Розробка та аналіз алгоритмів

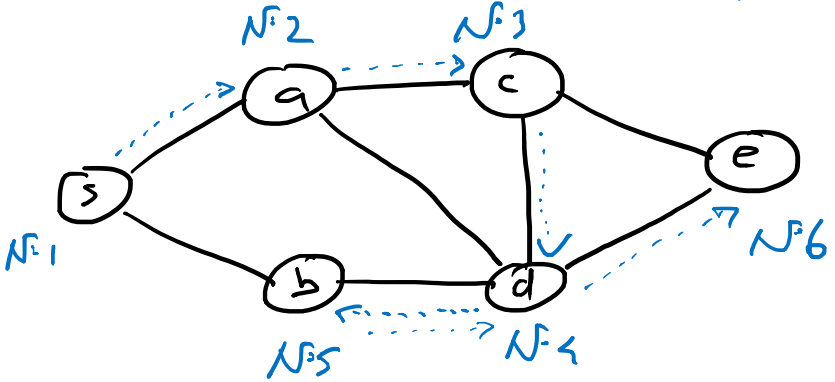
13. Пошук вглиб

- Алгоритм пошуку вглиб
- Топологічне сортування
- Компоненти сильної зв’язності

Пошук вглиб (Depth-First Search, DFS)

Вхід: граф G, початкова вершина S

Вихід: обійти всі досяжні вершини з S
та надрукувати їх номери за порядком обходу
(DFS-номери)



$T(n,m) = O(n+m)$

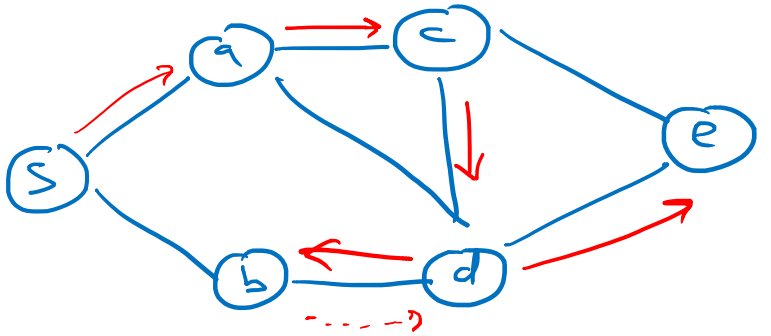
поточний DFS-номер

```
DFSR(G, s, k)
1. позначити s як досліджену
2. DFS[s] = k
3. для кожного ребра (s,u) в G:
4.     якщо u ще не досліджена: // DFS[u]=NULL
5.         k = DFSR(G, u, k+1)
6. return k
```

Основний виклик: DFSR(G, s, 1)

Пошук вглиб. Алгоритм (друга версія)

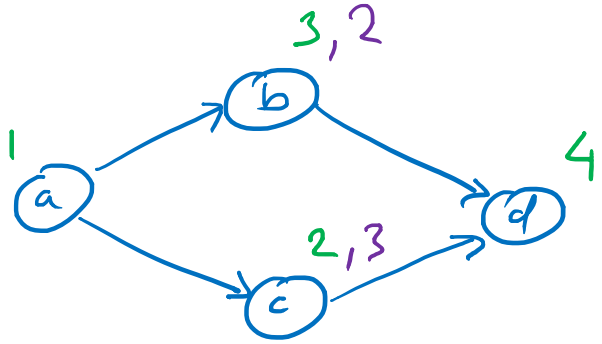
```
DFS(G, s)
1. Позначити вершину s як досліджену
2. DFS[s] = 1
3. Створити стек S: S = [s]
4. k = 1
5. while S ≠ []:
6.     v - вершина з голови стеку
7.     якщо існує ребро (v,u) в G і u ще не досліджена:
8.         k = k+1
9.         позначити u як досліджену; DFS[u] = k
10.        здати u в голову стеку
11.    else:
12.        видалити вершину v з голови стеку
```

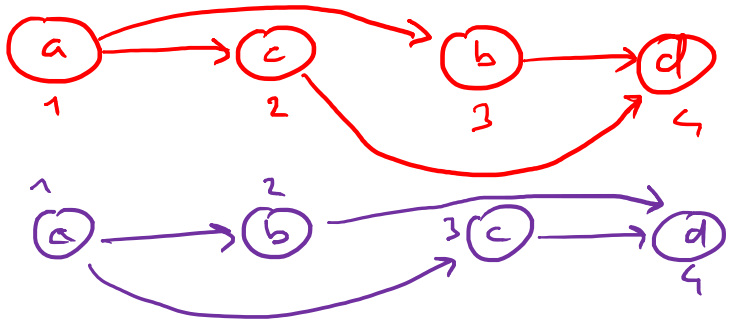


$T(n,m) = O(n+m)$

Топологічне сортування

- Топологічне сортування орієнтованого графу $G=(V, E)$ називається таке впорядкування всіх його вершин, що якщо граф містить ребро (v, u), то вершина v в цьому впорядкуванні буде знаходитись перед вершиною u.

