Esercizio

Data una lista di n interi A, un'inversione di A è una coppia $(A[i], A[j])$ con $0 \le i < j \le n$ per cui $A[i] > A[j]$

Es. $A = (1, 3, 4, 2, 5, 3)$

Progettare un algoritmo che data la lista A, in tempo $O(n \log n)$ determini il # di inversioni in A.

CASO INTUITIVO: Data una lista di lunghezza n dovremmo esamire
potremmo esaminare tutte le $\binom{n}{2}$ coppie

$\rightarrow$ Complessità $O(n^2)$

Un possibile algoritmo:

1) Separa la lista in 2 metà X e Y

2) Conta ricorsivamente le inversioni in ognuna delle 2 metà

3) Conta le inversioni tra le due metà cioè con $x \in X, y \in Y$

4) Restituisci la somma dei 3 numeri

Esempio: $A = [1, 5, 4, 8, 10, 2, 6, 9, 3, 7]$

1)

X $[1, 5, 4, 8, 10]$ $[2, 6, 9, 3, 7]$ Y

Se confrontiamo le due metà membro a membro avremmo

$\frac{n}{2} \times \frac{n}{2}$ confronti $\rightarrow O(n^2)$

ANCORA NON VA BENE

Esempio: $A = [1, 5, 4, 8, 10, 2, 6, 9, 3, 7]$

1)

X $[1, 5, 4; 8, 10]$ $\qquad$ $[2, 6, 9, 3, 7]$ Y

Se confrontiamo le due metà membro a membro avremmo

$\frac{n}{2} \times \frac{n}{2}$ confronti $\rightarrow O(n^2)$

ANCORA NON
VA BENE

Esempio: $A = [1, 5, 4, 8, 10, 2, 6, 9, 3, 7]$

Supponiamo di saper fare il passo 2) e che X e Y siano ordinati.

$$X = [1, 4, 5, 8, 10] \qquad [2, 3, 6, 7, 9] = Y$$

- Se $x_i < y_j$ allora x_i non è invertito con nessun elemento a destra di y_j → incrementiamo i
- Se $x_i > y_j$ allora y_j è invertito con ogni elemento alla destra di x_i. → incrementiamo j

Ho $n+1-i$ inversioni

Esempio: $A = [1, 5, 4, 8, 10 \,|\, 2, 6, 9, 3, 7]$

Supponiamo di saper fare il passo 2) e che X e Y siano ordinati.

$$X = [1, 4, 5, 8, 10] \qquad [2, 3, 6, 7, 9] = Y$$

$x_1 < y_1 \to 0$ inv.

$x_2 > y_1 \to 4$ inv.

$x_2 > y_2 \to 4$ inv.

$x_2 < y_3 \to 0$ inv.

$x_3 < y_3 \to 0$ inv.

$x_4 > y_3 \to 2$ inv.

$x_4 > y_4 \to 2$ inv.

$x_4 < y_5 \to 0$ inv.

$x_5 > y_5 \to 1$

$\boxed{13}$

Contare le inversioni costa $O(n)$

Fare merge di questi due array costa $O(n)$

Esempio: $A = [1, 5, 4, 8, 10, 2, 6, 9, 3, 7]$

Supponiamo di saper fare il passo 2) e che X e Y siano ordinati.

