Олимпиадное программирование

Занятие 18. Графы. Поиск кратчайщих путей

Труфанов Павел Николаевич

Онлайн-школа  Фоксфорд

Foxford.ru 2019-2020

Постановка задачи

Имеется невзвешенный граф, в котором задана стартовая вершина s. Требуется найти длины кратчайших путей от вершины s до всех остальных вершин, при этом надо уметь восстанавливать сами кратчайшие пути.

Основная теорема

Идея – давайте выберем некую вершину v, и обновим расстояния вдоль всех выходящих из нее ребер. Но тогда требуется, чтобы у этой вершины v уже было оптимальное расстояние. Теорема – давайте каждый раз выбирать еще необработанную вершину с минимальным расстоянием до нее.

Реализация

Требуется некая структура данных, которая позволит за быстро выбирать необработанную вершину с минимальным расстоянием.
И это – ...

Реализация

Требуется некая структура данных, которая позволит за быстро выбирать необработанную вершину с минимальным расстоянием.
И это – очередь

Алгоритм

Повторяем пока новая вершина находится – берем необработанную вершину с минимальным расстоянием и обновляем расстояния вдоль всех выходящих из нее ребер.

Восстановление ответа

Вспомним идею предков, которую мы использовали в динамике: для каждой вершины запоминаем ту вершину, из которой мы оптимальнее всего обновились. Теперь идя назад по предкам мы сможем восстановить оптимальный путь.

Несколько стартовых вершин

Пусть теперь выбрано несколько стартовых вершин. Хотим для каждой вершины найти расстояние до ближайшей стартовой.
Давайте добавим фиктивную вершину, до которой поставим расстояние -1, и проведем из нее ребра во все стартовые. Тогда запустимся алгоритмом из фиктивной и получим корректные ответы.
Можно даже не создавать фиктивную, а сразу положить в нашу очередь все стартовые.

0-1 BFS

Дан граф, где вес каждого ребра 0 или 1. Хотим по прежнему находить кратчайшие расстояние. Посмотрим на очередь. В начале очереди всегда лежат вершины с расстоянием d, а в конце с расстоянием $d + 1$. Мы берем вершину с расстоянием d. Если ребро веса 1, то надо положить его в компанию вершин с расстоянием $d + 1$. Если же ребро веса 0, то нужно положить в компанию вершин с расстоянием d. Значит, можно положить в начало очереди. Для этого используем структуру deque.

Алгоритм Дейкстры

Та же самая задача, но теперь граф взвешенный и все ребра неотрицательного веса. Хотим делать то же самое.
Можно каждый раз выбирать вершину простым циклов. А можно использовать встроенную структуру set.

Алгоритм Форда-Беллмана

Дан взвешенный граф с любыми весами ребер. Требуется найти длины кратчайших путей от стартовой вершины v, или найти отрицательный цикл.

Алгоритм Форда-Беллмана

Обозначим за один шаг алгоритма проход по всем ребрам, где для каждого ребра мы пытаемся обновить расстояние для его конца с большим расстоянием, через второй конец. Утверждается, что за один шаг алгоритма мы установим оптимальное расстояние для хотя бы одной вершины. Изначально оптимальное расстояние установлено только для стартовой вершины, а значит требуется сделать $n - 1$ итерацию.

Поиск отрицательного цикла

В процессе алгоритма легко проставлять предков. Как понять, что есть отрицательный цикл? Давайте сделаем еще один шаг, и если после него расстояние до какой-нибудь вершины обновится, то она лежит в отрицательном цикле. Используя предков, мы сможем восстановить сам цикл.

Алгоритм Флойда

Дан взвешенный граф с любыми весами ребер. Требуется найти длины кратчайших путей между всеми парами вершин.

Алгоритм Флойда

Воспользуемся динамическим программированием. Пусть $dp[i][j][k]$ - длина кратчайшего пути между вершинами i, j, который посещает промежуточные вершины только с номерами $0, \ldots, k-1$. $dp[i][j][0] = matr[i][j]$, где $matr$ - матрица смежности. Как посчитать $dp[i][j][k]$? Если новый путь не посещает вершину $k-1$, то $dp[i][j][k] = dp[i][j][k-1]$. Иначе кратчайший путь можно поделить на участки $dp[i][k-1][k-1]$ и $dp[k-1][j][k-1]$. Так и пересчитаем динамику.

До встречи!

FOXFORD.RU

Онлайн-школа Фоксфорд

Фоксфорд