Algorytmy i Struktury Danych

Wykład 5

Algorytmy grafowe

Graf skierowany:

$G = (V, E)$, gdzie

$V = \{v_1, \dots, v_n\}$ — zbiór wierzchołków

$E = \{e_1, \dots, e_m\}$ — zbiór krawędzi } na ogół nie dopuszczamy pętli (v, v)

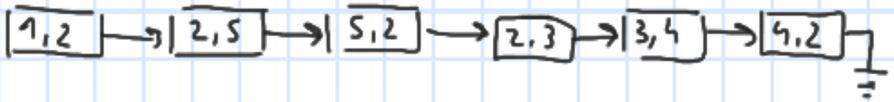
$E \subseteq V \times V$

Z każdą krawędzią / wierzchołkiem można związać dodatkowe informacje (np. wagę lub długość krawędzi)

Grafy nieskierowane — graf w którym krawędzie to zbiory dwuelementowe (często reprezentowane jako grafy skierowane gdzie istnieją krawędzie są w obie strony

Reprezentacje grafów

① lista / tablica krawędzi



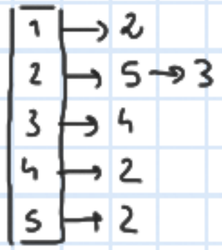
② Reprezentacja macierzowa

	1	2	3	4	5
1	-	1	0	0	0
2	0	-	1	0	1
3	0	0	-	1	0
4	0	1	0	-	0
5	0	1	0	0	-

Zalety
- prostota
- dostęp $O(1)$ do dowolnej krawędzi

Wady
- złożoność pamięciowa $\Theta(V^2)$
- dostęp w czasie $O(V)$ do krawędzi z danego wierzchołka v

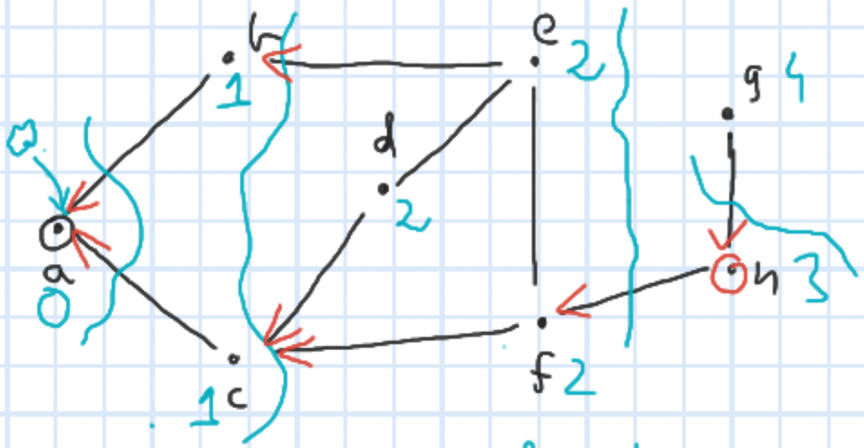
③ Listy sąsiedztwa



Zalety
- złoż. pamięciowa $\Theta(V+E)$
- dostęp do krawędzi wychodzących z wierzchołka v w czasie $O(d(v))$ (stopień wierzchołka)

Wady
- brak dostępu w czasie $O(1)$ do konkretnej krawędzi

Algorytm przeszukiwania grafu wszerz (Breadth-First Search; BFS)



Q: | a | b | c | e | d | f | h | g |
0 1 1 2 2 2 3 4

Zastosowania BFS

- najkrótsze ścieżki z danego źródła
- spójność
- wykrywanie cykli
- testowanie dwuspójności

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

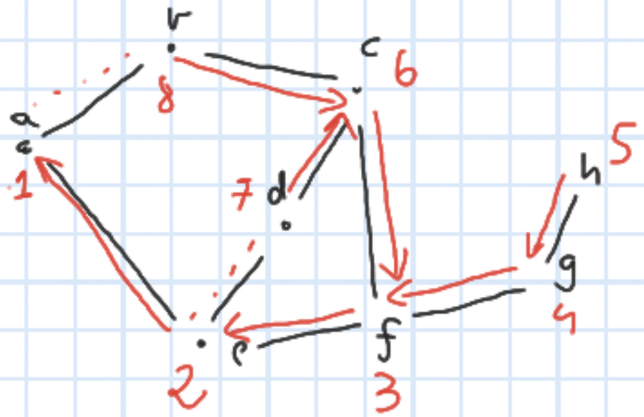
typowa implementacja
```
n = len(V)
d = [-1 for v in range(n)]
visited = [False for v in range(n)]
parent = [None for v in range(n)]
```
↓
-1