$X = [1, 4, 5, 8, 10] \qquad [2, 3, 6, 7, 9] = Y$

Contare le inversioni costa $O(n)$

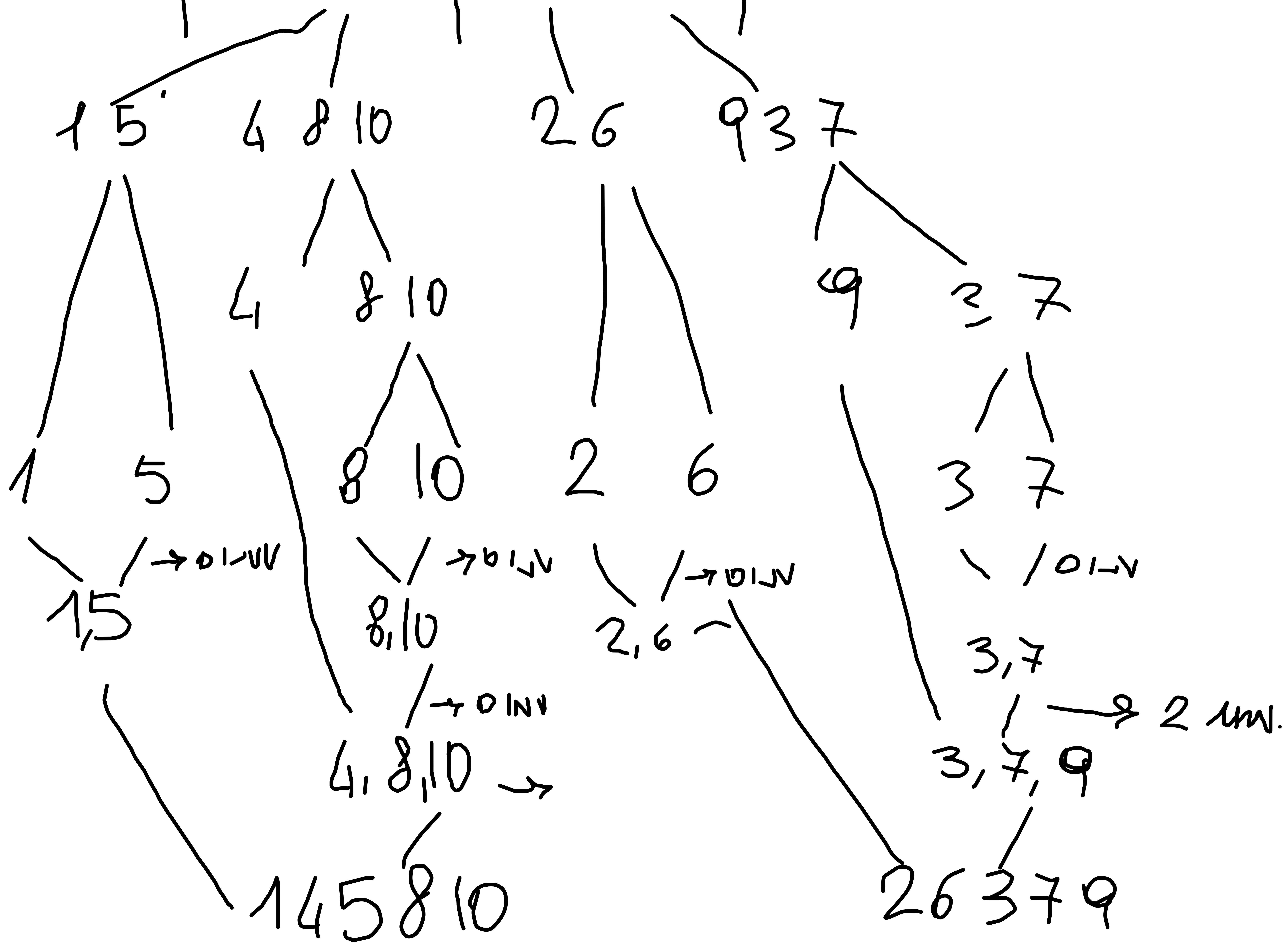
Fare merge di questi due array costa $O(n)$

