

grafi

# definizioni

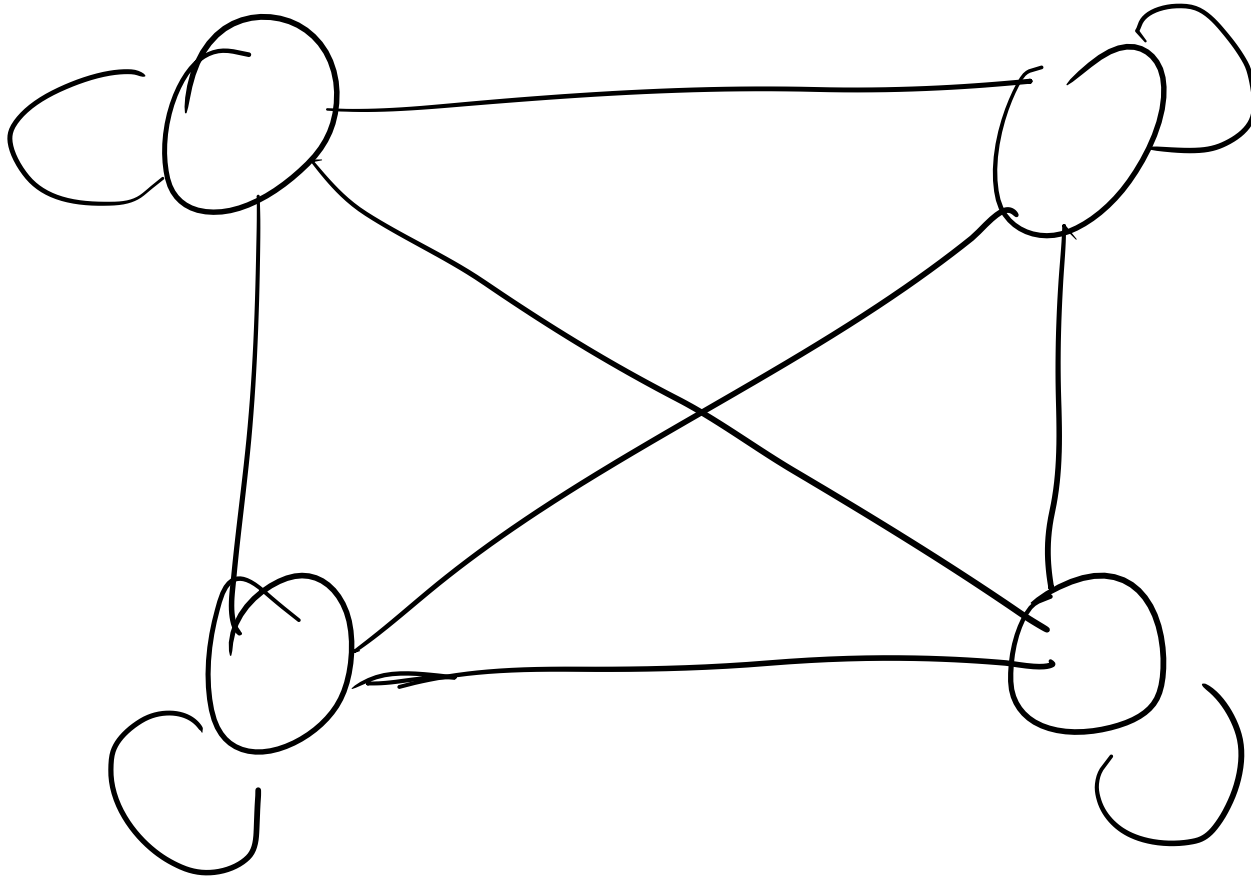
- Un grafo è un insieme di nodi  $V$  detti **vertici** connessi da collegamenti chiamati **archi** raggruppati in un insieme  $E$
- Formalmente, definiremo

$$G = (V, E)$$

$$\bar{E} : V \times V \rightarrow \mathbb{R}$$

# definizioni

- $|V|$  = numero di vertici nel grafo
- $|E|$  = numero di archi nel grafo
- Grafo **sparso**: un grafo in cui  $|E| \ll |V|^2$
- Grafo **denso**: un grafo in cui  $|E| \approx |V|^2$
- Grafo **orientato**: un grafo in cui gli archi hanno un verso
- Grafo **non orientato**: un grafo in cui gli archi non hanno un verso



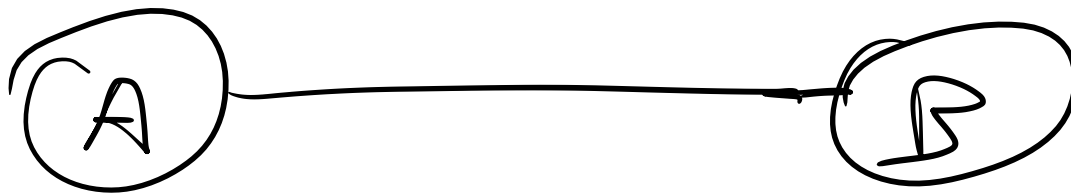
$$|V| = 4$$

$$|E| = 7$$

# definizioni

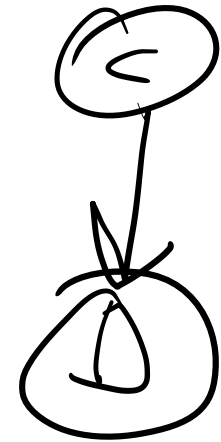
Un nodo  $v_i$  si dice **adiacente** a un nodo  $v_j$  se esiste un arco  $e_{ij} = (v_i, v_j)$  che li collega

