Динамическое транзитивное замыкание

Кузнецов Илья Александрович
371 группа

18.05.2022

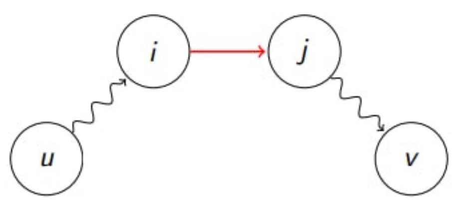
1. На лекции мы научились поддерживать инкрементальное транзитивное замыкание ориентированных графов. Придумайте алгоритм декрементального транзитивного замыкания, работающий за $O(n^2(m+n))$ суммарно за все обновления.

Решение. Инициализация: строим начальную матрицу достижимости M (выполняем транзитивное замыкание), а также списки смежности для исходного графа и инвертированного графа (направление ребер в другую сторону).

При реализации удаления ребра (i, j) нас будут интересовать все такие вершины v, которые стали недостижимы из i после удаления ребра, но которые были достижимы из j до удаления. Для этого используем поиск в глубину (DFS).

Из всех найденных вершин v запустим DFS на графе с обратными ребрами. Так мы определим достижимость каждой вершины u из v. Для всех вершин u, не достижимых из v, необходимо показать это в матрице достижимости $M[u, v] = 0$.

Рассмотрим алгоритм на примере следующего графа.



Если удалить ребро (i, j) в данном графе, то вершиной, которая стала недостижима из i, но была достижима из j, является вершина v. После запуска DFS на ней в инвертированном графе недостижимыми вершинами будут u и i, поэтому в матрице достижимости нужно будет показать $M[u, v] = 0$ и $M[i, v] = 0$.

Algorithm 1 Декрементальное обновление матрицы достижимости

function REMOVE(i, j)
    $adj[i].remove(j)$
    $adj_{rev}[j].remove(i)$
    $res \leftarrow dfs(adj, i)$
    for $v : res[v] = 0 \wedge M[j, v] = 1$ **do**
        $res_{rev} \leftarrow dfs(adj_{rev}, v)$
        for $u \in V$ **do**
            $M[u, v] \leftarrow res_{rev}[u]$

Доказательство. Оценим сложность алгоритма. Сложность DFS — $O(n+m)$ времени, а максимальное число вызовов функции на все обновления равно m, где m — число ребер (если удалятся все ребра в графе).

Строки 6-8 имеют сложность $O(n + m)$ времени, а в худшем случае, вершин, соответствующих условию в строчке 5, может быть порядка n, то есть мы получаем $O(n(n+m))$ времени в строчках 5-8 на одно удаление ребра. А при m вызовах — $O(mn(n + m))$ времени.

Если учесть оценку, что $m <= n^2$, то итоговая сложность равна $O(n^3(m+n))$ времени. Но, на самом деле строки 6-8 не будут выполняться n^3 раз, а могут выполниться не более n^2 раз, так как алгоритм декрементальный и если $M[i, j]$ однажды стало 0, значит оно никогда не станет 1.

Таким образом, суммарная сложность на все обновления составляет $O(n^2(m + n))$ времени. $\square$