La complessità si può calcolare con il master theorem

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

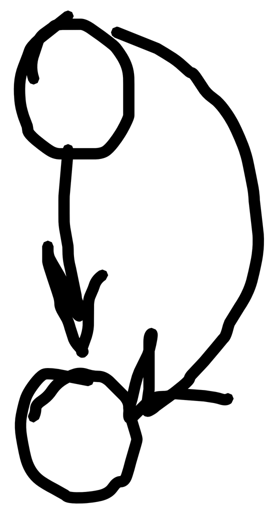
$$\rightarrow T(n) = O(n \log n)$$

Esempio: $A = [1,5,4,8,10 \mid 2,6,9,3,7]$

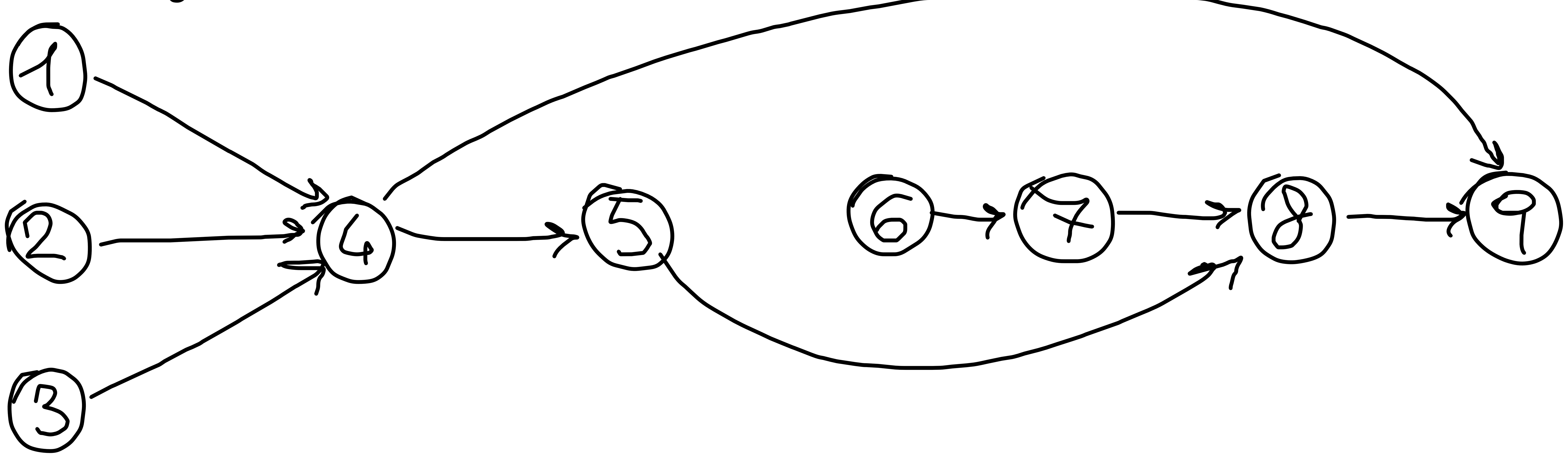


Calcolare il tempo di esecuzione di:
ORECCHIETTE ALLE CIME DI RAPA

OPERAZIONE	TEMPO	PRECEDENZE
1) scaldare aglio	1	—
2) schiacciare acciughe	2	—
3) tagliare peperoncino	1	—
4) preparare soffritto	2	1, 2, 3
5) pulire le rape, conservare cime	20	—
6) scaldare acqua	10	—
7) avviare cottura orecchiette	11	6
8) aggiungere cime e terminare cottura	7	5, 7
9) condire le orecchiette	1	4, 8



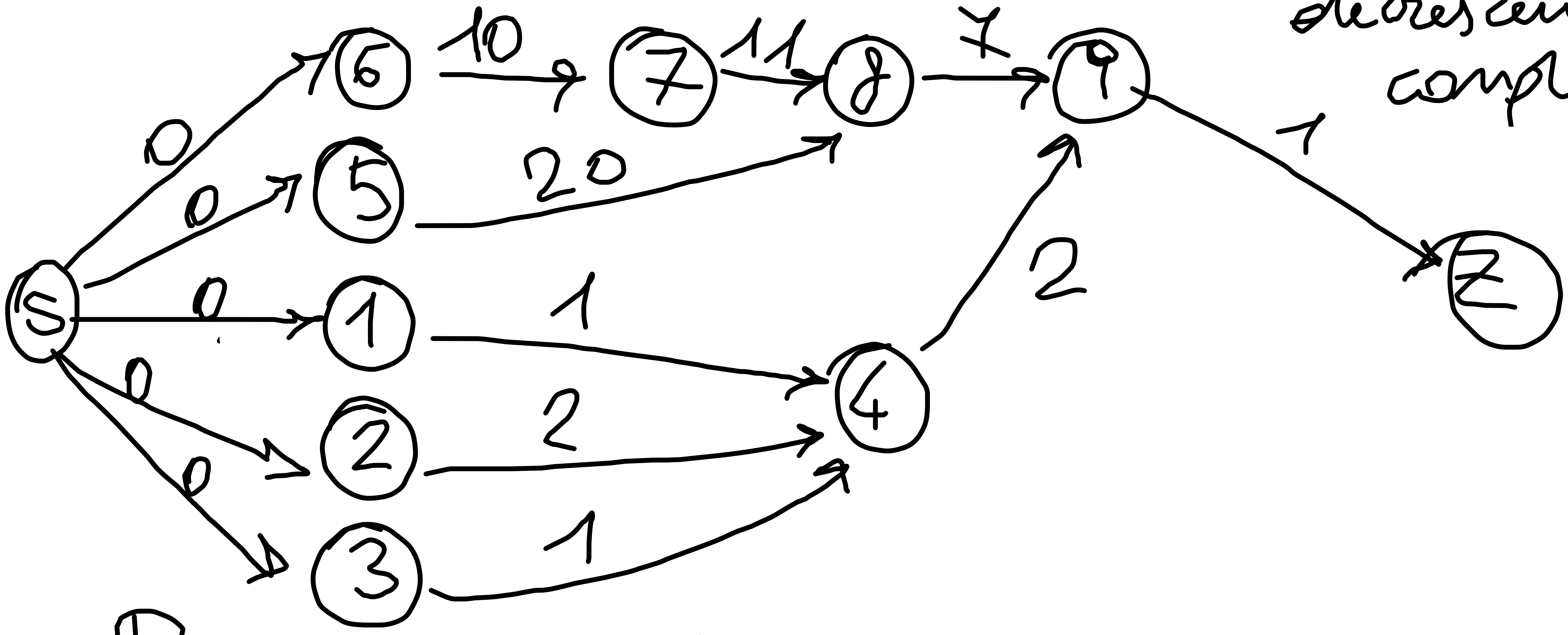
Il grafo delle attività



Intuitivamente il cammino + lungo mi darà il tempo di completamento.