$$v_i, v_j \in V, (v_i, v_j) \in E$$



A è adiacente a B

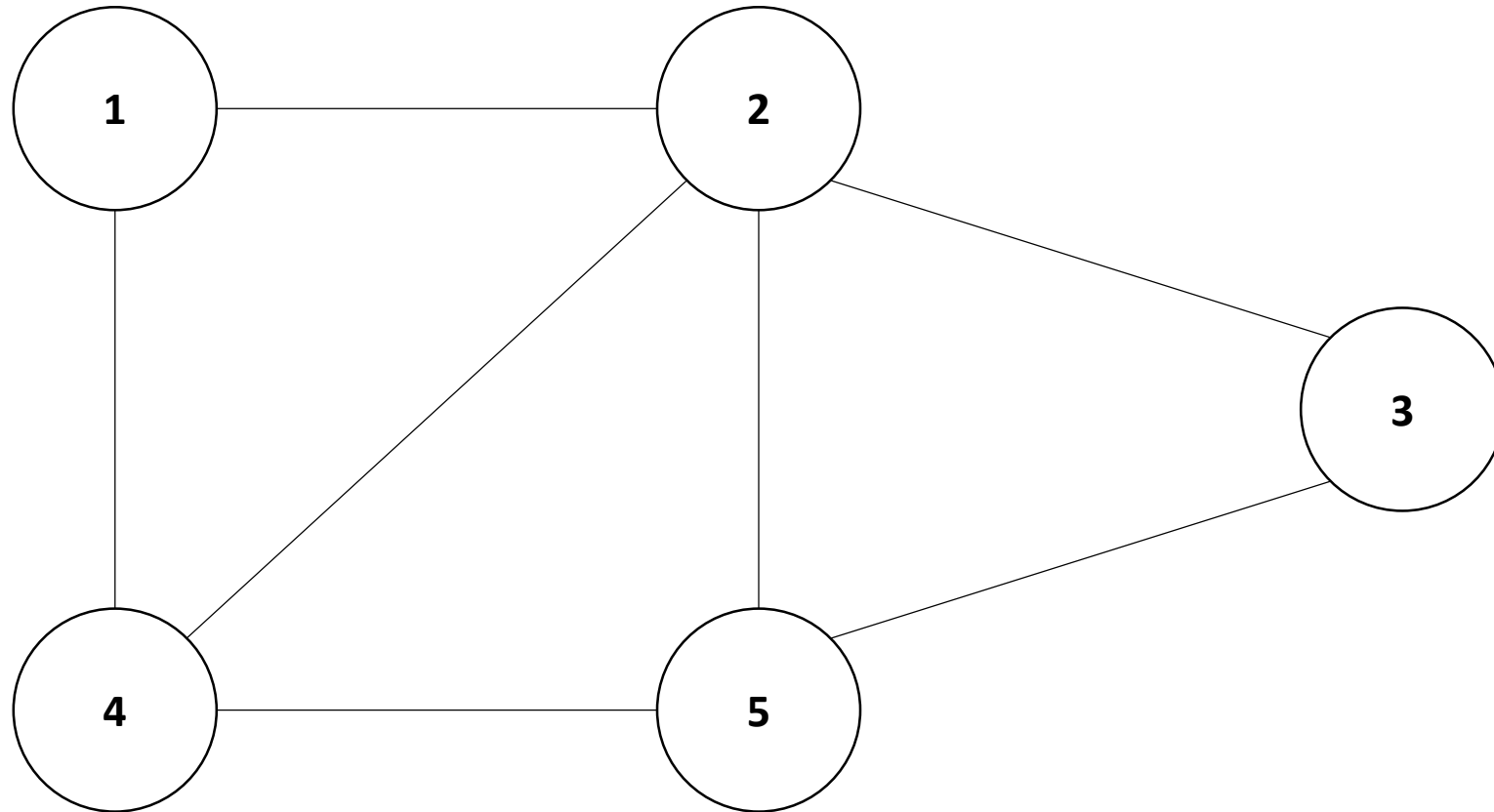
B è adiacente a A



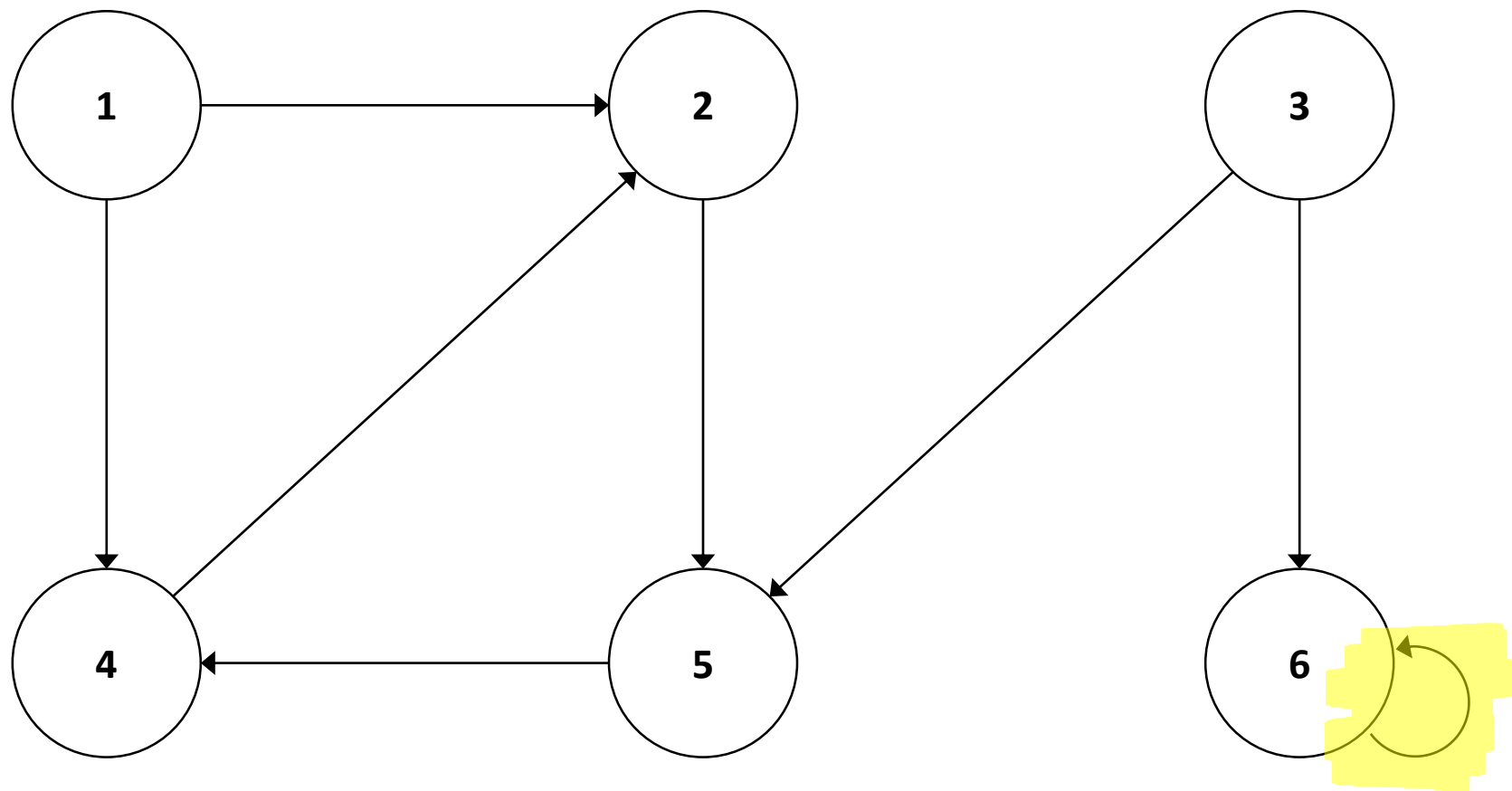
C è-adj. D

