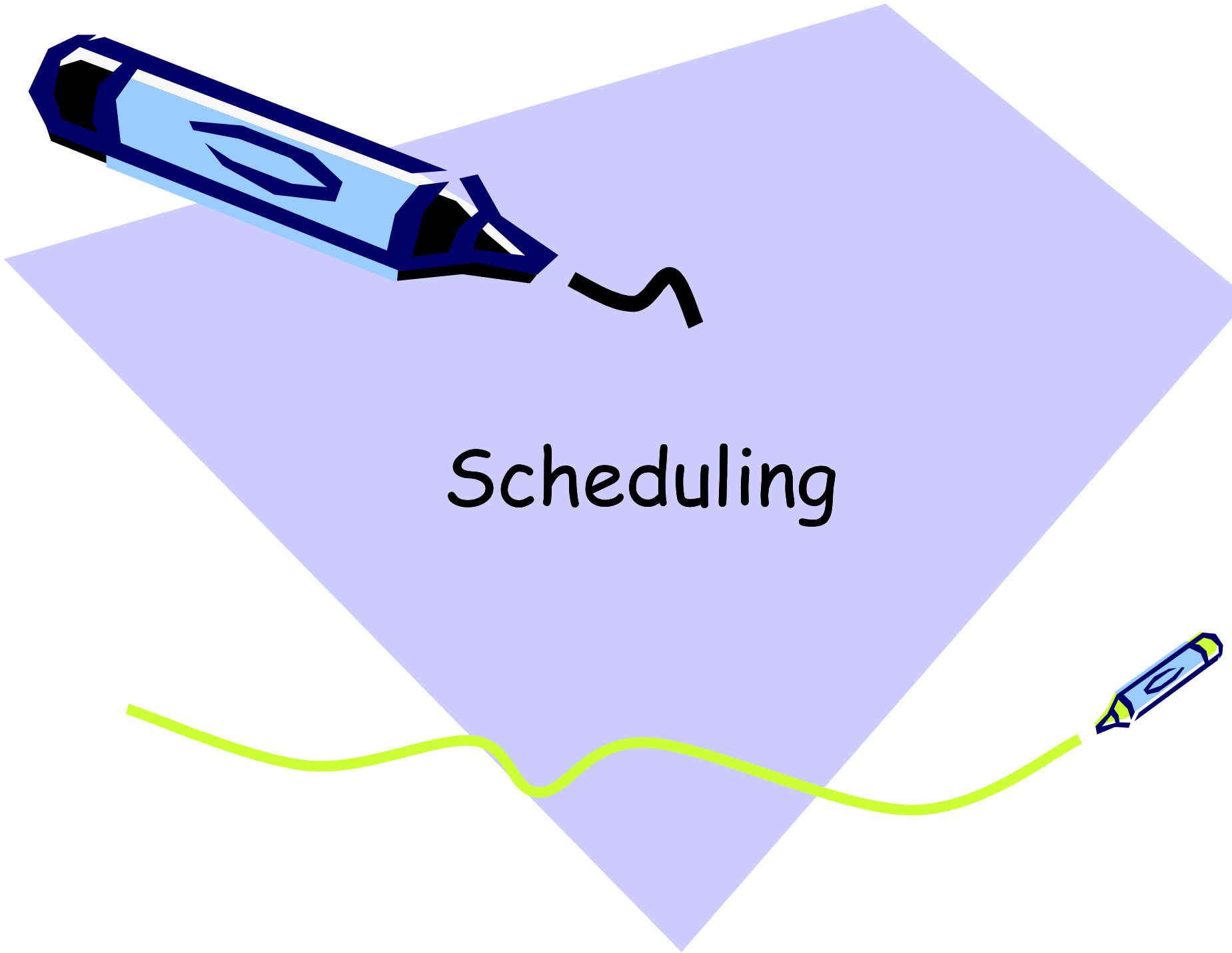

Scheduling

# Organizzazione della produzione

# Pianificazione della produzione: schedulazione di dettaglio

Sono dati:

- Un insieme di lavori (*job*): ognuno costituito da una o più operazioni
- Un insieme di risorse (*macchine*) che devono essere utilizzate per eseguire i lavori

# Scheduling delle operazioni

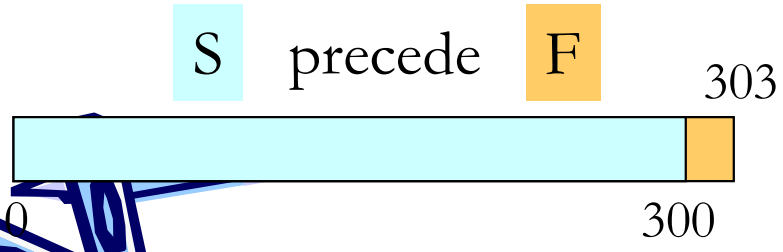
Scelta dei tempi di inizio e fine di ogni operazione su ogni macchina

# Esempio: fotocopie

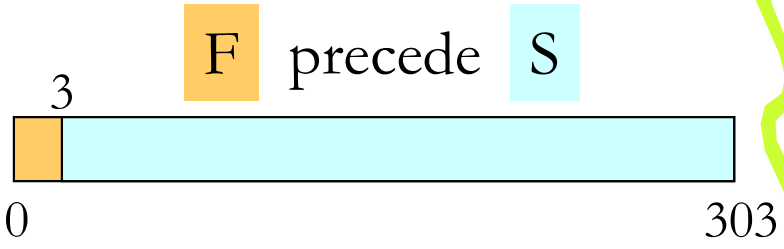
Silvia (S) e Francesca (F) giungono nello stesso momento ad una macchina fotocopiatrice. F deve fare 1 fotocopia, S ne deve fare 100.
Il galateo imporrebbe a S di lasciar passare per prima F, che impegna la macchina per un tempo breve e poi la lascia libera.

È questa una buona idea?

Analisi. Supponiamo che l'esecuzione di una fotocopia richieda 3 secondi. Due casi:



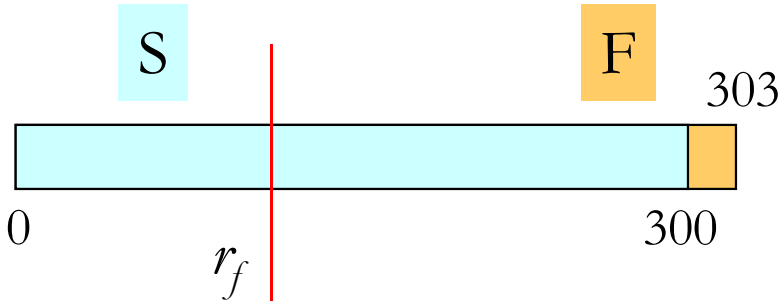

Attesa totale = 300 + 303 = 603




Attesa totale = 3 + 303 = 306

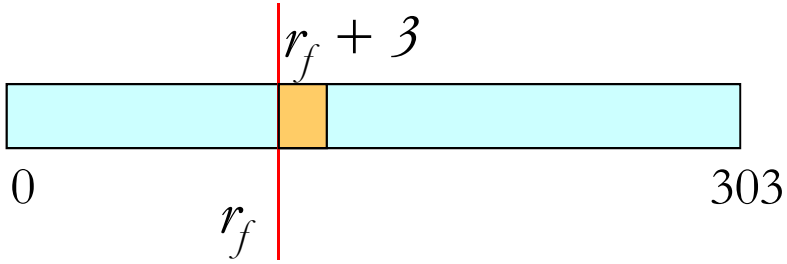
# Esempio: fotocopie

E se Silvia (S) giungesse prima di Francesca (F) (che arriva all'istante $r_f$)?



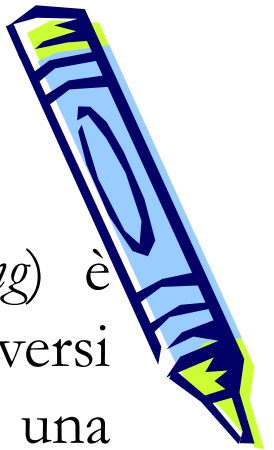

Attesa totale = $300 + 303 - r_f = 603 - r_f$

Di nuovo, il galateo imporrebbe a Silvia di interrompere le proprie copie in favore di Francesca:




Attesa totale = $3 + 303 = 306$

# Esempio: CPU

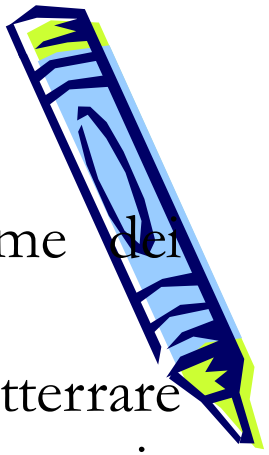


• uno dei compiti del sistema operativo (*multitasking*) è quello di disciplinare l'accesso alla CPU dei diversi programmi di calcolo (eventualmente associati ad una priorità).

• il processamento di un programma può essere interrotto per consentire il completamento di altri. Ciò evita che un programma "lungo" ne blocchi altri molto brevi che hanno priorità più bassa.