Złożoność BFS

- listowa: $\Theta(V+E)$
- macierz: $\Theta(V^2)$

← zwraca d, parent, visited

Algorytm przeszukiwania w głąb (Depth-First Search; DFS)




```
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   ← czas odwiedzenia
    u.visited = True
    for v - sąsiad u:
        if not v.visited:
            v.parent = u
            DFSVisit(G, v)
    time += 1   ← czas przetworzenia
```

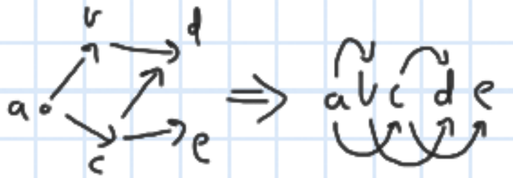
Złożoność

reprezentacja listowa $\Theta(V+E)$

reprezentacja macierzowa $\Theta(V^2)$

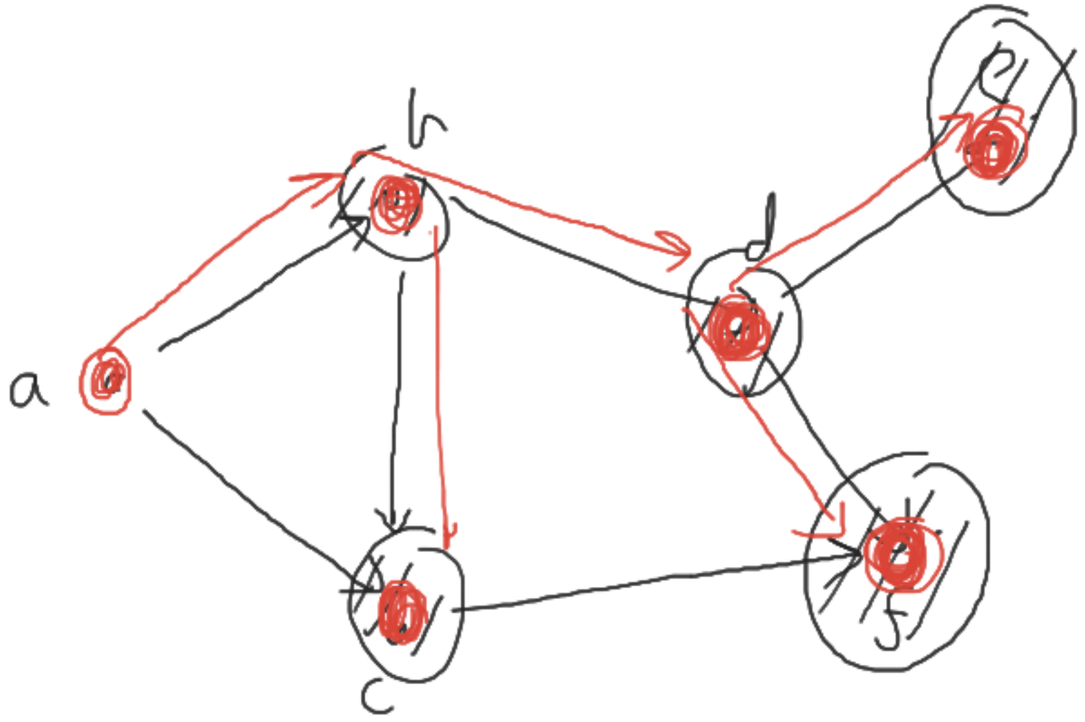
Zastosowania

- spójność
- dwudzielność
- wykrywanie cykli



- sortowanie topologiczne
- silnie spójne składowe
- cykl Eulera
- mosty / punkty artykulacji

DAG



$O(V^2)$