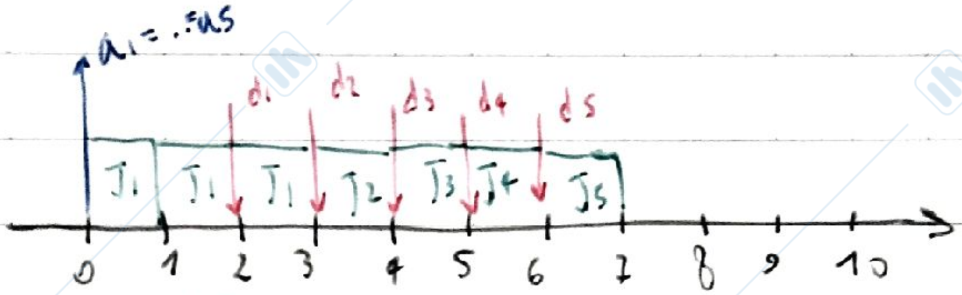
N_{LATE} = # PROCESSI IN RITARDO

L_{MAX} = RITARDO MASSIMA



$C_1 = 3 \quad C_2 = C_3 = \dots C_5 = 1$

PROBLEMA SCHEDULABILE (SE SODDISFA TUTTI I VINCOLI)

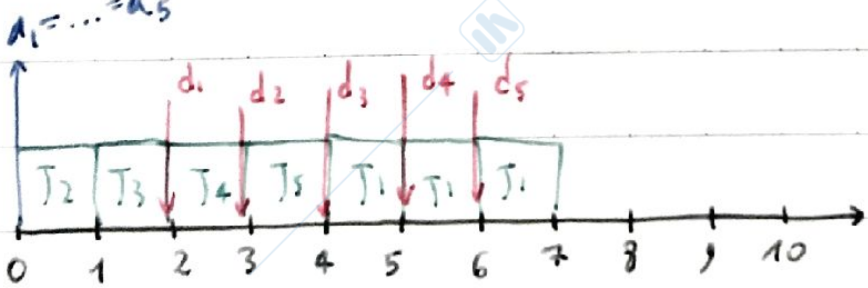
AMMISSIBILE (SOLUZIONE DEL PROBLEMA SCHEDULABILE)

SCHEDULABILITÀ (ANALISI)

1. MIN L_{MAX} → $L_{MAX} = 1$, $N_{LATE} = 5$ (processi in ritardo)

2. MIN N_{LATE} → $N_{LATE} = 1$, $L_{MAX} = 5$ — Qual è meglio?

possiamo fare così



Dipende da come modelliamo il problema, ma in generale è meglio la soluzione 2

Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

Date 25.09.19

PROPRIETÀ
(associate agli algoritmi)

- PREEMPTIVE/NON PREEMPTIVE: Se un processo una volta in esecuzione può essere interrotto e rimesso oppure no.

- STATICI/DINAMICI: Se consideriamo un algoritmo di scheduling, ossia che prende il processo pronto.
Come ordino i processi?
Se si ordinano usando solo parametri al tempo 0 si dice che l'algoritmo è STATICO.
Se l'algoritmo tiene conto della storia nel tempo del processo l'algoritmo si dice DINAMICO.

- ONLINE/OFFLINE: L'input al problema diventa noto nel tempo → ONLINE
In alcuni casi specifici è già dato l'input al tempo 0.
↓
OFFLINE

- OTTIMI / EURISTICI: Ottimi → Garanzia teorica
Euristico → funzionano bene sperimentalmente e non danno una garanzia

SCHEDULING DI PROCESSI APERIODICI

Vuol dire che tutte le attivazioni sono uguali

	NO PRECEDENZE	PRECEDENZE	
ATTIVAZIONE SINCRONA	EDD (Earliest Due Date)	LDF (Latest Deadline First)	
ATTIVAZIONE ASINCRONA	EDF (Earliest Deadline First)	EDF*	PREEMPTIVE
	BRATLEY	SPRING	NON PREEMPTIVE

nell'attivazione sincrona non c'è bisogno della distinzione poiché in una situazione ottima non devo fare PREEMPTION

EDD (processi aperiodici)

