16.12.21.

Романчина
Татьяна,
БО5-031.

① Найти в графе количество треугольников за $O(E\sqrt{E})$.

Разобьём вершины на 2 множества:

(1) $\{x \mid \deg(x) > \sqrt{E}\}$ и $\{x \mid \deg(x) \leq \sqrt{E}\}$ (2)

Сначала перебираем вершины x (1) множества (таких в графе $N = \frac{2E}{\deg(x)} \leq \frac{2E}{\sqrt{E}} = 2\sqrt{E}$),

внутри перебираем ребра (y,z) и проверяем, есть ли ребра (x,y), (y,z), (x,z).

Найдем какое-то кол-во треугольников, содержащих хотя бы 1 вершину x ($\deg(x) > \sqrt{E}$).

за $O(E\sqrt{E})$.

Затем в цикле перебираем ребра $(y,z) \in E$.

Если $\deg(y) \leq \sqrt{E}$ и $\deg(z) \leq \sqrt{E}$, то внутри перебираем x-соседей y/z (таких $\leq \sqrt{E}$) и смотрим, есть ли ребро $(x,z)/(x,y)$ за $O(E\sqrt{E})$.

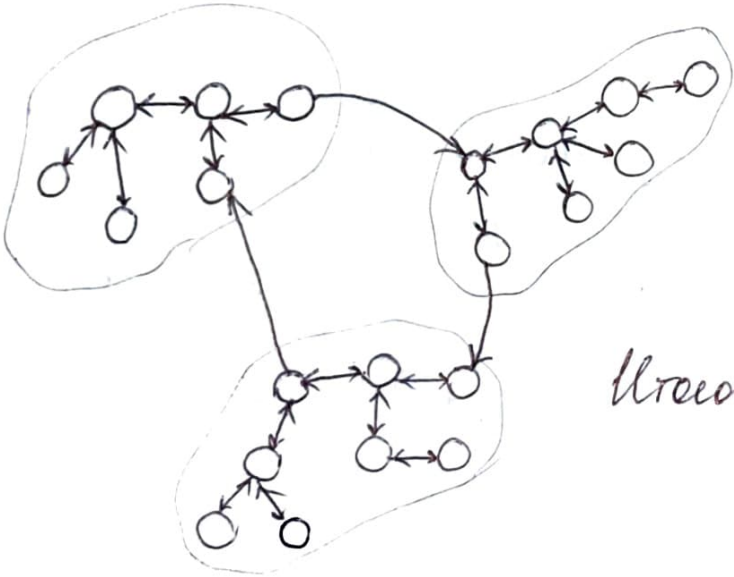
Итоговая ассимптотика: $O(E\sqrt{E}) + O(E\sqrt{E}) = O(E\sqrt{E})$.

(2) Ориентировать неор. граф так, чтобы он стал сильносвязным за $O(V+E)$ или сказать, что это невозможно.

Сначала, если граф не связен в смысле неор. графа, работаем с каждой компонентой отдельно. Запустим на графе dfs и на ребрах, принадлежащих дереву обхода dfs поставим ориентацию в обе стороны. $O(V+E)$.

Теперь запустим алгоритм Косарайю поиска компонент сильной связности. $O(V+E)$.

Соединим однонаправленными ребрами все компоненты по циклу. $O(V) \leq O(V+E)$.



Теперь граф сильно связен

Итоговая ассимптотика: $O(V+E)$

③ Разбить все ребра неор. графа на минимальное число путей за $O(V+E)$.

Нам нужно дополнить граф до Эйлерова минимальным числом ребер.* После этого путь останется только 1, если мы удалим добавленные ранее ребра, то этот путь распадется в k реберно-непересекающихся путей. И наоборот, если мы обратно добавим эти ребра, то получим единственный путь. Поэтому задача сводится к минимизации кол-ва ребер, добавленных в граф.

* Если наш граф связен, то парасочетаем нечетные вершины min числом ребер. (1)
Если несвязен, то добиваемся связности за $O(V+E)$ (алгоритм Косарайю), затем переходим к (1).

Итоговая асимптотика: $O(V+E)$

④ Для каждой пары вершин в графе найти $W[a,b]$ - такой минимальный вес, что a и b в нем путь по ребрам веса $\leq W[a,b]$. $O(V^2)$

Наша задача - минимизировать макс. вес ребра в каждом пути от v до u, $v,u \in V$.

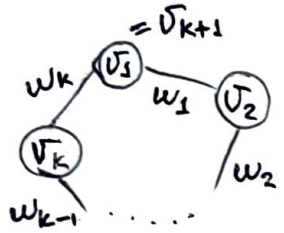
Воспользуемся алгоритмом Прима для построения минимального остовного дерева нашего графа. Все пути в дереве определяются однозначно. Так как на каждой итерации алгоритма Прима мы имеем мн-во вершин T (те, которые мы уже добавили в дерево) и мн-во V (те, которые мы ещё в дерево не добавили), то путь для какой $t \in T$ до какой $v \in V$ будет проходить через одно из ребер, соединяющих вершину из T с вершинами из V. По ходу алгоритма мы выбираем из них ребро с min весом, поэтому минимизируем вес ребра в каждом пути. $O(V^2)$.

