ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

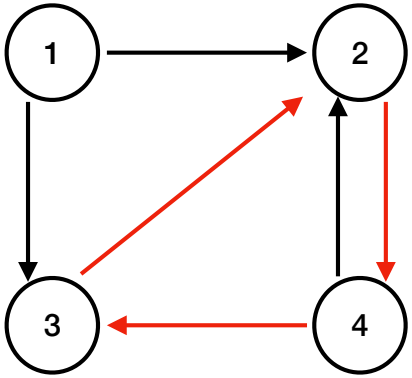
A.A. 2022/23

STRUTTURE DATI:
DAG e ordinamento topologico

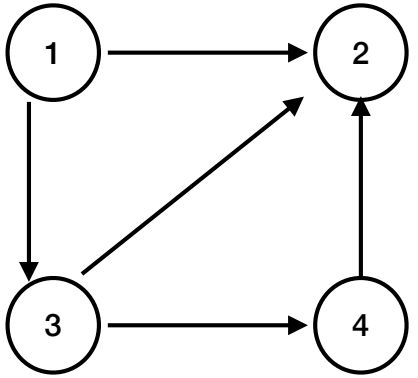
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DAG: Directed Acyclic Graph

(Grafo Diretto Acicliclo)



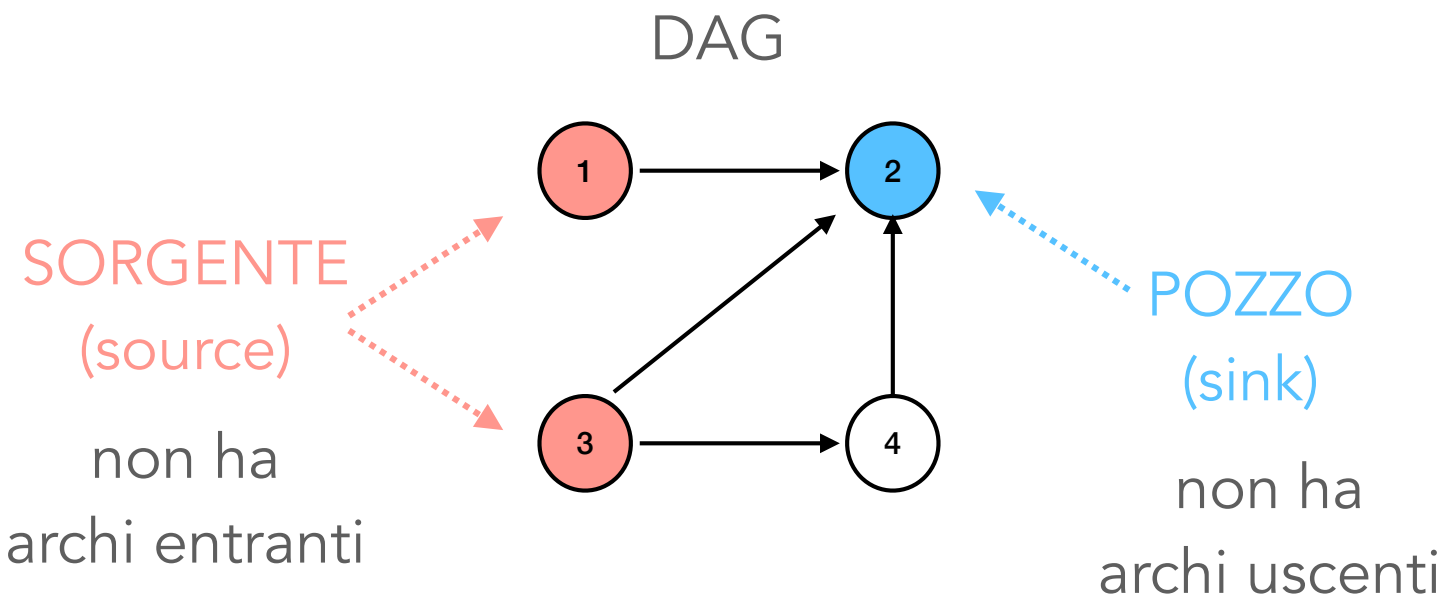
NO



SI

DAG: Directed Acyclic Graph

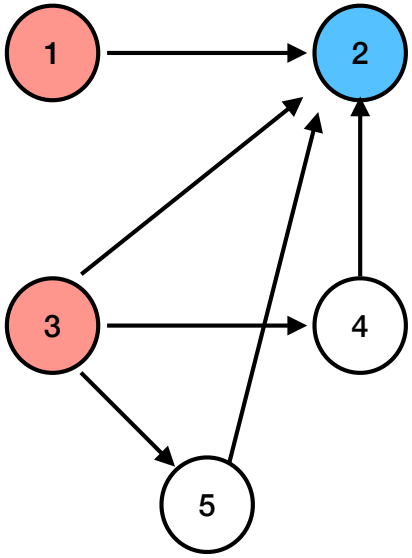
(Grafo Diretto Acicliclo)