•L'obiettivo tipico consiste nel minimizzare l'attesa complessiva dei programmi.

# Esempio: voli in atterraggio



• L'area terminale di un aeroporto contiene l'insieme dei velivoli prossimi all'atterraggio.

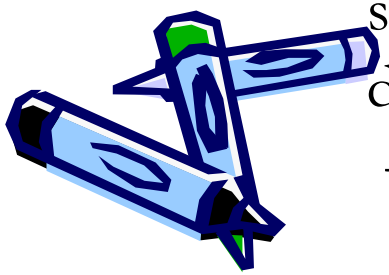
• Il sistema di controllo dell'aeroporto permette di far atterrare al più un volo alla volta e la durata della manovra di atterraggio è nota per ogni velivolo.

• Ad ogni velivolo è inoltre associato un tempo previsto di atterraggio

• Obiettivo del controllore può essere quello di decidere la sequenza in modo da minimizzare la somma (pesata) dei ritardi.

•Vincoli aggiuntivi:

- a causa della turbolenza, si deve garantire una certa separazione in atterraggio fra un aeromobile di grandi dimensioni ed di piccole dimensioni
- precedenze fra voli



# Scheduling delle operazioni

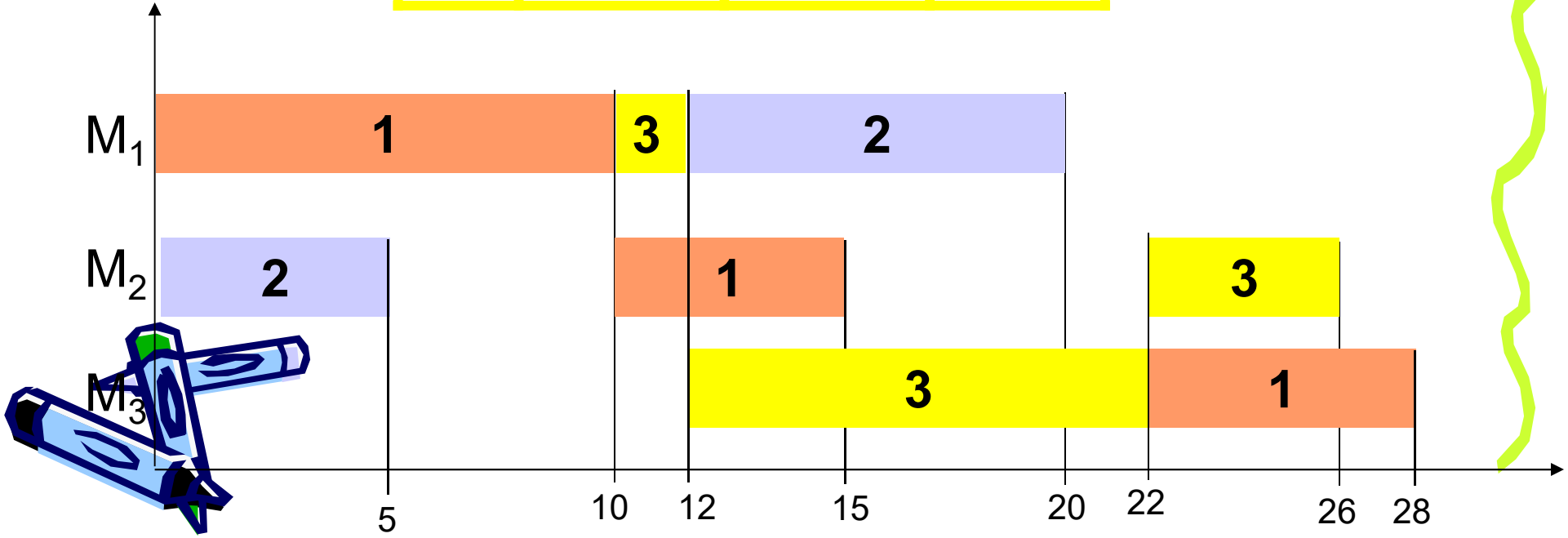
Consideriamo:

3 lavori e 3 macchine

| Job | Sequenza delle operazioni operazione=(macchina, tempo) | | |
|---|---|---|---|
| $J_1$ | $(M_1,10)$ | $(M_2,5)$ | $(M_3,6)$ |
| $J_2$ | $(M_2,5)$ | $(M_1,8)$ | - |
| $J_3$ | $(M_1,2)$ | $(M_3,10)$ | $(M_2,4)$ |

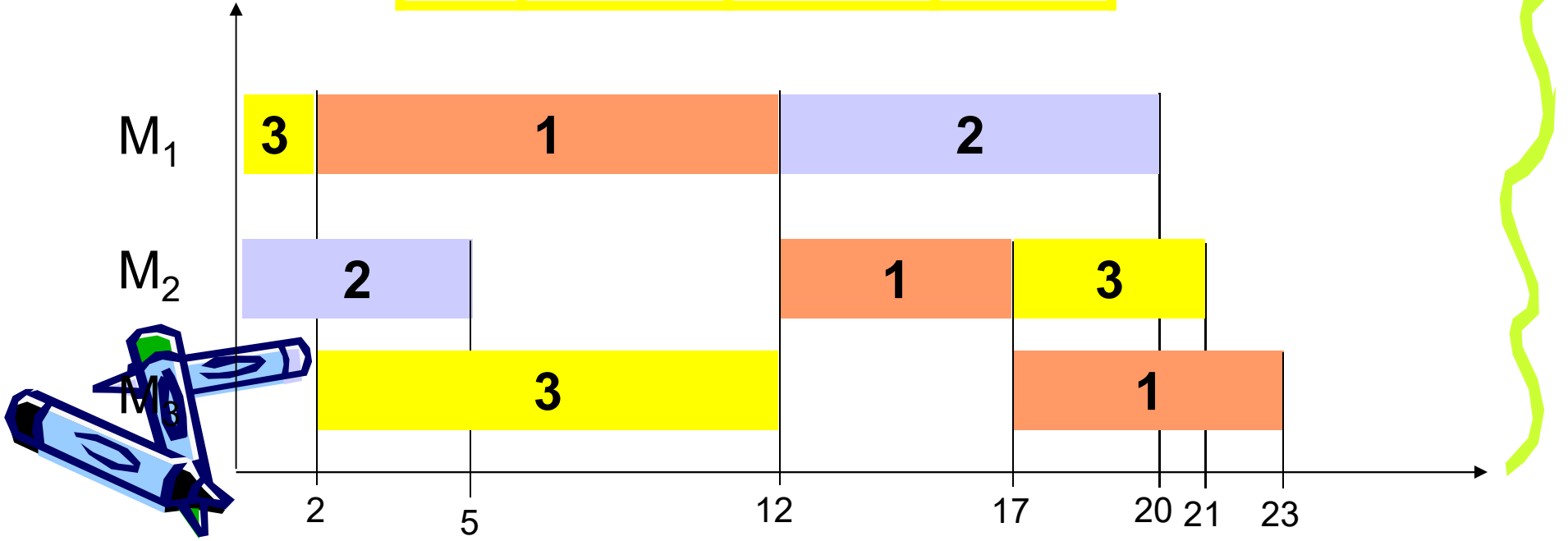
# Diagramma di Gantt

| Job | Sequenza Operazioni | | |
|---|---|---|---|
| $J_1$ | $(M_1,10)$ | $(M_2,5)$ | $(M_3,6)$ |
| $J_2$ | $(M_2,5)$ | $(M_1,8)$ | - |
| $J_3$ | $(M_1,2)$ | $(M_3,10)$ | $(M_2,4)$ |



# Diagramma di Gantt

| Job | Sequenza Operazioni | | |
|---|---|---|---|
| $J_1$ | $(M_1,10)$ | $(M_2,5)$ | $(M_3,6)$ |
| $J_2$ | $(M_2,5)$ | $(M_1,8)$ | - |
| $J_3$ | $(M_1,2)$ | $(M_3,10)$ | $(M_2,4)$ |



# Esempio

Aldo (A), Bruno (B), Carlo (C), Duilio (D) condividono un appartamento. Ogni mattino ricevono 4 giornali: Financial Times (FT), Guardian (G), Daily Express (DE), Sun (S).
Ciascun lettore inizia la lettura ad una certa ora, ha la propria sequenza fissata di lettura dei gionali e legge ciascun giornale per un tempo prefissato:

| lettore | Ora inizio | Sequenza lettura (tempo in min.) | | | |
|---|---|---|---|---|---|
| Aldo | 8.30 | FT(60) | G (30) | DE (2) | S(5) |
| Bruno | 8.45 | G(75) | DE(3) | FT(25) | S(10) |
| Carlo | 8.45 | DE(5) | G(15) | FT(10) | S(30) |
| Duilio | 9.30 | S(90) | FT(1) | G(1) | DE(1) |

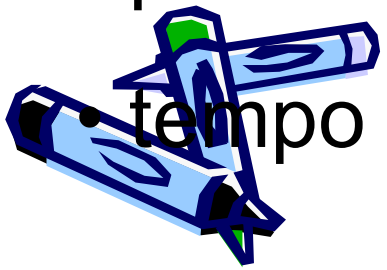
# Esempio

- tutti i lettori preferiscono (eventualmente) aspettare che un giornale (il prossimo nella propria sequenza) sia disponibile anziché modificare la sequenza prefissata.
- nessun lettore rilascia un giornale prima di averlo letto completamente.
- ciascun lettore termina la lettura di tutti i giornali prima di uscire.
- i quattro lettori attendono che tutti abbiano terminato di leggere prima di uscire di casa

Problema:

Qual'è la minima ora in cui A, B, C, D possono uscire?

