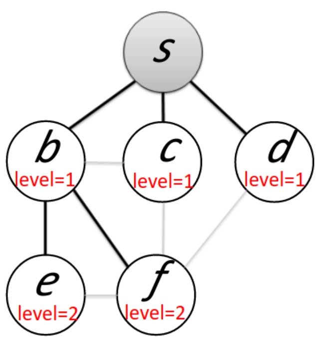
ДЗ по теме "Динамическая связность"

10 апреля 2023 г.

- **Мягкий дедлайн:** 17.04.2023, 23:59
- **Жёсткий дедлайн:** 21.05.2023, 23:59

По умолчанию число вершин во входном графе равно n, а число рёбер — m.



1. **(5 баллов)** Давайте научимся решать задачу динамического *декрементального* SSSP (single source shortest paths) на неориентированном невзвешенном графе, используя идею с уровнями немного в другом ключе.

   На вход дан граф $G = (V, E)$ и его фиксированная вершина-источник s. Нужно поддерживать запросы вида: дана вершина v, каково расстояние $d(s, v)$? Так как алгоритм декрементальный, рёбра только удаляется (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 ($d(s, v)$) (см. картинку). 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 $i + 1$.

Наш алгоритм должен поддерживать удаления с помощью обновлений множеств N_1, N_2, N_3 для некоторых вершин и изменений их уровня в BFS-дереве. На запрос у нас уходит константное время — достаточно спросить у вершины её уровень.

Рассмотрим, что происходит, если мы удаляем из графа ребро (u, v). Если $l(u) = l(v)$ (вершины на одном уровне), то удаление данного ребра не меняет расстояния от s, значит нужно просто удалить v из $N_2(u)$ и u из $N_2(v)$. Пусть $l(v) = i$ и $l(u) = i - 1$ (другой случай работает симметрично). Нам нужно удалить u из $N_1(v)$ и v из $N_3(u)$. Если во множестве $N_1(v)$ остались вершины, то расстояния не изменились (подумайте, почему). Если же $N_1(v)$ стало пустым, то v должна"провалиться" вниз на новый уровень. И более того, если v провалилась, то все вершины w, для которых $N_1(w) = \{v\}$ должны провалиться, и так далее!

(a) Придумайте рекурсивную процедуру $fall(v)$, которая для вершины v, такой, что $N_1(v) = \emptyset$, "роняет" v на правильный уровень BFS-дерева, корректно обновляет уровни соседей v и "роняет" те вершины, чей уровень изменился при падении v. Предполагаем, что до процесса падения происходит проверка на связанность.

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

(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 $O(md)$.

2. **(10 баллов)** На лекции мы научились поддерживать декрементальную динамическую связность неориентированного графа за $O(\log^2 n)$ амортизированно. Придумайте, как усовершенствовать алгоритм, чтобы научиться поддерживать декрементально (только удаления рёбер) минимальный остовный лес во взвешенном неориентированном графе также за $O(\log^2 n)$ амортизированно. Внимание: значения весов уникальны для каждого ребра (если у двух рёбер вес одинаковый, можно присвоить им разные веса с учётом их уникального идентификатора).

   Какими операциями мы должны дополнить структуру Эйлеров обход + BST для работы с весами рёбер?