PROBLEMA: n PROCESSI SINCRONI, NON CI SONO VINCOLI DI PRECEDENZA, PREEMPTION

ALGORITMO: I PROCESSI VENGONO ESEGUITI IN ORDINE NON DECRESCENTE DI DEADLINE

PROPRIETÀ: OTTIMO PER MINIMIZZARE L_{MAX}

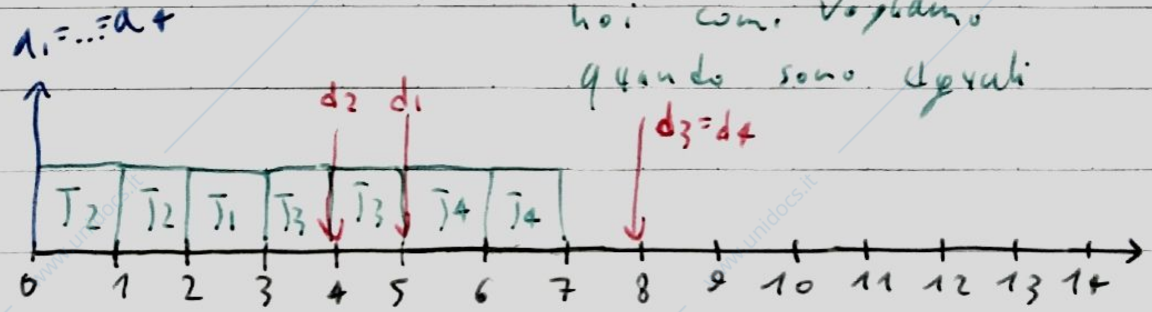
es.

	J_1	J_2	J_3	J_4
a_i	0	0	0	0
c_i	1	2	2	2
d_i	5	4	8	8

→ Ordiniamo i processi:

