Динамическая связность

Салью Артур
371 группа

Задача 1

Дан неориентированный невзвешенный граф $G = (V, E)$, необходимо уметь решать задачу *динамического декрементального SSSP* из фиксированной вершины-источника s. Нужно поддерживать запросы вида: дана вершина v, каково расстояние $d(s, v)$? Так как алгоритм декрементальный, рёбра только удаляются (без вставок).

Для начала мы препроцессим граф следующим образом: посчитаем BFS-дерево с корнем в s (просто запустим BFS из вершины s, и выпишем получившееся дерево). Каждой вершине v получившегося дерева присвоим уровень $l(v)$, значение которого есть расстояние от вершины s ($d(s, v)$). Очевидно, что $l(s) = 0$. С этим деревом будем работать как со структурой данных.

Также BFS посчитает для каждой вершины посчитает нам три множества её соседей N_1, N_2, N_3. Пусть $l(v) = i$, тогда $N_1(v)$ — соседи v, имеющие уровень $i - 1$; N_2 — соседи v с уровнем i; N_3 — соседи v с уровнем $i + 1$.

Задача 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. Пусть мы удаляем из графа ребро (u, v). Если $l(u) = l(v)$, то удаление ребра не меняет расстояния от s, а значит нужно просто удалить v из $N_2(u)$ и u из $N_2(v)$. Пусть не умаляя общности $l(v) = i$ и $l(u) = i - 1$. Нам нужно удалить u из $N_1(v)$ и v из $N_3(u)$. Случай когда после удаления $N_1(v) \neq \varnothing$ нам неинтересен, так как расстояния из s до вершин остаются неизменными (кратчайший путь из s в v в данном случае будет проходить через ребро (u', v), где $u' \in N_1(v)$).

Положим $N_1(v) = \varnothing$. Вершина v должна "провалиться" по дереву вниз на новый уровень. Более того, провалиться должны и все вершины $w : N_1(w) = \{v\}$. Ниже приведен код, реализующий процедуру "падения" уровня вершины.

Algorithm 1

```
procedure FALL(v)
    level[v] ← level[v] + 1
    for w ∈ N_2(v) do
        N_2(w).remove(v)
        N_3(w).add(v)
    for w ∈ N_3(v) do
        N_1(w).remove(v)
        N_2(w).add(v)
    N_1(v) ← N_2(v)
    N_2(v) ← N_3(v)
    N_3(v) ← ∅
    for u ∈ {w | w ∈ N_2(v) and N_1(w) = ∅} do
        fall(u)
```

Пояснение: Дети вершины v становятся с ней на одном уровне, а вершины, с которыми v была на одном уровне, теперь становятся её предками. Так же особое внимание представляет ситуация, когда удаленное ребро (u, v) является мостом. Так как мы рассматриваем только декрементальную задачу, то можем считать $d(s, w) = \inf$ для всех $w \in G'$, где G' новая компонента связности, образованная удалением (u, v).

Задача 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. При обработке вершины v мы рассматриваем все инцидентные ей ребра. Максимальное число вызовов процедуры, инициированных падением данной вершины достигается, когда высота BFS-дерева сравнима с n. Принмая сложность удаления ребра константной, получаем верхнюю оценку $O(\sum_{v \in V} n \ deg(v)) = O(nm)$ □

Задача 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 $d(s, v) \leq d$. Докажите, что суммарное время на все апдейты в этом случае равно $O(md)$.

Доказательство. Аналогично предыдущему пункту, однако здесь число вызовов процедуры не может превышать количество уровней d BFS-дерева. Таким образом, верхняя оценка — $O(md)$ □