Алгоритмы и Структуры Данных КР1 Вариант 07

Гарипов Роман М3238

05.11.2020

Задача №1

Алгоритм

1. Запустим алгоритм Форда-Беллмана, вычислим массив $d[i]$ - длина кратчайшего пути от вершины s до вершины i

2. Далее будем работать на графе, ребра которого - ребра исходного графа лежащие на кратчайших путях, то есть добавим ребро в граф (uv), если $d[u] + e_{uv} = d[v]$. В таком графе могут быть только циклы веса ноль.

3. Сконденсируем граф, чтобы циклы весом ноль сжались в отдельные вершины, будем называть такие вершины помеченными. Те вершины в сконденсированном графе, которые были обычными вершинами в исходном назовём – непомеченными.

4. Для всех помеченных вершин, запустим из них `dfs` и запишем в массиве ответов, что для всех вершин достижимых из текущей помеченной, кратчайший путь не единственный(если достигли помеченной вершины, то понятно что надо записать все вершины которые находятся в этой компоненте сильной связности и саму компоненту сильной связности из которой стартуем `dfs`-ом тоже надо записать).

5. Осталось обработать вершины, которые недостижимы из помеченных. Для каждой такой вершины x будем запускать `dfs` по обратным рёбрам(грубо говоря будем подниматься вверх по графу) и смотреть, если из какой-то вершины можно пойти в две какие-то, то для вершины x запишем, что кратчайший путь не единственный.

Ассимптотика первого пункта – $\mathcal{O}(VE)$, пятого – $\mathcal{O}(V(V+E)) = \mathcal{O}(VE)$, остальные пункты – $\mathcal{O}(V+E)$.

Доказательство

1. Будем считать, что из стартовой вершины существуют пути во все остальные вершины графа. Понятно что для тех, для которых не существует пути, не будет существовать ни одного кратчайшего. Для всех остальных вершин кратчайшее расстоние будет существовать.

2. Доказательство корректности построения графа кратчайших путей будет показано в Задаче №2, где будет строится такой же граф кратчайших путей на неориентированном невзвешанном графе, но такое же доказательство пройдет и для нашего случая.

3. Действительно, в таком графе могут быть циклы только веса ноль, так как отрицательных нет по условию, а цикл положительного веса не может существовать в графе кратчайших путей.

4. Можем покрутиться n раз по циклу веса 0 и пойти дальше, и покрутиться $n+1$ раз по циклу веса 0, это будут разные пути, а поскольку мы работаем в графе кратчайших путей – это будут разные кратчайшие пути. Таким образом покроем все случаи, когда до вершины есть несколько кратчайших путей проходящих по циклу веса 0. Осталось обработать все случаи когда цикла на пути не будет, это и говорит нам о корректности алгоритма, так как обработаны все возможные циклы.

5. Рассмотрим случай, когда по обратному ребру из какой-то вершины x можем пойти в две или более вершины, а `dfs` по обратным рёбрам запущен из вершины y. Существует два или более пути $s \rightsquigarrow x$, так же существует $x \rightsquigarrow y$. Ну тогда существует и два или более путей $s \rightsquigarrow y$. Наш `dfs` ищет “развилки” в

графе. И он всегда их найдёт, ведь если по пути от вершины s существует развилка и там хотя бы два пути, которые потом встречаются(до остальных просто будет существовать единственный путь, если до этого не была обнаружена неединственность), то в какую-то вершину входит два или более ребра.

Задача №2

Решение с ассимптотикой $\mathcal{O}(E+V)$.

(a) Запустим `bfs` в нашем графе и выделим все кратчайшие пути.

Алгоритм

1. Предварительно запустим `bfs` на нашем графе, чтобы посчитать массив расстоний $d[i]$.
2. Далее запустим обход с очередью от вершины t, достанем очередную вершину u из очереди, пройдемся по её соседям v, если $d[u] = d[v] + e_{vu}$, то добавим ребро (vu) в граф кратчайших путей, а вершину v в очередь. Так мы добавим в граф все рёбра которые лежат хотя бы на одном кратчайшем пути и пройдя по ним можно достичь вершины t.

