Филиппова Анастасия

Домашнее задание 8

1

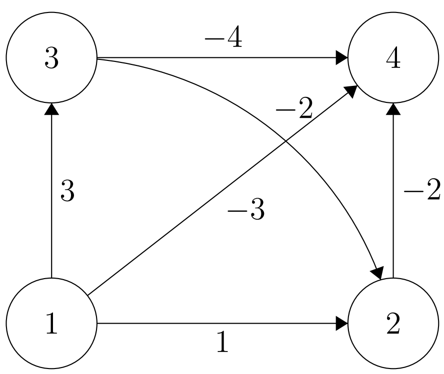
Пусть G — исходный граф. Пусть данное ребро соединяет пару вершин (u, v) в графе. Определим, лежит ли ребро (u, v) хотя бы на одном кратчашем пути из a в b. Для этого найдем кратчайшее расстояние из вершины a в b, пусть оно равно $r(a, b)$. Далее найдем $r(a, u), r(a, v), r(b, u), r(b, v)$. Если $r(a, u) + r(b, v) + 1 = r(a, b) \vee r(a, v) + r(b, u) + 1 = r(a, b)$, тогда данное ребро лежит хотя бы на одном из кратчайших путей из a в b.
Сложность: так как граф не взвешенный, с помощью алгоритма BFS находим кратчайшие расстояния между парами вершин за $O(|E| + |V|)$, арифметические операции за $O(1)$. Общая сложность: $O(|E| + |V|)$.
Корректность: докажем, что ребро (u, v) лежит хотя бы на одном крачайшем пути тогда и только тогда, когда $[r(a, u) + r(b, v) + 1 = r(a, b)] \vee [r(a, v) + r(b, u) + 1 = r(a, b)]$.
1) $\Longrightarrow$ Пусть ребро (u, v) лежит на кратчайшем пути и $[r(a, u) + r(b, v) + 1\overline{r}(a, b)] \wedge [r(a, v) + r(b, u) + 1\overline{r}(a, b)]$. Но тогда, очевидно, рассматриваемое ребро не является частью кратчайшего пути. Противоречие.
2) $\Longleftarrow$ Выполнено условие $[r(a, u) + r(b, v) + 1 = r(a, b)] \vee [r(a, v) + r(b, u) + 1 = r(a, b)]$, тогда очевидно, что ребро лежит на кратчайшем пути.

2

Запустим BFS из вершины $s \in G$. Чтобы получить кратчайшее расстояние между данной вершиной и остальными, достаточно проверять при добавлении вершин в очередь, вес текущего ребра равен нулю или нет. Если равен — добавляем в начало очереди, если нет — в конец. Таким образом, сначала расмотрим все пути из рассматриваемое вершины нулевой длины (и больше эти ыершины не посети).

4

1)Если запустить алгоритм Флойда-Уоршела в графе с циклами отрицательного веса, тогда алгоритм будет работать не корректно, так как будут существовать такие вершины, между которыми не существует кратчайшего пути ($-\infty$). Моцифицируем алгоритм, чтобы найти все такие пары вершин (i, j). Очевидно, что кратчайший путь (i, j) будет неопределен, если существует такая вершина u, достижимая из i и j, что $d[u][u] < 0$. Тогда на 3 цикле алгоритма (при выборе вершин, через которые может проходить кратчайший путь) будем также проверять условия:
if $(d[u][u] < 0 \wedge d[i][k] < \infty \wedge d[j][k] < \infty) = 1$,
then $d[i][j] = -\infty$.
Корректность алгоритма следует из того, что u лежит в цикле отрицательного веса тогда и только тогда, когда $d[u][u] < 0$, что, на мой взгляд, очевидно (то есть из i вершины можем пройти в j, проходясь по циклу отрицательного веса, но по нему ведь можно пройтись сколько угодно раз, если устремить количество к бесконечности, получем, что расстояние между i и j стремится к $-\infty$).
2)



$$D^{(0)} = D^{(1)} = D^{(2)} = D^{(3)} = D^{(4)} = \begin{bmatrix} 0 & 1 & 3 & -3 \\ \infty & 0 & \infty & -2 \\ \infty & -2 & 0 & -4 \\ \infty & \infty & \infty & 0 \end{bmatrix}.$$

(рассчитываем матрицы по алгоритму для всех пар $i, j \in 1...4$, на каждом шаге увеличивая k на 1: $d[i][j]^k = min(d[i][k]^{k-1} + d[k][j]^{k-1})$.

