Девятое задание по алгоритмам

Ситдиков Руслан

16 апреля 2018 г.

Задача 1

Легко поддерживать дополнительную информацию — так называемых **"предков"**, по которым можно будет восстанавливать сам кратчайший путь между любыми двумя заданными вершинами в виде **последовательности вершин.**
Для этого достаточно кроме матрицы расстояний $d[][]$ поддерживать также матрицу предков $p[][]$, которая для каждой пары вершин будет содержать номер фазы, на которой было получено кратчайшее расстояние между ними. Понятно, что этот номер фазы является не чем иным, как "средней"вершиной искомого кратчайшего пути, и теперь нам просто надо найти кратчайший путь между вершинами i и $p[i][j]$, а также между $p[i][j]$ и j. Отсюда получается простой рекурсивный алгоритм восстановления кратчайшего пути.

Задача 2

Утверждение: При наличии цикла отрицательного веса в матрице D появятся отрицательные числа на главной диагонали.
Так как алгоритм Флойда последовательно релаксирует расстояния между всеми парами вершин (i, j) том числе и теми, у которых $i = j$ а начальное расстояние между парой вершин (i, i) равно нулю, то релаксация может произойти только при наличии вершины k такой, что $d[i][k]+d[k][i] < 0$ что эквивалентно наличию отрицательного цикла, проходящего через вершину i.
Из доказательства следует, что для поиска цикла отрицательного веса необходимо, после завершения работы алгоритма, найти вершину i для которой $d[i][i] < 0$, и вывести кратчайший путь между парой вершин (i, i). При этом стоит учитывать, что при наличии отрицательного цикла расстояния могут уменьшаться экспоненциально. Для предотвращения переполнения все вычисления стоит ограничивать снизу величиной $-\infty$, либо проверять наличие отрицательных чисел на главной диагонали во время подсчета.

Задача 3

Алгоритм
Найдём это ребро с отрицательным весом. Пусть его вес x. Тогда увеличеним веса всех рёбер на $|x|$. Далее найдём нужный путь, например Дейкстрой, не забываем учесть, что мы увеличили все веса на $|x|$.

Корректность
Алгоритм корректен, так как при увеличении весов всех ребёр, мы будем работать только с неотрицательными весами, а при неотрицательных весах, можно спокойной применять Дейкстру.
Сложность
Так как мы только применили алгоритм Дейсктры и нашли это ребро, с орицательным весом, то сложность нашего алгоритма составляет $O(n^2 + m)$.

Задача 5

Алгоритм
Найдём транзитивное замыкание, многократно применив поиск в ширину. Сохранил свою реализацию на $https://paste.ubuntu.com/p/RJ3pbRyBNQ/$
Сложность
Алгоритм работает за $O(nm)$, так как мы пробегаемся по каждой вершине.