D non è-adj. C

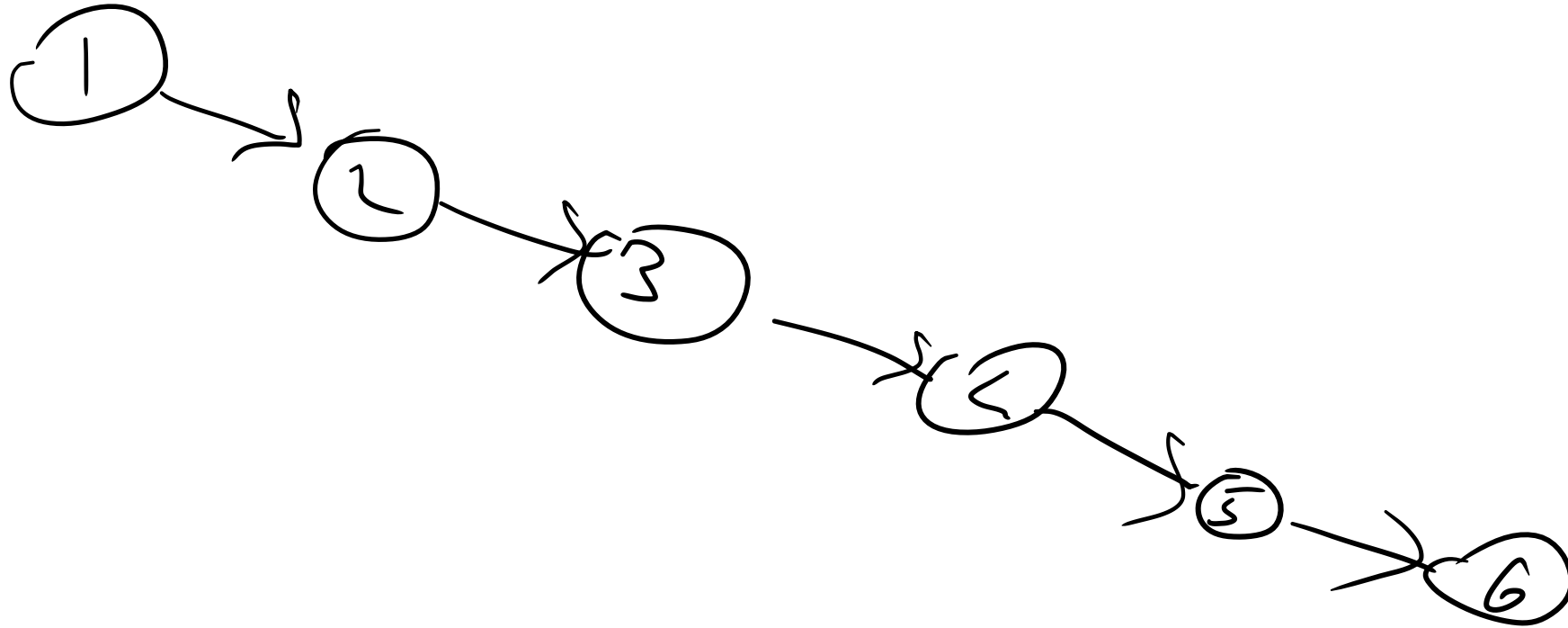
# Grafo non orientato



# Grafo orientato



- 1,2,3,4,5,6,7,8,9,10



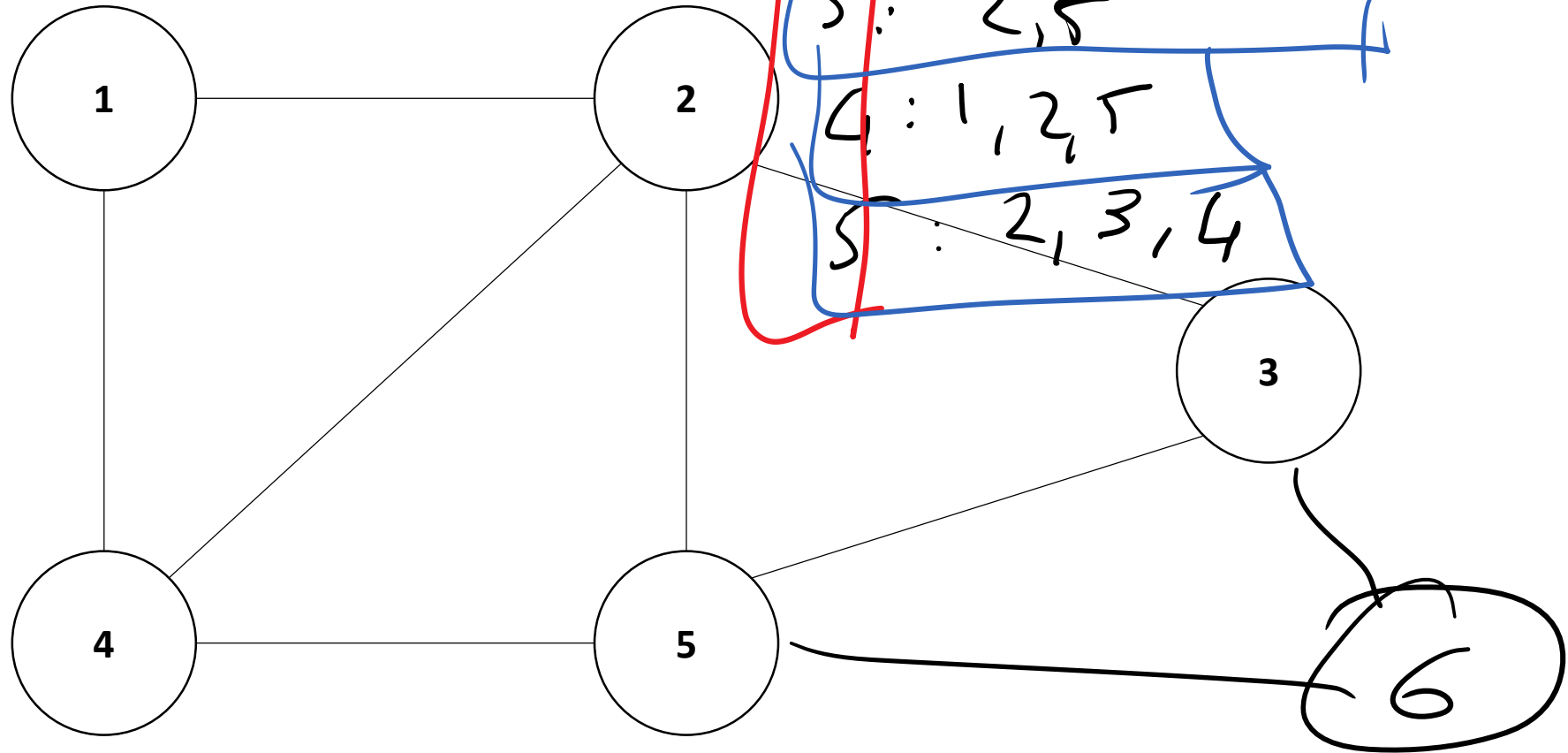


Vertici (nodi)

