Курс с++

Царьков Олег

Описание тикета OTSN-15

**DSU.** Будем снова решать задачу разбиения графа на компоненты связности.

Только на этот раз поиск компонент связности сделаем за константное время, не зависящее от количества вершин в графе!

Конечно, чтобы добиться такого неожиданного результата, надо чем-то пожертвовать, а именно: мы запретим удалять из графа ребра, их можно только добавлять.

Для этого используем структуру Disjoined Set Union(объединение непересекающихся множеств).

Алгоритм прост: представим себе что у нас есть много Splay-деревьев(ничего общего с рассматриваемым графом они почти не имеют), и будем считать, что две вершины в одной компоненте связности, если они в одном *Splay*-дереве.

• добавление ребра: если мы соединяем вершины 1 и 2 ребром, то если они были в одной компоненте связности, то не надо ничего делать, а если в разных, то надо объединить эти компоненты в одну. Делаем так: берем *Splay*-дерево вершины 1 и ищем в нем корень $r_1$, берем *Splay*-дерево вершины 2 и ищем в нем корень $r_2$. Если корни совпали, то 1 и 2 в одном и том же дереве и ничего делать не надо. Если корни не совпали, то надо слить эти два *Splay*-дерева, вот и все.

• определить, являются ли вершины в одной компоненте связности: найти у обеих корень их *Splay*-деревьев и сравнить.

• найти корень — это значит подняться по ребрам до корня, это будет происходить за $ln(n)$, поэтому добавление ребра будет занимать $ln(n)$, зато поиск количества компонент связности ничего не занимает, потому что оно итак известно.

**Задание 1.** Написать DSU с функциями:

• конструктор, задающий дерево без ребер

• AddEdge – добавление ребра

• CheckUnited принимает две вершины и определяет в одной ли они компоненте связности за $ln(n)$.

• GetComponentsNumber – выдает количество компонент связности.

**Задание 2.** Отлично, теперь есть две программы, которые считают одно и то же — количество компонент связности, что надо сделать?... Тестировать, что они возвращают одинаковый результат на произвольных деревьях.