# Classificazione dei problemi di *scheduling*



Caratterizzazione dei lavori:

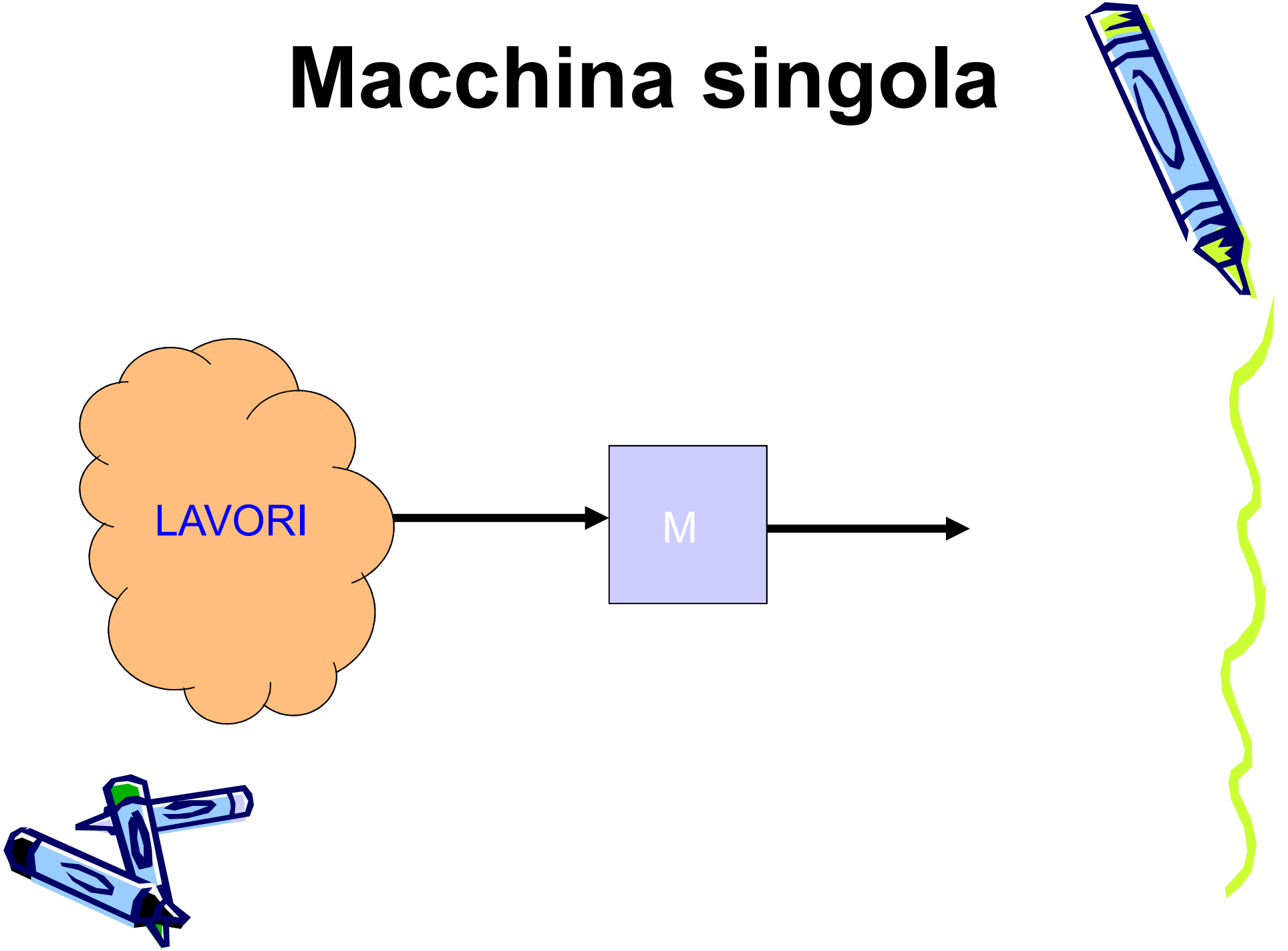
- tempo di processamento $p_j$ ($p_{kj}$)
- data di consegna (*duedate* o *deadline*) $d_j$
- data di rilascio (*release date*) $r_j$
- peso del lavoro (priorità) $w_j$
- tempo di set-up tra due lavori $s_{ij}$

# Classificazione dei problemi di *scheduling*

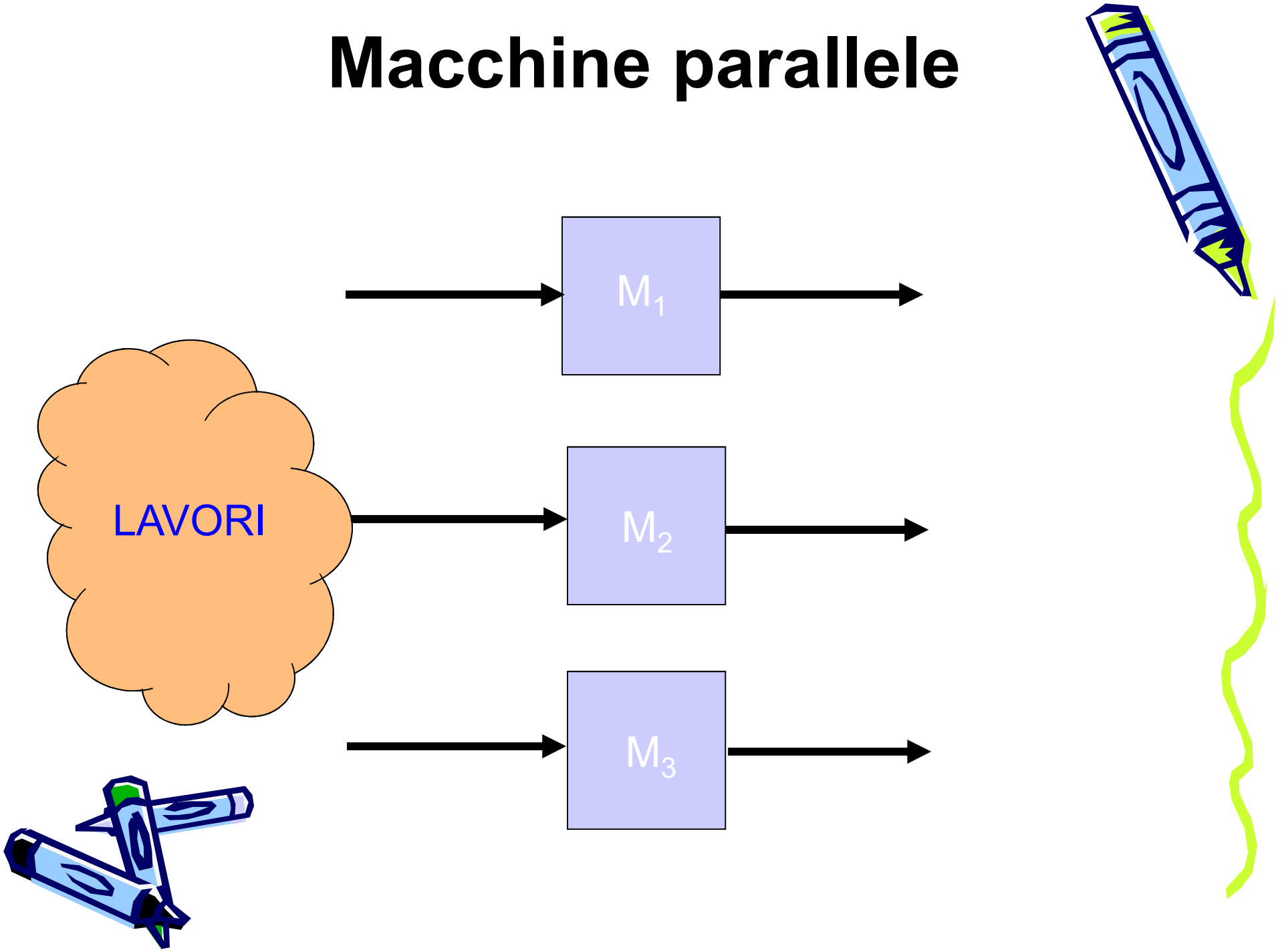
Caratterizzazione delle risorse e dell'ambiente produttivo:

- macchina singola
- macchine parallele
  - *identiche*
  - *scorrelate*
  - *uniformi*
- Flow shop
- Job shop

