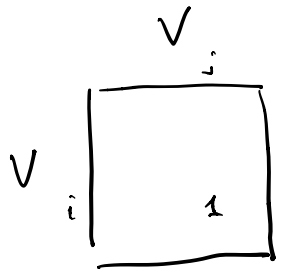
Хранение графов

1. Матрица смежности



⊕ Проверка ребра за $O(1)$

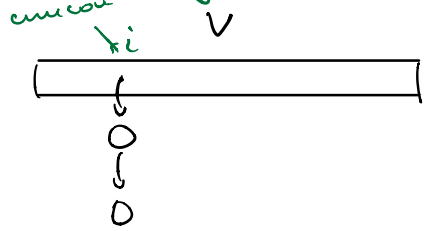
⊖ Размер $O(V^2) = O(|V|^2)$

⊖ Перебор всех рёбер из вершины за $O(V)$

$G = (V, E)$ — рёбра, вершины

"Размер графа" $= |V| + |E|$

2. Списки смежности

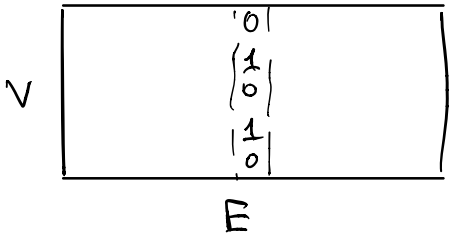


⊕ Размер $O(V+E)$

⊕ Перебор всех соседей за $O(\#\text{ соседей})$

⊖ Проверка ребра за $O(V)$

3*. Матрица инцидентности



Размер $O(V \cdot E)$

Поиск в глубину (DFS, depth-first search)

```
def Explore(v):
    visited[v] = True
    previsit(v)
    for (v,u) ∈ E:
        if not visited[u]:
            Explore(u)
    postvisit(v)
```

Сложность:

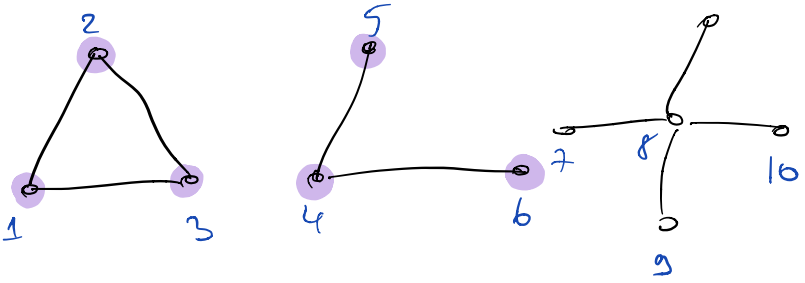
O(V) вызовов Explore

перебор соседей

```
def DFS(G):
    for v=1 to |V|:
        visited[v] = False
    for v=1 to |V|:
        if not visited[v]:
            Explore[v]
```

В итоге:

- O(V+E) для списков смежности
- $O(V^2)$ для матрицы смежности



Поиск компонент связности в неор. графе

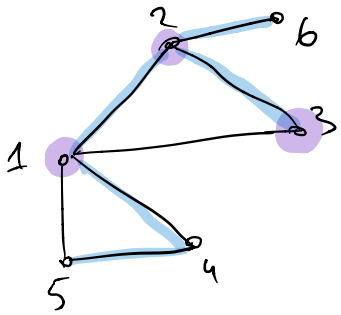
CC[v] - номер компоненты связности

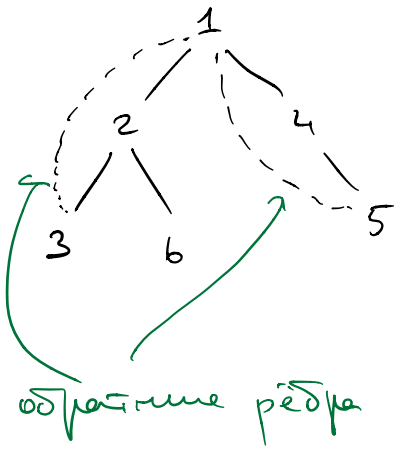
component – глобальный счётчик

```
previsit(v):
    cc[v] = component

def CC(G):
    for v=1 to |V|:
        visited[v] = False
    component = 1
    for v=1 to |V|:
        if not visited[v]:
            Explore[v]
            component = component+1
```

