Disjoint set union. Хотим реализовать следующий интерфейс:

- $get(x)$ — найти компоненту, в которой лежит x
- $unite(x, y)$ — объединить две компоненты
- $check(x, y)$ — проверить, что две вершины лежат в одной компоненте

$check(x, y)$ обычно выполняется как проверка $get(x) = get(y)$.

Под капотом мы будем хранить ориентированный лес. А именно, из каждой вершины будет идти ребро в предка. Компоненту будем идентифицировать номером корня в соответствующем дереве.

Как делать $get(x)$? Подниматься по ребрам $x \to parent_x \to \ldots \to root$. Тогда $get(x) = root$

Для $unite(x, y)$ надо подвесить корень одной компоненты к какой-то вершине из другой компоненты.

Эвристики. Понятно, что в наивной реализации очень легко достигается время работы $O(n)$ на запрос. Но существует две легкие эвристики, которые значительно улучшают время работы. Формально, они являются не эвристиками, а просто оптимизациями алгоритма.

Эвристика размеров. При операции *unite* будем подвешивать корень одного дерева к корню другого дерева (это значит, что нам понадобится в начале сделать две операции *get*). При этом мы будем подвешивать корень меньшего дерева к большему. Таким образом, любой подъем в *get* по ребру увеличивает размер компоненты в два раза. Мы получаем оценку $O(\log n)$ на запрос.

Эвристика рангов. Зададим каждой вершине ранг r_x. При подвешивании будем сравнивать корни не по размерам, а по рангам — к большему подвешивать меньший. Если два ранга совпали, то после подвешивания ранг корня увеличивается на 1. Тогда количество вершин с рангом x не больше, чем $\frac{n}{2^x}$. Это можно показать по индукции. Поскольку вершина с рангом $x+1$ получается как комбинация двух вершин с рангом x (а вершины с рангом x после этого становятся неактивными), то $A_{x+1} \le \frac{A_x}{2}$, где A_x — число вершин ранга x. Считаем, что ранг одной вершины — 0.

Очевидное утверждение — ранг вдоль пути к предку возрастает. Поскольку рангов не больше, чем $O(\log n)$, то мы опять получаем оценку $O(\log n)$ на запрос.

Эвристика сжатия путей. Идея эвристики такая: если мы прошли по пути до корня, то мы для каждой вершины на этом пути узнали текущую компоненту. Это значит, что можно запомнить эту информацию и не подниматься в промежуточные вершины в дальнейшем — подниматься сразу в текущий корень компоненты. Утверждается, что эта эвристика без какой-либо эвристики на *unite* работает амортизированно за $O(\log n)$ на запрос.

Как такую оценку показать? Разобьем ребра на легкие и тяжелые. Ребро называется *тяжелым*, если на нем висит хотя бы половина поддерева. Подняться по легкому ребру больше логаифма раз мы не можем. Утвердается, что суммарных проходов по тяжелым ребрам будет $O(n \log n)$.

Каждый раз, когда мы проходим по тяжелому ребру, оно становится легким (кроме, может быть, ребра в корень). Иногда легкие ребра опять становятся тяжелыми.

Будем рассматривать все вершины, кроме корня (все ребра, связанные с корнем, работают за $O(1)$, поэтому нам не важно). Для оставшихся вершин верно, что они только теряют тяжелых сыновей. При этом мы можем считать, что новые вершины не появляются — пририсовывается что-либо только к корню, а наша вершина уже никогда не будет корнем. Тогда к v присоединено не более $O(\log n)$ тяжелых ребер за все время ее существования. Это верно, потому что при удалении тяжелого ребра (переподвешивания его к корню), суммарный размер поддерева уменьшается в два раза. Таким образом, верна оценка $O(n \log n)$ на число тяжелых ребер, которые мы суммарно рассмотрим.

Объединение эвристик. Утверждается, что при объединении ранговой эвристики и эвристики сжатия путей достигается время $O(n\alpha)$. Это какое-то сложное доказательство, которое (возможно) будет потом. Мы покажем $O(n \log^* n)$.