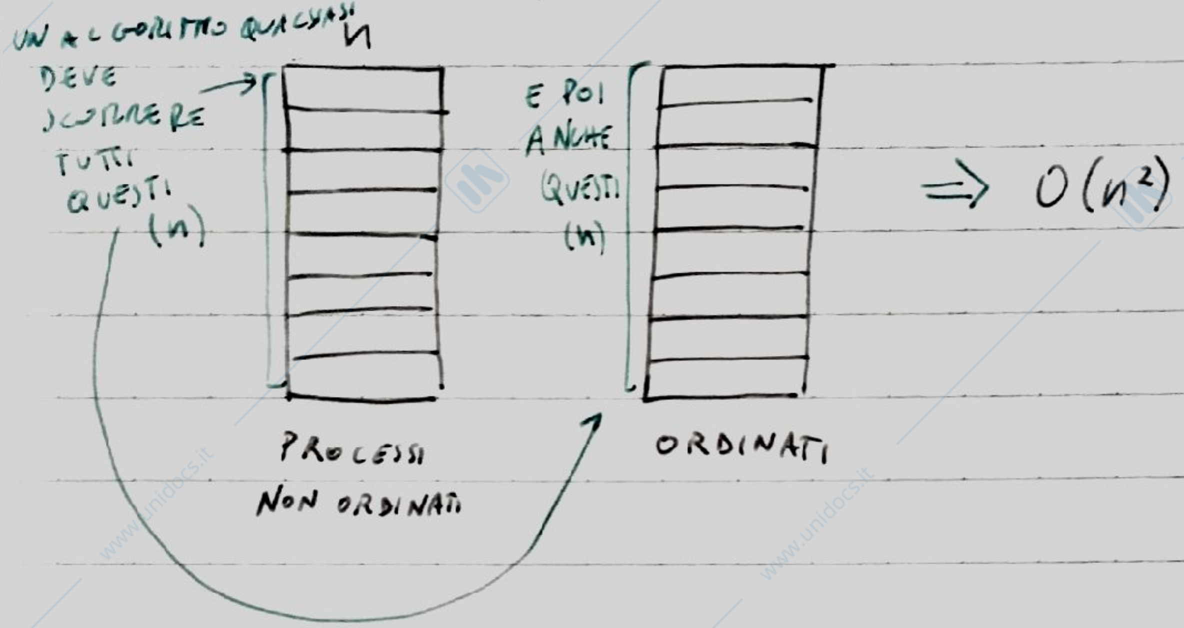
$J_2 > J_1 > J_3 > J_4$

possiamo scegliere noi come vogliamo quando sono uguali



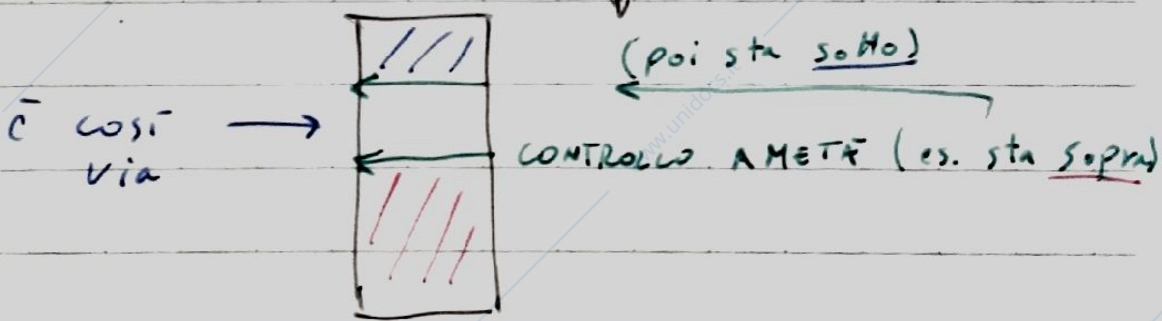
$L_1 = -1 \quad L_2 = -2 \quad L_3 = -4 \quad L_4 = -1 \quad \rightarrow L_{MAX} = -1$

→ Complessità algoritmo

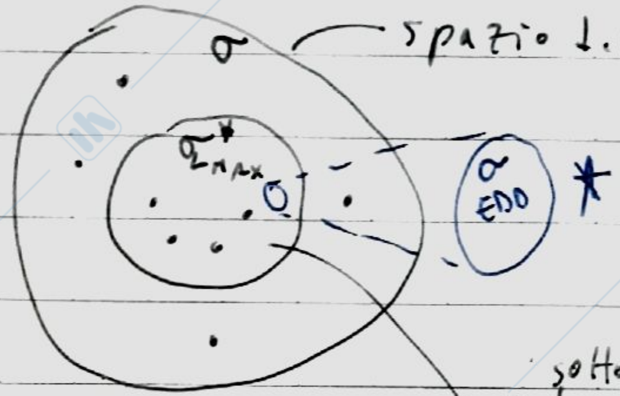


→ Algoritmo efficiente: bisezione

$O(n \lg n)$

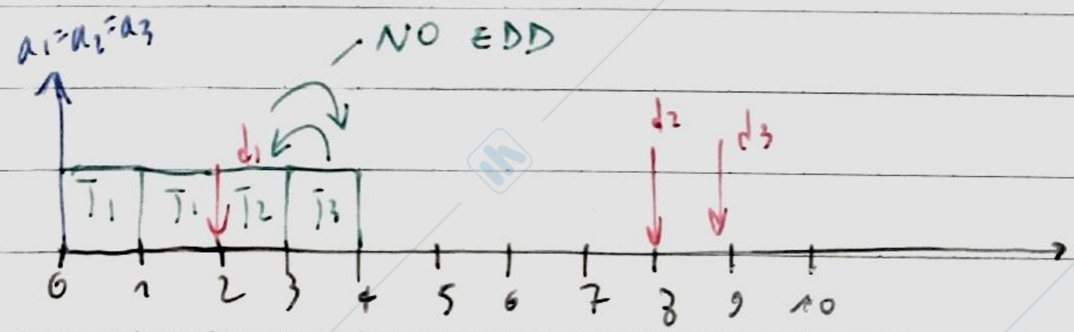


→ Dimostrazione ottimalità per min L_{MAX}



→ Se 2 processi hanno la stessa deadline non importa chi va prima in esecuzione perchè tutte le soluzioni possibili daranno lo stesso L_{MAX}.

→ Esiste una schedulazione in cui L_{MAX} è il minimo possibile ma non è conforme a EDD?



$C_1=2$
$C_2=1$ → $L_{MAX}=0$ (lo otteniamo anche scambiando J2 e J3 perchè L_{MAX} rimane sempre =0.
$C_3=1$

Ma la soluzione $J1 > J3 > J2$ NON è EDD)

Quindi esiste un sotto-sottoinsieme di soluzioni EDD *

→ Dobbiamo ~~dimostrare~~ dimostrare che L_{MAX} di qualsiasi soluzione non EDD deve essere maggiore o uguale di un L_{MAX} delle soluzioni EDD.

