Algorytmy i Struktury Danych

Wykład 6

Algorytmy grafowe

na ogół nie dopuszczamy pętli, czyli krawędzi (u,u)

Graf skierowany:

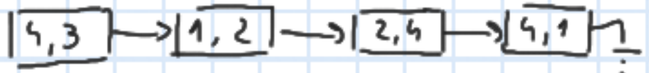
- $G = (V, E)$
- $V = \{v_1, \ldots, v_n\}$ - zbiór wierzchołków
- $E = \{e_1, \ldots, e_m\} \subseteq V \times V$

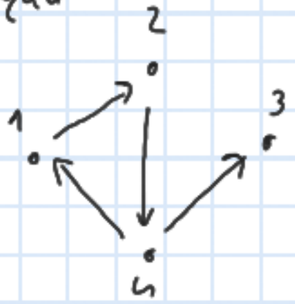
Graf nieskierowany jest zdefiniowany analogicznie, ale każda krawędź to dwuelementowy zbiór wierzchołków

↳ na ogół reprezentujemy jako graf skierowany z krawędziami w dwie strony

Często z krawędziami i wierzchołkami wiążemy dodatkowe informacje (np. wagi / długości)

Reprezentacja przez listę/tablicę krawędzi



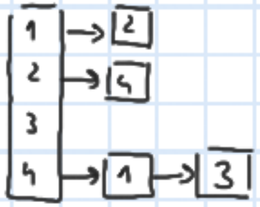


Reprezentacja macierzowa

	1	2	3	4
1	-	T	F	F
2	F	-	F	T
3	F	F	-	F
4	T	F	T	-

- testowanie czy istnieje krawędź $O(1)$
- przegląd krawędzi wychodzących z danego wierzchołka $O(n)$

Reprezentacja przez listy sąsiedztwa



- sprawdzenie czy istnieje krawędź ma złożoność $O(n)$
- przegląd krawędzi wychodzących z danego wierzchołka v ma złożoność $O(d(v))$

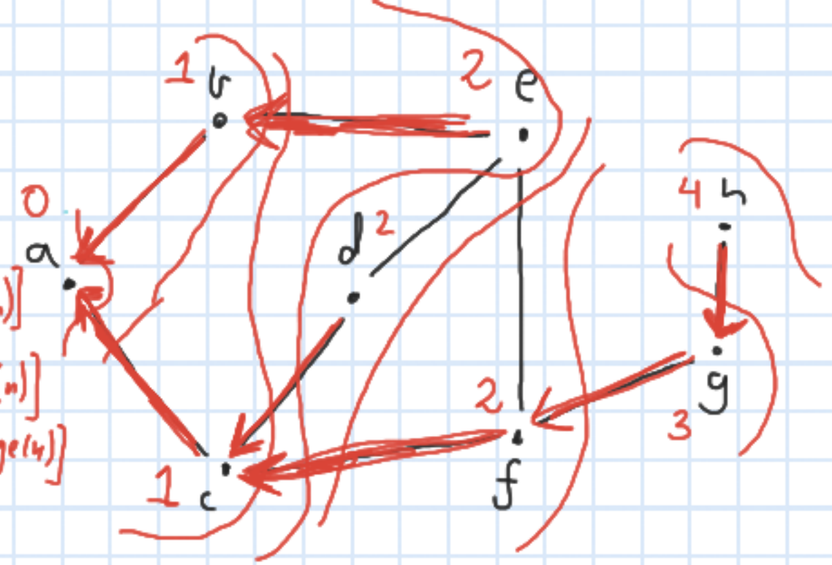
stopień v → liczba wychodzących krawędzi

Algorytm BFS (Breadth-First Search)

przejście grafu wszerz

```
def BFS(G, s)
# G=(V,E), s∈V
    Q = Queue()
    for v∈V:    v.visited = False
    s.d = 0
    s.visited = True
    s.parent = None
    Q.put(s)
    while not Q.isEmpty():
        u = Q.get()
        for v - sąsiad u:
            if not v.visited:
                v.d = u.d + 1
                v.parent = u
                v.visited = True
                Q.put(v)
    return d, parent, visited
```

```
n = len(V)
d = [-1 for v in range(n)]
visited = [False for v in range(n)]
parent = [None for v in range(n)]
```



Q: a | b c | e d f | g | h

0 1 2 3 4

BFS - znajduje najkrótsze ścieżki w sensie liczby krawędzi

Inne zastosowania:
- testowanie spójności grafu
- dwudzielność
- wykrywanie cykli