5

$$1)\ \begin{array}{cc} & \begin{matrix} d & c & h & e & g & f & b & a \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & 7 & 14 & \infty & \infty & \infty & \infty & \infty \\ 0 & 7 & 11 & 9 & 16 & \infty & 15 & \infty \\ 0 & 7 & 11 & 9 & 13 & 16 & 15 & 19 \\ 0 & 7 & 11 & 9 & 13 & 14 & 15 & 19 \end{bmatrix} \end{array}.$$

Данную матрицу можно получить если итерироваться по всем ребрам (u, v), сравнивая величины $A[k-1][u]$ и $A[k-1][v] + w(v, u)$. После 4 фазы метки меняться не будут.
2) Каким должен быть граф, чтобы метки менялись на каждой фазе, при условии, что все ребра равны 1? Чтобы выполнялось написанное выше условие, необходимо и достаточно, чтобы в графе хотя бы один кратчайший путь был длины $|V| - 1$. От стартовой вершины u до v должно лежать $|V| - 1$ ребро, причем путь должен быть кратчайшим. Тогда, получается, что такие графы — простые цепи из V вершин.

6

Так как введен такой частичный порядок, получается, что можем занумеровать вершины таким образом, чтобы в отсортированном списке вершин по номеру ребра шли только от вершин с меньшим номером к вершинам с большим. Таким образом получаем, что граф, на котором задано такое отношение порядка является ациклическим, следовательно, каждая вершина образует собственную компоненты сильной связности. Тогда число компонент равно числу вершин. Ответ: $|V|$.

7

1) Предположим противное, $f(u) < f(v)$, причем v достижима из u, u не достижима из v. Рассмотрим возможные варианты:
1.$d(u) < d(v)$. То есть, получается, сначала посетили вершину u, закрыли ее, затем вершину v. Но так как мы знаем, что вершина v достижима из u, то после открытия вершины u, мы должны были открыть вершину v так как она достижима и ранее была не открыта. Противоречие.
2.$d(u) > d(v)$. То есть открыли вершину v, затем вершину u, закрыли u, затем закрыли v. Но тогда, получается $d(v) < d(u) < f(u) < f(v)$, из чего следует, что u достижима из v. Получили противоречие. Следовательно, $f(u) > f(v)$.
2) Будем при закрытии вершины класть ее в начало некоторого списка, тогда на выходе получим список вершин, отсортированных по времени закрытия (от большего к меньшему). К моменту выхода из вызова $DFS(v)$ все вершины, достижимые из v уже посещены обходом (это следует из доказанно выше леммы). Следовательно, получаем, что при таком порядке вершин, ребра могут идти только от больших вершин к меньшим.

8

1) Пусть дан граф G, G' — инвертируемый граф. Докажем, что $v, u \in K \Longleftrightarrow v, u \in K'$, где K, K' — компоненты сильной связности в графах G и G' соответственно.
Пусть вершины u и v лежат в одной компоненте сильной связности графа G, то есть есть пути в графе (u, v) и (v, u). Очевидно, что в графе G' u и v лежат в одной компоненте сильной связности, так как в инвертированном графе, очевидно, будут пути $(v, u), (u, v)$. Доказали, что $v, u \in K \Longrightarrow v, u \in K'$. В обратную сторону: пусть в графе G' u, v лежат в разных компонентах сильной связности. Предположим противное: v, u лежат в разных компонентах сильно связности графа G, получаем противоречие, так как выше доказано обратное.
2) Алгоритм Косараю:

1. Вызываем поиск в глубину на графе G.
2. Получаем список вершин, отсортированный по убыванию времени закрытия вершин.
3. Инвертируем граф (G').
4. На графе G' запускаем DFS, но выбираем вершины из полученного выше списка.
Сложность: $O(|V| + |E|)$ — сложность DFS. Дополнительная память: $O(|V|)$ — храним список времени закрытия каждой вершины.
3) Конденсация графа — ациклический граф (нет обратных ребер), следовательно, запустив DFS на конденсации и отсортировав вершины по времени закрытия, получим топологически отсортированный граф.