Что такое $\log^* n$? Это функция, обратная к $f(a,b) = \begin{cases} 1 & , b = 0 \\ a^{f(a,b-1)} & else \end{cases}$. На всех тестовых данных, влезающих в современный компьютер, эта функция примерно равна четырем (кстати, обратная функция Аккермана тоже равна 4 на адекватных тестовых данных, — *прим. автора*).

Назовем ребро **крутым** (или жестким), если проход по ребру увеличивает ранг экспоненциально. А именно, $c^{r(v)} \le r(parent_v)$ для какой-то c. Тогда, если мы пройдем по пути с a крутыми ребрами, то ранг будет не меньше, чем $f(c,a)$. А значит, количество крутых ребер $O(\log^* n)$.

Действия разбиваются на три типа:

1. Проходы по крутым ребрам — $O(\log^* n)$ на запрос амортизированно.
2. Проходы в корень — $O(1)$ на запрос.
3. Проходы по не крутым ребрам — сейчас докажем линейную амортизированную оценку.

Сколько раз нужно пройти по не крутому ребру, чтобы оно стало крутым? Каждый раз, когда мы переподвешиваем вершину v от предка $parent_v$ к какой-то другой вершине, мы увеличиваем ранг предка. При этом ранг v зафиксирован. Это значит, что проходов из каждой вершины по не крутому ребру будет не более чем $c^{r(v)}$. Тогда, просуммировав по всем вершинам, получаем $\sum_{x=0}^{\log n} A_x \cdot c^x \le \sum_{x=0}^{\log n} \frac{nc^x}{2^x} \le n\frac{1}{1-\frac{c}{2}}$. Это будет верно при таких c, которые меньше, чем 2 (чтобы мы получили убывающую геометрическую прогрессию). При этом $f(c,a)$ все еще должна уходить в бесконечность. $c = 1.9$ подойдет. Таким образом, число проходов по не крутым ребрам будет линейно.

Задача о двудольном максимальном паросочетании. Пусть нам дан двудольный граф. Это такой граф, в котором вершины делятся на два класса, а ребра проводятся только между вершинами разных классов. Паросочетанием называется такое множество ребер, что никакие два ребра из этого множества не смежны. Максимальным по включению паросочетанием называется такое паросочетание, которое нельзя дополнить никаким ребром (которое, аккуратно, не является максимальным). Двудольной кликой называется два подмножества вершин (левой и правой доли), чтобы между любой парой вершин разных подмножеств было ребро.

Удлиняющие цепочки. Удлиняющей цепочкой называется чередующийся путь из свободной вершины левой доли в свободную вершину правой доли. А именно, все нечетные ребра на пути не принадлежат паросочетанию, а четные, наоборот, принадлежат.

Очевидно, что если есть удлиняющая цепочка, то паросочетание можно увеличить — заменить статус всех ребер на противоположный. Количество ребер в паросочетании увеличится на единицу.

Оказывается, если удлиняющих цепочек нет, то паросочетание максимально. Рассмотрим такое M, а также M^*, для которого верно, что $|M^*| > M$. То есть, одно паросочетание не содержит удлиняющих цепочек, а другое больше по размеру.

Посмотрим на симметрическую разность этих множеств. То есть, на те ребра, которые принадлежат только одному из множеств. В таком графе степени вершин до двух. Тогда этот граф состоит из циклов четной длины и путей. В цикле четной длины одинаковое количество ребер из M и M^*. На путях четной длины количество ребер из M и M^*, аналогично, совпадает. Теперь посмотрим на нечетные пути. Ребер

одного из двух типов на нем будет больше. Значит, на каком-то из путей, ребер из M^* будет больше, чем из M. Но такой путь обязательно будет чередующейся цепочкой! Получаем противоречие.

Алгоритм Куна. Мы не обсудили, но если на следующей лекции его не будет, то я напишу