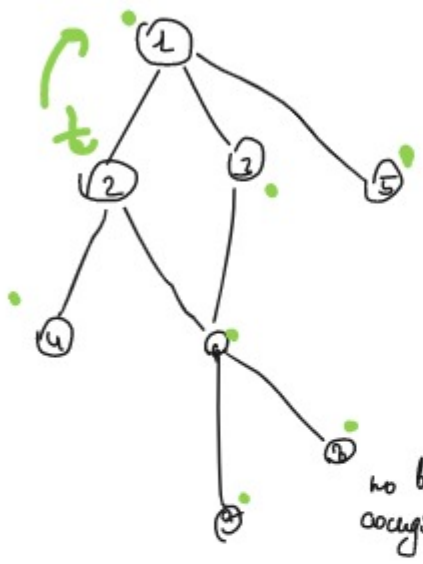
DFS

алгоритм „поиск в глубину"

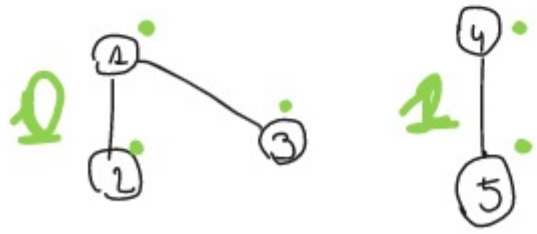


used[1] == true – уже ходили
used[1] == false – ещё не ходили

used t f t t t f

O(V + E)

```
dfs (x)
  used[x] = true
  for v : graph[x]          ← по ребрам → по вершинам
      if not (used[v])
          dfs (v)
```



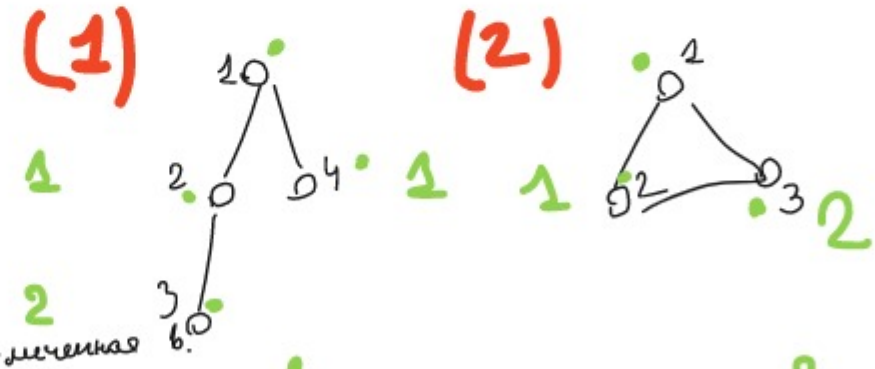
```
main:
  for (1 .... n)        ← проход по вершинам
      if (used[i] == false)   ← был ли в этой вершине dfs
          dfs (i)
```

Задача: выделить компоненты связности.

```
main
  int number = 0
  used( ) = -1  → false
  for (1 ... n)
      if (used[i] == -1)
          dfs(i, number)
          number++
```

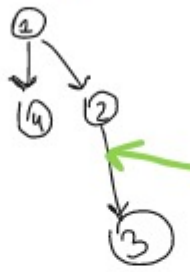
Задача: найти цикл

```
dfs (v, parent):
  used[v] = true
  for (u : graph[v])
      if not (used[u])        ← неотмеченная
          dfs (u, v)
      else if (u != parent)
          print ("cycle")
          break
```

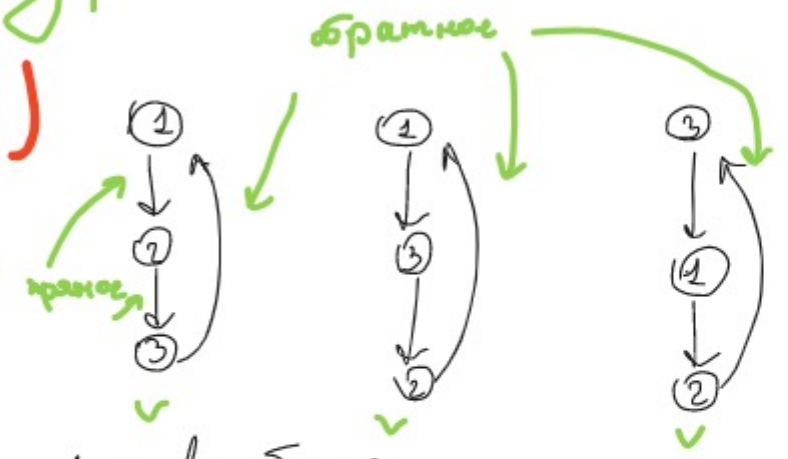


Дерево обхода

(1)



(2)



1 Запуск dfs делает 1 дерево обхода

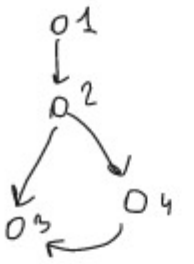
3 типа ребер в деревьях обхода:

1) прямое (из предка в потомка)

2) обратное (из потомка в предка)

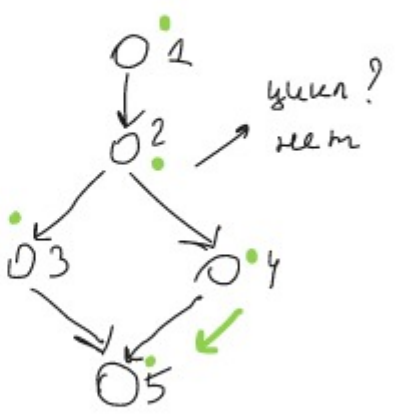
3) перекрестное (из правой ветки в левую)

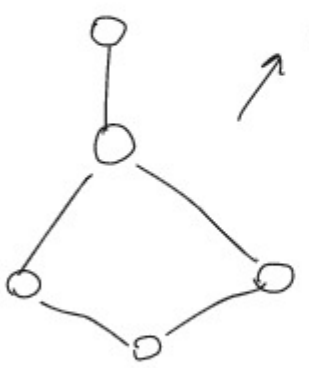
только для ориент.



из 3 в 4 мы не можем перейти, а из 4 в 3 можем

Задача: найти цикл в ориентир графе





Новые категории:

1. белая – мы не заходили 0
2. серая – в текущей процедуре, не вышли из вершины 1
3. чёрная – вершина пройдена 2

used → color

изначально белые

```
dfs ( v )
   color [v] = 1
   for ( u : graph [v] )
      if ( color [u] == 0)
         dfs(u)
      if ( color [u] == 1)
         print ( Cycle )
         break
   color [v] = 2
```