

95. Определение изоморфизма графов. Алгоритм проверки изоморфности двух ориентированных или неориентированных деревьев за $O(n \log n)$.

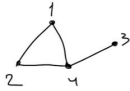


Рис. 1: Пример изоморфизма.



Рис. 2: Изоморфизм ориентированных деревьев.

Пусть G и H - два неориентированных графа. Функция $\varphi : V(G) \rightarrow V(H)$ - **изоморфизм**, если:

- 1) φ - биекция
- 2) $(u, v) \in E(G) \Leftrightarrow (\varphi(u), \varphi(v)) \in E(H)$

Замечание. Для ориентированных аналогично.

Пункт 1. Проверка ориентированных (подвешенных/корневых) деревьев на изоморфность.

0. См. рис. 2 - совсем тривиальный алгоритм, сравнивающий по количеству сыновей, не совсем работает, валится на совпадениях по количеству.

1. Пусть G - корневое дерево с корнем в r . Тогда для всех поддеревьев найдём "номер класса эквивалентности": мы разбиваем вершины на классы эквивалентности по количеству детей в каждом из поддеревьев, которые являются их сыновьями. Получаются вектора размеров поддеревьев, и по их равенству (вплоть до порядка элементов) вводятся классы эквивалентности.

```
1 map<vector<int>, int> num;  
2 vector<int> classes; // вектор классов экв. по изоморфизму для вершин деревьев  
3 ll c = 0; // количество классов  
4  
5 void dfs (int v) {  
6     vector<int> ans;  
7     for(int u: сын v) {  
8         dfs(u);  
9         ans.push_back(classes[u]);  
10    }  
11    sort(ans.begin(), ans.end());  
12    if (!num.count(ans)) {  
13        num[ans] = c++;  
14    }  
15    classes[v] = num[ans];  
16 }
```

3. Для двух деревьев - запускаем $\text{dfs}[r_1]$, $\text{dfs}[r_2]$, после чего сравниваем classes .

Де-факто это быстро, хотя точную асимптотику сказать сложно (map работает за логарифм длины сравнения, а там сравниваются вектора; но с хешированием векторов можно ускорить, если надо, и тогда будет $O(n)$ - время работы хеш-таблицы.)

Пункт 2. Проверка неориентированных деревьев на изоморфность: раньше у нас были фиксированные пары корней, которые мы сопоставляли: в биекции корень всегда сопоставляется корню - тут же мы берём пару центроидов (все возможные пары центроидов), потому что аналогично, центроид в первом графе останется центроидом во втором, вот наша гарантированная пара, а дальше - пункт 1.