Динамическая связность

Мирошников Владислав
371 группа

17.05.2022

1 (a). Придумайте рекурсивную процедуру $fall(v)$, которая для вершины v, такой, что $N_1(v) = \varnothing$, "роняет"v на правильный уровень BFS-дерева, корректно обновляет уровни соседей v и "роняет" те вершины, чей уровень изменился при падении v.

Решение

Считаем, что перед вызовом рекурсивной процедуры $fall(v)$ выполняется проверка на непустоту $N_2(v)$. В противном случае будет образовываться новая компонента связности, то есть у нас фактически произойдет отделение поддерева от построенного BFS-дерева. Если при этом мы хотим также по условию задачи отвечать на запрос, каково расстояние $d(s, v)$ при фиксированной вершине-источнике s, то в таком случае расстояние до любой вершины из новой компоненты связности будет ∞, так как эта вершина станет недостижима из вершины-источника s. Для этого, например, можно задать level у всех вершин новой компоненты связности равный -1 и за константное время запроса к вершине определять, достижима ли она.

Algorithm 1

function FALL(v)
    $l(v) \leftarrow l(v) + 1$
    for $u \in N_2(v)$ **do**
        $N_2(u) \leftarrow N_2(u) \setminus \{v\}$
        $N_3(u) \leftarrow N_3(u) \cup \{v\}$
    for $u \in N_3(v)$ **do**
        $N_1(u) \leftarrow N_1(u) \setminus \{v\}$
        $N_2(u) \leftarrow N_2(u) \cup \{v\}$
    $N_1(v) \leftarrow N_2(v)$
    $N_2(v) \leftarrow N_3(v)$
    $N_3(v)_{old} \leftarrow N_3(v)$
    $N_3(v) \leftarrow \varnothing$ ▷ При вызове $fall$ от детей происходит заполнение
    for $u \in \{w \mid w \in N_3(v)_{old}$ and $N_1(w) = \varnothing\}$ **do**
        $fall(u)$

1 (b). Докажите, что если в графе n вершин и m рёбер изначально, на все обновления суммарно при удаления m рёбер уйдет время $O(mn)$.

Доказательство. Положим степень вершины $v := deg(v)$. Обработка одной вершины внутри рекурсивной процедуры $fall(n)$ занимает $O(deg(v))$ времени. Это обусловлено тем, что в алгоритме нам фактически нужно обработать все инцидентные для данной вершины ребра, что и есть по определению степень вершины. Далее оценим максимальное количество вызовов процедуры $fall$, инициированных вершиной v. В случае, если BFS-дерево почти вырождается в список, то есть его высота сравнима с n, то максимальное количество вызовов процедуры сравнимо с данной величиной n. Таким образом, работа процедуры $fall(n)$ для вершины v занимает $O(n\ deg(v))$ времени, а удаление ребра занимает $O(1)$ времени. В итоге получаем следующее выражение:

$O(m + \sum_{v \in V} n\ deg(v)) = O(m + n \sum_{v \in V} deg(v)) = O(m + nm) = O(mn)$ □

1 (c). Пусть вместо всего BFS-дерева нам разрешено хранить только BFS-дерево с d уровнями, т.е. структура будет поддерживать только расстояния до вершин v, такие, что $d(s, v) \le d$. Докажите, что суммарное время на все апдейты в этом случае равно $O(md)$.

Доказательство. Положим степень вершины $v := deg(v)$. Далее оценим максимальное количество вызовов процедуры $fall$, инициированных вершиной v. В случае, если BFS-дерево почти вырождается в список, то есть его высота сравнима с d (d — количество уровней BFS-дерева), то максимальное количество вызовов процедуры сравнимо с данной величиной d. Таким образом, работа процедуры $fall(n)$ для вершины v занимает $O(d\ deg(v))$ времени, а удаление ребра занимает $O(1)$ времени. В итоге получаем следующее выражение:

$O(m + \sum_{v \in V} d\ deg(v)) = O(m + d \sum_{v \in V} deg(v)) = O(m + dm) = O(md)$ □