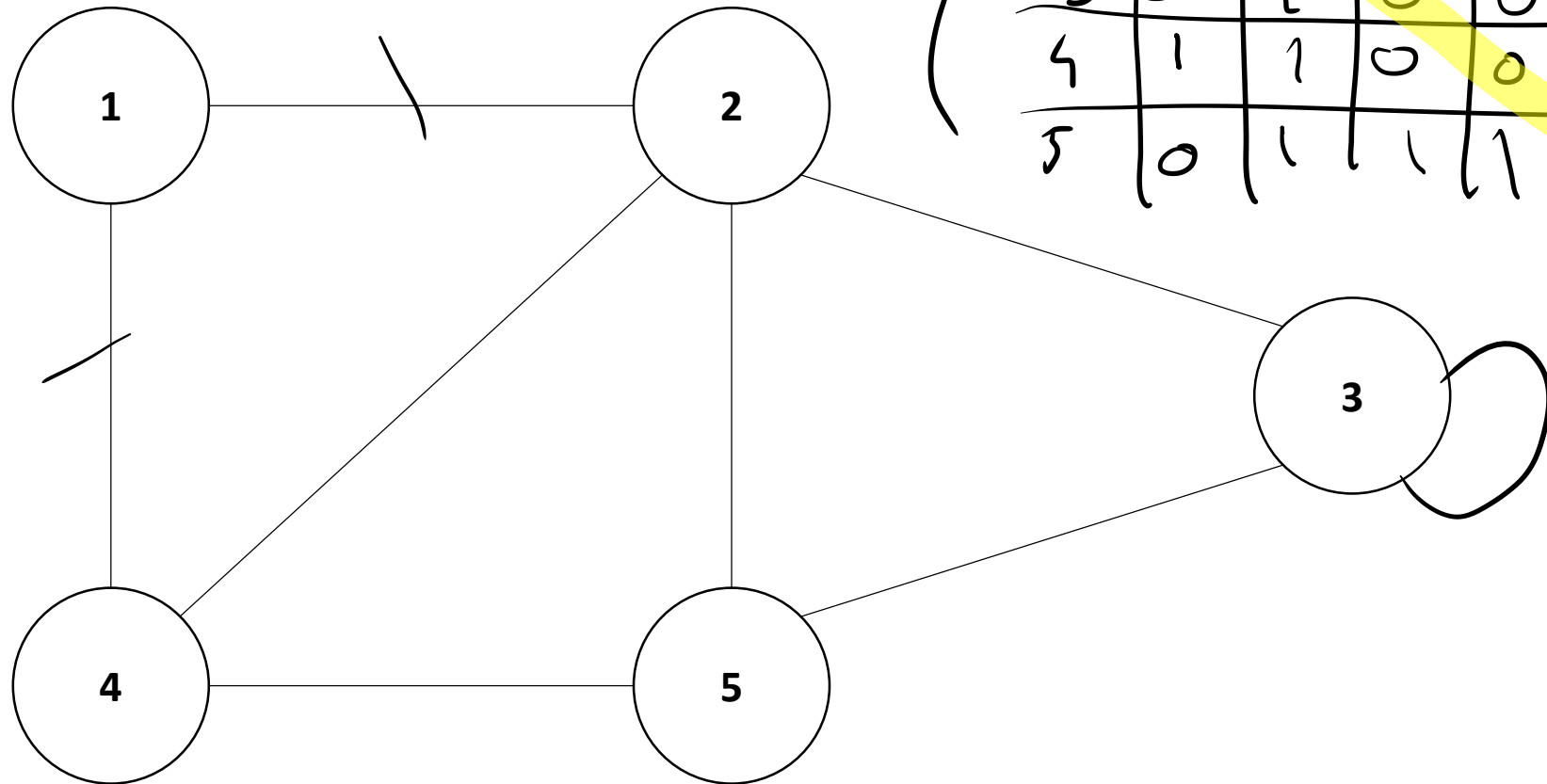
Archi

- ORIENTAMENTO    • PESO
- VALORE
- PUNTERO AI NODI  
ADACENI

# Grafo non orientato



# Grafo non orientato



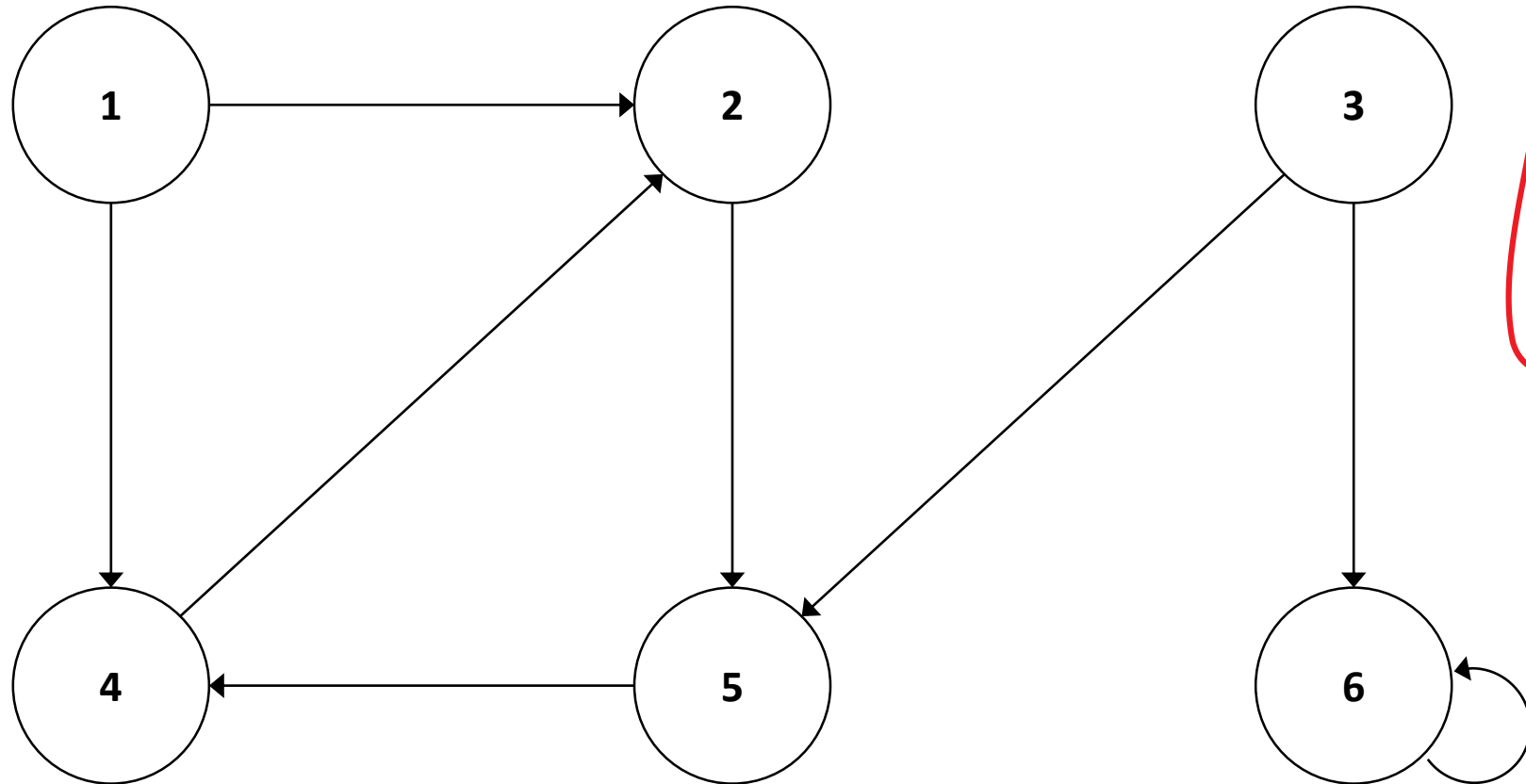
$i \setminus j$

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

# Grafo orientato

1: 2, 4  
2: 5  
3: 5, 6

4: 2  
5: 4  
6: 6

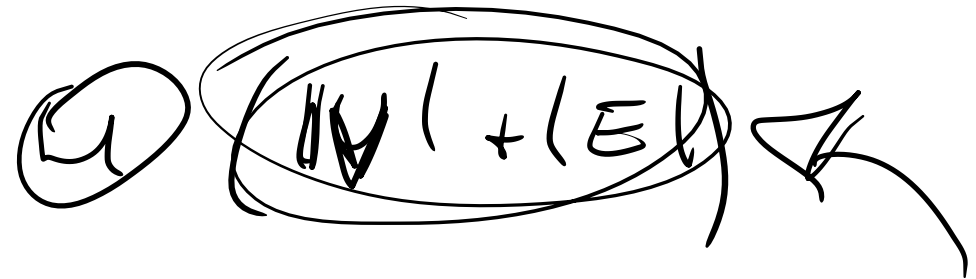


1: 2, 4  
2: 1, 3  
3: 2, 5  
4: 1, 2, 5  
5: 2, 3

# rappresentazioni

- Liste di adiacenza:

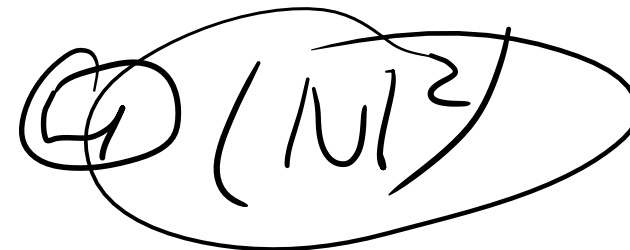
- Per ogni vertice nel grafo, si inseriscono i nodi adiacenti in una lista linkata
- Spazio richiesto in memoria:
  - Grafo non orientato:  $2|E|$
  - Grafo orientato:  $|E|$



- Matrice di adiacenza:

- Questa matrice ha dimensione  $|V| \times |V|$
- Se esiste un arco  $(v_i, v_j)$ , l'elemento in posizione  $(i,j)$  avrà valore 1.
- Spazio richiesto in memoria:  $|V|^2$

$$E \ll V^2$$

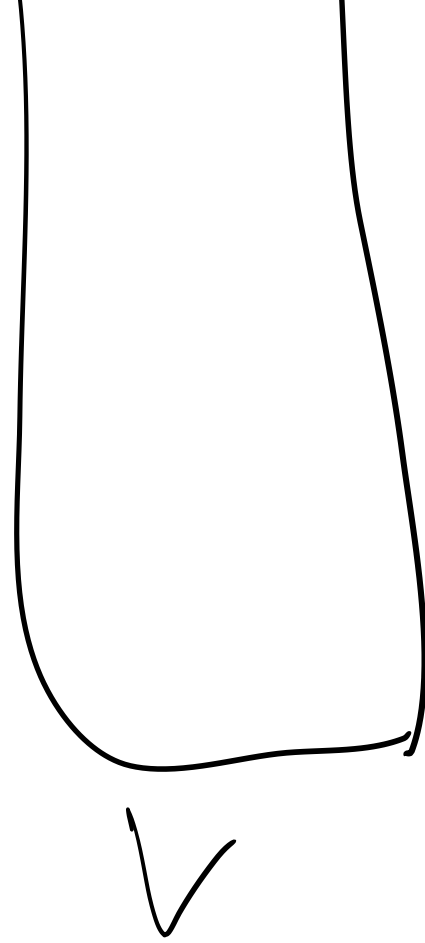
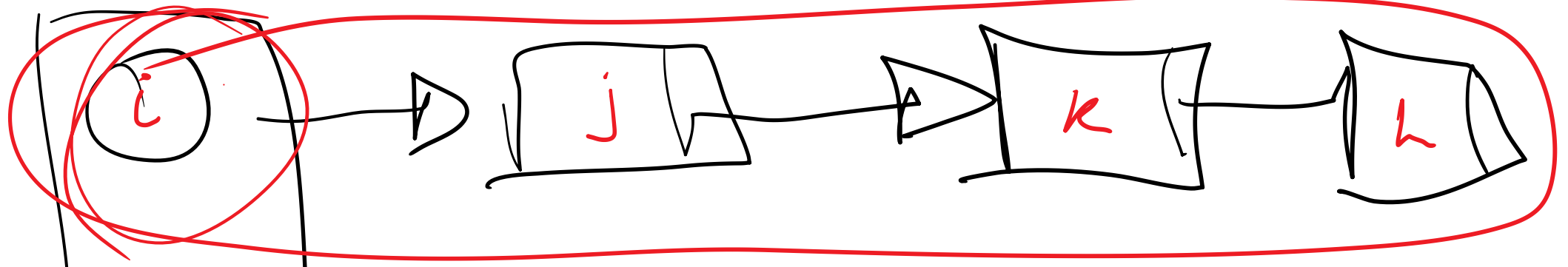