Доказательство

Рассмотрим момент, когда добавляем ребро (vu). Покажем, что это ребро лежит на кратчайшем пути от s до u, если выполняется описанное условие. $d[u]$ - длина кратчайшего пути до вершины u. Пусть путь с ребром (vu) на конце – не кратчайший. Тогда $\exists v' \in V : d[v'] + e_{v'u} < d[v] + e_{vu} = d[u]$, получаем противоречие, ведь $d[u]$ – кратчайший. Взяли произволное ребро (vu), значит для всех рёбер это будет справедливо.

(b) Теперь понадобится модифицированный `bfs`.

Алгоритм

1. Будем поддерживать очередь Q с таким инвариатом :

   $$\forall v \in Q \ \exists \text{ путь } s \rightsquigarrow v = way : way \text{ – префикс минимального лексикографически пути } s \rightsquigarrow t$$

   причем все вершины в очереди находятся на одинаковом расстоянии от вершины s (так будет проще доказывать корректность).
2. Положим в очередь вершину s. Запустим `bfs` по графу кратчайших путей, но немного изменим условие добавления новой вершины в очередь, а так же будем работать с двумя очередями, одна - текущий слой, а вторая - следующий слой, в которые ещё добавляем соседей вершин из текущего.
3. Формально, будем делать так: рассматриваем вершину x которую достали из очереди в цикле `bfs'a`. Пусть $min\_value = \min\limits_{xy \in E} xy_{cost}$, где xy_{cost} – число написанное на этом ребре. В очередь соответствующую новому слою, добавим всех соседей всех вершин из текущего слоя, если мы не были в очередном соседе, а так же число написанное на ребре в соседа – xy_{cost}.
4. Когда очередь с вершинами из текущего слоя станет пустой, применим операцию swap для текущей пустой очереди и очереди следующего слоя.

Доказательство

Стоит сказать, что, когда мы используем две очереди в нашем `bfs'e` по сути ничем не отличается от использования одной в обычной версии алгоритма, потому что по прежнему все вершины будут посещены в порядке неубывания расстояния от стартовой вершины.

Так будет удобнее доказывать корректность алгоритма, потому что тогда можем посмотреть на последовательность очередей, i-ая очередь соотвествует вершинам в графе по пути до которых мы получаем префикс оптимального ответа длины i.

Достаточно доказать что при корректно сформированном текущем слое(i -ый слой), мы наберём новый слой который будет содержать вершины – путь до которых будет являться оптимальным префиксом длины $i+1$

Получаем доказательство по индукции.

База: 0-ой слой, имеем пустой префикс, который является оптимальным.

Переход: Минимизоровав префикс(что уже имеем по предположению индукции) и добавив минимально возможный символ получим минимальный префикс длины $i+1$, обозначим его за $pref_{i+1}$. Пусть не так. Предположим существует префикс $pref'_{i+1}$ меньше нашего лексикографически. Он не может отличаться в одном из первых i символов, потому что тогда он не меньше $pref_{i+1}$. Тогда только в $i+1$ позиции. Но в $pref_{i+1}$ стоит минимальный символ из всех возможных, поэтому $pref'_{i+1}$ – не будет минимальным, противоречие.

Значит $pref_{i+1}$ – так выглядят все префиксы длины $i+1$, как раз все такие мы и добавим в очередь для следующего слоя.

Поскольку наш обход запускается на графе кратчайших путей, мы выберем минимальный префикс , что нас и просили.

Задача №3

Обозначим за X – правую долю, за Y – левую. Знаком равностильности $\Leftrightarrow$ будем обозначать равносильность максимизации двух выражений.

$$|A| - |F(A)| \Leftrightarrow |A| - |F(A)| + |X| - |X| \Leftrightarrow |A| + |X \setminus f(A)| - |X| \Leftrightarrow |A| + |X \setminus f(A)|$$

Последнее преобразование справедливо, так как $|X|$ – фиксированная величина, которая не зависит от выбора A.

