29. Топологическая сортировка ориентированного ациклического графа: определение и алгоритм поиска (с доказательством корректности).

Топологическая сортировка ориентированного ациклического графа $G(V, E)$ представляет собой упорядочивание вершин таким образом, что для любого ребра $(u, v) \in E$ номер вершины u меньше номера вершины v.

Предположим, что граф ацикличен, т.е. решение существует. Что делает обход в глубину? При запуске из какой-то вершины v он пытается запуститься вдоль всех рёбер, исходящих из v. Вдоль тех рёбер, концы которых уже были посещены ранее, он не проходит, а вдоль всех остальных — проходит и вызывает себя от их концов.

Таким образом, к моменту выхода из вызова $DFS(v)$ все вершины, достижимые из v как непосредственно (по одному ребру), так и косвенно (по пути) — все такие вершины уже посещены обходом. Следовательно, если мы будем в момент выхода из $DFS(v)$ добавлять нашу вершину в начало некоего списка, то в конце концов в этом списке получится топологическая сортировка.

```
for (int i = 0; i < n; ++i)
    if (color[v] == "white")
        dfs(i);
// вывести вершины в порядке убывания tout
```

Корректность: ▲ Покажем, что если $(u, v) \in E$, то u выведется раньше v. u выведется раньше чем $v \Leftrightarrow tout[v] < tout[u]$ Рассмотрим 2 случая

1. v была замечена DFSом раньше чем u. Тогда к моменту выхода из v вершина u все еще не посещена, иначе нашелся бы цикл $\Rightarrow tout[v] < tout[u]$

2. u была замечена раньше чем v. Тогда мы перейдем по ребру (u, v) в v, а до выхода из u необходимо выйти из всех вершин, в которые мы попали из нее (по лемме о белых путях) $\Rightarrow tout[v] < tout[u]$ ■

Асимптотика: $O(n+m)$ (dfs проходит каждому ребру и вершине по 1 разу. Сортировку tout можно сделать прям в dfs идею см. в билете 31)

30. Отношение сильной связности между вершинами. Компоненты сильной связности. Сильно связный граф.

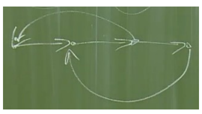
Определение: В ориентированном графе вершины u, v - *сильно связаны*, если из u есть путь в v и из v есть путь в u.

Утверждение: сильная связность является отношением эквивалентности

▲ Рефлексивность и симметричность очевидны из определения. Транзитивность получается из того что просто склеиваем пути ■

Определение: Классы эквивалентности относительно отношения сильной связности, на которые разбивается граф, называются *компонентами сильной связности*.

Определение: Ориентированный граф *сильно связен*, если для каждой вершины все остальные вершины достижимы из нее.



Пример: компонента сильной связности (сама является сильно связным графом)

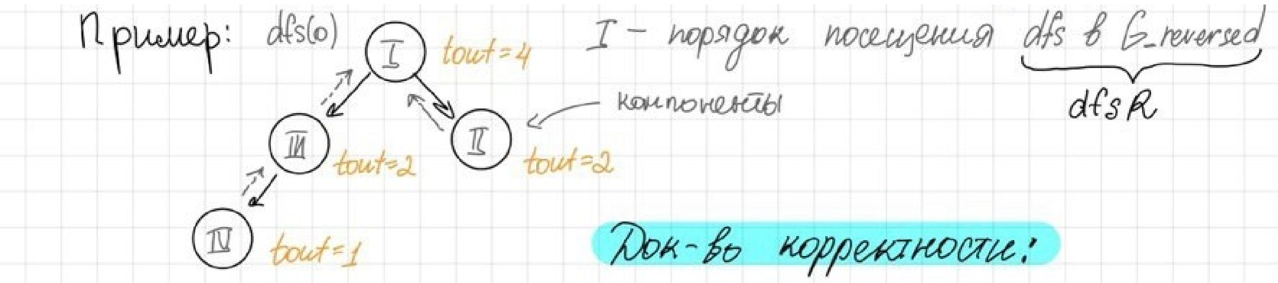
31. Алгоритм Косарайю. Корректность и время работы.

1. Выполним DFS, вычисляющий для каждой вершины время выхода DFS из нее (этот массив обозначим за p).

2. Строим граф H с инвертированными ребрами

3. Выполняем DFS на H, перебирая вершины в порядке убывания $p(u)$.

```
used.assign(n, false);
for (int s = 0; s < n; ++s)
    if (!used[s])
        dfs(s);
// p - список вершин в порядке убывания tout
// можно получить его например добавив в конец dfs строчку
// p.pushback(v)
// (только потом надо реверснуть весь массив)
used.assign(n, false);
for (int v: p)
    if (!used[s])
        reversed_dfs(s); // dfs в графе, где все ребра инвертированы
// каждое дерево обхода обратного dfs будет являться КСС
```



▲ 1. Если v - вершина с максимальным tout в своей КСС, то $reversed\_dfs(v)$ посетит как минимум всю эту КСС (так как все остальные вершины из нее еще не used) $\Rightarrow$ целиком содержит эту КСС, так как $reversed\_dfs$ проходит по всем вершинам, из которых достижимо v в исходном графе

2. Покажем, что не посетили больше чем одну КСС. Пусть u, v лежат в разных КСС. Тогда имеет место один из двух случаев

   – нет путей из u в v и из v в u

     Очевидно, что в этом случае ни одна из вершин не достижима из другой ни по прямым, ни по инвертированным ребрам $\Rightarrow$ алгоритм причислит их к разным КСС

   – Есть путь из u в v, но нет пути из v в u (симметричный случай доказывается аналогично)

     Тогда $tout[u] > tout[v]$ (доказательство аналогично доказательству корректности в топологической сортировке, билет 29). Значит $reversed\_dfs(u)$ запустится раньше чем от v. Так как по предположению из v в u нет пути, то $reversed\_dfs(u)$ не попадет в v. Но тогда $reversed\_dfs(v)$ уже не попадет в u, так как она $used \Rightarrow$ алгоритм причислит их к разным КСС ■

Асимптотика: $O(n + m)$ (обходим все вершины и ребра по одному разу dfsом, потом по одному разу reversed_dfsом)