Далее от каждой вершины построенного MST запускаем dfs (т.к. пути в дереве опр-ны однозначно), в процессе которого обновляем max веса ребра на пути. Пусть $MST = (V', E')$, где $V' = V$, $E' = V-1$. $T(V', E') = O(V' + E') = O(V)$ для каждого dfs.
Итого: $V \cdot O(V) = O(V^2)$

Итоговая ассимптотика: $O(V^2)$.

⑥ Найти в графе цикл минимального среднего веса. $E \le 2000$, $|w_i| \le 10^9$.

Бинарным поиском будем искать такое min значение V, что при вычитании V из весов всех ребер существует отрицательный цикл. Этот цикл и будет циклом минимального среднего веса



$$\frac{\sum_{i=1}^{k} w_i}{k} < V \Leftrightarrow \sum_{i=1}^{k} (w_i - V) < 0.$$

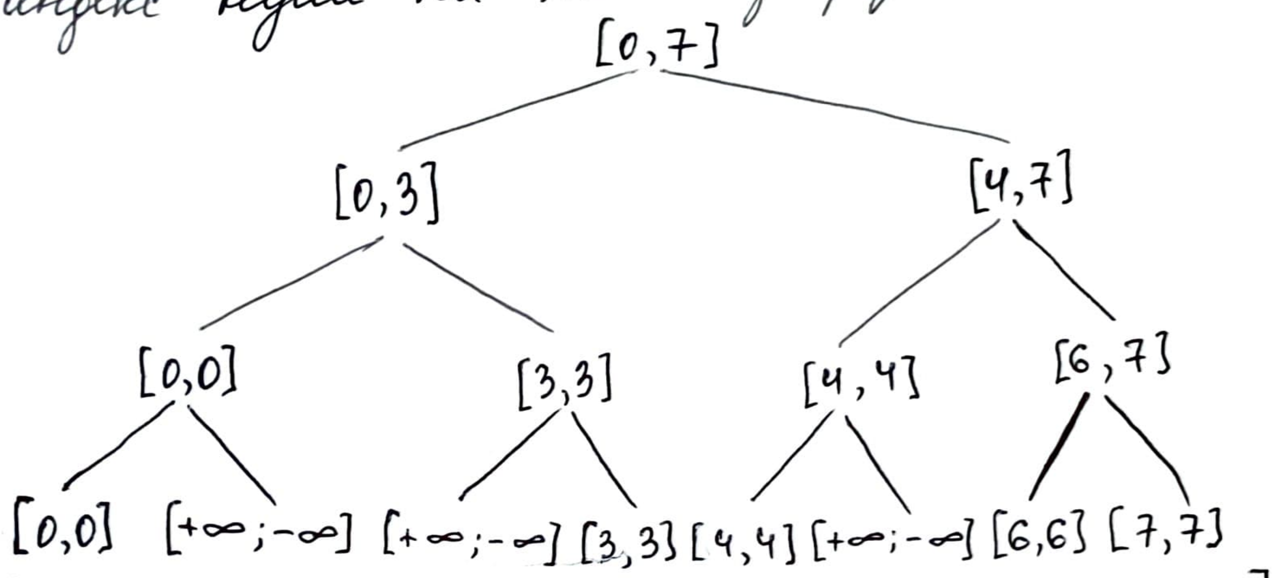
Искать цикл отрицательного веса будем алгоритмом Форда-Беллмана ($O(VE)$).

Итоговая асимптотика: $O(VE \cdot \log 10^9)$.

(8) Есть массив из нулей и единиц. В online за $O(\log n)$ отвечать на запросы: поменять элемент, найти ближайший слева/справа ноль к позиции i.

Допустим есть массив: $[0, 1, 1, 0, 0, 1, 0, 0]$

Построим по нему следующее ДО: в каждой вершине будем хранить min и max индекс нуля на этом подотрезке, то есть:



// $[0, 1, 1, 0, 0, 1, 0, 0]$

(1) Поменять значение в позиции i =

$$= \begin{cases} [+\infty; -\infty] \longrightarrow [i, i], \text{ если 1 меняем на 0.} \\ [i, i] \longrightarrow [+\infty; -\infty], \text{ если 0 меняем на 1.} \end{cases}$$

+ Поднимаемся из элемента к вершине и обновляем все значения [min, max] за $O(\log N)$.

(2) Найти ближайший 0 справа от позиции i = найти min из значений a (node = $[a, b]$, a — min, b — max) на отрезке $[i+1, N-1]$ за $O(\log N)$.

(3) Найти ближайший 0 слева от позиции i = найти max из значений b (node = $[a, b]$) на отрезке $[0, i-1]$ за $O(\log N)$.