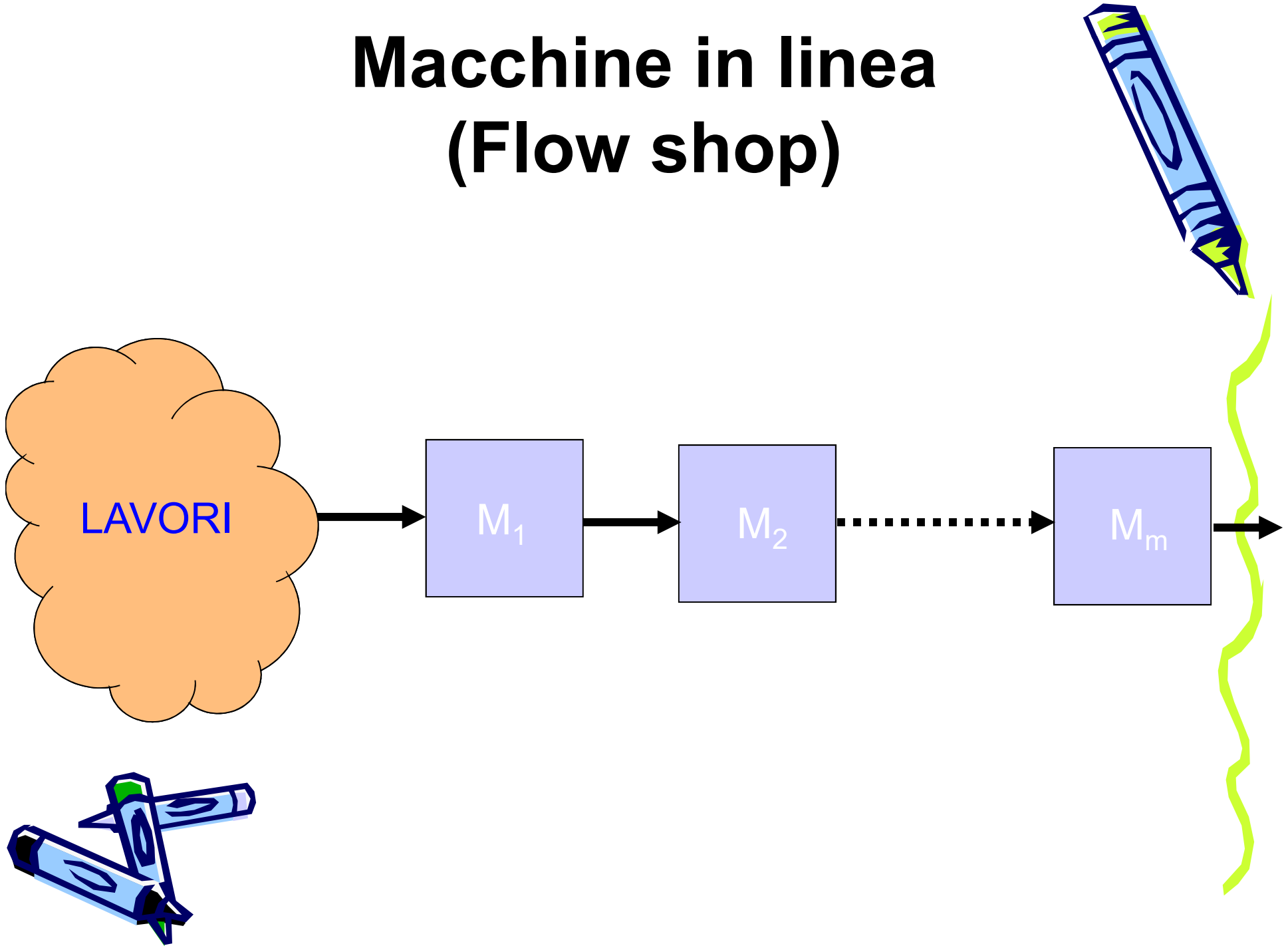
# Macchina singola



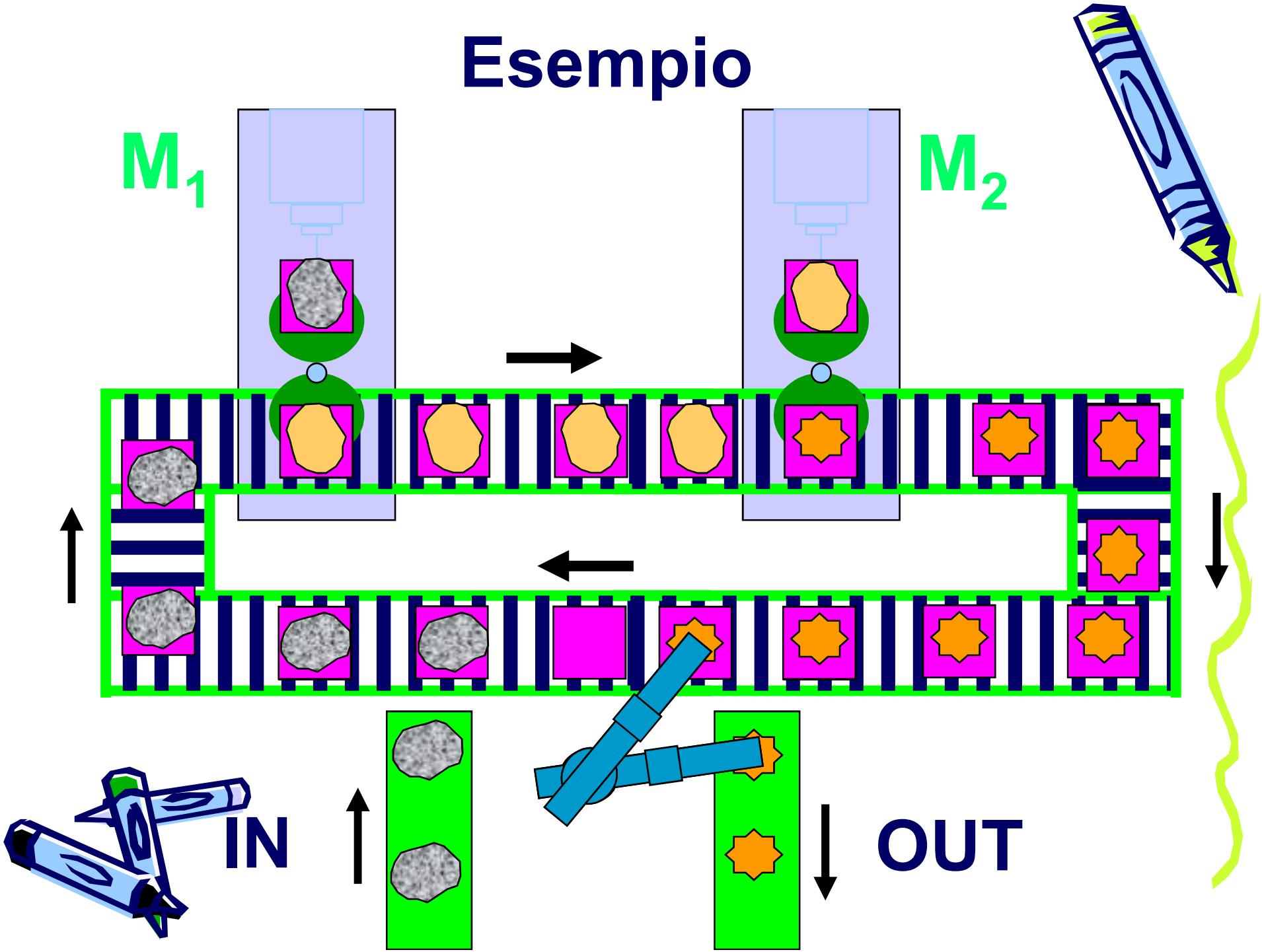
# Macchine parallele



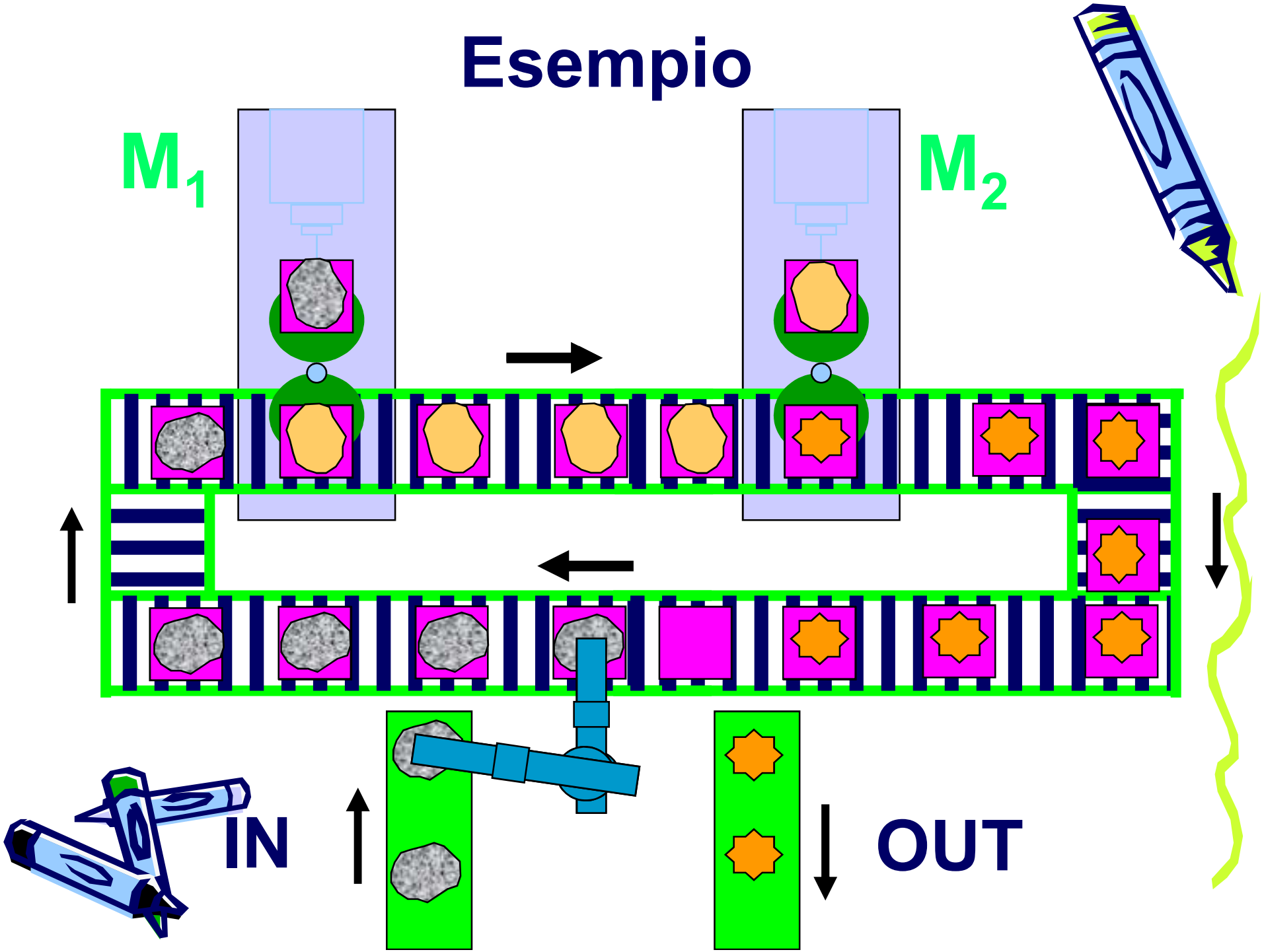
# Macchine in linea (Flow shop)



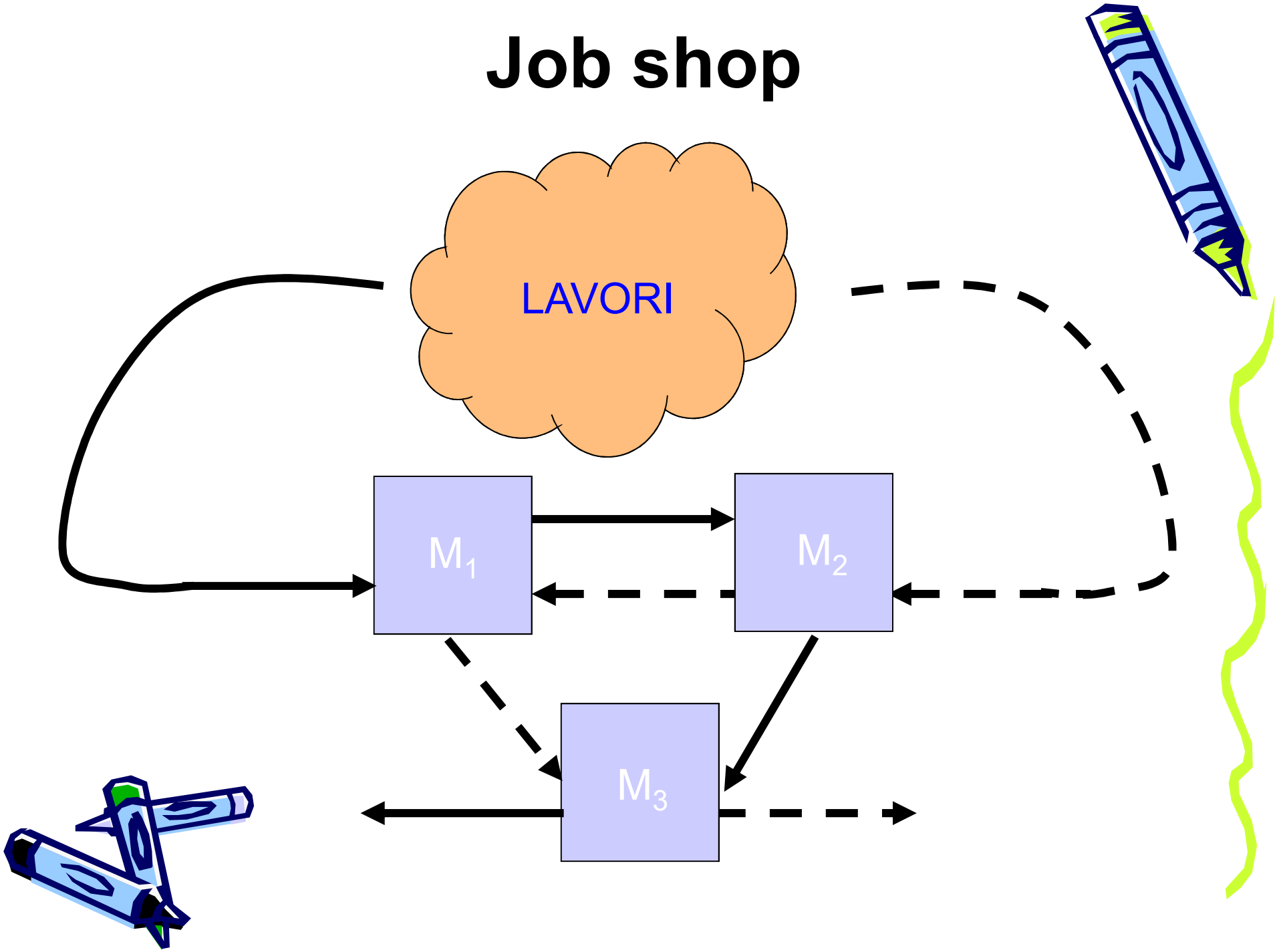
# Esempio



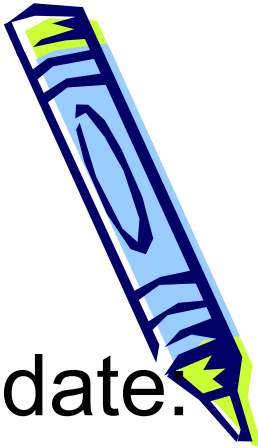
# Esempio



# Job shop



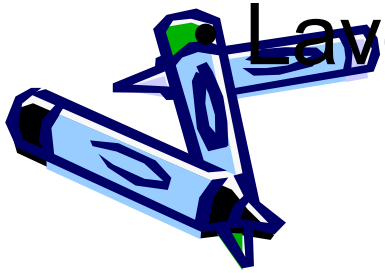