Поиск циклов в неор. графах

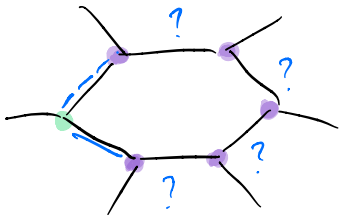
Обход Explore строит остовное дерево



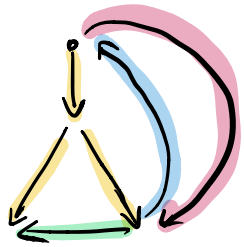


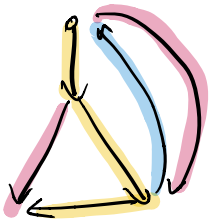
Утв: ∃ обратное ребро ⟺ есть цикл

⟹ тривиально



Поиск в глубину на ор. графах



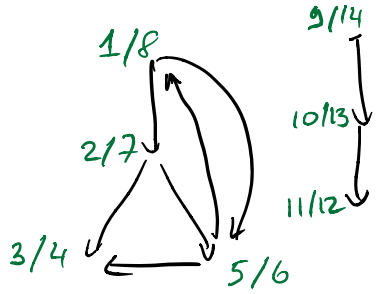


1. рёбра обхода (рёбра дерева)
2. перекрёстное ребро
3. обратное ребро
4. прямое ребро

pre[v], post[v] – время входа и выхода

time – глобальное время

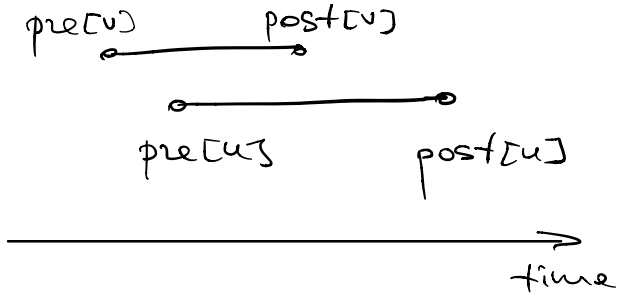
```
previsit(v):
    pre[v] = time
    time = time + 1

postvisit(v):
    post[v] = time
    time = time + 1
```

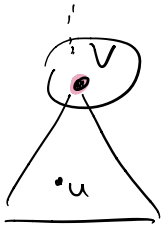


NB! ∀ вершина ассоц. с отрезком [pre[v], post[v]]

Утв $\forall u, v \in V$ их отрезки либо вкладываются друг в друга, либо не пересекаются

т.е. запрещено!



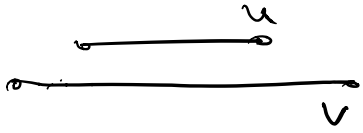


□

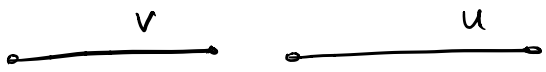
Для # ребра $(u, v) \in E$ ∠ соотв. отрезки:

1. 

ребро дерева / прямое ребро

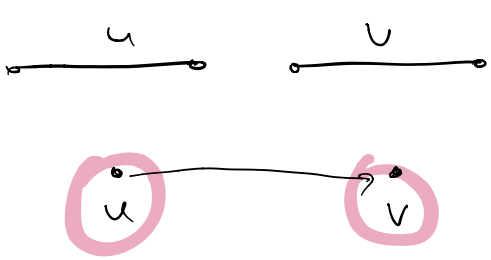
2. 

обратное ребро

3. 

