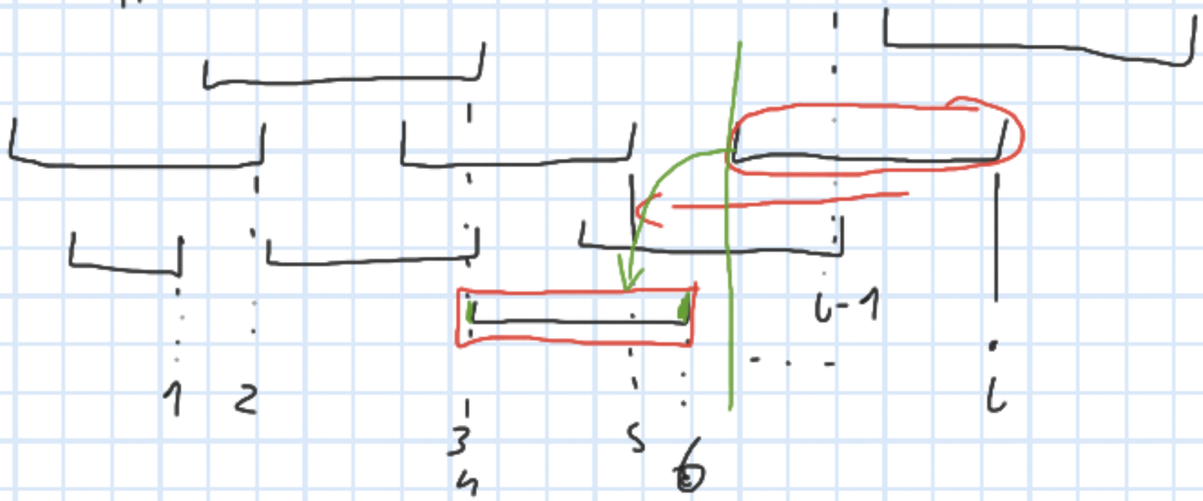
ASD - Wykład 9

Zadanie offline 4



$f(i,t)$ = maks. pojemność budynków na terenie do końca i-go, kosztujących $\leq t$

$$f(i,t) = \max\Big(f(i-1,t),\ \text{pojemność}(i) + f\big(\text{prev}(i), t - \text{cena}(i)\big)\Big)$$

bud. j nie nachodzi na i

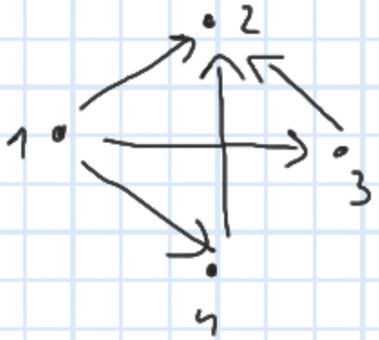
$O(n^2 p)$

$O(n^2 + np)$

$O(n \log n + np)$

ASD - Wykład 9

Algorytmy grafowe



Graf skierowany:

$G = (V, E)$

$V = (v_1, \ldots, v_n)$ – zbiór wierzchołków

$E = \{e_1, \ldots, e_m\}$, $E \subseteq V \times V$

na ogół nie dopuszczamy krawędzi (u,u)

Z każdą krawędzią / wierzchołkiem można związać dodatkowe informacje

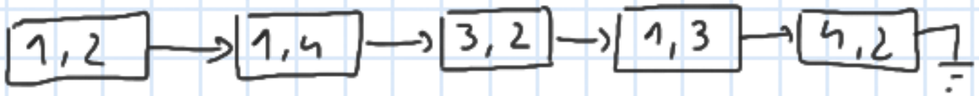
Graf nieskierowany – krawędzie są zbiorami dwuelementowymi

↓

często realizowane jako grafy skierowane, gdzie krawędzie są "w obie strony"