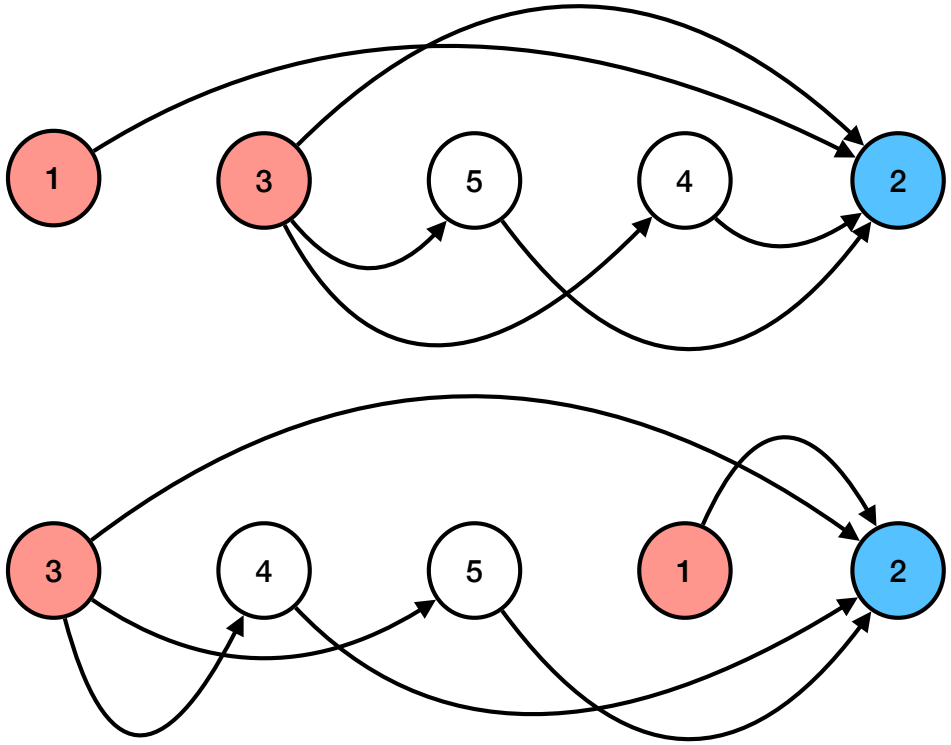


DAG: ordinamento topologico (linearizzazione)

ORDINAMENTO TOPOLOGICO:

Dato un DAG `G=(V,E)`, un ordinamento topologico è un ordinamento lineare dei suoi nodi tali che, se `(u,v)` ∈ `E`, allora `u` viene prima di `v` nell'ordinamento.

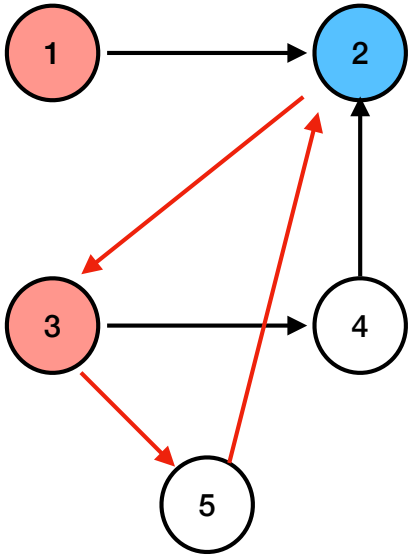


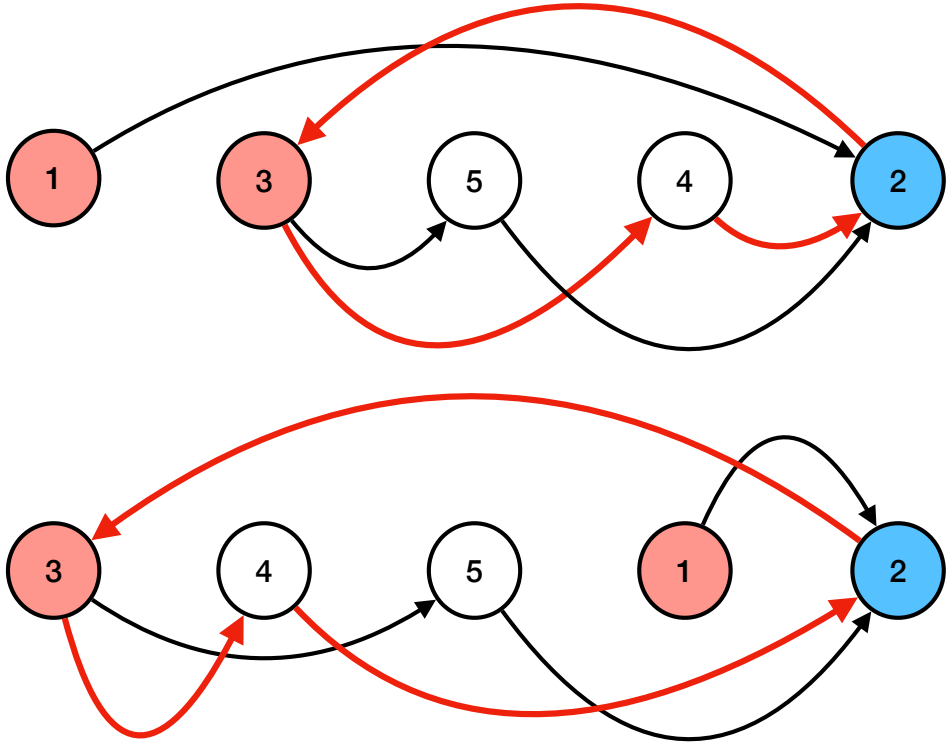


Gli archi sono sempre diretti **da sinistra verso destra**

DAG: ordinamento topologico (linearizzazione)

Se il grafo non è un DAG, ma ha un ciclo,
non è possibile trovare un ordinamento topologico





Ci sara' sempre un arco che "torna indietro"

DAG: ordinamento topologico (linearizzazione)

ORDINAMENTO TOPOLOGICO:

Dato un DAG `G=(V,E)`, un ordinamento topologico è un ordinamento lineare dei suoi nodi tali che, se `(u,v)` $\in$ `E`, allora `u` viene prima di `v` nell'ordinamento.

