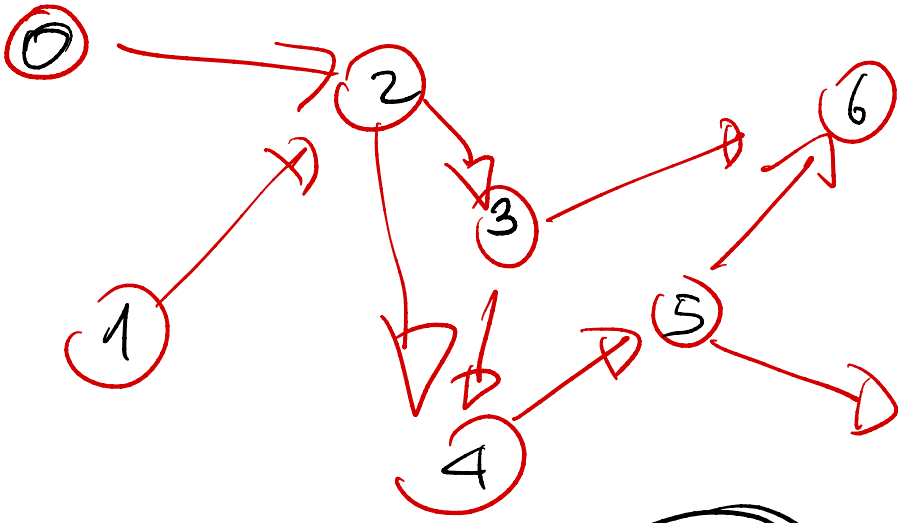
ORDINAMENTO TOPOLOGICO → DAG (Directed Acyclic Graph)

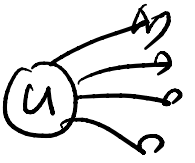


DAG ∃ sempre nodo con grado di uscita zero

```
1  OrdinamentoTopologico( ):
2    FOR (s = 0; s < n; s = s + 1)
3      raggiunto[s] = FALSE;
4    contatore = n - 1;
5    FOR (s = 0; s < n; s = s + 1) {
6      IF (!raggiunto[s]) DepthFirstSearchRicorsivaOrdina( s );
7    }
```

VISITA DFS

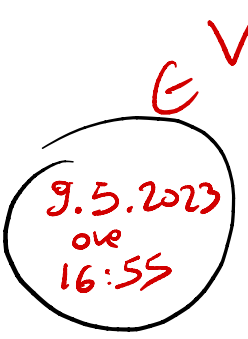
```
1  DepthFirstSearchRicorsivaOrdina( u ):
2    raggiunto[u] = TRUE;
3    FOR (x = listaAdiacenza[u].inizio; x != null; x = x.succ) {
4      v = x.dato;
5      IF (!raggiunto[v]) DepthFirstSearchRicorsivaOrdina( v );
6    }
7    eta[u] = contatore;
8    contatore = contatore - 1;
```



PROGETTO: GRAFO (DAG) PER BLOCKCHAIN/BITCOIN
(versione semplificata)

$G = (V, E, W)$

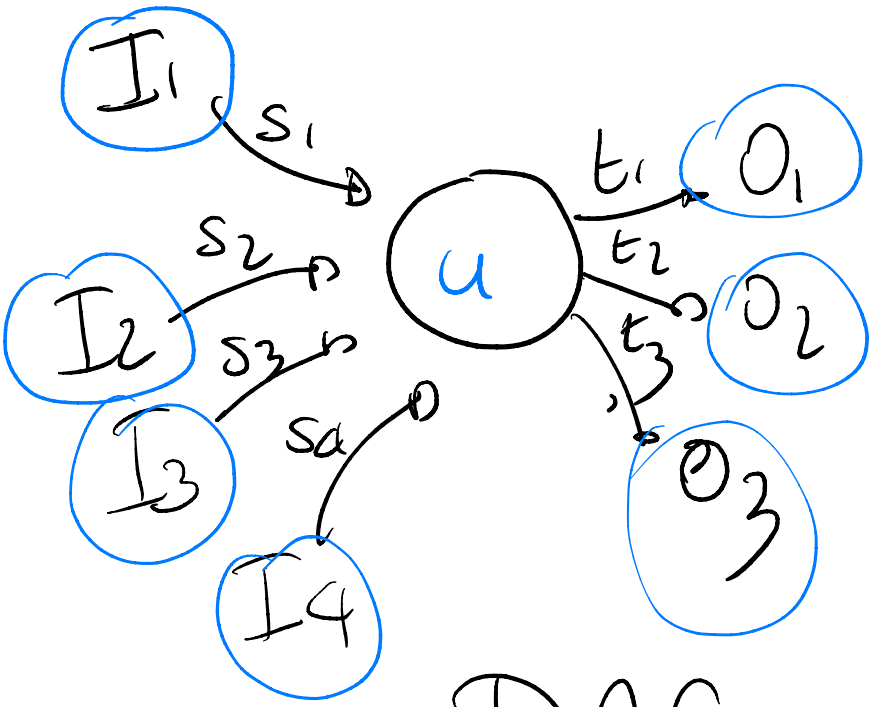
V = transazioni con timestamp



E = spostamenti di denaro

$W : E \longrightarrow R$: ammontare spostato

Esempio



DAG

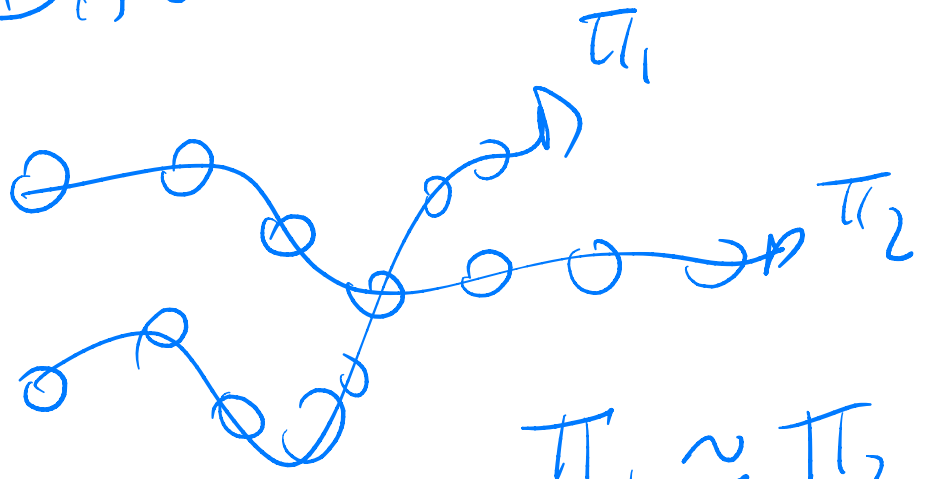
$I_1, I_2, I_3, I_4, O_1, O_2, O_3 \in V$

$s_1 + s_2 + s_3 + s_4$

$\parallel$

$t_1 + t_2 + t_3$

DAG



$\pi_1 \simeq \pi_2$

↑

Similitudine

- quantità di denaro
- tempistica (timestamp)