Reprezentacje grafów

Lista / tablica krawędzi

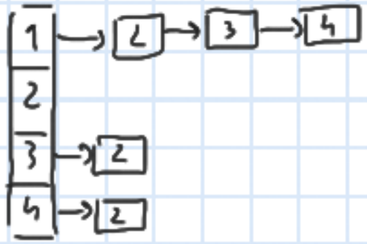


Reprezentacja macierzowa

	1	2	3	4
→ 1	–	1	1	1
2	0	–	0	0
3	0	1	–	0
4	0	1	0	–

ten fragment wystarcza dla grafów nieskierowanych

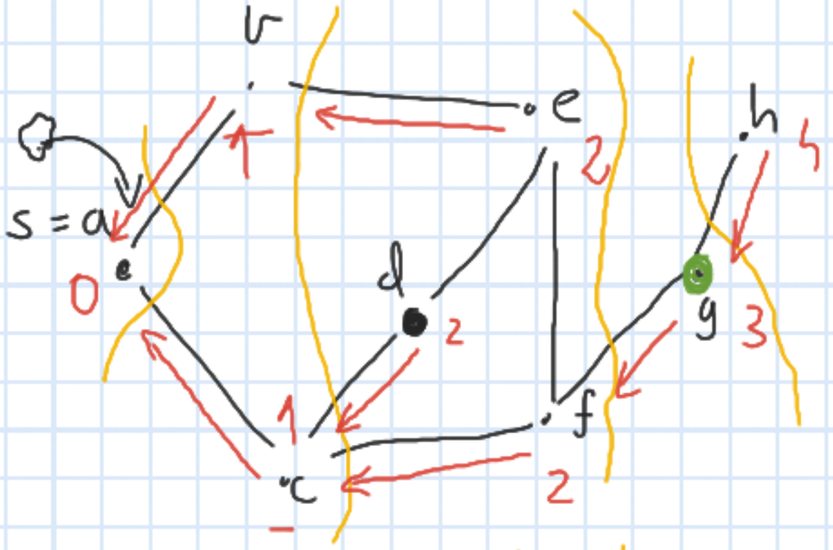
Reprezentacja przez listy sąsiedztwa



Algorytm BFS (breadth-first search)

Przeszukiwanie w szerz

```
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.empty():
    u = Q.get()
    for v - sąsiedzi u:
      if not v.visited:
        v.visited = True
        v.d = u.d + 1
        v.parent = u
        Q.put(v)
```

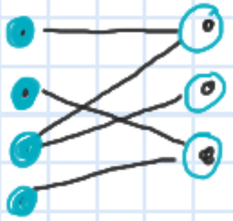


Q: a | b | c | e | d | f | g | h

0 ↑ 1 2 3 4

Zastosowania

- najkrótsze ścieżki (w grafie bez wag)
- spójności grafu
- wykrywanie cykli
- dwudzielności



Złożoność

rep. macierzowa

$O(V^2)$

rep. listowa

$O(V+E)$

Algorytm DFS (depth-first search)

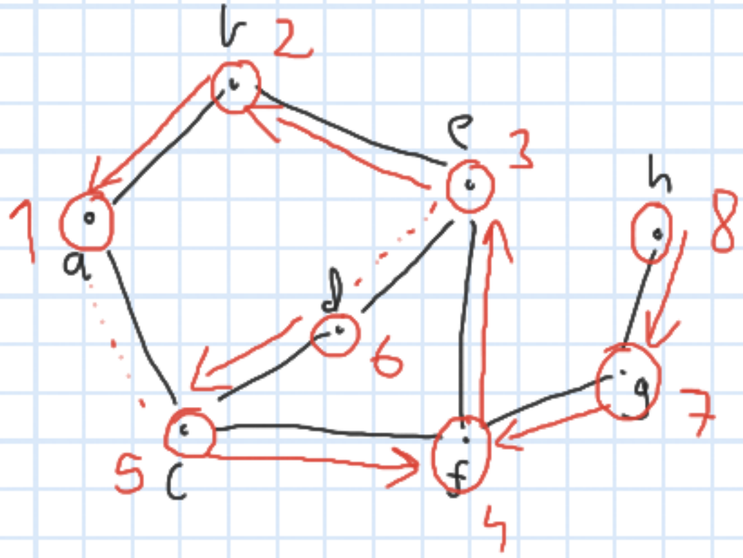
Przeszukiwanie w głąb

```
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)
```

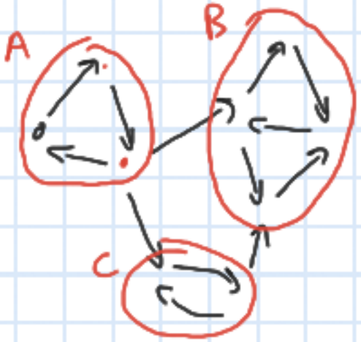
Złożoność

reprezentacja macierzowa $O(V^2)$

reprezentacja listowa $O(V+E)$

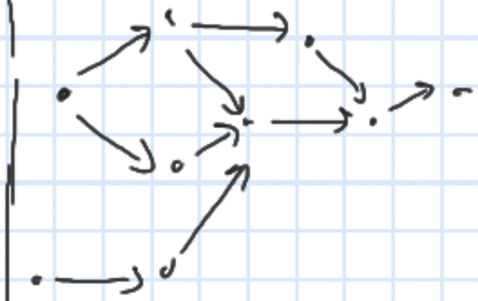



```
def DFSVisit(G, u):
    nonlocal time
    time += 1
    u.visited = True      <- wierzchołek został odwiedzony / czas odwiedzenia
    for v - sąsiad u:
        if not v.visited:
            v.parent = u
            DFSVisit(G, v)
    time += 1             <- u został przetworzony / czas przetworzenia
```



Zastosowania

- spójność
- dwudzielność
- wyłuskiwanie cykli
- sortowanie topologiczne



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