# Misure di prestazione (lavori)



Dato il lavoro *i* con release date e duedate:

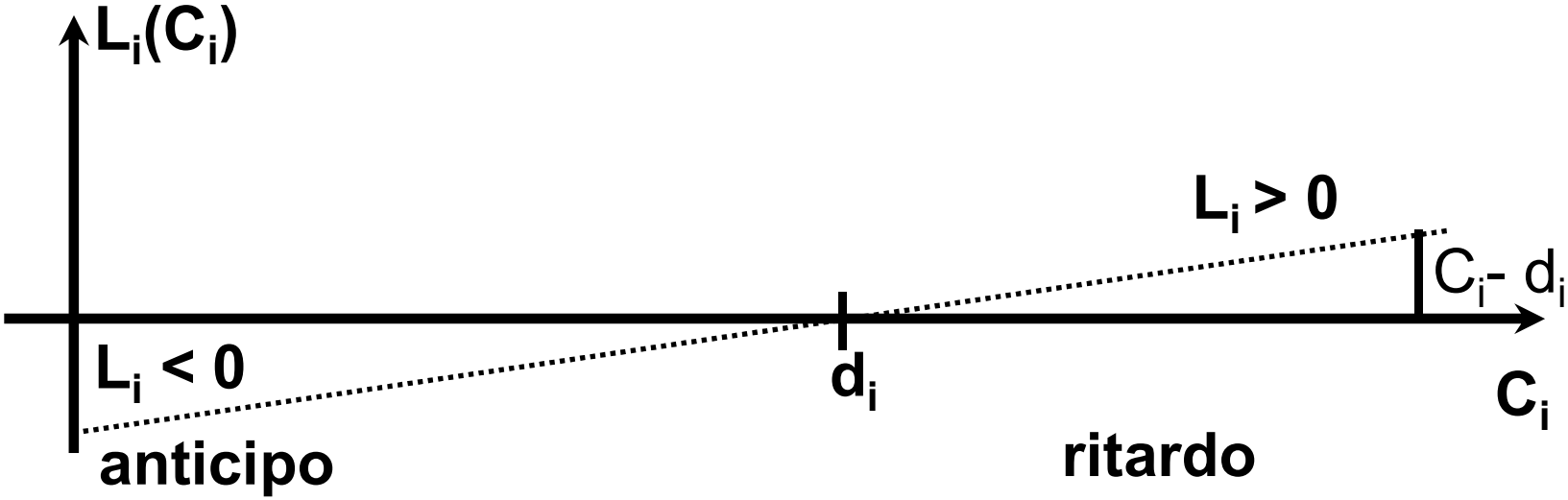
- tempo di completamento $C_i$
- tempo di attraversamento $F_i = C_i - r_i$
- *Lateness* $L_i = C_i - d_i$
- *Tardiness* $T_i = \max\{ 0, C_i - d_i \}$
- *Earliness* $E_i = \max\{ 0, d_i - C_i \}$
- Lavori in ritardo $U_i = 1$ se $C_i > d_i$

