Динамическая транзитивное замыкание

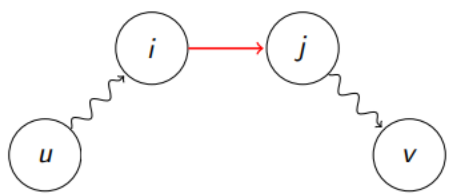
Мирошников Владислав
371 группа

19.05.2022

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

Решение

Инициализация: строим начальную матрицу достижимости M (выполняем транзитивное замыкание), а также списки смежности для исходного графа и реверснутого (направление ребер в другую сторону). Общая идея следующая: если удалилось ребро (i, j), то необходимо обработать вершины v, которые стали недостижимы из i после удаление ребра (i, j) и при этом были достижимы в графе из вершины j до удаления ребра. Для таких вершин v необходимо запустить DFS на реверснутом графе, чтобы посчитать достижимость каждой вершины из данной вершины v. Далее для каждой такой вершины u, которая недостижима из v в реверснутом графе нужно в матрице достижимости $M[u, v] := 0$. В качестве примера рассмотрим следующий граф:



В случае, если удалится ребро (i, j) в данном графе, подходящей вершиной, которая стала недостижима из i, но была достижима из j является v. После запуска DFS на ней на реверснутом графе недостижимыми вершинами будут u и i и в матрице достижимости нужно будет $M[u, v] := 0$ и $M[i, v] := 0$.

Algorithm 1 Декрементальное обновление матрицы достижимости

function REMOVE(i, j)
    $adjacencyList[i].remove(j)$
    $adjacencyList_{reverse}[j].remove(i)$
    $dfsResult = dfs(adjacencyList, i)$
    for $v : dfsResult[v] = 0 \wedge M[j, v] = 1$ **do**
        $dfsResult_{reverse} = dfs(adjacencyList_{reverse}, v)$
        for $u \in V$ **do**
            $M[u, v] \leftarrow dfsResult_{reverse}[u]$

Так как алгоритм декрементальный, то если M[i, j] однажды стало 0, значит оно никогда не станет 1 (по аналогии с инкрементальным, где если стало 1, то не станет 0).

Оценим сложность алгоритма: сложность DFS равна $O(n + m)$, а максимальное число вызовов функции на все апдейты равно m, где число ребер (если удалятся все ребра в графе). Таким образом суммарная сложность на все апдейты будет $O(m(n+m+n))$. При этом если учесть оценку, что $m <= n^2$, то итоговая сложность равна $O(n^2(m + n))$.