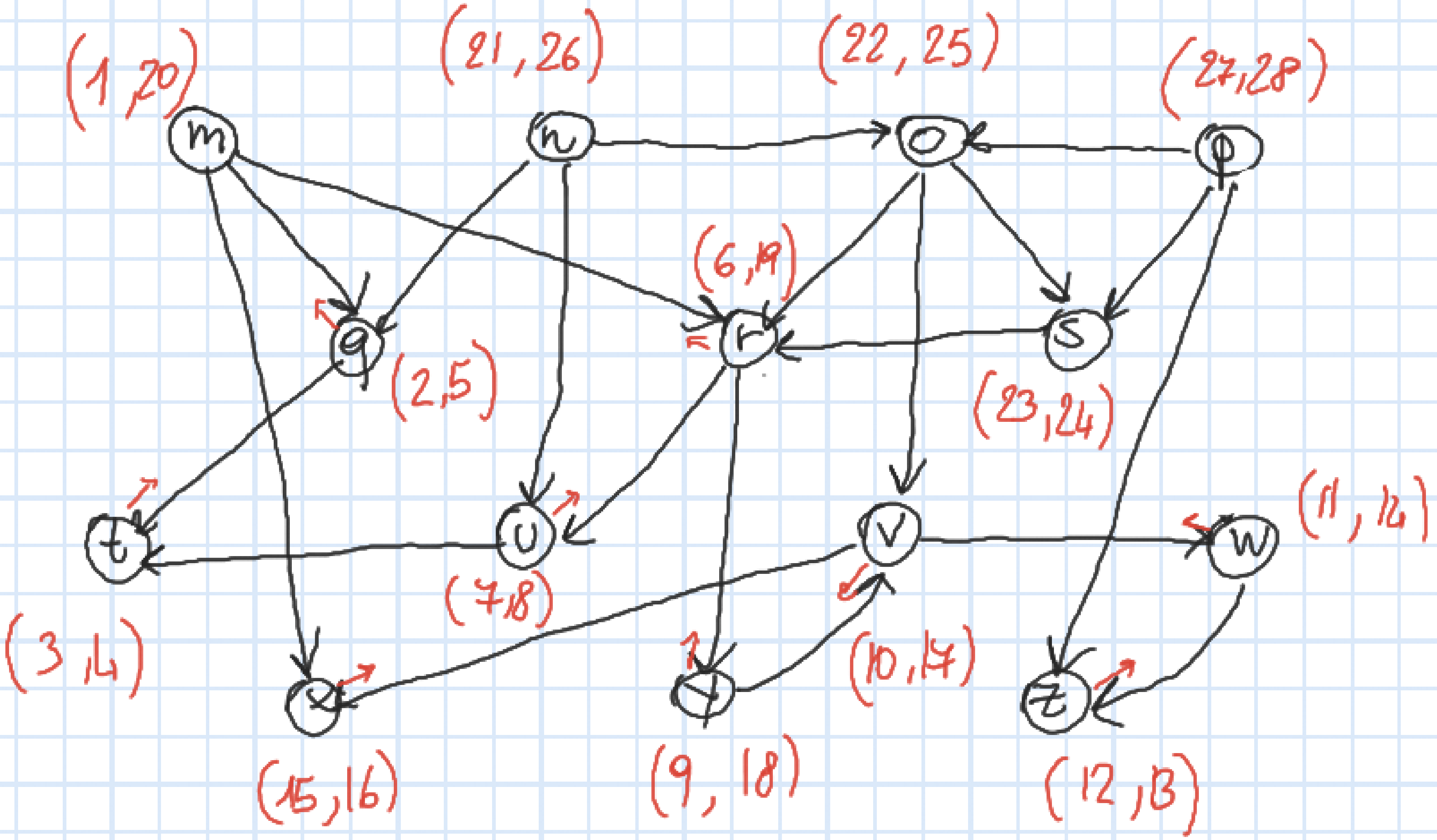
22.4.2. Descrivete un algoritmo che in tempo lineare riceve come input un DAG $G=(V,E)$ e due vertici s,t e restituisce il numero di cammini semplici da s a t.