  $U_i = 0$ se $C_i \leq d_i$



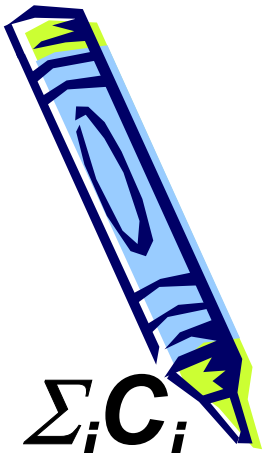
# Lateness (Ritardo)

Ritardo del lavoro *i* : $L_i = C_i - d_i$

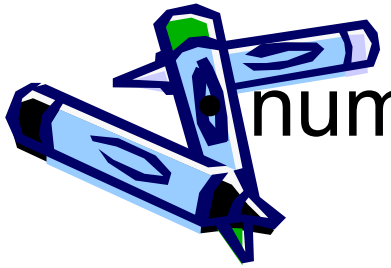



**$d_i$: tempo di consegna (duedate) per il lavoro *i***

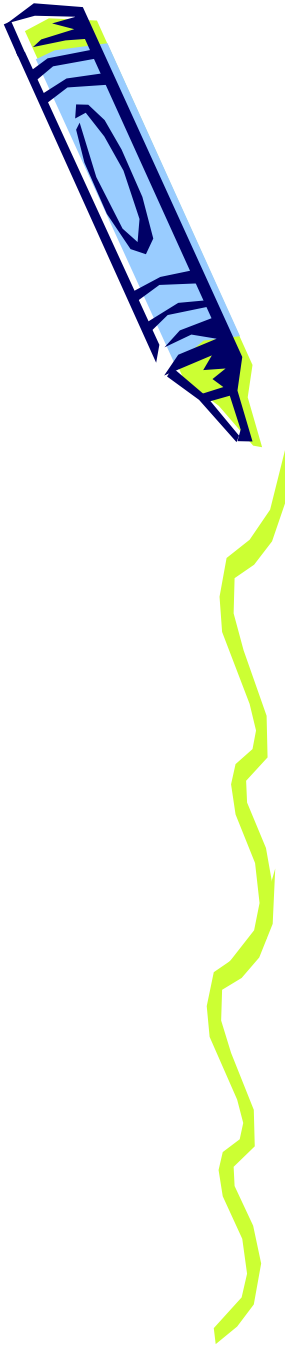
# Misure di prestazione (sistema)



- somma dei tempi di completamento: $\Sigma_i C_i$
- *flow time* totale: $\Sigma_i F_i$
- massima *Lateness:* $L_{max} = max_i L_i$
- massima *Tardiness:* $T_{max} = max_i T_i$
- *Tardiness* totale pesata $\Sigma_i w_i T_i$
- makespan $C_{max} = max_i C_i$
- numero di lavori in ritardo $\Sigma_i U_i$



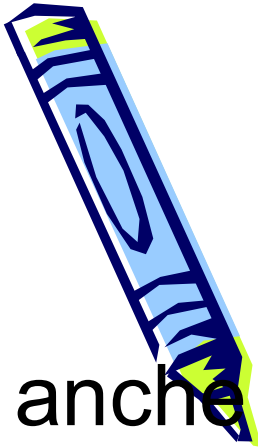
# Misure di prestazione



Equivalenza tra misure

$$\sum_{i=1}^{n} L_i = \sum_{i=1}^{n} C_i - \sum_{i=1}^{n} d_i = \sum_{i=1}^{n} F_i + \sum_{i=1}^{n} (r_i - d_i)$$

# Misure di prestazione



Una sol. che minimizza $L_{max}$ minimizza anche $T_{max}$ (ma, in generale, non è vero il viceversa):

$$T_{\max} = \max\{T_1,\ldots,T_n,0\} =$$
$$\max\{\max\{L_1,0\},\ldots,\max\{L_n,0\}\} =$$
$$= \max\{L_1,\ldots,L_n,0\} = \max\{L_{\max},0\}$$

# Esempio

| Lavori | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $p_i$ | 8 | 16 | 10 | 7 | 2 |

Sequenza ottima (5, 4, 1, 3, 2)

# Esempio

Aldo (A), Bruno (B), Carlo (C), Duilio (D) condividono un appartamento. Ogni mattino ricevono 4 giornali: Financial Times (FT), Guardian (G), Daily Express (DE), Sun (S).
Ciascun lettore inizia la lettura ad una certa ora, ha la propria sequenza fissata di lettura dei gionali e legge ciascun giornale per un tempo prefissato:

| lettore | Ora inizio | Sequenza lettura (tempo in min.) | | | |
|---|---|---|---|---|---|
| Aldo | 8.30 | FT(60) | G (30) | DE (2) | S(5) |
| Bruno | 8.45 | G(75) | DE(3) | FT(25) | S(10) |
| Carlo | 8.45 | DE(5) | G(15) | FT(10) | S(30) |
| Duilio | 9.30 | S(90) | FT(1) | G(1) | DE(1) |

# Esempio

- tutti i lettori preferiscono (eventualmente) aspettare che un giornale (il prossimo nella propria sequenza) sia disponibile anziché modificare la sequenza prefissata.
- nessun lettore rilascia un giornale prima di averlo letto completamente.
- ciascun lettore termina la lettura di tutti i giornali prima di uscire.
- i quattro lettori attendono che tutti abbiano terminato di leggere prima di uscire di casa

Problema:

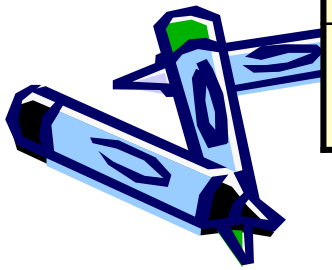
Qual'è la minima ora in cui A, B, C, D possono uscire?

# Soluzioni

• Il problema equivale a: determinare in quale ordine ciascun giornale deve esser letto dai quattro lettori in modo da minimizzare il tempo di lettura totale.

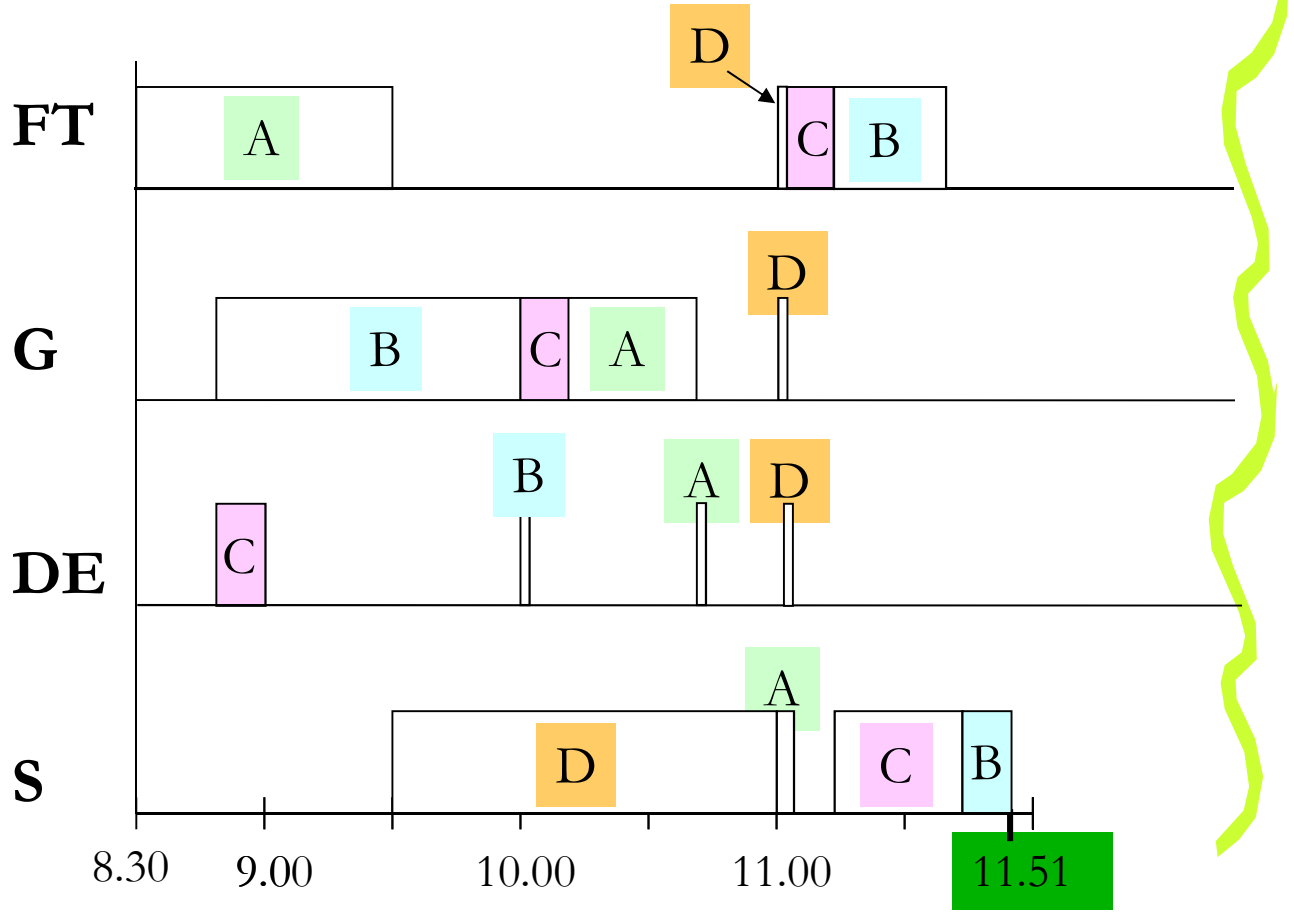
una possibile soluzione:

