Динамическое транзитивное замыкание

Фаст Никита

4 мая 2023 г.

1 Формулировка задачи

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

2 Решение

Решение основано на статье[TI83].

Будем поддерживать матрицу достижимости M. Также будем поддерживать матрицу смежности ориентированного графа adj и матрицу смежности реверснутого графа (т.е. у которого каждое ребро повернуто в противоположную сторону) adj_{rev}.

Идея алгоритма в том, чтобы рассмотреть множество вершин $target$, которые достижимы из вершины v и которые более недостижимы из вершины u по-причине удаления ребра (u, v). Таким образом $target = \{y | y \in not\_reachable\_from\_u \text{ and } M[v, y] = 1\}$. Множество вершин недостижмых из u – $not\_reachable\_from\_u$ может быть вычислено с помощью обхода графа в глубину из вершины u следующим образом: {i for i, x in enumerate($DFS(adj, u)$) if x == 0}.

Для каждой вершины $y \in target$ и каждой вершины $x \in V(G)$ требуется установить, достижима ли y из x, а после положить в $M[x, y]$ соответствующее значение. Для проверки того, достижима ли вершина y из вершины x воспользуемся тем, что данный вопрос эквивалентен вопросу достижимости вершины x из вершины y в реверснутой версии графа. Это позволяет применить алгоритм обхода в глубину на вход которому подается матрица смежности реверснутого графа и вершина y с которой начинается обход $DFS(adj_{rev}, y)$.

Таким образом для удаления ребра (u, v) применяется следующая процедура.

Algorithm 1 DELETE(u, v)

$adj[u, v] = 0$
$adj_{rev}[v, u] = 0$
$not\_reachable\_from\_u =$ {i for i, x in enumerate($DFS(adj, u)$) if x == 0}
for y in $not\_reachable\_from\_u$ **do**
    if $M[v, y] == 1$ **then**
        $reachable\_from\_y = DFS(adj_{rev}, y)$
        for x in $V(G)$ **do**
            $M[x, y] = reachable\_from\_y[x]$
        end for
    end if
end for

Докажем, что вышеописанный алгоритм имеет временную сложность $O(n^2(m+n))$ суммарно на все апдейты.

1. DFS работает за $O(m+n)$

2. Т.к. это декрементальный алгоритм, то как только ячейка матрицы M изменила значение с 1 на 0, то значение в ячейке будет 0 все оставшееся время.

3. Вычисление $M[x, y]$ для всех x (строки 7-8) будет происходить не более n^2 раз, поскольку в ходе каждого вычисления хотя бы один элемент M (тот самый M[u, y] поскольку по условию строки 4 y не достижим из u) изменит значение с 1 на 0, а всего в матрице n^2 элементов.

4. Строка 5 исполняется за константу, строка 6 за $O(m + n)$ поскольку используется DFS, строки 7-8 за $O(n)$. В итоге строки 5-8 исполняются за $O(m + n)$.

5. С учетом предыдущего пункта суммарная временная сложность на все апдейты будет равна $O(n^2(m + n))$

Список литературы

[TI83] N. Katoh T. Ibaraki. On-line computation of transitive closure for graphs. 1983.