Домашнее задание №9

Томинин Ярослав, 778 группа

5 мая 2018 г.

1.

Сопоставим каждой вершине матрицы массив из вершин, которые составляют наш путь, так для к-ой матрицы на $D_{ij}^{(}k)$ месте будет еще массив из вершин, которые составляют кратчайший путь, состоящий из всех вершин, не больших к. Докажем, что мы сможем поддерживать этот инвариант по индукции.

База: для к=1 в вершинах записан массив, состоящий из i и j.

Переход: Допустим верно для к-1, докажем для к. Для этого посмотрим на $\min(D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)})$. Если длина не поменялась, то и массив не поменяется, иначе мы просто сопоставим $D_{ij}^{(k)}$ массив, который является объединением массивов $D_{ik}^{(k-1)}$ и $D_{kj}^{(k-1)}$. В силу того, что у нас нет отрицательных путей на к-ой итеррации массив будет иметь не больше, чем K+2 элемента.

Сложность по времени: Оценим, сколлько мы тратим времени не это улучшение. На каждой итеррации для каждой вершины мы тратим O(1) на то, чтобы слить два массива в один или же оставить тот же самый массив. Так мы делаем n^3 раз. Именно поэтому наше улучшение не влияет на асимтотику и алгоритм будет работать за $O(n^3)$.

2.

Достаточно просто посмотреть на главную диагональ матрицы $D_{ii}^{()}$. Если там будут стоять 0, то это будет означать, что в нашем графе нет пути из i в i , который был бы меньше, чем 0. А это и означает, что у нас нет цикла отрицательной длины.

3.

Алгоритм: Реализуем алгоритм дейкстры, но когда мы будем делать релаксацию из вершины, которая стоит первая в куче, будем наблюдать за тем, чтобы взятое нами ребро не оказалось отрицательным. Если во время работы алгоритма мы не нашли такого ребра, то это означает, что оно не достижимо из a и мы уже нашли наш кратчайший путь. В противом случае проведем алгоритм Дейксты от вершины U до V(мы уже знаем эти врешины), предварительно исключив отрицательное ребро(соединяет V и U). Если же полученная длина пути больше, чем модуль длины отрицательно ребра, то у нас нет цикла отрицательной длины и мы можем утверждать, что это ребро может встречаться в кратчайшем пути только один раз: тогда выберем min из кратчайшего пути из а в б без отрицательного ребра и с отрицательным ребром(для этого используем два раза алгоритм Дейкстры и найдем сначала путь из а в V, а потом из U в б)(пользуемся тем фактом, что каждый подпуть кратчайшего пути-кратчайший и отрицательное ребро встречается максимум 1 раз).

Разберем случай, когда у нас есть цикл отрицательной длины. Теперь применим два раза алгоритм Дейкстры и найдем сначала путь из а в V, а потом из U в б, чтобы понять достижимы ли вершины а и б. Если не достижимы, то отрицательное ребро не участвует в кратчайшем пути и ответ получается из первого алгоритма Дейкстры. В противном случае мы должны вывести ошибку, так как мы сможем уменьшать расстояние, если будем проходить по циклу бесконечное количество раз.
Корректность: Наше решениеразбирает 3 различных сидуации и их достаточно для того, чтобы описать всевозможные варианты:
1) Если вершина V не достижима из а. Это мы можем понять проделывая первый алгоритм Дейкстры. В этой ситуации мы просто заканчиваем алгоритм Дейкстры выводим расстояние между вершинами а и б, так как мы знаем, что в нашей компоненте связности нет ребер отрицательной длины и, следовательно, наш алгоритм будет работать корректно.
2) Если же мы запустили первый алгоритм Дейкстры и при релаксации нашли это ребро. Тогда мы запоминаем эти вершины и наше решение распадается на две принципально различных ситуации: достижима из U вершина б или нет. Это мы сможем понять, если запустим алгоритм Дейкстры из вершины U(опять же не рассматриваем отрицательное ребро). Если же вершина б не достижима, то это означает, что отрицательное ребро не участвует в поиске кратчайшего пути между а и б. Поэтому мы можем его выкинуть и первый алгоритм Дейкстры нашел верный ответ. Но что же будет, если вершина б достижима из U? Это 3 ситуация.
3)В этой ситуации есть два варианта: либо у нас есть цикл отрицательной длины, либо его нет. Это можно проверить запустив алгоритм Дейкстры из U в V. Если же полученная длина пути больше, чем модуль длины отрицательно ребра, то у нас нет цикла отрицательной длины и мы можем утверждать, что это ребро может встречаться в кратчайшем пути только один раз: тогда выберем min из кратчайшего пути из а в б без отрицательного ребра и с отрицательным ребром(мы уже знаем эти значения, так как мы запускали алгоритмы Дейкстры из а в V и из U в б)(пользуемся тем фактом, что каждый подпуть кратчайшего путикратчайший и отрицательное ребро встречается максимум 1 раз). В противном случае(если же есть цикл отрицательной длины) мы должны вывести ошибку, так как мы сможем уменьшать расстояние, если будем проходить по циклу бесконечное количество раз.
Сложность по времени: Мы используем алгоритм Дейкстры константное количество раз, поэтому асимтотика нашего алгоритма $O((\mid V \mid + \mid E \mid) \log \mid V \mid)$
4.
1) Понятно, что алгебраическая структура полукольца устроена почти