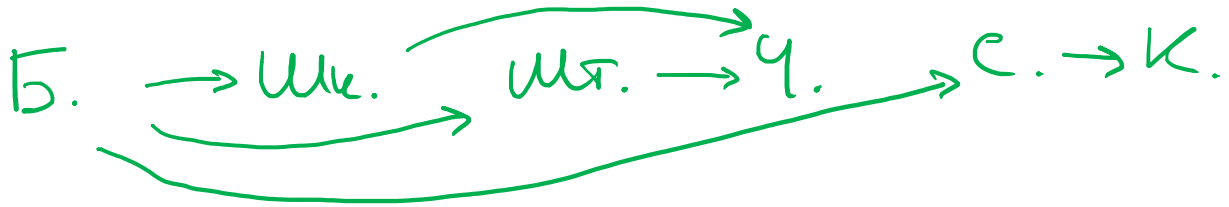




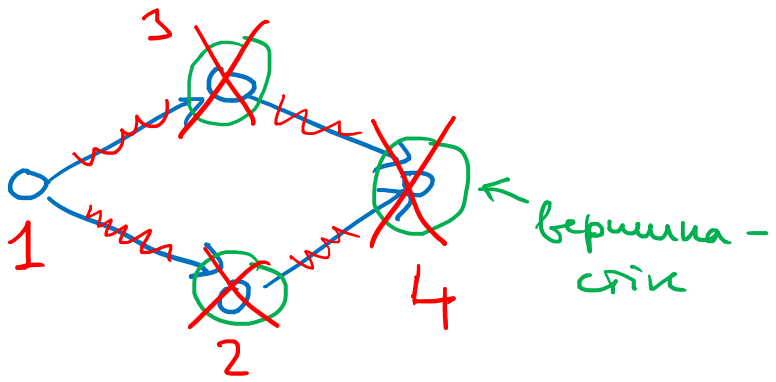
Умова існування Т.С. в орграфі:
граф не повинен містити циклів

Топологічне сортування. Приклад застосування

Впорядкування задач у плані (Які задачі (роботи) виконуються наступними?)

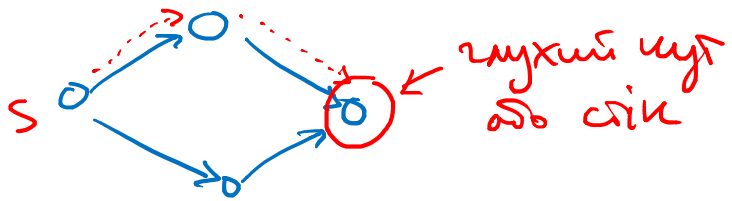


Топологічне сортування. Ідея алгоритму



В будь-якому орієнтованому ациклічному графі існує вершина-стік : всі ребра в неї тільки заходять.

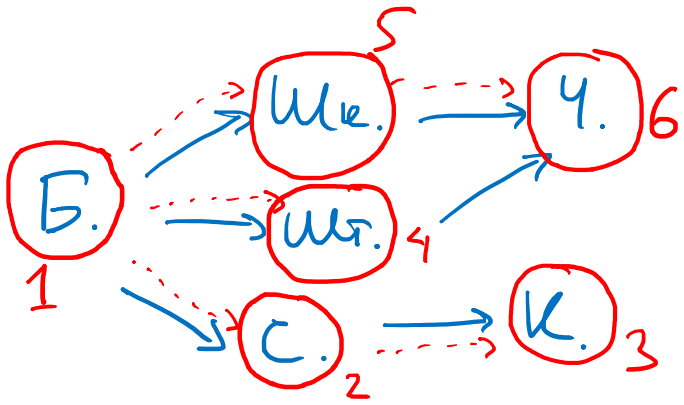
1) Знайти вершину-стік → v
2) Встановити TS[v] = n
3) Запустити процедуру для графу G - {v}