Последнее выражение описывает размер независимого множества, действительно, выбрали какие-то вершины в левой доле, взяли всех кроме соседей выбранных вершин в правой доле. Это соотвествует оптимальному набору вершин при фиксированном множестве A. Тогда понятно, что максимизировав это значение, получим максимальное независимое множество. Поскольку имееют место указанные равносильные преобразования при максимизации, значение $|A|-|F(A)|$ – максимально, когда в качестве A выбираются все вершины левой доли, лежащие в максимальном независимом множестве.

Задача №4

Решение с ассимптотикой $\mathcal{O}(E+V)$.

Будем считать что имеем дело со связным графом, на случай несвязных графов алгоритм и доказательство обобщаются без особых проблем.

Лемма 1. *Если каждую компоненту рёберной двусвязносъти в графе сжать в отдельную вершину, то получим дерево.*

Доказательство. Действительно, пусть не так. Тогда имеем в сжатом графе цикл. Поскольку какие-то вершины образуют цикл, они должны быть в одной компоненте рёберной двусвязности. Но мы уже сжали их, и эти вершины оказались в разных. Противоречие. □

Эта лемма говорит нам о том, что можно рассматривать компоненты рёберной двусвязности по отдельности, так как циклов соединяющих две разные компоненты не может быть.

Лемма 2. *Компонента рёберной двусвязности является рёберным кактусом $\Leftrightarrow$ в компоненте не более одного цикл.*

Доказательство. ($\Rightarrow$) Если компонента – вершинный кактус, она будет содержать не более одного цикла. Пусть их два или более. Возьмём любые два. Пусть они не пересекаются, тогда эти два цикла либо связаны ребром, либо нет, в обоих случаях они были бы в разных компонента рёберной двусвязности. Значит они пересекаются хотя бы по одному вершине. Возьмём эту вершину, она принадлежит двум циклам, по определению вершинного кактуса такого быть не может. Противоречение. Значит больше одного быть не может.

($\Leftarrow$) Имеем не более одного цикла в компоненте, все вершины в одной компоненте, значит только этому единственному циклу и принадлежит каждая вершина, либо не принадлежать никакому – определение выполняется. $\square$

Лемма 2 – критерий для проверки компоненты на "вершинную кактусовость". Лемма 1 – говорит что можно проверять компоненты по отдельности. Получаем алгоритм который позволяет является ли граф вершинным кактусом.

Сожмём вершины рёберной двусвязности в отдельные вершины за $\mathcal{O}(E+V)$ операций. Проверим что в каждой ровно один цикл(степень каждой вершины должна равняться двум) за $\mathcal{O}(E+V)$.

В случае несвязного графа надо проверить то же самое в каждой компоненте по отдельности.

Задача №5

Решение с ассимптотикой $\mathcal{O}(E+V^4)$

Алгоритм

1. Сожмём кратные рёбра минимального веса между всеми парами вершин и запоминим сколько таких рёбер с минимальным весом существует между каждой фиксированной парой вершин.

2. Запустим Алгоритм Флойда, найдем кратчайшее расстояние от каждой вершины до любой другой за $\mathcal{O}(V^3)$

3. Переберём пару вершин u и v, между которых хотим посчитать количество рёбер лежащих хотя бы на одном кратчайшем пути $u \rightsquigarrow v$. Перерём пару вершин x и y, между которыми возьмём все рёбра минимального веса. Если $d[u][x] + xy_{cost} + d[y][v] = d[u][v]$, то прибавим к ответу $ans[u][v]$ количество ребёр минимального веса ведущих из вершины x в вершину y.

Таким образом, получили матрицу, в которой записан ответ для каждой пары вершин.

Доказательство

Покажем корректность 3-го пункта. Конкретно, покажем, что наш критерий для принадлежности ребру кратчайшему пути $u \rightsquigarrow v$ правильный. В Задаче №2 мы показали, что если при переходе по ребру кратчайший путь увеличивается на стоимость его ребра, то ребро принадлежит какому-то кратчайшему пути. Точно такое же доказательство будет работать и для нескольких последовательных рёбер. Ведь можно представить что мы сначала добавили к пути $d[u][x]$ ребро xy, а потом к получившемуся пути - путь $y \rightsquigarrow v$ Поэтому предложенный критерий является корректным.