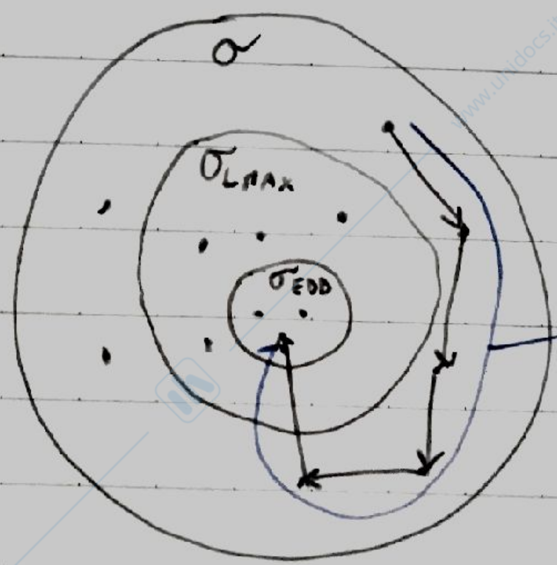
$$\hookrightarrow L_{MAX}(\sigma_{NOT\ EDD}) \geq L_{MAX}(\sigma_{EDD})$$

$$\forall\, \sigma_{NOT\ EDD}$$

$\downarrow$

Tesi:

Qualunque sia la soluzione da cui parto per costruire una sequenza e questa finisce con una soluzione EDD, L_{MAX} NON cresce.



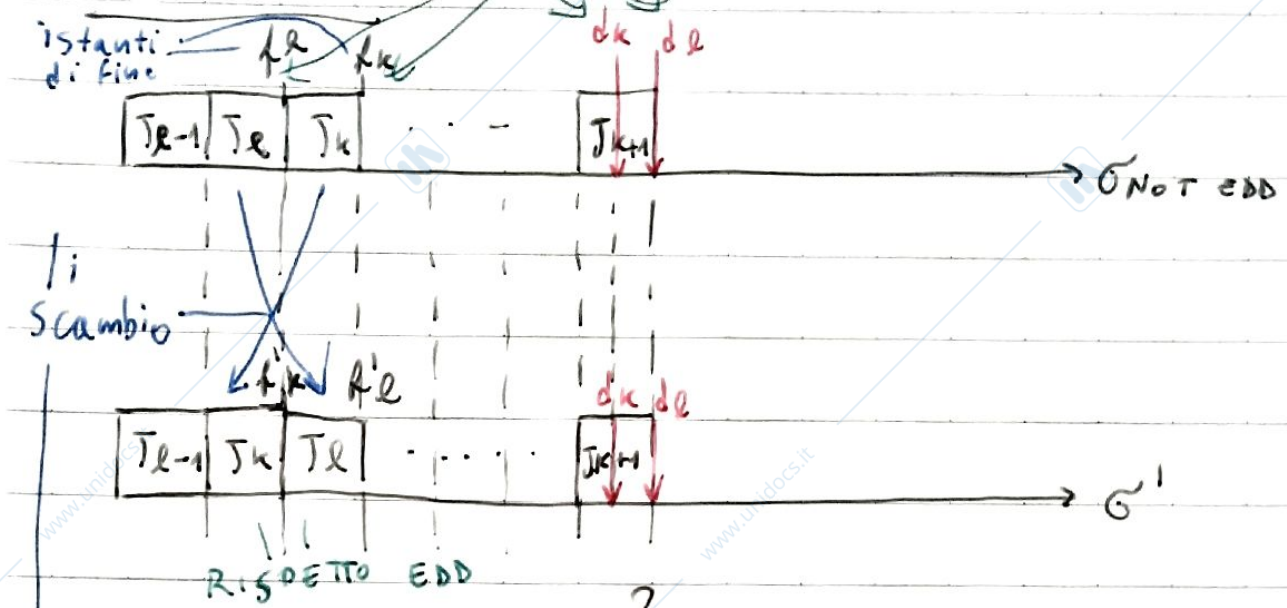
L_{MAX} NON CRESCE

Costruisco una relazione tra una soluzione di partenza e una in σ_{EDD}

QUINDI L_{MAX} di qualsiasi soluzione non può essere più grande di un L_{MAX} in σ_{EDD}

NON RISPETTA EDD

⇒ Dimostrazione



$$L_{MAX}(\sigma_{NOT\ EDD}) \overset{?}{\geq} L_{MAX}(\sigma')$$

l'idea dello switch è che continuo a scambiare finché i processi non soddisfano EDD.