Топологічне сортування. Алгоритм

```
TopologicalSort(G)
1. Позначити всі вершини як не досліджені
2. Label = n        // поточний номер у топ.сорт.
3. for i=1 to n:
4.     Якщо вершина i ще не досліджена:
5.         DFSR(G, i)
```

```
DFSR(G, s)
1. Позначити s як досліджену
2. Для кожного ребра (s,u) в G:
3.     Якщо u ще не досліджена:
4.         DFSR(G, u)
5. TS[s] = Label
6. Label = Label - 1
```

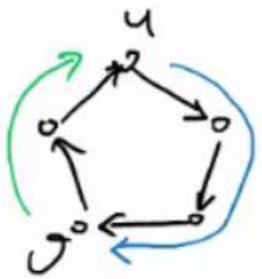


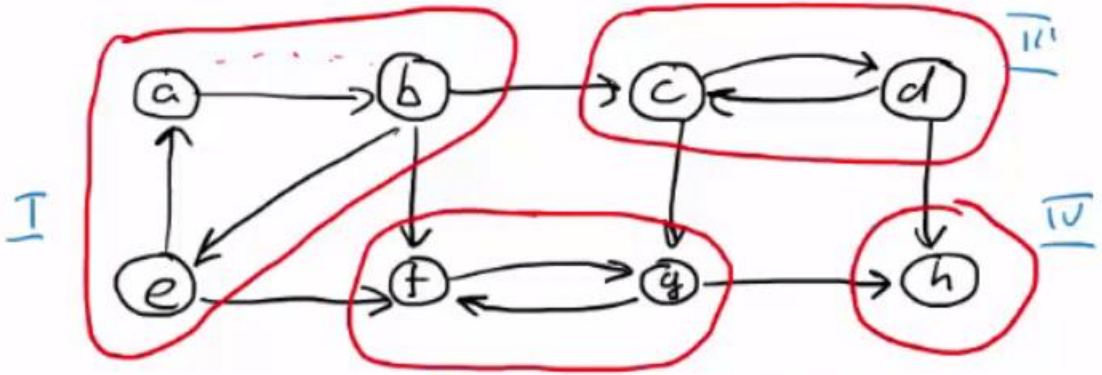
Label = 6

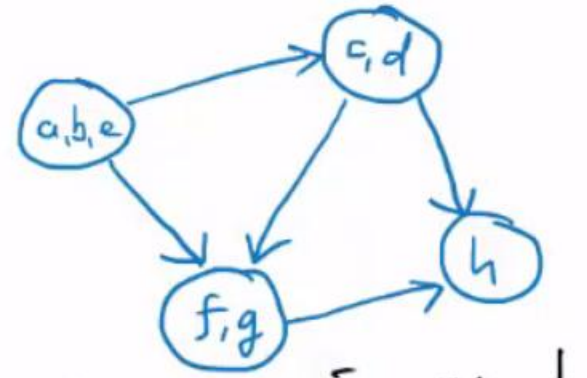
$T(n,m) = O(n+m)$

Зв’язність в орієнтованих графах

- Компонентою сильної зв’язності орієнтованого графу $G=(V,E)$ є максимальна множина вершин C, така що для кожної пари u та v з C існує орієнтований шлях як з u до v, так і з v до u.





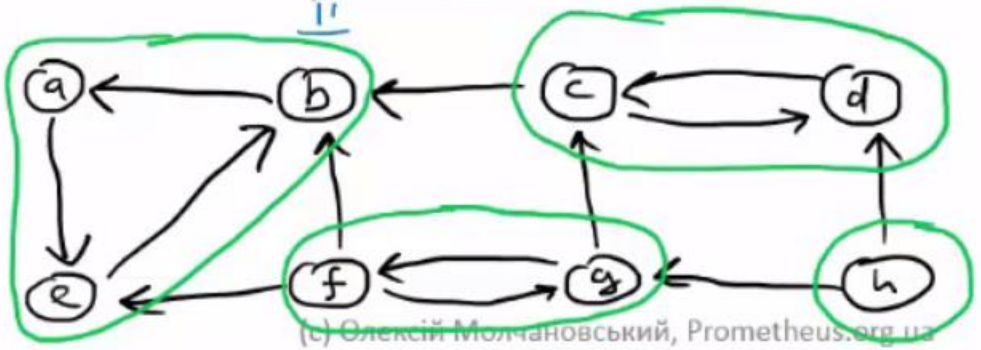


ациклічний граф

G^T – транспонований граф

$V^T = V$

$E^T = \{(u,v) \mid (v,u) \in E\}$



Пошук компонент сильної зв'язності

$T(n,m) = O(n+m)$

StrongConnectedComponent(G)