аналогично кольцу, но есть одно но: элементы полуколца могут не содержать противоположных и в этом вся проблема, ведь, как можно было заметить, в алгоритме Штрассена мы неявно используем этот факт, когда пытаемся вычесть одну матрицу из другой. Следовательно алгоритм Штрассена не работает для полуколец.

2)

Заменим все числа неравные 0 в матрице A+E на 1, а оставшиеся на 0. Тогда после возведения матрицы в степень получим какие-то числа, но мы знаем, что если число не равно 0, то это означает, что путь существует. Следовательно если мы опять проведем операцию замены чисел на 0 и 1, то мы с уверенностью можем сказать, что все операции будут стоить константу, из этого следует, что сложность по времени будет состоять из применения алгоритма Штрассена в алгоритме поиска транзитивного замыкания, основанном на быстром возведении в степень($O(n^{\log_2 7} \log n)$), и замены чисел в полученной матрице n раз.($O(n^3)$) Тогда общая сложность $O(n^3)$

Корректность доказывается аналогично тому, как мы доказывали возведение матрицы A+E в степень. Инвариантом будет- если в к-ой матрице на каком-то месте стоит 1, то это означает, что существует путь между двумя вершинами, реберная длина которого не превосходит к.(если сформулировать по друугому, то это путь длины к, который использует петли)

База: для к=1 это очевидно(ведь путь длины 1 - ребро)

Переход : Пусть верно для к-1, тогда путь длины к из i в j мы можем получить из всевозможных комбинаций $a_{ik}^{(k-1)}$ и $a_{kj}^{(1)}$. Если хотя бы один равен 1, то то мы можем составить путь, длины не более, чем к. Так как в сумме нет отрицательных чисел, то это означает, что если наше число не 0, то есть путь, иначе его нет.

5.

Проделаем поиск в глубину следующим образом: возьмем вершину и будет делать алгоритм до того момента, пока стек не станет пустым. Это означает, что мы прошли все вершины, которые достижимы из этой вершины. После этого занесем те вершины, которые побывали в стеке в матрицу транзитивности. Так сделаем для каждой вершины и в итоге получим ответ в задаче.

Корректность: Уже доказывали для поиска в глубину

Асимтотика по времени: |V| раз используем поиск в глубину(но не полный, его асимтотика по времени |E|). Общая сложность O(|V||E|).

6.