# Matrice di adiacenza

- Vertice (template)
- Grafo con matrice di adiacenza

Vertice → solo valore

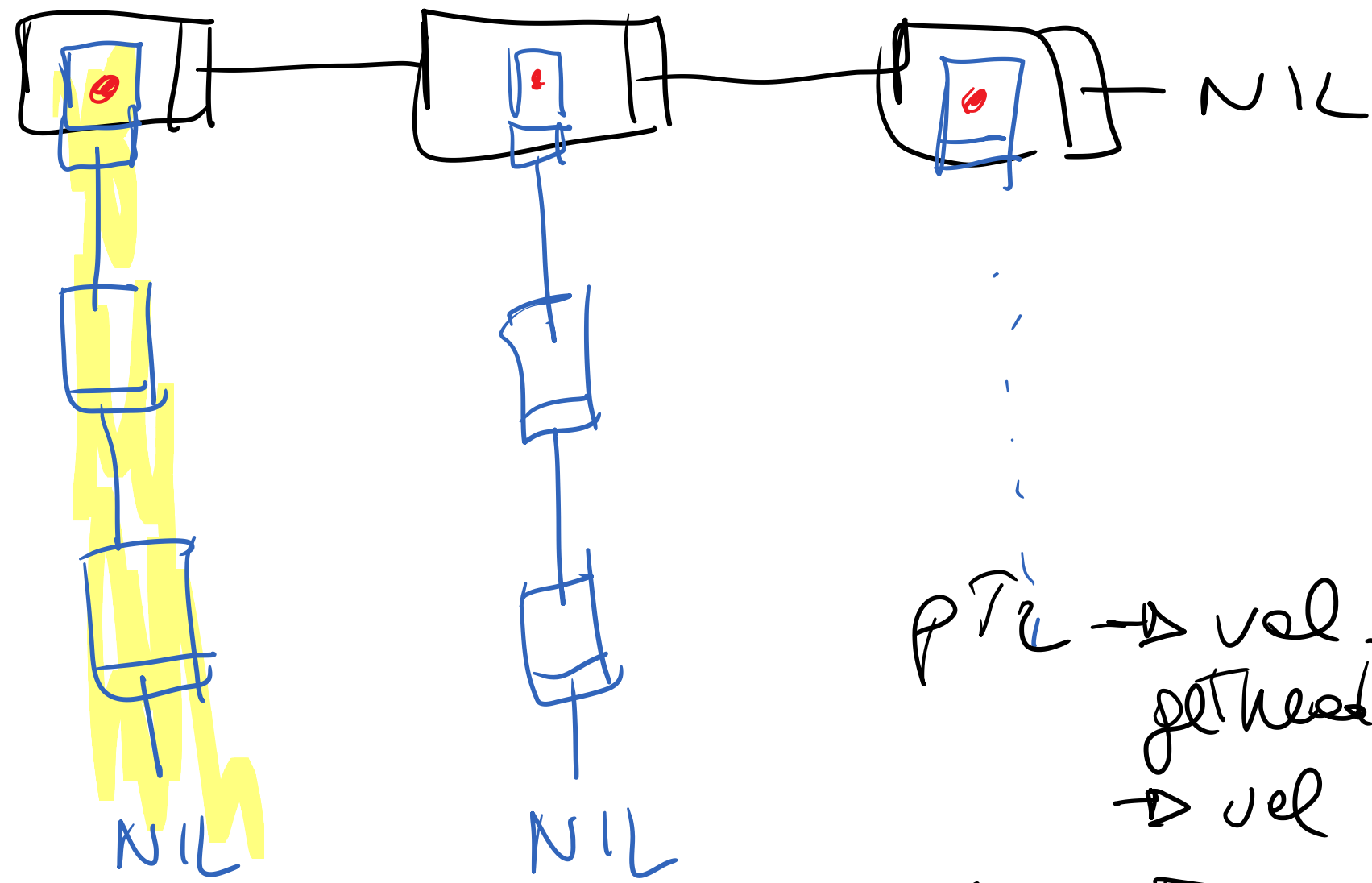
Grafo: → matrice  
ARRAY di vertice,



$2E$  (NON ORIENTED)

$E$  (ORIENTED)

vertices  
GraphVertex  $\leftrightarrow$

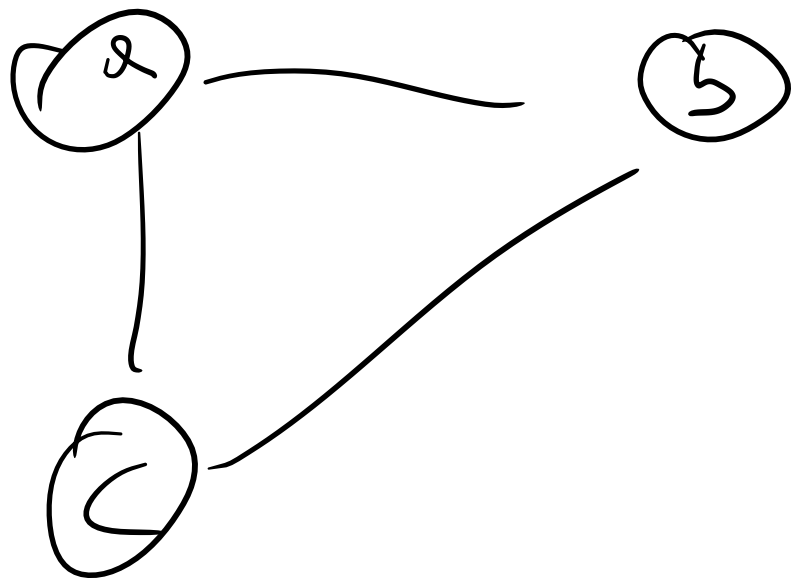


$PT \rightarrow val, \text{gethead}() \rightarrow val$

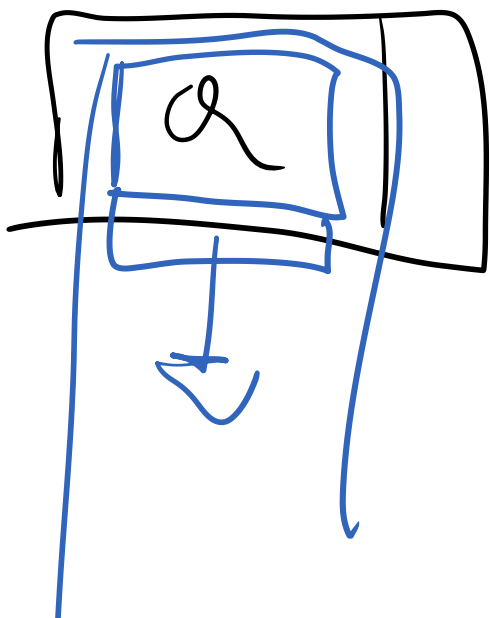
Node  $\langle$  GraphVertex  $\langle T \rangle \rangle$