1. Викликати DFSLoop для обчислення часу завершення $f[v]$ для кожної вершини ← $O(n+m)$
2. Побудувати транспонований граф G^T ← $O(m)$
3. Викликати DFSLoop(G^T) в якому вершини розглядаються в порядку зменшення значень $f[v]$, які були отримані на кроці 1. ← $O(n+m)$

(компоненти сильної зв'язності – вершини, які були пройдені за одну ітерацію в DFSLoop(G^T))

DFSLoop(G)

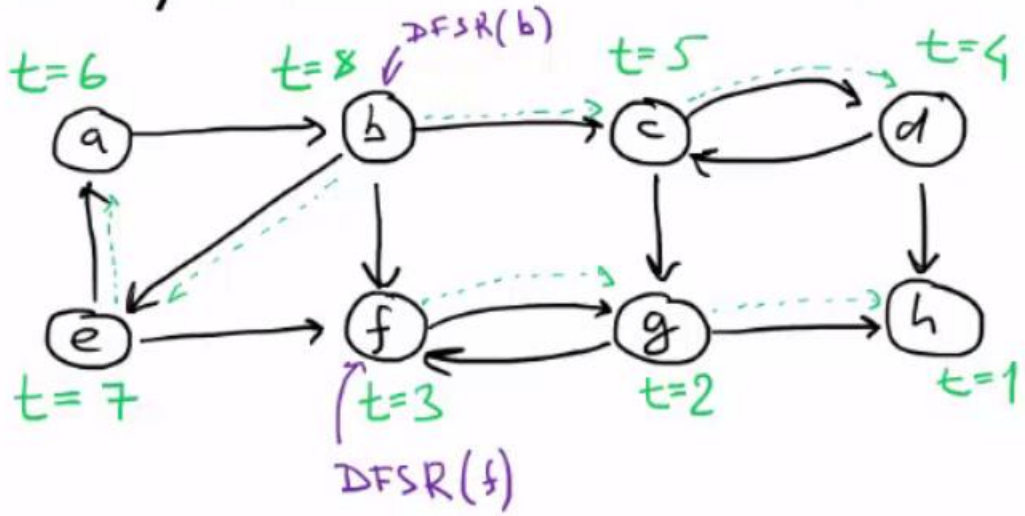
1. позначити всі вершини не досліджені.
2. $t = 0$
3. для кожної вершини v в G:
4. якщо вершина v не досліджена:
5. DFSR(G, v)

під час 2-го виклику DFSLoop(G^T) вершини переглядаються за зменшенням $f[v]$

DFSR(G, s)

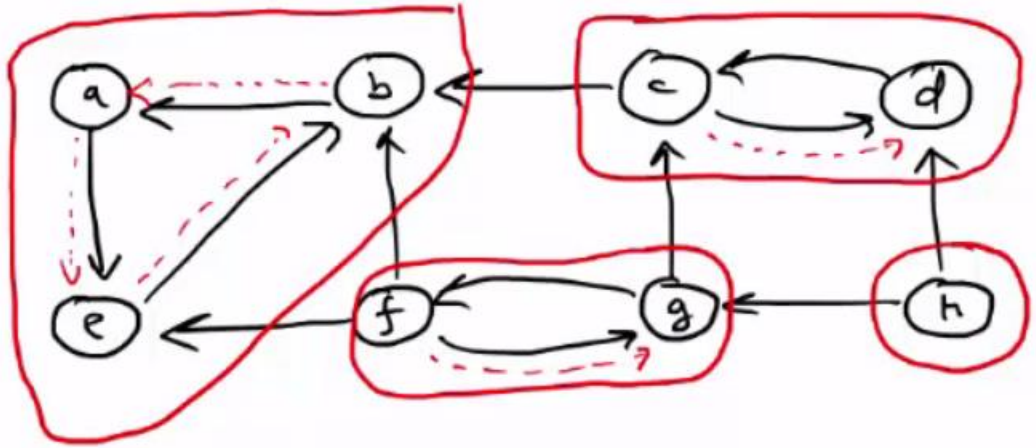
1. позначити s як досліджену
2. для кожного ребра (s,u) в G:
3. якщо u ще не досліджена:
4. DFSR(G, u)
5. $t = t+1$
6. $f[s] = t$ ← час завершення обробки вершини s

Пошук компонент сильної зв'язності. Приклад



1) Запустити DFSLoop(G)

2) Побудувати G^T



3) Запустити DFSLoop (G^T)
і розглядаємо вершини у порядку зменшення часу f[v] (t)