Calcolare il cammino + lungo su un grafo non è fattibile, a meno che sia un DAG.

Il nostro grafo è un DAG (faccio DFS, ordino in maniera decrescente rispetto ai tempi di completamento)

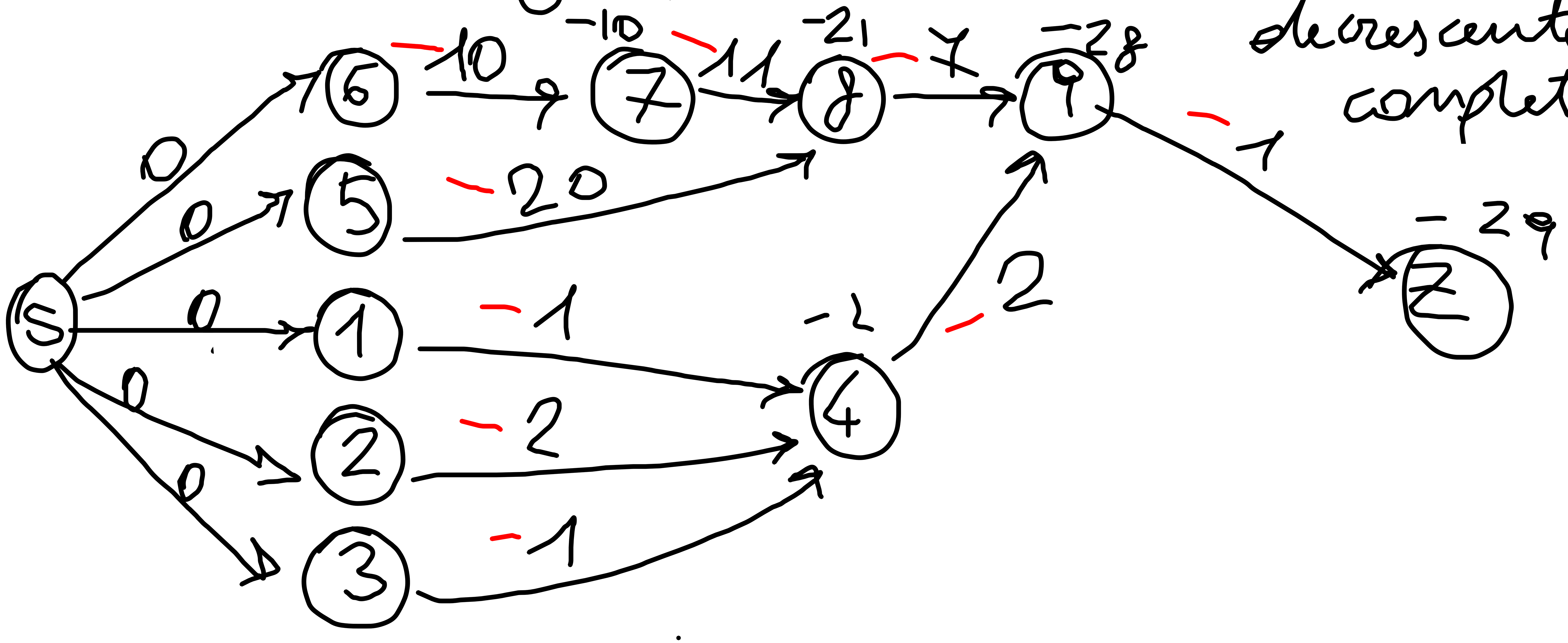


Per inserire i pesi degli archi su questo dag dobbiamo inserire 2 nodi pozzo e sorgente e $w_{ij} = t_i$

→ connessa a tutti i nodi con degree-in $= 0$

→ connessa a tutti i nodi con dout$= 0$

Il nostro grafo è un DAG (faccio DFS, ordino in maniera decrescente rispetto ai tempi di completamento)



Per trovare il DAG-SHORTEST PATH inverto i pesi degli archi.
E applico il rilassamento

$v.d > u.d + w(u,v)$

Sappiamo che il tempo per eseguire le operazioni è 29.

Non tutte le attività devono essere svolte al + presto.

La differenza tra tempo di inizio minimo t_i e massimo T_i è detto tempo di slack $m_i = T_i - t_i$

$\rightarrow T_i = T_j - w_{ij}$

tempo DAG shortest path

	t_i	T_i	SLACK $= m_i$
0	0	0	0
1	0	25	25
2	0	24	24
3	0	25	25
4	2	26	24
5	0	1	1
6	0	0	0
7	10	10	0
8	21	21	0
9	28	28	0
z	29	0	0

Le attività con tempo di slack = 0 sono dette attività critiche ed il cammino che le contiene si chiama CAMMINO CRITICO.

→ Il metodo del cammino critico (CPM) è un metodo fondamentale del project management