Złożoność

$O(V+E)$ - rep. listowa

$O(V^2)$ - rep. macierzowa

Algorytm DFS (depth-first search)
przeszukiwanie w głąb

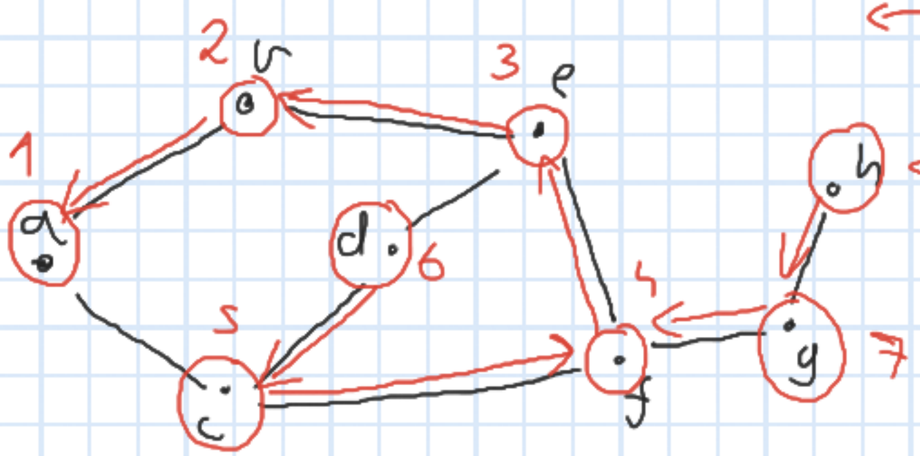
```
def DFS(G)
# G = (V, E)
    for v ∈ V:
        v.visited = False
        v.parent = None

    time = 0
    for u ∈ V:
        if not u.visited:
            DFSVisit(G, u)
```

```
def DFSVisit(G, u)
    nonlocal time
    time += 1
    u.visited = True
    for v - sąsiad u:
        if not v.visited:
            v.parent = u
            DFSVisit(G, v)
    time += 1
```

wierzchołek u zostaje odwiedzony i to jest czas odwiedzenia

u zostaje przetworzony / czas przetworzenia



Złożoności

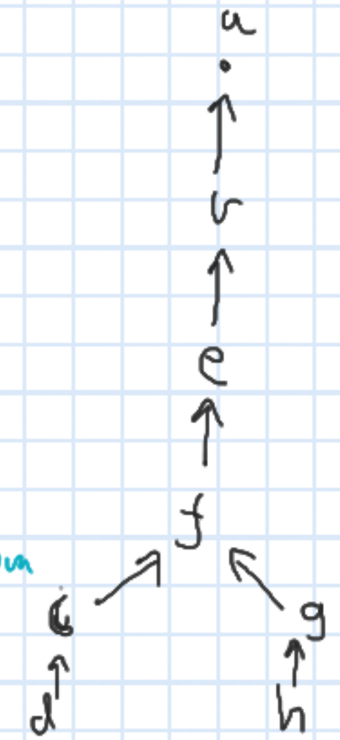
$O(V+E)$ - reprezentacja listowa

$O(V^2)$ - reprezentacja macierzowa

Zastosowania DFS

- spójności
- dwudzielności
- wykrywanie cykli
- sortowanie topologiczne
- silnie spójne składowe
- cykl Eulera
- mosty / punkty artykulacji

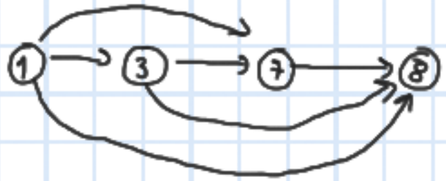
drzewo DFS



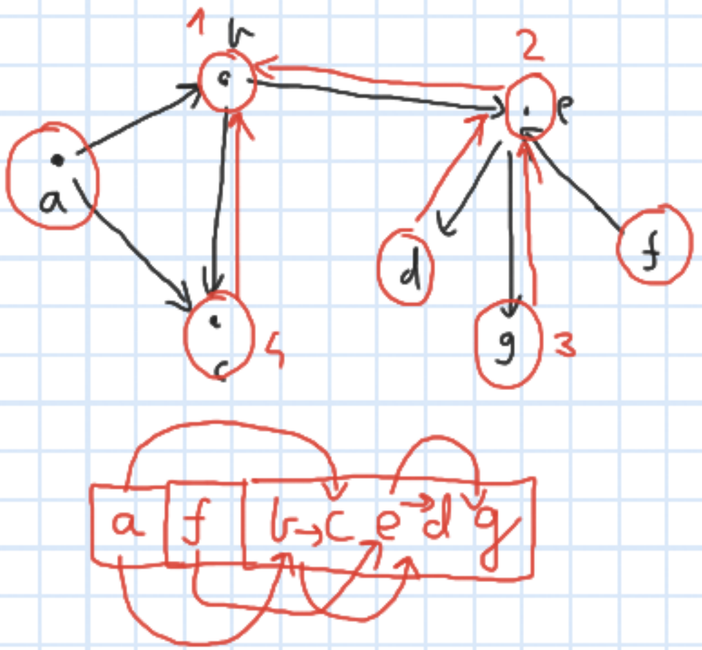
Sortowanie topologiczne

dag - directed acyclic graph
(skierowany graf acykliczny)

sortowanie topologiczne dagu: ułożenie jego wierzchołków w takiej kolejności, że krawędzie wskazują wyłącznie z lewej na prawą



zastosowanie: wyznaczenie kolejności realizacji zadań jeśli niektóre muszą być wykonane przed innymi



Algorytm

- uruchomić DFS
- po "przetworzeniu" każdego wierzchołka dopisać go na początek tworzonej listy