Динамическая транзитивное замыкание

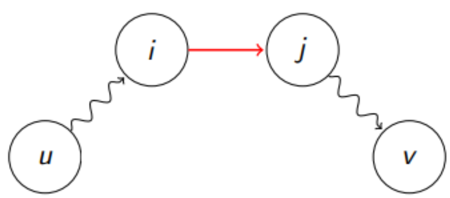
Мирошников Владислав
371 группа

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 реверстнутом 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

1: **function** REMOVE(i, j)
2: $\quad adjacencyList[i].remove(j)$
3: $\quad adjacencyList_{reverse}[j].remove(i)$
4: $\quad dfsResult = dfs(adjacencyList, i)$
5: $\quad$ **for** $v : dfsResult[v] = 0 \land M[j, v] = 1$ **do**
6: $\quad\quad dfsResult_{reverse} = dfs(adjacencyList_{reverse}, v)$
7: $\quad\quad$ **for** $u \in V$ **do**
8: $\quad\quad\quad M[u, v] \leftarrow dfsResult_{reverse}[u]$

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). Строки 6-8 имеют сложность $O(n+m)$, а в худшем случае, вершин, соответствующих условию в строчке 5, может быть порядка n, то есть мы получаем $O(n(n+m))$ с 5-8 строчку на одно удаление ребра и при m вызовах $O(mn(n+m))$. При этом если учесть оценку, что $m <= n^2$, то итоговая сложность равна $O(n^3(m+n))$. Но, на самом деле строки 6-8 не могут выполняться n^3 раз, а могут выполниться не более n^2 раз из-за декрементальности алгоритма. Таким образом, итоговая суммарная сложность на все апдейты будет $O(n^2(m+n))$.