$\langle$  next  
val  $\rangle$

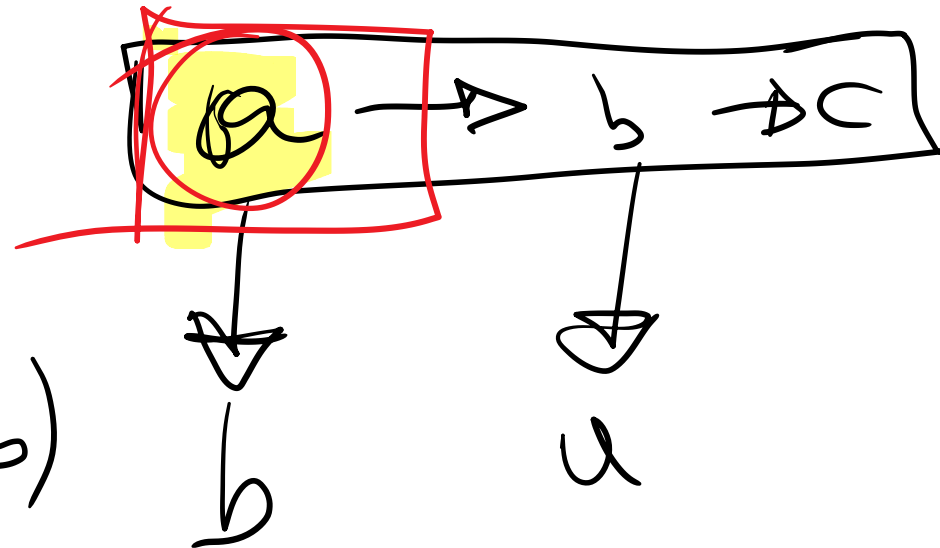




node 1



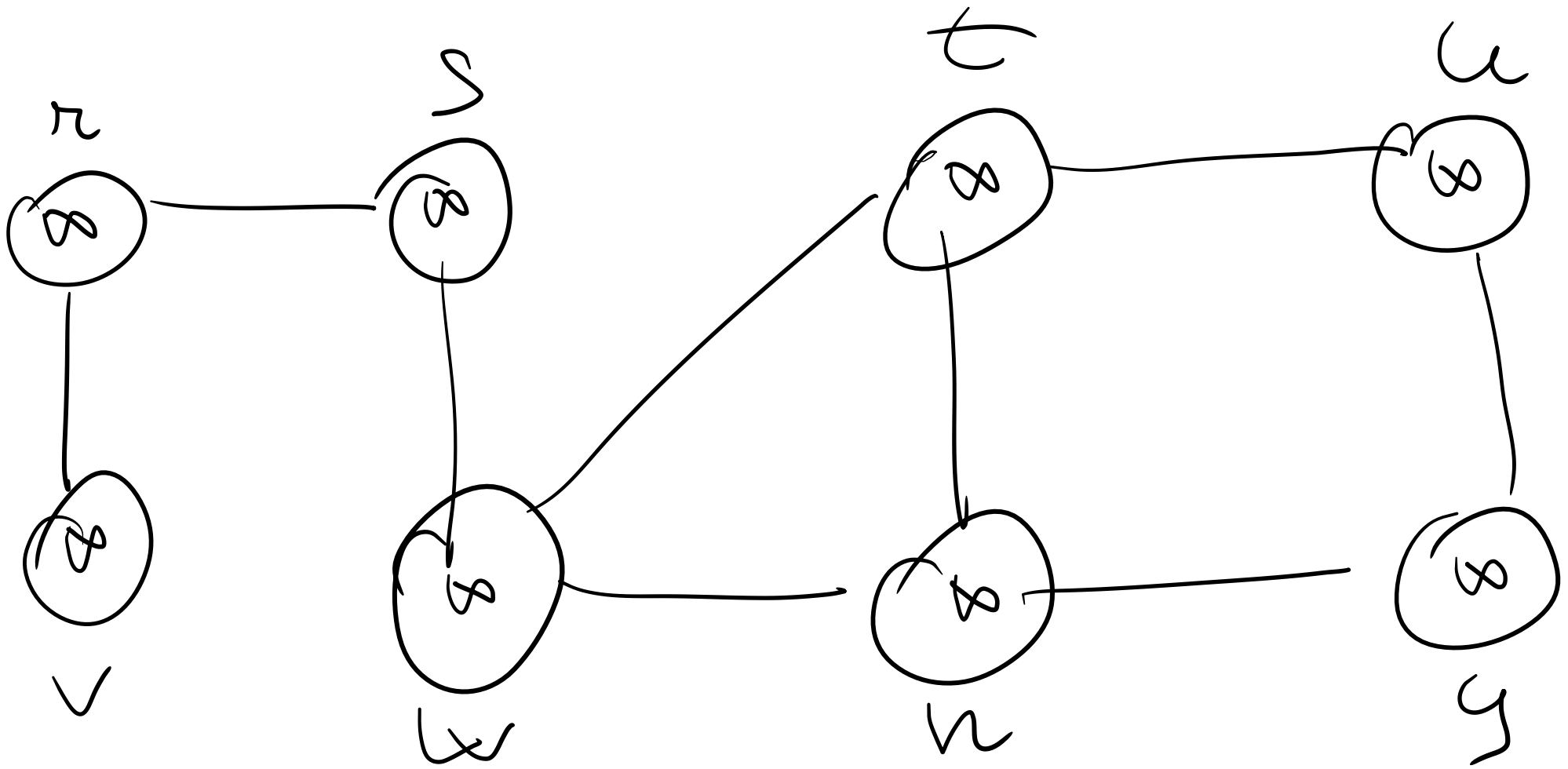
(2, 5)



VISITA IN  
AMPIEZZA

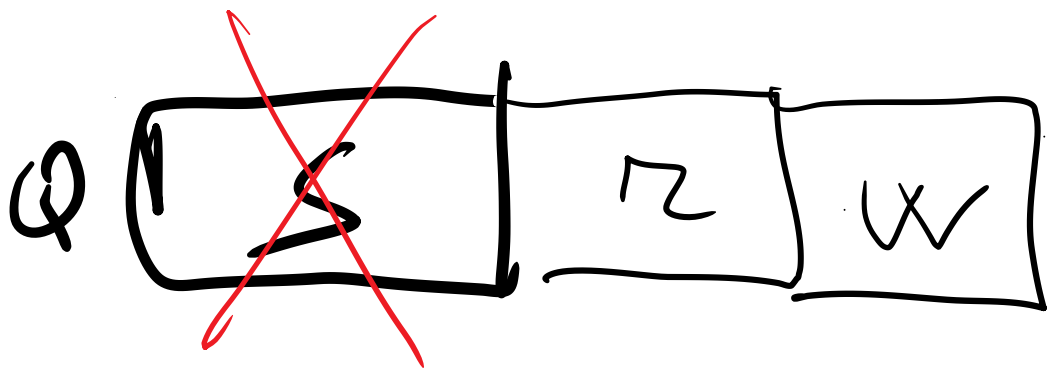
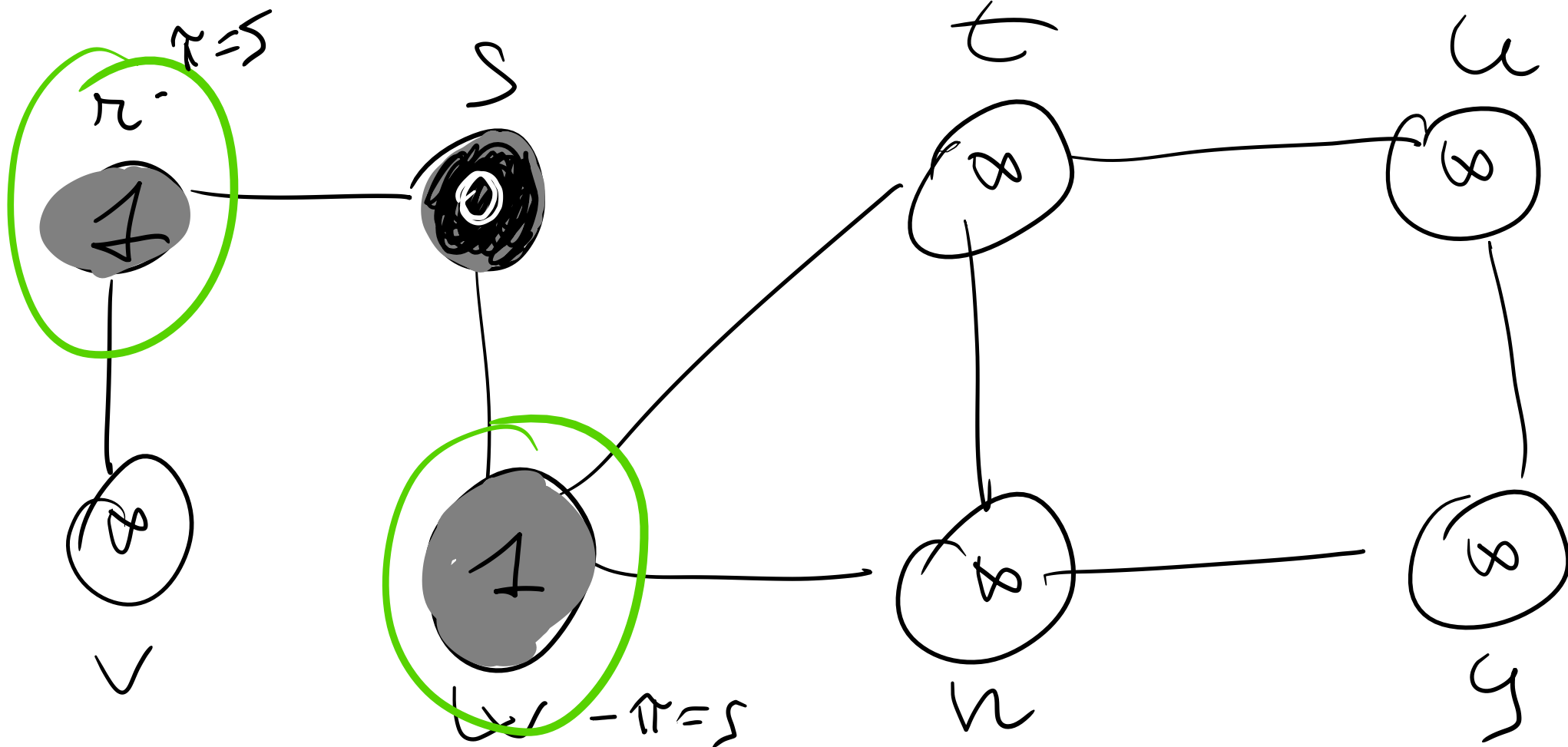
BREADTH-FIRST SEARCH