| giornale | Lett. 1 | Lett. 2 | Lett. 3 | Lett. 4 |
|---|---|---|---|---|
| FT | A | D | C | B |
| G | B | C | A | D |
| DE | C | B | A | D |
| S | D | A | C | B |



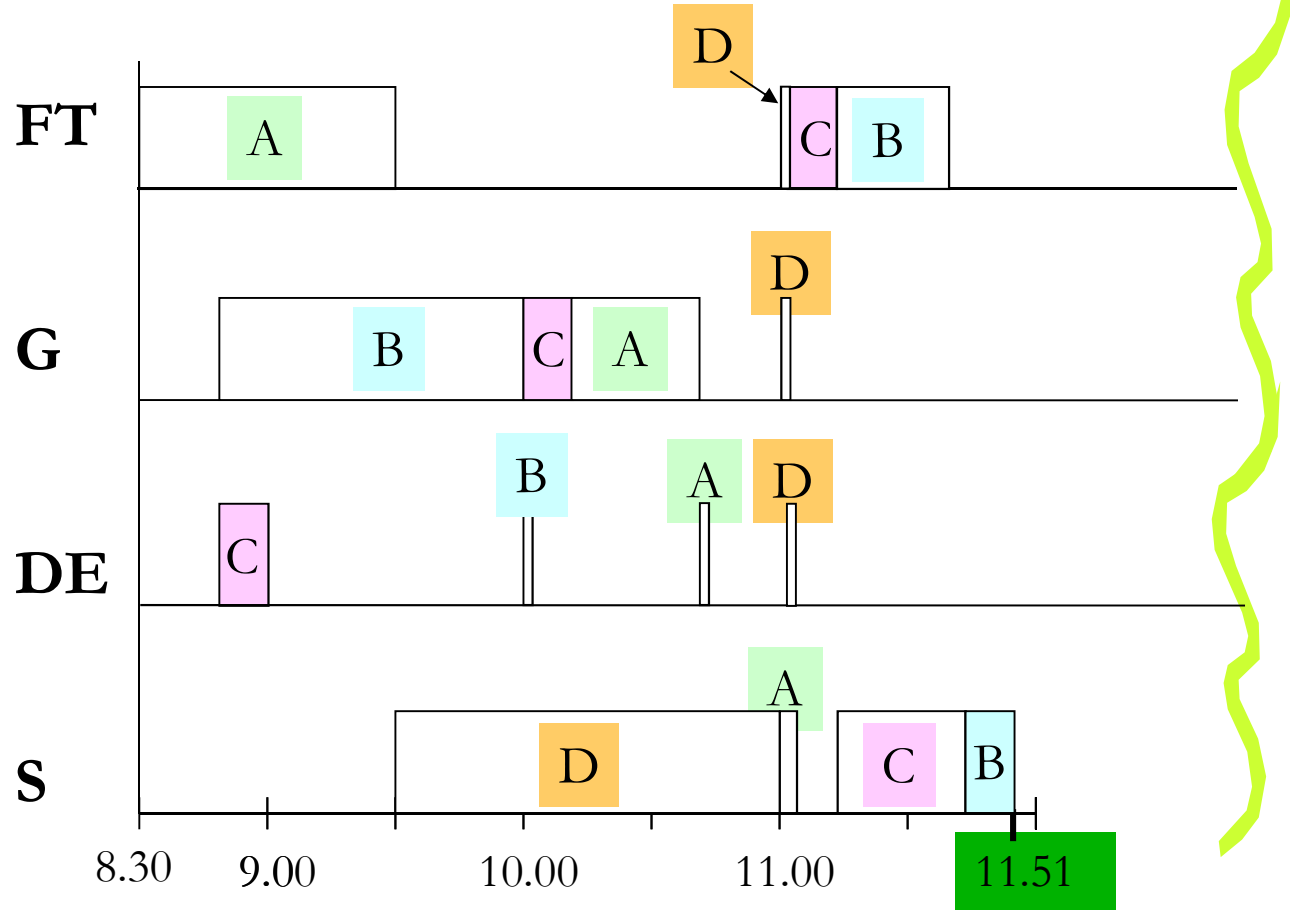
| Giorn. | L1 | L2 | L 3 | L4 |
|---|---|---|---|---|
| **FT** | A | D | C | B |
| **G** | B | C | A | D |
| **DE** | C | B | A | D |
| **S** | D | A | C | B |

# Diagramma di Gantt

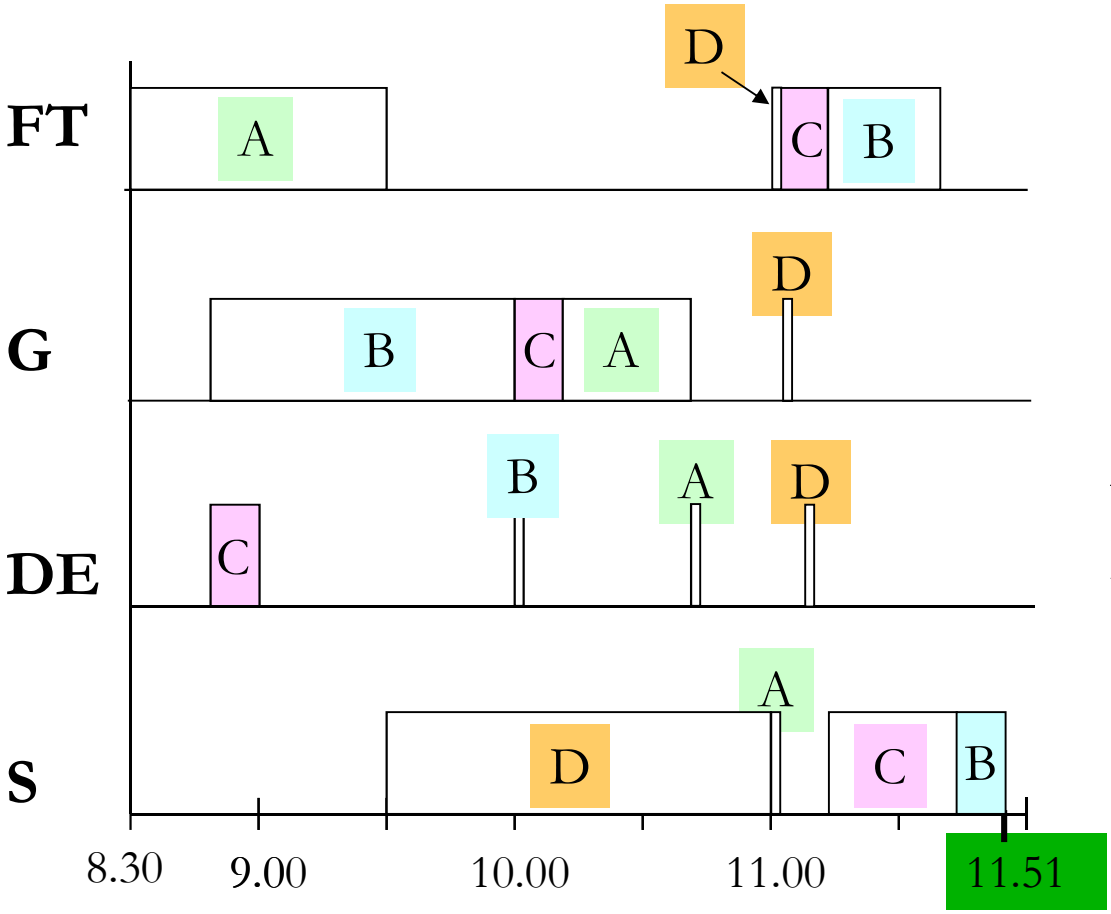


# Vincoli tecnologici

| lettore | inizio | Sequenza lettura | | | |
|---|---|---|---|---|---|
| Aldo | 8.30 | FT | G | DE | S |
| Bruno | 8.45 | G | DE | FT | S |
| Carlo | 8.45 | DE | G | FT | S |
| Duilio | 9.30 | S | FT | G | DE |



