Алгоритмы. Домашнее задание №11

Горбунов Егор Алексеевич

4 декабря 2015 г.

Задача №1 (online-соединение вершин)

Задача: *Изначально в графе G $|V|$ вершин и нет рёбер. Нужно online обрабатывать запросы*

- *соединить ребром пару вершин*
- *по вершине узнать размер связной компоненты, в которой вершина лежит и самую лёгкую вершину в это компоненте*

Амортизированное время на запрос — $\mathcal{O}(\alpha(|V|))$

Решение: Для решения будем хранить вершины в СНМ с использованием эвристики сжатия путей и объединения по рангу. Вначале для каждой $v \in V(G)$ выполним $MakeSet(v)$:

procedure $\textsc{MakeSet}(v)$
  $parent(v) \leftarrow v$
  $size(v) \leftarrow 1$ // размер компоненты связности, в которой v лежит
  $lightest(v) \leftarrow v$ // самая лёгкая вершина в компоненте, где лежит v

Изначально у нас $|V|$ компонент связности размера 1. Каждое множество — дерево, которое содержит вершины одной компоненты связности. Будем всегда хранить валидную информацию о компоненте (размер и самую лёгкую вершину) в корне дерева, соответствующего этой компоненте. Изначально ($|E(G)| = 0$) вся информация валидна по построению процедуры $MakeSet$. Научимся правильно её обновлять при добавлении ребра (добавление ребра есть объединение деревьев — компонент связности)...

procedure $Connect(v, u)$
  $c_v,\ c_u \leftarrow Find(v),\ Find(u)$ // стандартная реализация $Find$ со сжатием путей
  if $c_v = c_u$ **then return**
  if $size(c_v) > size(c_u)$ **then**
    $swap(c_u, c_v)$
  $parent(c_v) \leftarrow c_u$
  $size(c_u) \leftarrow size(c_v) + size(c_u)$
  $lightes(c_u) \leftarrow \min_{w(v)}(lightest(c_u), lightest(c_v))$ // выбор вершины с min весом

Таким образом для добавления ребра (v, u) вызывается процедура $Connect(v, u)$. А для ответа на запрос про размер компоненты, которой принадлежит вершина v нужно найти $c_v = Find(v)$ и в качестве ответа вернуть $size(c_v)$. Аналогично для самой лёгкой вершины, только возвращаем $lightet(c_v)$. Собственно мы добавили в стандартную реализацию СНМ лишь дополнительное свойство $lightest$...В качестве ранга используем размер компоненты связности. Таким образом известно, что все операции работают за $\mathcal{O}(\alpha(|V|))$, где α — обратная функция Аккермана. ■

Задача №2 (Существование пути по рёбрам не тяжелее x)

Задача: *Дан неор. граф $G(V, E)$ с весами на рёбрах и n запросов вида (v, u, x) — существует ли путь между вершинами v и u по рёбрам не тяжелее x? В offline нужно ответить на эти запросы за $\mathcal{O}(|E|\log|V| + n\log n + n\alpha(|V|))$.*

Решение: Рассмотрим какой-то один запрос (v, u, x). Давайте удалим из графа G все рёбра, вес которых $> x$. Но тогда, если вершины v и u попали в разные компоненты связности в этом новом графе, то ответ на запрос, соответственно, «нет», иначе ответ «да», т.е. путь из v в u по рёбрам веса $\leq x$ существует.

Окей. Теперь нам нужно как-то быстро уметь находить компоненты связности графа, если из него выкидываются рёбра более определённого веса. Воспользуемся тем, что все запросы (v, u, x)нам известны и будем последовательно на них отвечать по мере увеличения x. Отсортируем все запросы по мере увеличения x: (v_i, u_i, x_i), $x_{i-1} \leq x_i$. Рассмотрим изначально граф на вершинах из V, но без рёбер. Вспомним задачу 1 и будем последовательно добавлять в граф рёбра, по мере увеличения их веса. Как только вес рассматриваемого для добавления ребра стал больше x_1, так сразу заканчиваем добавлять и отвечаем на запрос (v_1, u_1, x_1), т.к. на данный момент в нашем строящемся графе содержатся лишь те рёбра, чей вес $\leq x_1$. На запрос мы отвечаем за $\mathcal{O}(\alpha(|V|))$ в силу того, что каждая компонента связности у нас есть множество в СНМ и если $Find(v_1) = Find(u_1)$, то путь есть, а иначе его нет. Далее продолжим добавлять рёбра в граф (по возрастанию весов рёбер) и остановимся, когда вес рассматриваемого для добавления ребра стал больше x_2, и.т.д...

Таким образом нам потребовалось:

- Отсортировать запросы по возрастанию x: $\mathcal{O}(n\log n)$
- Отсортировать рёбра по весу: $\mathcal{O}(|E|\log|E|)$
- Ответить последовательно на n запросов выполнив на каждый запрос 2 вызова процедуры $Find$: $\mathcal{O}(n\alpha|V|)$

Итого: $\mathcal{O}(|E|\log|V| + n\log n + n\alpha(|V|))$. ■