Основные алгоритмы, комментарии по одиннадцатому листку (От кратчайших путей к динамическому программированию)

Шибаев Иннокентий

April 21, 2021

1 По поводу теории

1.1 О том что можно делать с матрицами смежности

Рассмотрим граф $G = (V, E)$ и его матрицу смежности A, т.е. матрицу где

$$A_{uv} = \begin{cases} 1, & (u, v) \in E \\ 0, & \text{иначе.} \end{cases}$$

Рассмотрим степени матрицы A. Какой смысл они несут? Верно следующее утверждение:

Утверждение 1.1. Пусть есть граф $G = (V, E)$ и его матрица смежности A. Тогда A^t_{uv} – число маршрутов из вершины u в вершину v длины t.

Proof. Вообще доказательство этого факта по индукции рассматривалось на лекции, но мне хотелось бы рассмотреть более грубое решение через суммы. Рассмотрим для начала случай A^2. Надо доказать что в этой матрице записано число маршрутов длины 2 из одной вершины в другую.

Рассмотрим две вершины, пусть u и v. Если есть какой-то маршрут из u, в v длины 2, то он проходит через еще какую-то вершину $k \in V$, т.е. имеет вид $u \to k \to v$.

Таким образом, чтобы посчитать число маршрутов нам надо для каждого k проверить что есть ребра (u, k) и (k, v), и просуммировать такие маршруты по k. С учетом того что у нас есть матрица смежности A, определенная выше, мы можем задавать это условие буквально рассматривая произведение $A_{uk} \cdot A_{kv}$ – это будет равно 1 если оба ребра есть, и будет равно 0 в противном случае. Итак, число маршрутов длины 2 из вершины u в вершину v есть

$$B_{uv} = \sum_{k \in V} A_{uk} A_{kv}.$$

Осталось заметить что то что написано это ровно то что получается при возведении матрицы A во вторую степень – A^2_{uv}.

Теперь рассмотрим маршрут длины t. Аналогично тому что мы говорили раньше, он должен проходить через некоторый набор вершин $i_1, \dots, i_{t-1} \in V$, и чтобы найти число таких маршрутов мы должны, соответственно, найти

$$B_{uv} = \sum_{i_1, \dots, i_{t-1} \in V} A_{ui_1} A_{i_1 i_2} \dots A_{i_{t-1} v},$$

т.е. мы именно просматриваем все маршруты длины t – маршрут есть тогда и только тогда, когда есть все ребра входящие в него. Осталось сказать что то что записано выше – ровно возведение в степерь, $B_{uv} = A^t_{uv}$ □

Примечание 1.1. Итак, возводя матрицу смежности в степень мы получаем число маршрутов какой-то фиксированной длины. Чтобы получить число маршрутов любой длины от 1 до t достаточно (считая что в графе нет петель) перейти к графу с петлями

$$A := A + E.$$

Правда в таком случае есть проблема – мы будем считать так же маршруты которые "крутятся" в петлях где-то внутри маршрута. А нам нужно чтобы они это делали только в конце – т.е. к примеру чтобы учесть маршрут длины l когда мы ищем маршруты длины не превышающей $t > l$ наш маршрут должен закончится в нужной вершине, после чего там "крутиться" – тогда мы учтем его корректное число раз.

Чтобы это сделать можно сделать немного более хитро – раздвоить каждую вершину, и добавить два ребра $v \to v'$ и $v' \to v'$ – и запустить возведение в степень $t+1$. Тогда интересующие нас значения будут находится в клетках $A^{t+1}_{uv'}$.

Примечание 1.2. Есть проблема связанная со скоростью. Обычное перемножение двух матриц размера $n \times n$ занимает $O(n^3)$ операций. И чтобы проверить достижимость надо возвести матрицу в степень $n-1$ (так как все пути не длиннее чем $|V| - 1 = n - 1$). Это уже $O(n^4)$.

Ну, это можно улучшить, как минимум заменив возведение в степень на бинарное возведение в степень, получив уже $O(n^3 \log n)$ операций. Можно пойти еще дальше, и заменить алгоритм перемножения матриц, к примеру, алгоритмом Штрассена, получив уже $O(n^{\log_2 7} \log n)$.

Примечание 1.3. Еще одна проблема связана с тем как быстро растут числа в матрице. Растут они экспоненциально, так что если мы реально хотим поддерживать число путей нам надо использовать уже длинную арифметику, так что на самом деле оценка сложности будет хуже. С другой стороны если вам нужна достижимость (т.е. ответ на вопрос "достижима ли v из u" то вам достаточно лишь поддерживать наличие/отсутствие пути из u в v, т.е. 1 если в какой-то момент путь появился, либо 0, иначе.

Но это не единственный способ использования матриц смежности. С их помощью можно так же реализовать поиск длин минимальных путей во взвешенном графе. Чтобы это реализовать надо модифицировать операции которые мы использовали, вместо умножения надо считать минимум из того что есть в клетке с новыми перебираемыми путями.

Разберем это в деталях. Пусть матрица A такова что

$$A_{uv} = \begin{cases} w_{uv}, & (u,v) \in E \\ +\infty, & \text{иначе}, \end{cases}$$

где w_{uv} – вес ребра (u,v). Тогда очевидно, что в матрице A записаны длины кратчайших путей длины 1 между всеми парами вершин.