PERCHÉ?

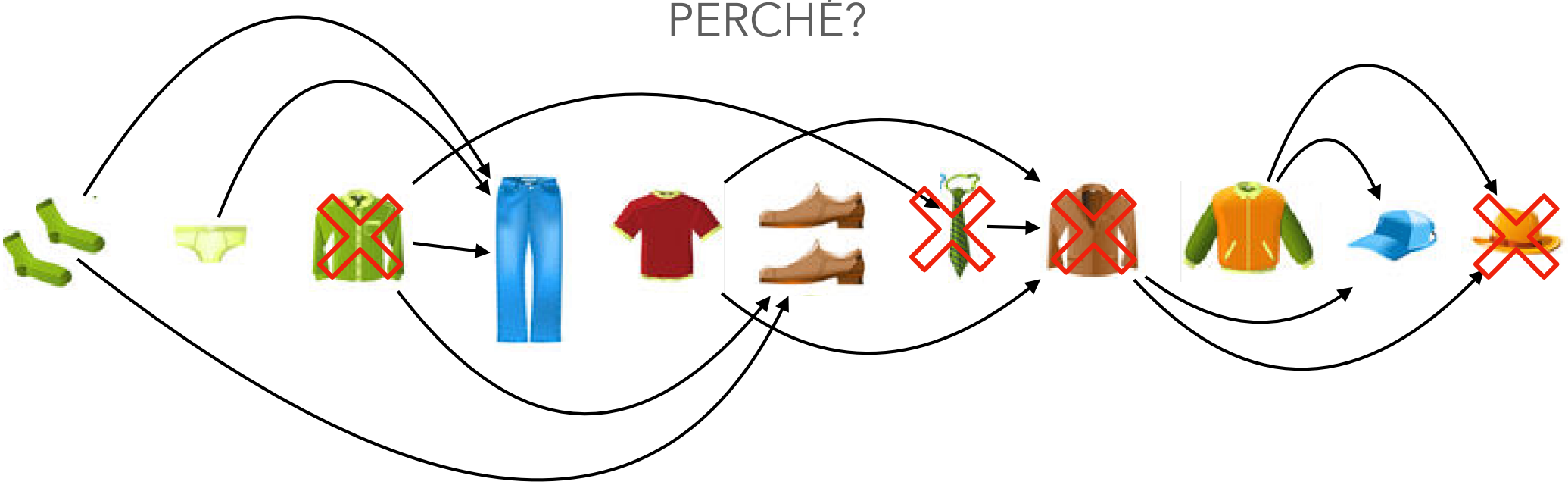
Un ordinamento topologico ci dice in che ordine indossare gli indumenti

DAG: ordinamento topologico (linearizzazione)

ORDINAMENTO TOPOLOGICO:

Dato un DAG `G=(V,E)`, un ordinamento topologico è un ordinamento lineare dei suoi nodi tali che, se `(u,v)` $\in$ `E`, allora `u` viene prima di `v` nell'ordinamento.

PERCHÉ?



DAG: ordinamento topologico (linearizzazione)

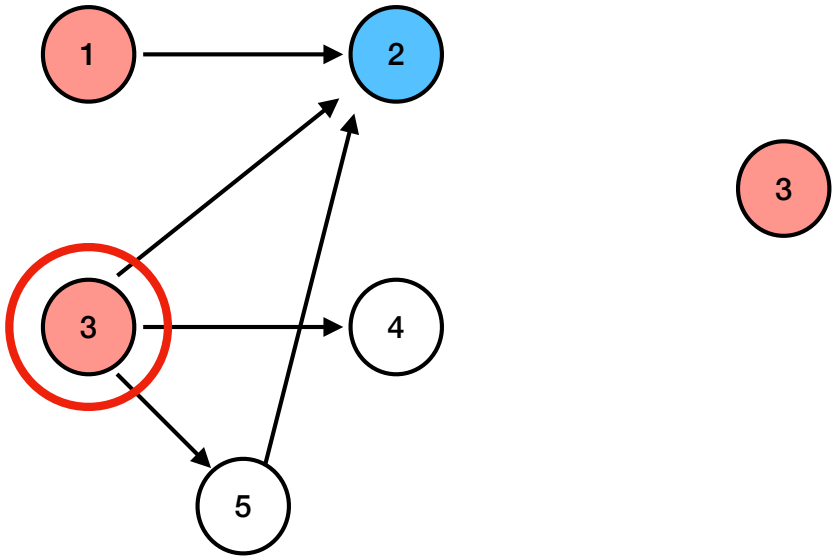
ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