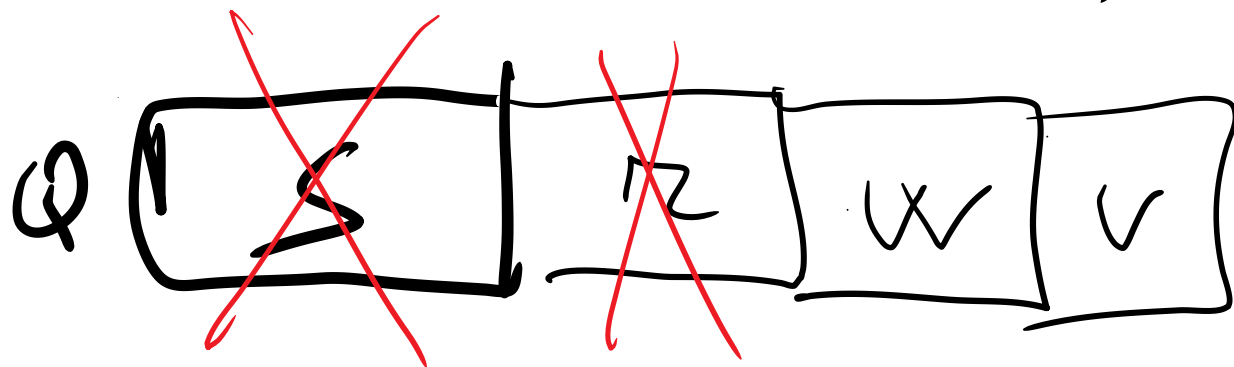
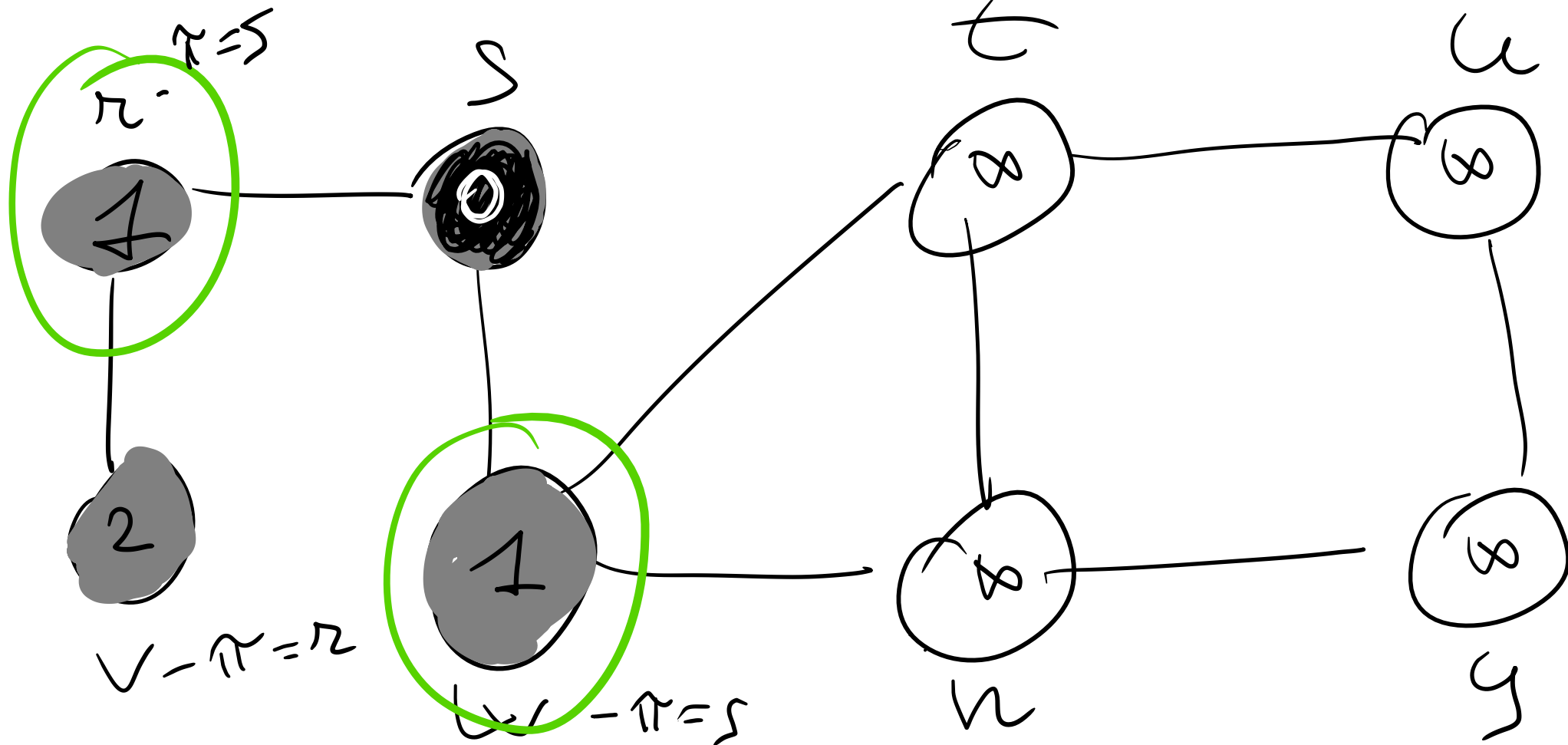
BFS