1)
a) Его сложность $O(log|V| \times (|V| + |E|) \times |V|)$. Но его можно чуть модифицировать, как и в 5 задаче, тогда сложность будет $O(log|V| \times |E| \times |V|)$.
b) В первом случае алгоритм Дейкстры будет работать некорректно(разбирались с этим в прошлый раз), а во втором случае наш алгоритм будет иметь большую асимтотику(настолько большую, что нам будет выгодней использовать алгоритм Беллмана-Форда, это мы тоже разбирали в прошлом дз).
2)
a) Рассмотрим кратчайший путь который соответствовал нашей функции w. Теперь поймем, что длина всех путей от вершины i до j увеличится на c(i)-c(j), так как во все остальные вершины мы будем сначала входить, а потом из них выходить. Поэтому наш путь останется минимальным.
b) Давайте рассмторим производьное ребро между вершинами i и j. Тогда w'(i,j)=c(i)+w(i,j)-c(j)=d(s,i)+w(i,j)-d(s,j). Поймем, что $d(s,j) \leq d(s,i)+w(i,j)$ Поэтому $d(s,i)+w(i,j)-d(s,j) \geq d(s,i)+w(i,j)-d(s,i)-w(i,j) = 0$. Что и требовалось доказать.
в) Просто добавим новую вершину и кинем от нее ребра длины 0 до остальных вершин. Теперь мы можем быть уверены в том, что мы сможем добраться до всех вершин.
г)
Алгоритм: Наш алгоритм заключается в следующем:
Шаг 1: Добавим в наш граф вершину s и ребра из нее во все остальные вершины длины 0.
Шаг 2: Проведем поиск кратчайших путей с помошью алгоритма Беллмана-Форда.
Важно! Если алгоритм находит отрицательный цикл, то сразу выводим ошибку!
Шаг 3: Изменим нашу меру на w', которая не поменяет кратчайших путей, но при этом сделает так, чтобы все ребра были положительны.
Шаг 4: Завершим наш алгоритм запусканием алгоритма Дейкстры |V| раз из каждой вершины.
Шаг 5: Составим нашу матрицу, в кооторой на месте i,j будет кратчайший путь.
Корректность: При добавлении вершины s мы не добавляем новых путей между двумя вершинами, так как ни из одной вершины графа невозможно попасть в вершину s. Следовательно мы можем искать кратчайшие пути нашего нового графа. Далее мы запускаем алгоритм Беллмана-Форда для нашего нового графа из s и, пользуясь леммой из пункта a), приводим наш граф к графу без отрицательных ребер, изменяя метрику. После чего мы можем утверждать, что алгоритм Дейкстры будет

работать корректно и мы действительно найдем кратчайшие пути в нашем новом графе между любыми двумя вершинами, которые будут соответствовать кратчайшим путям в изначальном графе в силу леммы из пункта a)(Рассмотрим кратчайший путь который соответствовал нашей функции w. Теперь поймем, что длина всех путей от вершины i до j увеличится на c(i)-c(j), так как во все остальные вершины мы будем сначала входить, а потом из них выходить. Поэтому наш путь останется минимальным.)

Сложность по времени: Добавление вершины O(|V|), алгоритм Белмана-Форда O(|E||V|), изменение меры O(|E|), алгоритм Дейкстры |V| раз $O(log|V|\times|E|\times|V|)$, составление матрицы $O(|V|^2)$.Суммарная сложность $O(log|V| \times |E| \times |V|)$

д) Теперь попытаемся улучшить наш алгоритм. Если же в нашем графе существует цикл отрицательной длины, то на втором шаге с помощью алгоритма Белмана-Форда найдем все вершины, которые учавствуют в отрицательных циклах.(достаточно посмотреть на диагонали матрицы $D^{(n)}$) Далее выкинем все эти вершины и ребра, в которых они принимают участие. В полученном графе у нас нет циклов отрицательныой длины и мы сможем корректно провести для него наш алгоритм. После этого мы должны посмотреть на достижимость. Для этого просто проведем поиск в глубину для каждой вершины из отрицательного цикла. На основе полученных данных составим массив вершин, которые достижимы из циклов отрицательной длины(это объединение результатов работы каждого алгоритма). После этого проведем поиск в глубину из оставшихся вершин и запишем в новый массив все те, из которых достижимы вершины

Составлять матрицу будем следующим образом:

1)Посмотрим на результат работы нового алгоритма, если он выдал не бесконечность, то мы должны посмотреть на результат поиска в глубину(если из этой вершины)