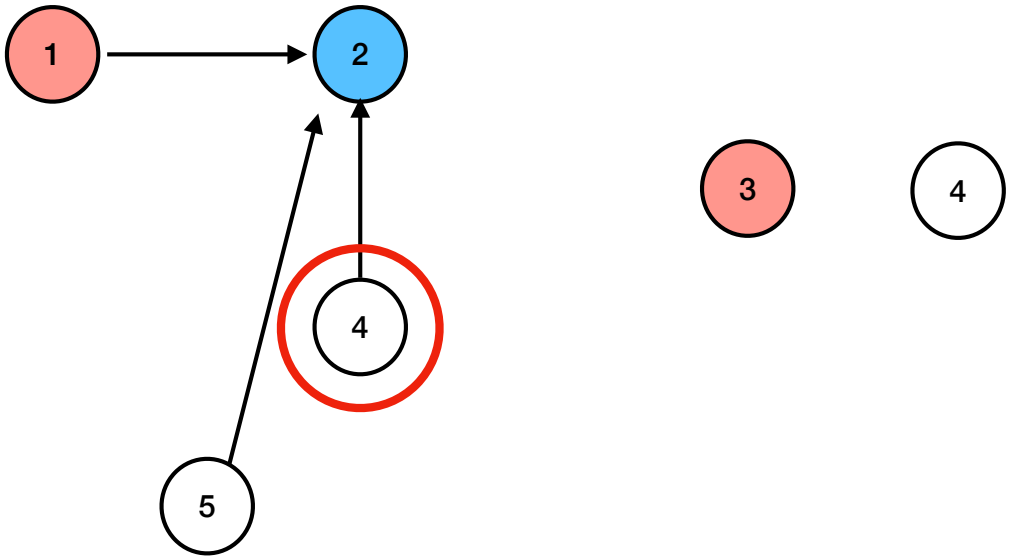
- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo



DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

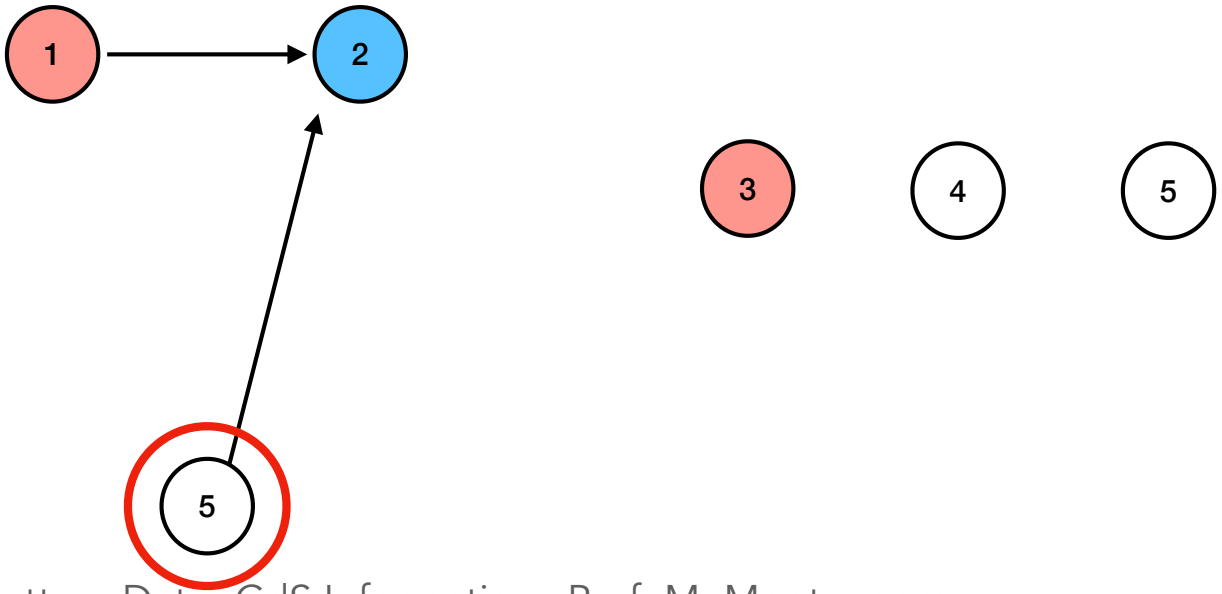
- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo



DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

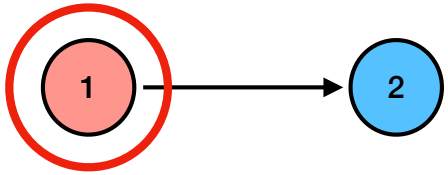
- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo



DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

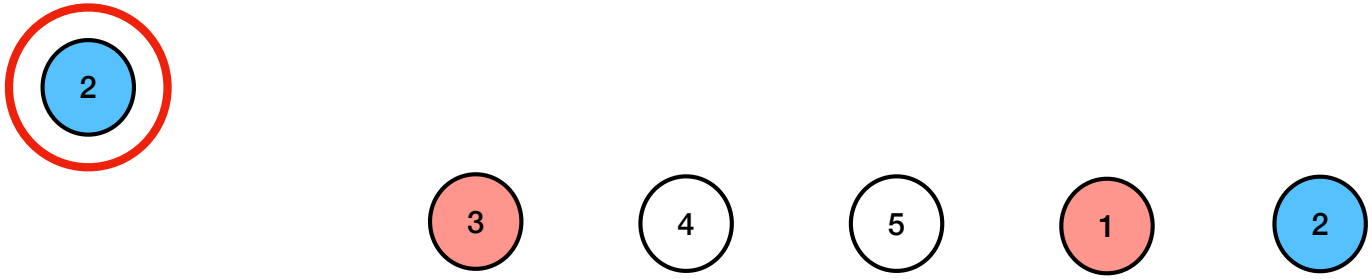
- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo



DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

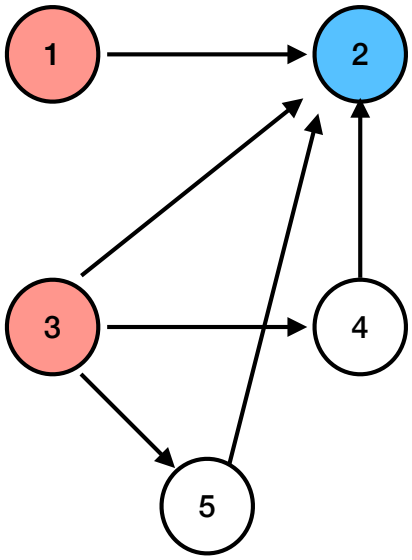
- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo

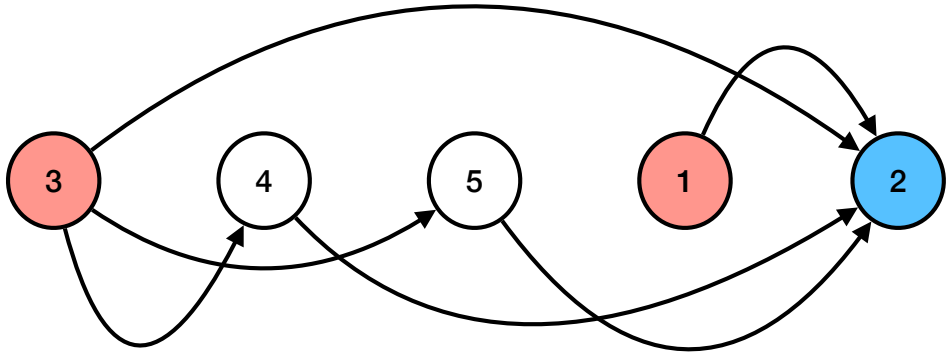


DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo





DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo

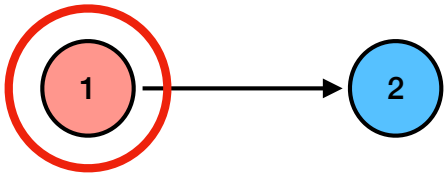
PERCHÉ FUNZIONA?

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo

PERCHÉ FUNZIONA?

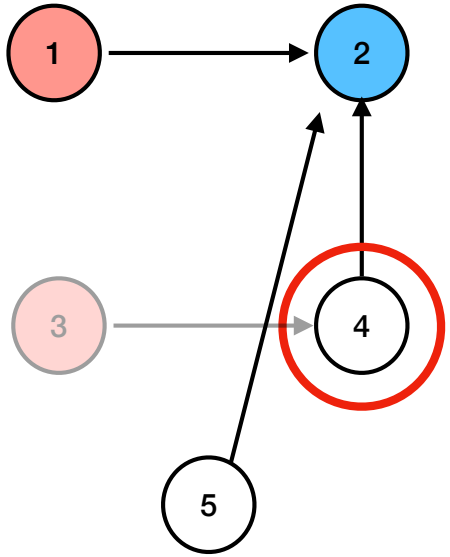


- Un nodo sorgente c'è sempre e non ha archi entranti
- I nodi raggiunti dagli archi uscenti da una sorgente non sono ancora stati ordinati

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo



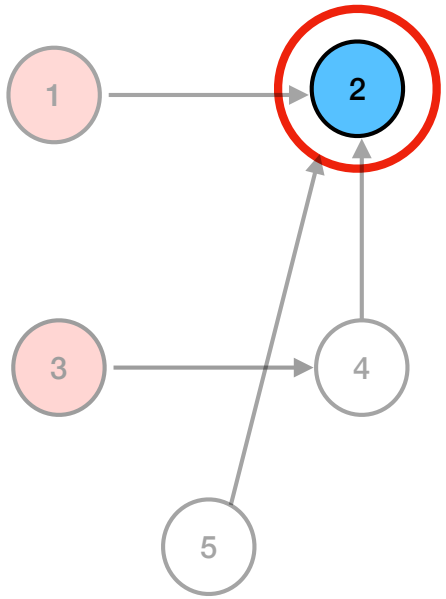
PERCHÉ FUNZIONA?

- Un nodo non sorgente **v** viene scelto quando non ha più archi entranti
- I nodi con un arco uscente verso **v** sono già stati ordinati

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo



PERCHÉ FUNZIONA?

- Un nodo pozzo non ha archi uscenti
- I nodi con archi incidenti nel pozzo sono già stati ordinati

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo

FUNZIONA? Ovvero esiste sempre una sorgente da selezionare?

- Un DAG ha sempre almeno una sorgente (vedere registrazione pillola sull'argomento)
- All'inizio, e dopo la rimozione di ogni nodo (e tutti gli archi uscenti), il grafo è (ancora) un DAG (sempre un pò più piccolo)

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 1:

- Cercare un nodo senza archi entranti
- Aggiungere il nodo all'ordinamento
- Rimuovere il nodo dal grafo e rimuovere tutti i suoi archi uscenti
- Ripetere questi tre passi fino a quando sono stati rimossi tutti i nodi dal grafo

COSTO COMPUTAZIONALE?