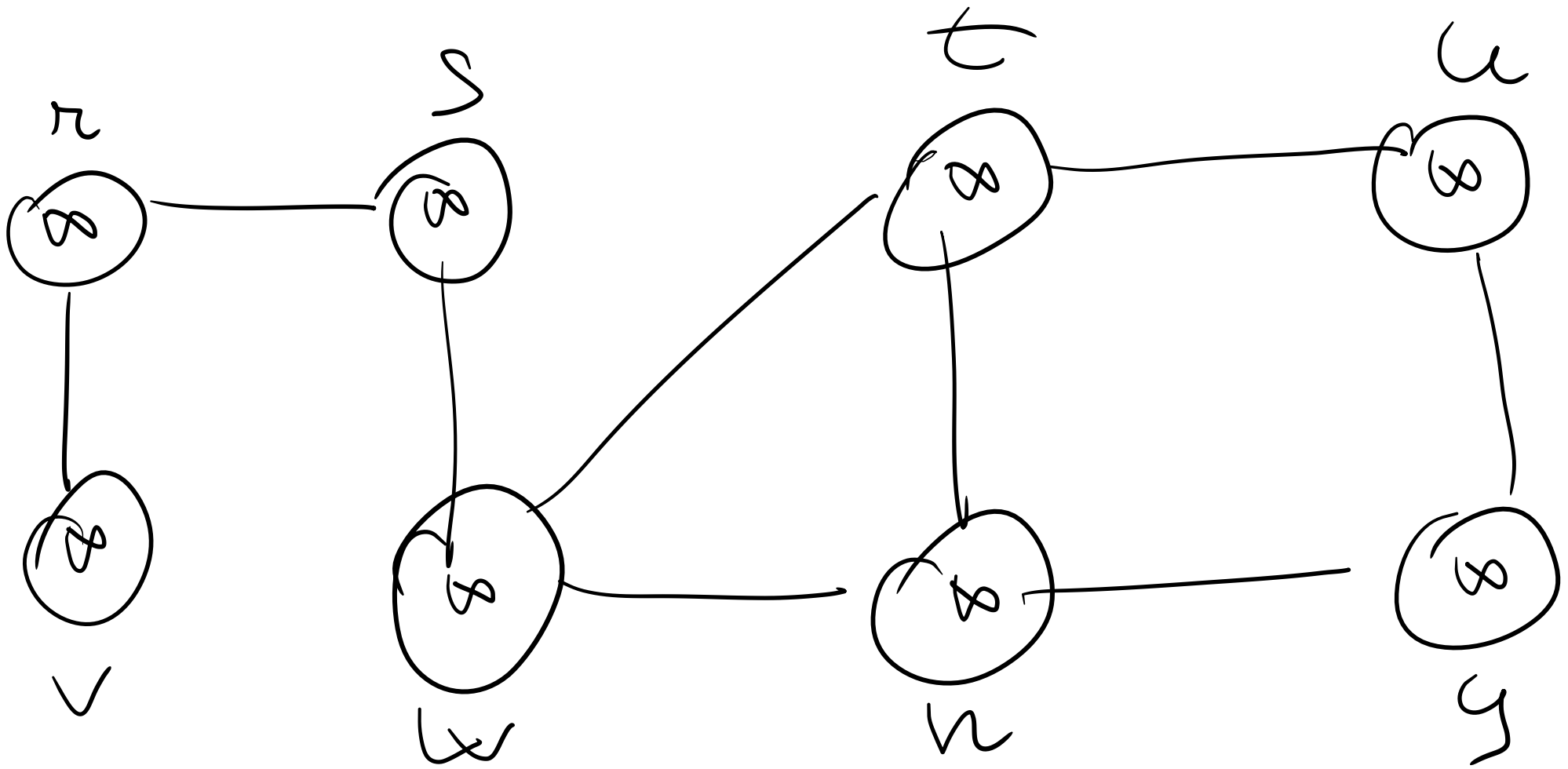


N	N	N	N	N	N	N	N	N

$d \approx$

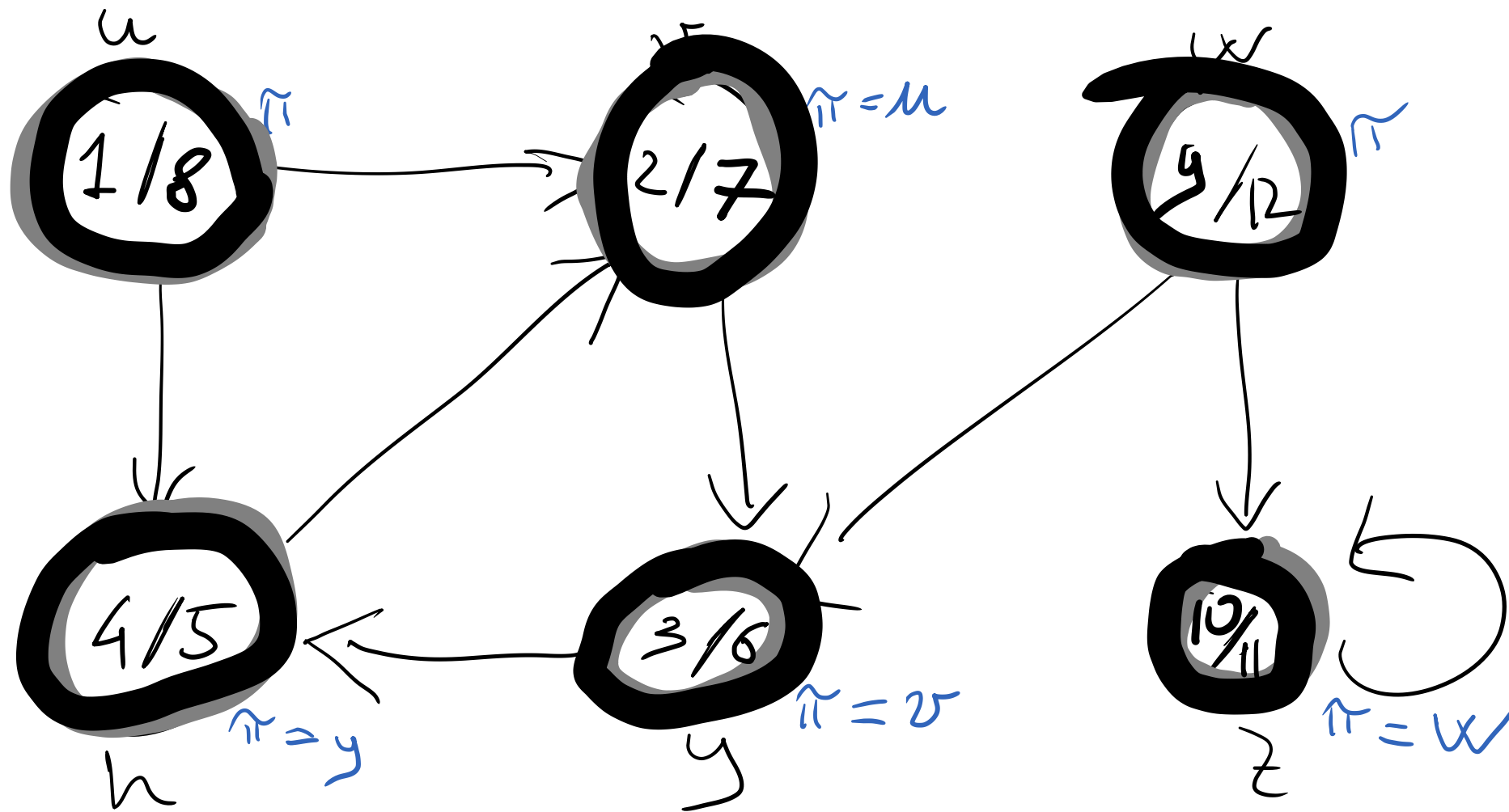






N	N	N	N	N	N	N	N	N

$d \approx$



time

0 1 2 3 4 5 6 7 8 9 10 11 12

# BFS – pseudo-codice

BFS( $G, s$ )

for ogni vertice  $u$  nel grafo  $G$ :

color[ $u$ ] = white

$d[u] = \text{inf}$

$p[u] = \text{null}$

color[ $s$ ] = gray

$p[s] = \text{null}$

$d[s] = 0$

INITIALIZAZIONE

$Q = /$

$Q.\text{enqueue}(s)$

while  $Q \neq /$

$u = Q.\text{dequeue}()$

for ogni vertice  $v$  in  $\text{adj}[u]$

if (color[ $v$ ] == white)

color[ $v$ ] = gray

$d[v] = d[u] + 1$

$p[v] = u$

$Q.\text{enqueue}(v)$

color[ $u$ ] = black

VISITA






# DFS - pseudocode

DFS(G)

```
for ogni vertice u in G:
    color[u] = white
    d[u] = inf
    f[u] = inf
    p[u] = null
global time = 0
for ogni vertice u in G:
    if color[u] == white
        DFS-visit(u)
```



DFS-visit(u)

```
color[u] = gray
time = time + 1
d[u] = time
for v in adj[u]:
    if color[v] == white
        p[v] = u
        DFS-visit(v)
color[u] = black
time = time + 1
f[u] = time
```