перекрёстное ребро

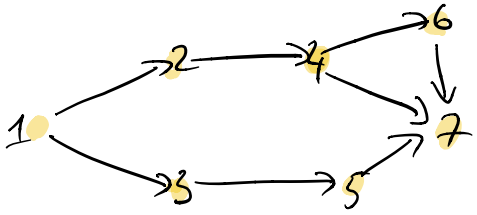
так не бывает

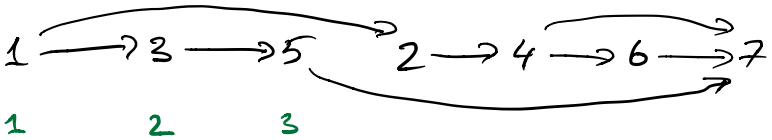


Топологическая сортировка

≡ топологическая сортировка DAG — такой порядок вершин T:

$$\forall (u,v) \in E \qquad T[u] < T[v]$$





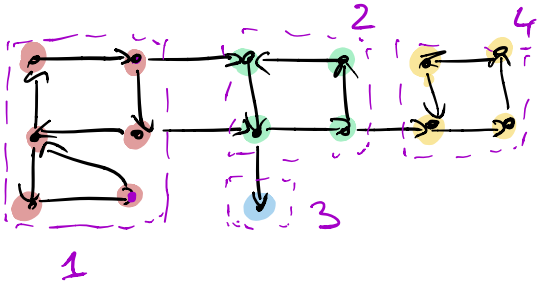
T 1 2 3

Утв: Топологическая сортировка задаётся порядком post[v] в порядке убывания

Вершина с max post — это исток

Выделение компонент сильной связности

≡ $\forall u, v \in SCC$: $u \to v$, $v \to u$



В DAG # вершина = SCC

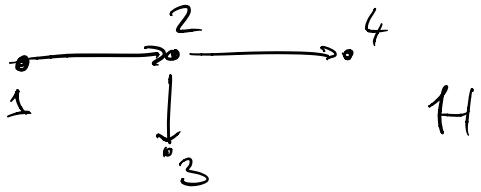
Утв В ∀ DAG есть сток

Утв В ∀ DAG есть исток

Утв G – DAG $\Leftrightarrow$ G^R – DAG

Утв Если G задан списками смежности, то можно построить G^R за $O(V+E)$

≡ Метаграф – граф из SCC



H

Утв Метаграф – ациклический

(*) Утв $\exists (U, V) \in H \Rightarrow$ (*)

$$\max_{u \in U} [post[u]] > \max_{v \in V} [post[v]]$$

Следствие:

Max время post находится в начальной вершине

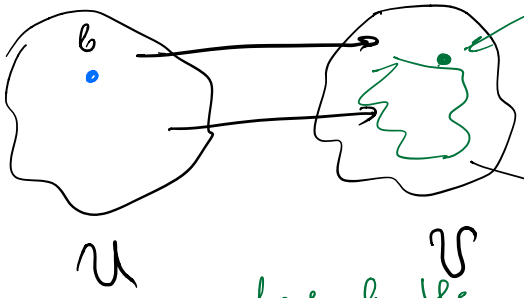
SCC(G): $O(V+E)$

DFS(G^R)

CC(G) # в порядке ↓↓ post

Доw-bo(*):

1.



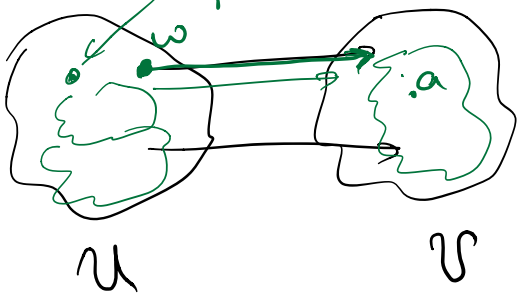
первое в dfs

$\forall a \in V \; \forall b \in U$:

$post[a] < pre[b]$

$<$ –

2.



$\forall a \in V$

$post[w] > post[a]$

$\Downarrow$

$\max_{u \in U} [post[u]]$

$\vee$

$\max_{v \in V} [post[v]]$