Dipende da come implementiamo il grafo e diversi passi dell'algoritmo

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 2 (basato su una DFS):

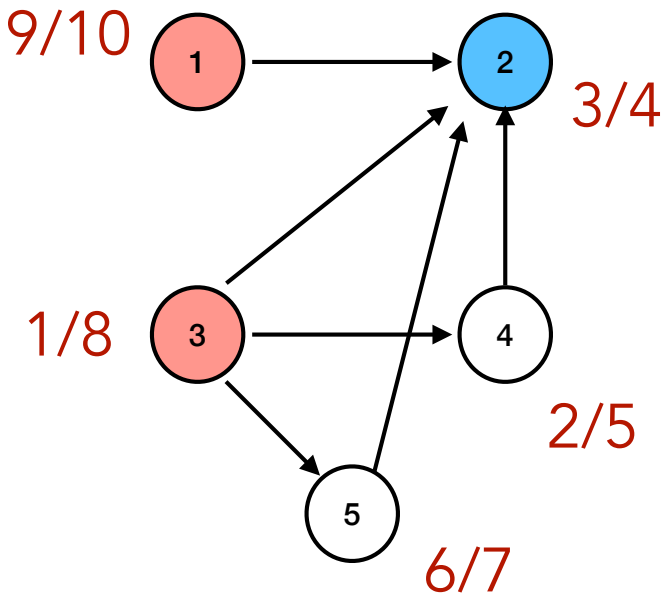
Eseguire una DFS sul grafo e inserire i nodi nell'ordinamento topologico in ordine inverso di valore di `post[]`

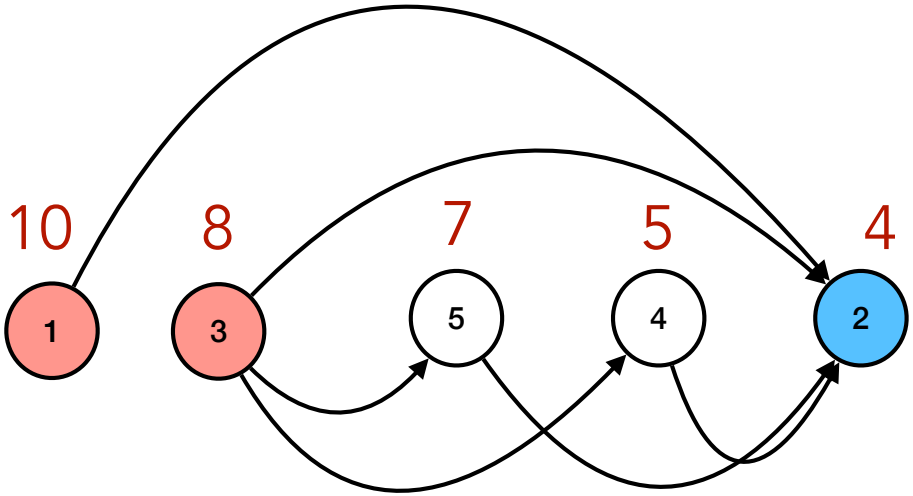
(dal più grande - a sinistra - al più piccolo - a destra)

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 2 (basato su una DFS):

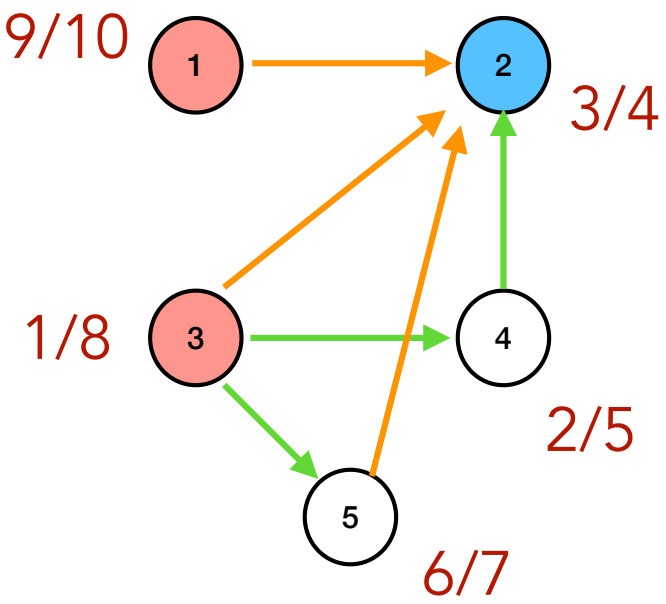
Eseguire una DFS sul grafo e inserire i nodi nell'ordinamento topologico in ordine inverso di valore di `post[]` (dal più grande al più piccolo)

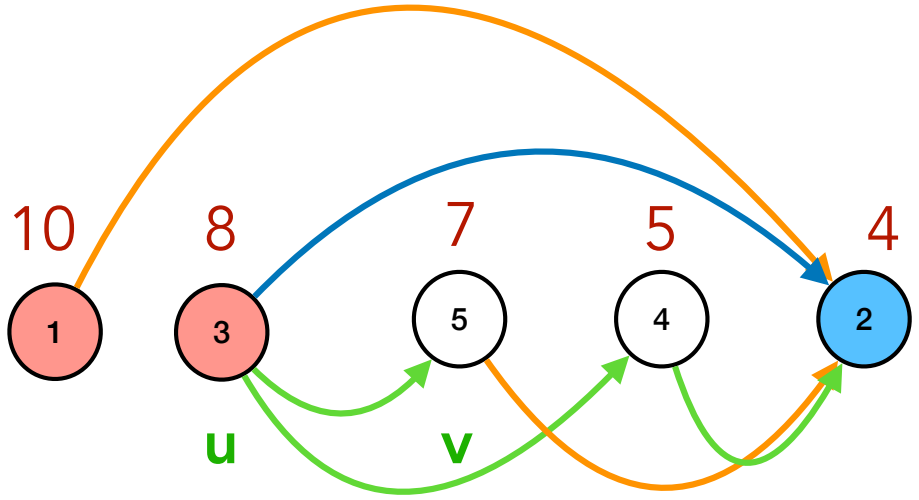




DAG: ordinamento topologico (linearizzazione)

PERCHÉ FUNZIONA?