↓ Soluzione finale $\in \sigma_{EDD}$

$$\boxed{\max\{L_l(\sigma_{NOT\ EDD}), L_k(\sigma_{NOT\ EDD})\} \geq \geq \max\{L_l(\sigma'), L_k(\sigma')\}}$$

→ $L_l(\sigma_{NOT\ EDD}) = f_l - d_l$
$L_k(\sigma_{NOT\ EDD}) = f_k - d_k$
$\max = L_k(\sigma_{NOT\ EDD})$ poiché $f_l < f_k$ e $d_l > d_k$

$L_l(\sigma') = f'_l - d_l$ → Verifica: $f'_l - d_l \leq f_k - d_k$, con $f'_l = f_k$ → ⇓ $\boxed{d_l \geq d_k \quad \text{VERO}}$

$L_k(\sigma') = f'_k - d_k$

↓

Verifica: $f'_k - \cancel{d_k} \leq f_k - \cancel{d_k}$

⇓

$\boxed{f'_k \leq f_k \quad \text{VERO}}$ ⇒ $L_{MAX}(\sigma_{NOT\ EDD}) \geq L_{MAX}(\sigma')$

No. SISTEMI INF.

Date 25 . 09 . 19

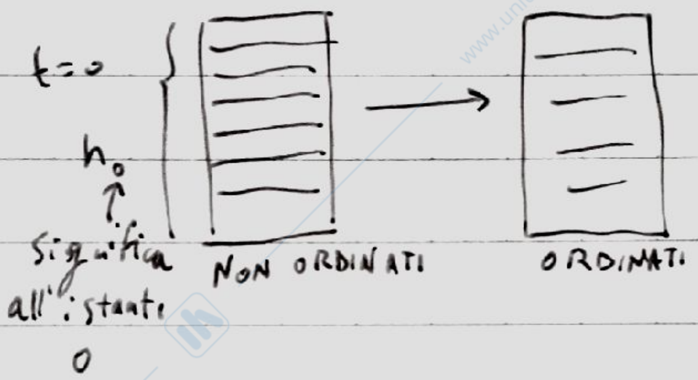
EDF

PROBLEMA: n PROCESSI (POTENZIALMENTE) ASINCRONI, NO VINCOLI DI PRECEDENZA, PREEMPTION

ALGORITMO: AD OGNI ISTANTE DI TEMPO, SI ESEGUE IL PROCESSO READY CON LA DEADLINE MINORE

PROPRIETÀ: ULTIMO PER MINIMIZZAZIONE L_{MAX}

es.



$O(n_0 \lg(n_0))$

$O(n_1 \lg(n_0 + n_1))$

come se li inserissimo in ordine tenendo conto degli n_0 precedenti

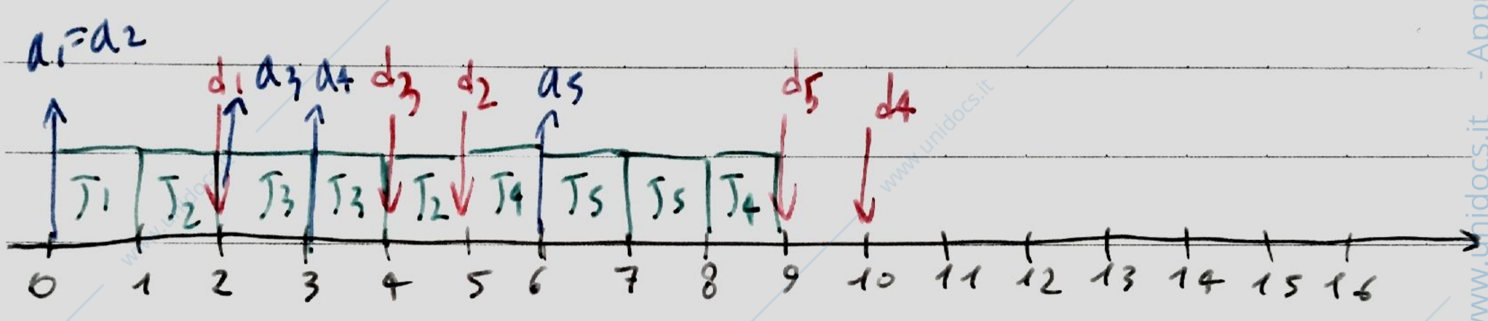
10 — We

No. SISTEMI INF.