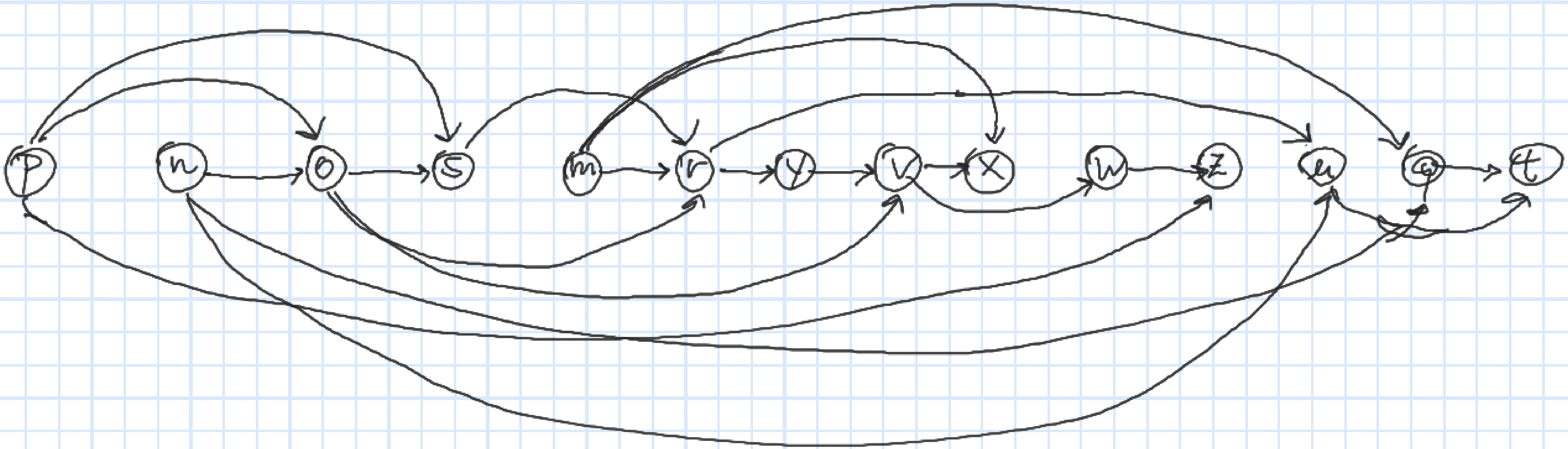
Dati ad esempio

p e v ci sono 4

$\downarrow$ s $\downarrow$ t

cammini:

p o v
p o r y v
p o s r y v
p s r y v



<u>DEF</u> Un cammino semplice è una sequenza di vertici non ripetuti

<u>CLAIM</u>: Il numero di cammini semplici distinti tra due nodi u e v è pari al numero di cammini semplici tra Adj(u) e v (cioè al # di cammini semplici tra i vicini di u e v)

[+1 se u e v sono connessi]

Il numero di cammini semplici da p a v sono:

p o v
p o r y v
p o s r y v
p s r y v

$N_{paths}(o \to v) = 3$

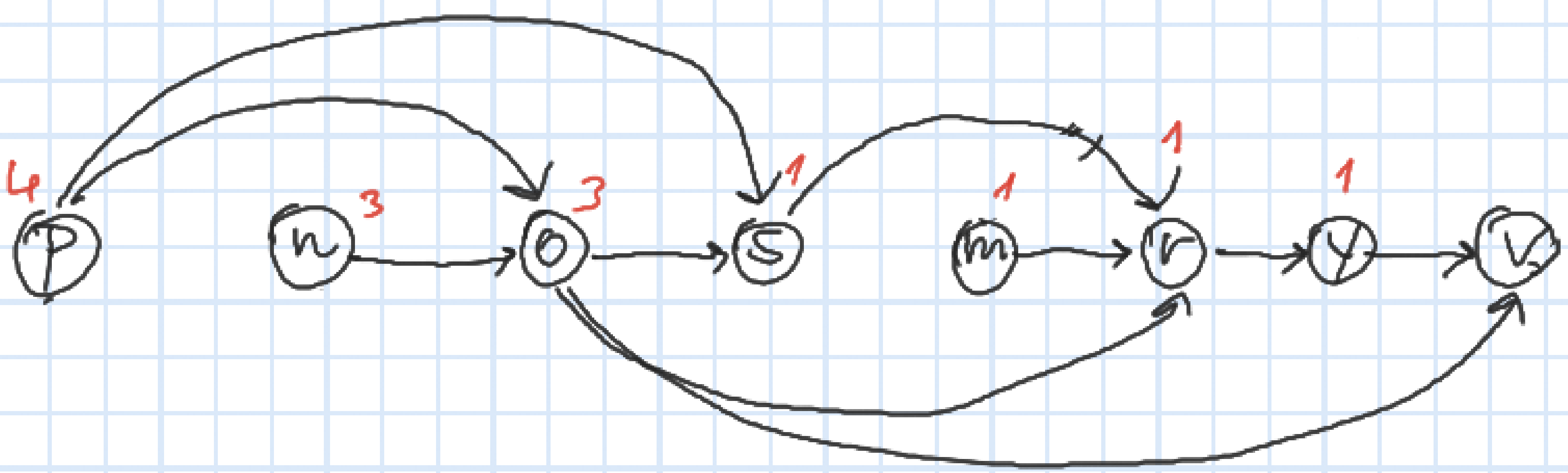
$\to 4$

$N_{paths}(s \to v) = 1$

DEF Un cammino semplice è una sequenza di vertici non ripetuti

CLAIM: Il numero di cammini semplici distinti tra due nodi u e v è pari al numero di cammini semplici tra Adj(u) e v (cioè al # di cammini semplici tra i vicini di u e v)

[+1 se u e v sono connessi]



1) Il numero di modi di arrivare da y a v è 1

2) Il numero di modi di arrivare da r a v è 1

3) Il numero di modi di arrivare da m a v è 1

4) Per s vale lo stesso di m

5) Il numero di modi di arrivare da o a v è il numero di modi di arrivare a v dai vicini di o: $\{s, r, v\}$ = 1 + 1 + 1

6) Il numero di modi per arrivare da n a v è 3

7) Il numero di modi per arrivare da p a v è 3 + 1 = 4

Dove sto sfruttando il fatto di essere su un DAG → ordinabile topologicamente?