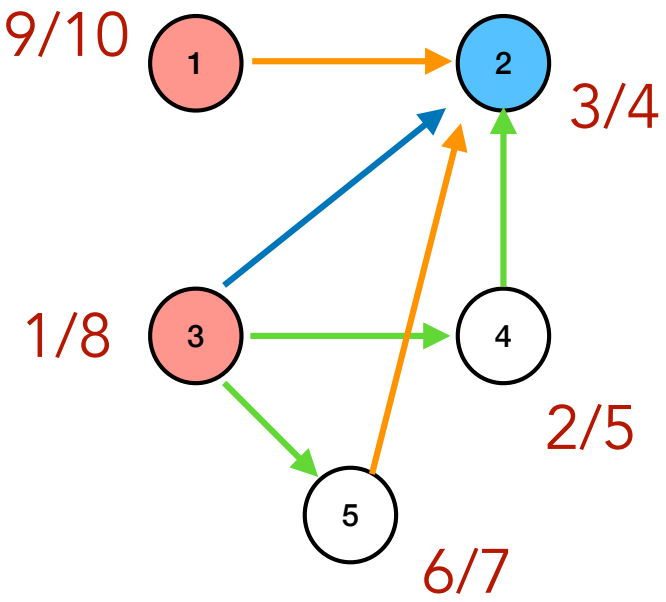


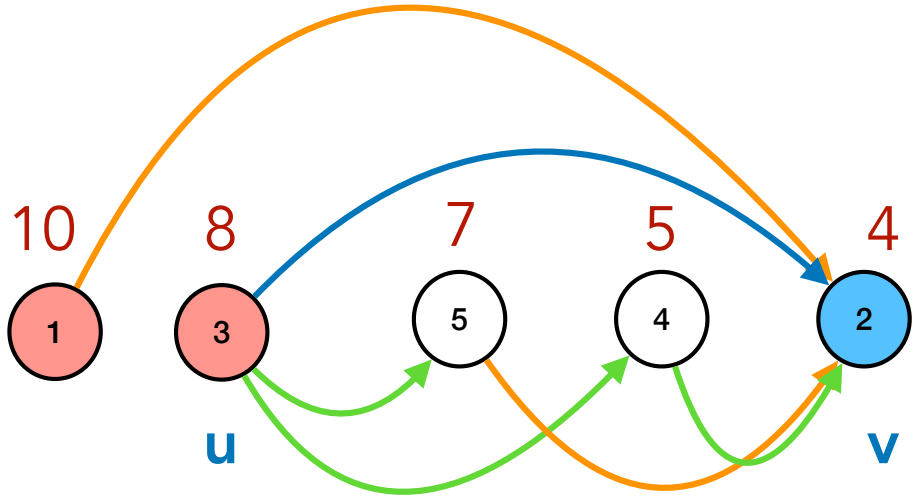
Arco `(u,v)` **TREE**:

`post[u] > post[v]` ⇒ il nodo u sta a sinistra del nodo v

DAG: ordinamento topologico (linearizzazione)

PERCHÉ FUNZIONA?



