижима из достижимого отрицательного цикла
3. Вершина не достижима
4. Вершина достижима, не содержится в отрицательном цикле и не достижима из него, то есть для неё корректно определено понятие расстояния

Оказывается, это NP-сложная задача. А всё из-за пункта 1 и 2. А вот если не пытаться разделить эти 2 пункта в разные, то задача решаема за $O(nm)$.

То есть хотим классифицировать следующим образом:

1. Вершина лежит в достижимом отрицательном цикле или достижима из достижимого отрицательного цикла
2. Вершина не достижима
3. Вершина достижима, не содержится в отрицательном цикле и не достижима из него

Всё что нам нужно сделать — это дополнительные n итераций Форда-Беллмана. Тогда гарантированно для всех вершин, достижимых из отрицательного цикла уменьшится расстояние относительно $(n-1)$-й итерации.

3 Потенциалы Джонсона

О них прекрасно написано на алгоритмике.