Теперь опять же рассмотрим маршруты длины 2. Они имеют вид $u \to k \to v$, и среди них нам надо выбрать минимальный, т.е.

$$B_{uv} = \min_{k \in V} \left(A_{uk} + A_{kv} \right),$$

и мы приходим к чему-то похожему на то что было раньше. Аналогично для маршрутов длины t надо рассматривать

$$B_{uv} = \min_{i_1, \ldots, i_{t-1} \in V} \left(A_{ui_1} + A_{i_1 i_2} + \ldots + A_{i_{t-1} v} \right),$$

но это можно записать как

$$B_{uv} = \min_{i_m} \left\{ \min_{i_1, \ldots, i_{m-1} \in V} \left(A_{ui_1} + \ldots + A_{i_{m-1} i_m} \right) + \min_{i_{m+1}, \ldots, i_{t-1} \in V} \left(A_{i_m i_{m+1}} + \ldots + A_{i_{t-1} v} \right) \right\},$$

т.е. мы можем воспользоваться тем же принципом что и при бинарном возведении в степень, и, значит, решать эту задачу за $O(n^3 \log n)$.

Примечание 1.4. Тут важно отметить, что так как мы уже используем не совсем умножение то тот же алгоритм Штрассена использовать не получится, так что оценка будет хуже $O(n^3)$. Далее мы разберем алгоритм Флойда-Уоршелла, который имеет как раз сложность $O(n^3)$.

Примечание 1.5. Еще один момент. То что мы разобрали выше – это поиск длины минимальных маршрутов (хотя с учетом того что мы говорим о случае без отрицательных циклов здесь можно говорить и про пути) **фиксированной** реберной длины t (если мы получаем, к примеру, матрицу A^t).

Чтобы получить здесь алгоритм для минимальных маршрутов реберной длины $\leqslant t$ надо просто заменить значения на диагонали нулями – добавить петли нулевого веса для каждой вершины.

1.1.1 Алгоритм Флойда-Уоршелла (Floyd–Warshall algorithm)

Этот алгоритм ищет длины кратчайших путей между **всеми парами вершин** за $O(n^3)$ (сравните это с $O(n^3 \log n)$ которое мы получили в прошлом параграфе).

Сам алгоритм довольно простой:

Алгоритм 1: Алгоритм Флойда-Уоршелла (Floyd–Warshall algorithm)

Input : Граф $G = (V, E)$; вершины V пронумерованы $\{1, \dots, n\}$; E задано матрицей A размера $n \times n$, где $A[i, j] = w_{ij}$ – весу ребра (i, j) если такое есть, и $+\infty$ если ребра нет

Output: Массив d, где $d[u, v]$ – длина кратчайшего пути в графе G от вершины u до вершины v

```
d := A
for i := 1 to n do
    d[i,i] := 0
for k := 1 to n do
    for i := 1 to n do
        for j := 1 to n do
            if d[i,j] > d[i,k] + d[k,j] then
                d[i,j] := d[i,k] + d[k,j]
```

Сложность алгоритма довольно понятна – у нас есть 3 вложенных цикла, получаем $O(n^3)$. Обсудим почему это вообще работает, и что алгоритм делает.

Алгоритм разбит на фазы (которые и перебираются параметром k). Утверждается, что на k-м шаге мы получаем самые короткие пути у которых внутренними могут являться только вершины с 1 по k.

Действительно, на первой фазе это верно (т.к. это просто матрица смежности). На k-й фазе мы для каждой пары вершин (i, j) смотри на кратчайший путь что проходит через вершину k (важно что мы смотрим на кратчайший путь без вершины k до нее и из нее, тем самым поддерживая условие), и, сравнивая его с кратчайшим путем который есть на данный момент, выбираем лучший. Таким образом мы получаем матрицу расстояний в которой уже записаны кратчайшие пути по вершинам с 1 по k.

После n-го шага мы получаем что внутренними могут быть все вершины, таким образом это есть кратчайшие пути.

Примечание 1.6. Здесь еще есть момент связанный с тем что мы пишем изменения сразу в матрицу d, а не создаем в начале матрицу d_{new} делая потом $d := d_{new}$, что вообще говорит нам теория. Но на самом деле разницы нет – если нет отрицательных циклов то на расстояния $d[i, k]$ и $d[k, j]$ такая перезапись повлиять не могла (у них не должно быть вершины k во внутренних путях т.е. они не могли измениться на этом шаге).

2 По поводу задач

Задача 2.1 (Задача 7 из листка 11). Дан орграф, вам нужно найти длины кратчайших путей от вершины v до всех остальных вершин. В графе могут быть ребра отрицательного веса, но только те, которые выходят из v. Предложите эффективный алгоритм, докажите его корректность и оцените асимптотику.

Решение. Если нет цикла отрицательного веса то мы можем добавить некоторую константу к весам ребер выходящих из v – на порядок путей по суммарным весам ребер такое не повлияет, и так мы сведем задачу к Дейкстре, к примеру. $\square$