Processando i nodi in ordine topologico inverso (cioè dalla fine) sono sicuro di aver esplorato tutto il vicinato di V. Cioè tutti i discendenti diretti di ~~u~~ sono stati considerati ⇒ che il claim di prima ci fornisce il # esatto di simple paths.

IN GENERALE, LA PROCEDURA FUNZIONA COSÌ:

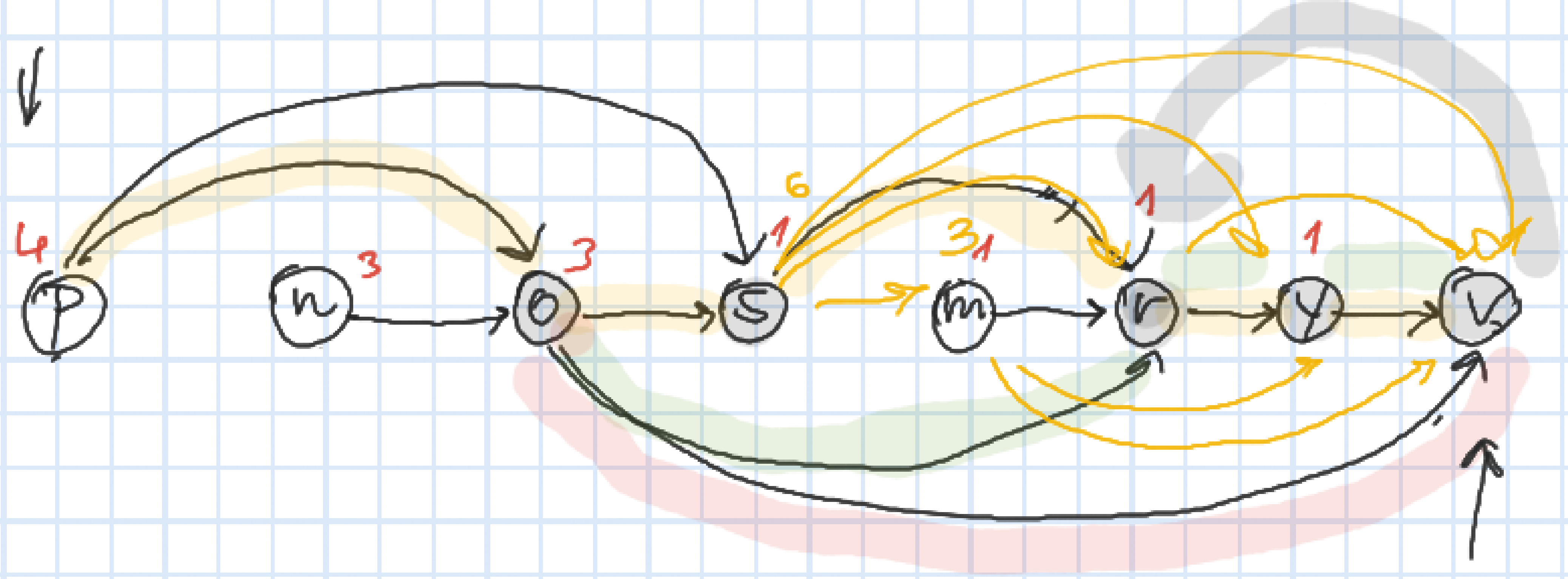
- Dato un ordinamento topologico di n vertici:
  $v(0), v(1), \ldots, v(i), \ldots, v(n)$

- Considera un vettore C di lunghezza n le cui entries conterranno i cammini da $V[i]$ a t.

- Sia i^* l'indice di t nell'ordine topologico, si calcola il contenuto di C andando all'indietro.

  - $C[i] = 0$ per $i > i^*$
  - $C[i^*] = 1$ per $i = i^*$
  - $C[i] = C[i(1)] + C[i(2)] + \ldots + C[i(t)]$ per $i < i^*$

  $$\underbrace{\text{dove } i(1) \ldots i(t) \text{ sono gli indici dei vertici raggiungibili da } v(i) \text{ in un passo.}}_{\text{cioè somma i label rossi}}$$



MASSIMO DI LINK SU UN DAG: $\frac{n^2-n}{2} = \frac{n(n-1)}{2}$

MASSIMO DI PATH SU UN DAG COMPLETO è assumendo che s e t siano in tutti i path 2^{n-2}

Se applicassimo DFS su un DAG completo ammettendo l'analisi di nodi "black" potremmo arrivare a chiamare la procedura un numero esponenziale di volte