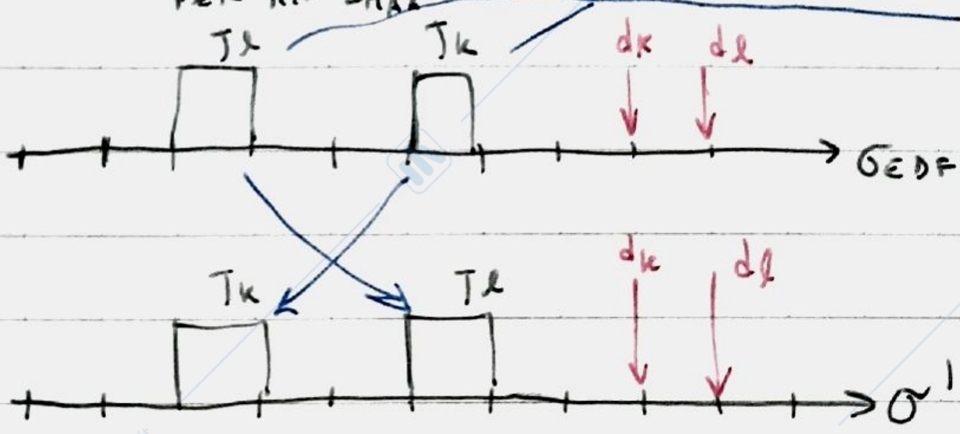
Date 25.09.19

es.

	J_1	J_2	J_3	J_4	J_5
a_i	0	0	2	3	6
c_i	1	2	2	2	2
d_i	2	5	4	10	9



EDF (OTTIMALITÀ) PER MIN L_{MAX} → Dimostrazione



devono essere entrambi attivati

↓

L_{MAX} non cresce

EDF*

PROBLEMA: n PROCESSI (POTENZIALMENTE) ASINCRONI, (POTENZIALMENTE) CON VINCOLI DI PRECEDENZA, PREEMPTION

ALGORITMO: 1° Fase: COSTRUZIONE DI UN PROBLEMA AUSILIARIO

↓

2° Fase: EDF
- Problema senza vincoli di precedenza
- Problema equivalente al problema originale

applicato al problema ausiliario

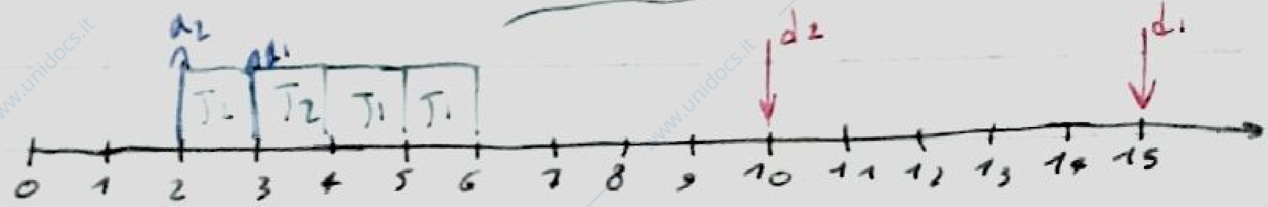
Supponiamo di avere:

	J_1	J_2
a_i	3	2
c_i	2	2
d_i	15	10

vincolo:

$J_1 \to J_2$

no rispetta la precedenza



dobbiamo creare un processo ausiliario:

① $a_2 = 2$

→ costruisco un $a_2^* = \max\{a_2, a_1 + c_1\}$

in generale processo con precedenza

② $d_1 = 15$

→ costruisco $d_1^* = \min\{d_1, d_2 - c_2\}$

questa trasformazione è un po' arbitraria:

se io facessi $a_2^* = \max\{a_2, a_1\}$

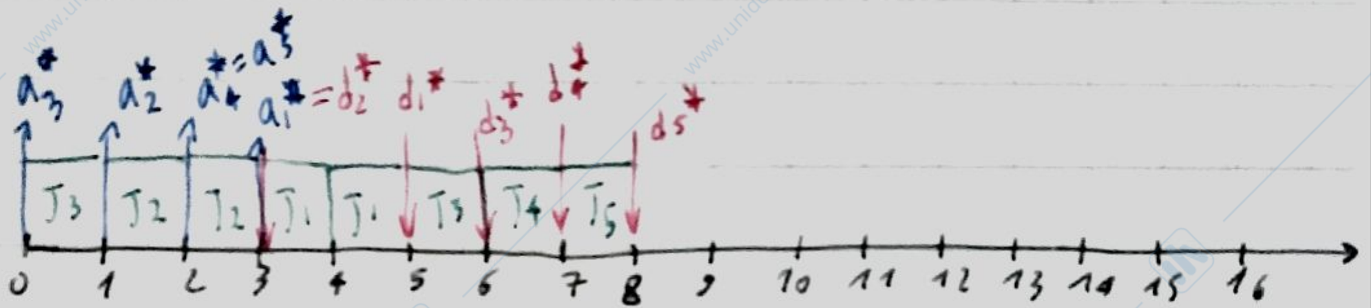
non cambierebbe nulla e

EDF funzionerebbe lo stesso

(All'esame usiamo ①)

es.

	J_1	J_2	J_3	J_4	J_5
a_i	1	1	0	0	2
c_i	2	2	2	1	1
d_i	5	7	9	7	8
a_i^*	3	1	0	2	2
d_i^*	5	3	6	7	8



→ $L_{max} = 0$

(dobbiamo utilizzare la formula $L_i = f_i - d_i$

Valore originale

$L_1 = 5 - 5 = 0$

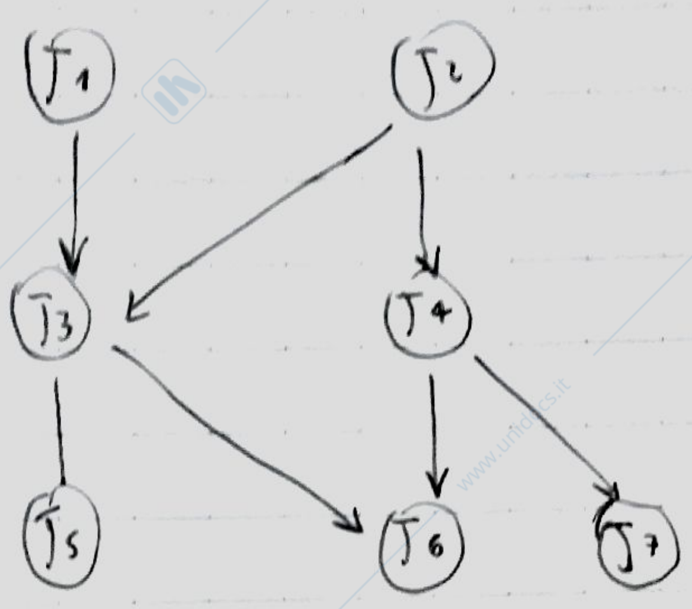
$L_2 = 3 - 7 = -4$

$L_3 = 6 - 9 = -3$

$L_4 = 7 - 7 = 0$

$L_5 = 8 - 8 = 0$

Consideriamo il caso in cui abbiamo:



$a_1^* = a_1 \quad a_2^* = a_2$

$a_3^* = MAX\{a_3, a_1 + c_1, a_2 + c_2\} \quad a_4^* = MAX\{a_4, a_2 + c_2\}$

↳ In generale:

così corretta

$a_k^* = MAX\{a_k, \underbrace{\{a_i^* + c_i\}}_{\text{di tutti quelli che precedono}}\}$

$a_5^* = MAX\{a_5, a_3^* + c_3\}$

$a_6^* = MAX\{a_6, a_3^* + c_3, a_4^* + c_4\}$

$a_7^* = MAX\{a_7, a_4^* + c_4\}$

PERÒ PIAN PIANO CHE SCENDO DEVO CONTARE I CAMBIAMENTI PRECEDENTI

$d_5^* = d_5$

$d_6^* = d_6$

$d_7^* = d_7$

$d_3^* = \min\{d_3, d_5^* - c_5, d_6^* - c_6\}$

$d_4^* = \min\{d_4, d_6^* - c_6, d_7^* - c_7\}$

$d_1^* = \min\{d_1, d_3^* - c_3\}$

$d_2^* = \min\{d_2, d_3^* - c_3, d_4^* - c_4\}$

$\downarrow$

CS.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7
a_i	0	0	0	0	0	0	0
c_i	2	3	3	5	1	2	5
d_i	25	25	25	25	25	25	25
a_i^*	0	0	3	3	6	8	8
d_i^*	20	15	23	20	25	25	25