Бабушкин А.

1 Обязательные задачи

1. Сортим рёбра по весу, идём двумя указателями. Хотим добавить новое ребро (v, u) – чекаем в ETT is_connected(v, u). Пока да, двигаем l, удаляя рёбра.

2. Из каждой не листовой вершины мы продолжим вниз вертикальный путь в ровно одну вершины – зачем не продолжать, если можно продолжать? Надо узнать, в какую. А надо в ту, в которую идёт больше всего путей. Узнать, в каком поддереве больше всего концов путей, можно как угодно. Например, переливайкой. Это всё оптимально, потому что на вершине v смена путей будет ровно у тех запросов, которые уходят через v вниз, но не в ту вершину, в которую мы продолжим путь. Значит, надо продолжить именно туда, куда идёт больше всего запросов.

3.

4. Первый рекурсивный вызов работает за F(n/8, m/2), второй – за F(n/8, n/8 + U), $U \le m/2$, $\to$ он работает за F(n/8, n/8 + m/2). Скажем, что первый тоже работает за F(n/8, n/8 + m/2), тогда $F(n, m) \le n + m + 2F(n/8, n/8 + m/2)$. Докажем, что $F(n, m) \le \min(n^2, m \log n)$ по индукции. База вроде очевидная с точностью до константы, дальше:
$F(n, m) \ge 2F(n/8, n/8 + m/2)$ *ge* $2\min(n^2/64, (n/8 + m/2)\log(n/8) \ge \min(n^2/32, m\log n)$. чтд.

2 Дополнительные задачи

1.

2. link, видимо, здесь ориентированный, иначе непонятно, чего от нас хотят. Тогда link это make_root + обычный link. cut совсем обычный, make_root тоже из обычного link-cut. Осталось разобраться с предками. Что такое быть предком? Это если я вошёл в вершину v раньше u, а вышел позже. Осталось находить первое и последнее ребро, где я вхожу и выхожу из вершины. Давайте хранить (на самом деле treap, но пока представим что это массив) массив рёбер, выходящих из вершины. Точнее, массив указателей на их pNode. Массив будет отсортирован. make_root будет выполнять циклический сдвиг этого массива, следовательно, когда мы захотим найти, какой же из элементов массива сейчас лежит левее всего в ETT, мы сможем бинарить в таком массиве и спрашивать позицию в ETT (если у нас есть циклически сдвинутый отсорченный массив, то его наименьший элемент можно найти за бинпоиск). Только запрос в ETT выполняется за лог, поэтому это $\log^2$ на поиск самого левого/правого вхождения вершины в ETT. Осталось заметить, что у нас есть link и cut, поэтому надо будет вставлять/удалять рёбра в эти массивы, поэтому заменим эти массиры на treap.

Теперь чтобы находить место, куда в них вставлять элемент, тоже нужен бинарь, поэтому link будет тоже за $\log^2$, а cut и make_root всё ещё за логарифм.