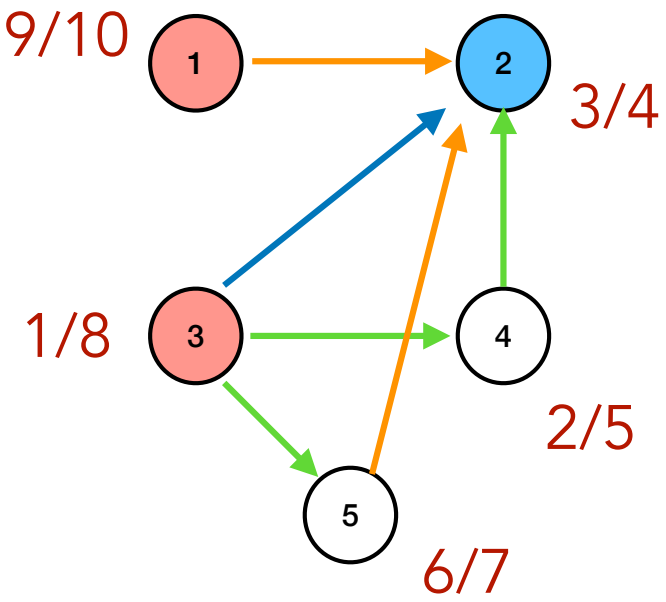


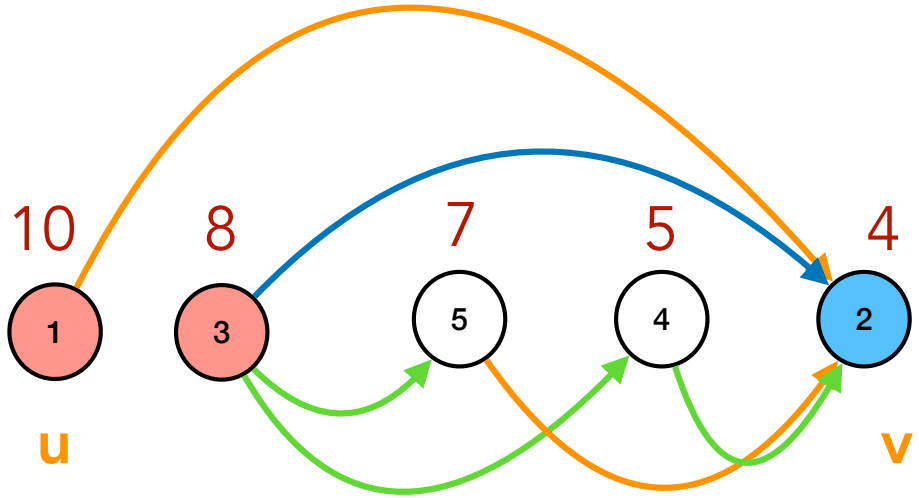
Arco `(u,v)` **FORWARD**:
`post[u] > post[v]`

⇒ il nodo u sta a sinistra del nodo v

DAG: ordinamento topologico (linearizzazione)

PERCHÉ FUNZIONA?





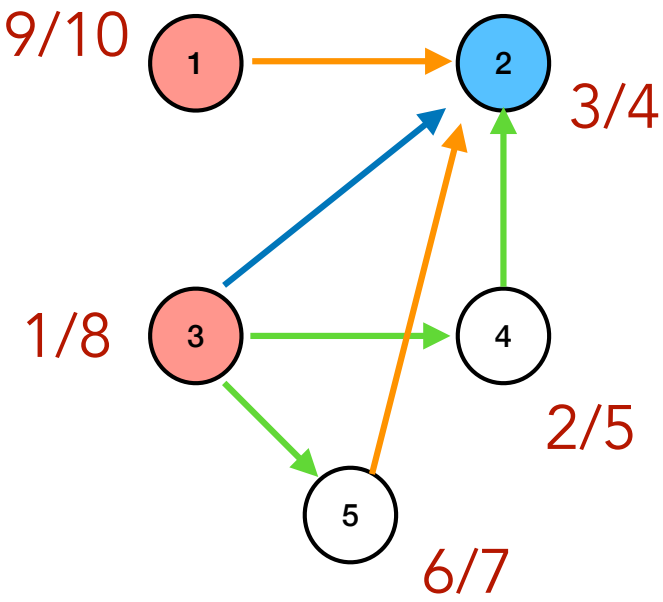
Arco `(u,v)` **CROSS**:

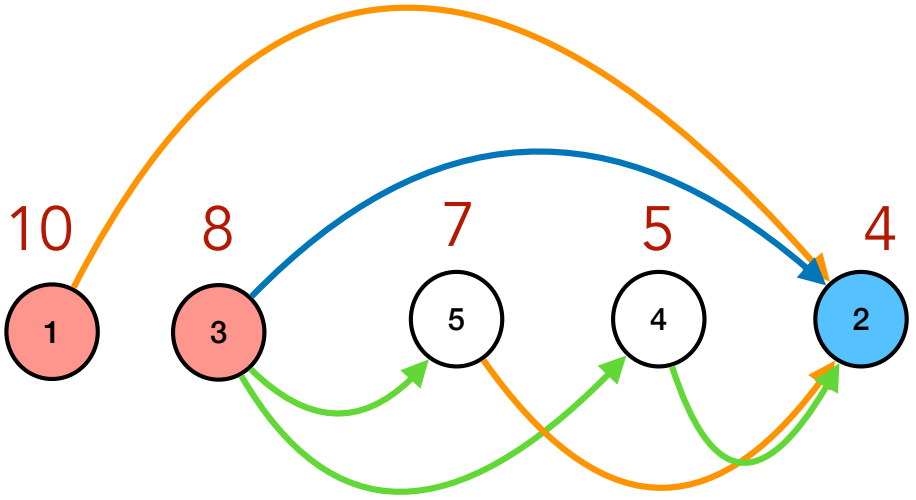
`post[v] < pre[u] < post[u]`

⇒ il nodo u sta a sinistra del nodo v

DAG: ordinamento topologico (linearizzazione)

PERCHE' FUNZIONA?





Arco `(u,v)` **BACK**:
NON CI SONO!

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 2 (basato su una DFS):

Eseguire una DFS sul grafo e inserire i nodi nell'ordinamento topologico in ordine inverso di valore di `post[]` (dal più grande al più piccolo)

COSTO COMPUTAZIONALE?

Si può definire l'ordinamento topologico contemporaneamente ad una visita DFS post-order

Quando termina la visita per il nodo u,
si mette u in una pila (o in testa ad una lista)

$$O(|V| + |E|)$$

DAG: ordinamento topologico (linearizzazione)

ALGORITMO 2 (basato su una DFS):

Funzione principale

```
TopologicalSort(G)
 S := new_stack()
 for all v ∈ V
  visited[v] := FALSE
 for all v ∈ V
  if visited[v] = FALSE
   DFS-TS(G,v)
 return S
```

Procedura ricorsiva

```
DFS-TS(G,v)
 visited[v] := TRUE
 for all (v,u) ∈ E
 // archi uscenti da v
  if visited[u] = FALSE
   then DFS-TS(G,u)
 push(S,v)
```

Costo computazionale $O(|V|+|E|)$