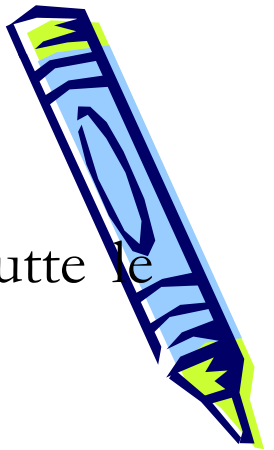
# Commenti




• un lettore può essere inattivo (in *idle*) in certi periodi (C in [10.15, 11.01])

• ci può essere un giornale "fermo" anche se c'è un lettore disponibile che deve ancora leggerlo: B potrebbe avere **S** 11.05, ma questo violerebbe la sua sequenza; FT potrebbe avere C alle 9.30 ma viola la soluzione

# Esercizi



• Determinare il valore del tempo di completamento di tutte le letture per la seguente soluzione:

| giornale | Lett. 1 | Lett. 2 | Lett. 3 | Lett. 4 |
|---|---|---|---|---|
| FT | D | B | A | C |
| G | D | C | B | A |
| DE | D | B | C | A |
| S | A | D | C | B |

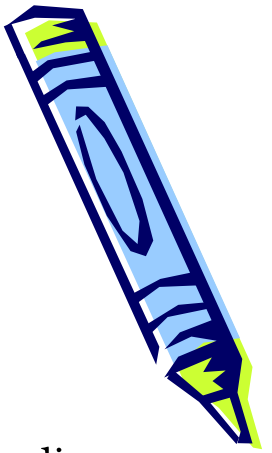
• Determinare uno schedule migliore di quello proposto in precedenza

# Enumerazione totale

- Il numero di soluzioni è $(4!)^4 = 331.776$
- per un problema con $n$ lettori e $m$ giornali diventa $(n!)^m$.
- esaminando 10.000 soluzioni al secondo (4 giornali e $n$ lettori):

| $n$ | tempo |
|---|---|
| 5 | 354 min |
| 10 | $2 \cdot 10^{17}$ giorni |

# Elementi di un problema di scheduling



- ***job***: attività da svolgere

  Fotocopia, esecuzione di un programma di calcolo, lettura di un giornale

Un job può rappresentare una singola attività o un insieme di attività (*task*) tecnologicamente legate

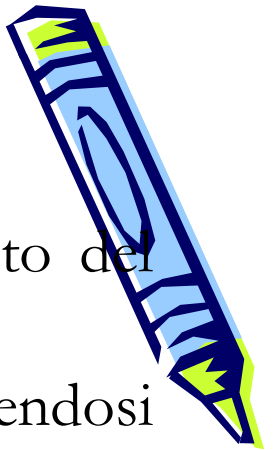
- ***macchine***: risorse che eseguono le attività

  Fotocopiatrice, CPU, giornali

L'esecuzione di un'attività su una macchina è detta ***operazione.***

I problemi di scheduling si classificano in base alle caratteristiche dei task e all'architettura delle macchine

# Attributi dei job



- ***tempo di processamento*** $p_{ij}$: durata del processamento del job *j* sulla macchina *i*
- ***release date*** $r_j$: tempo in cui *j* arriva nel sistema, rendendosi disponibile al processamento
- ***due date*** $d_j$: tempo entro il quale si desidera che il job *j* sia completato (data di consegna)
- ***peso*** $w_j$: indica l'importanza del job j

Notazione a tre campi: $\alpha/\beta/\gamma$:

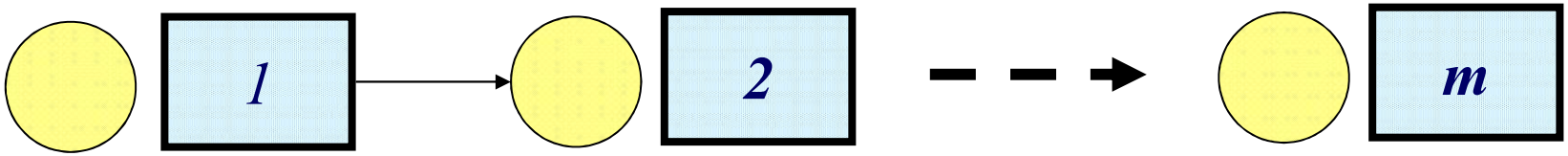
- $\alpha$ descrive il sistema di macchine
- $\beta$ rappresenta vincoli e modalità di processamento (0, 1 o più componenti)
- $\gamma$ indica l'obiettivo

# Campo α

• *Macchina singola (1)* : ciascun job richiede una singola operazione da eseguirsi sull'unica macchina disponibile

• *Macchine identiche parallele* ($P_m$) : ciascun job richiede una singola operazione da eseguirsi su una qualunque delle *m* macchine identiche.

• *Flow shop* ($F_m$) : *m* macchine in serie. Ciascun job deve essere processato su ciascuna di esse (*m* task). Tutti i job hanno lo stesso *routing*. Una volta terminata un'operazione il job viene posto nel buffer della macchina successiva




• *Job shop* ($J_m$) : ciascun job deve essere processato da un sottoinsieme di un insieme di *m* macchine, secondo un proprio routing.

# Campo β

• *Release dates* ($r_j$) : il job $j$ non può iniziare il processamento prima dell'istante $r_j$

• *Tempi di set-up* ($s_{jk}$) : tempo richiesto per il riattrezzaggio delle macchine fra i job $j$ e $k$. Se dipende dalla macchina $i$, si esprime con $s^i_{jk}$

• *Preemption (prmp)* : è ammesso interrompere un'operazione su una macchina prima del suo completamento per iniziare una nuova operazione. Il processamento eseguito prima dell'interruzione non va perso: quando l'operazione viene ripresa, essa richiede solo il tempo *rimanente* di processamento

• *vincoli di precedenza (prec)* : un job può essere processato solo se tutti i job di un certo insieme sono stati completati. Grafo delle precedenze. Se ciascun job ha al più un predecessore e un successore β = *chain*.

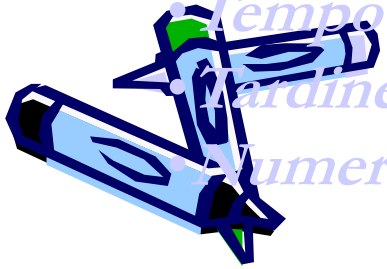
• *breakdown (brkdwn)* : le macchine non sono sempre disponibili, ma hanno periodi fissati di interruzione del servizio

# Campo γ

**Definizioni.**

| | |
|---|---|
| $C_j$ | tempo di completamento del job $j$ |
| $L_j = C_j - d_j$ | ***lateness*** del job $j$ |
| $T_j = \max(L_j, 0)$ | ***tardiness*** del job $j$ |
| $U_j$ = 1 se $C_j > d_j$ (*tardy* job) e *0* altrimenti | |

- *Makespan* ($C_{max}$) : $\max(C_1, \ldots, C_n)$ tempo di completamento dell'ultimo job
- *Massima Lateness* ($L_{max}$)
- *Tempo totale pesato di completamento* ($\Sigma w_j C_j$)
- *Tardiness totale pesata* ($\Sigma w_j T_j$)
- *Numero di tardy job* ($\Sigma U_j$)



# Definizioni di base

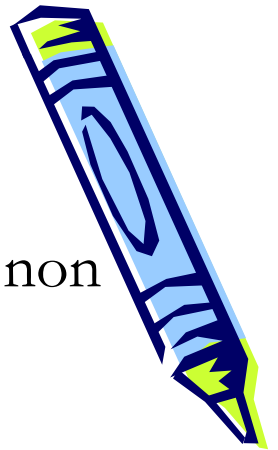
• Una funzione obiettivo si dice **regolare** se è non-decrescente in $C_1, \ldots, C_n$

• Una permutazione dei job che definisce l'ordine con cui i job sono processati su una certa macchina si dice **sequenza**

• Una soluzione di un problema di scheduling si dice **schedule**. Consiste nell' assegnamento di un istante di inizio a ciascuno dei task di ogni job (se *prmp*, istanti di inizio di ciascuna delle parti in cui viene suddiviso un task)

• uno schedule ammissibile è detto **nondelay** se nessuna macchina è ferma quando esiste un'operazione disponibile al processamento

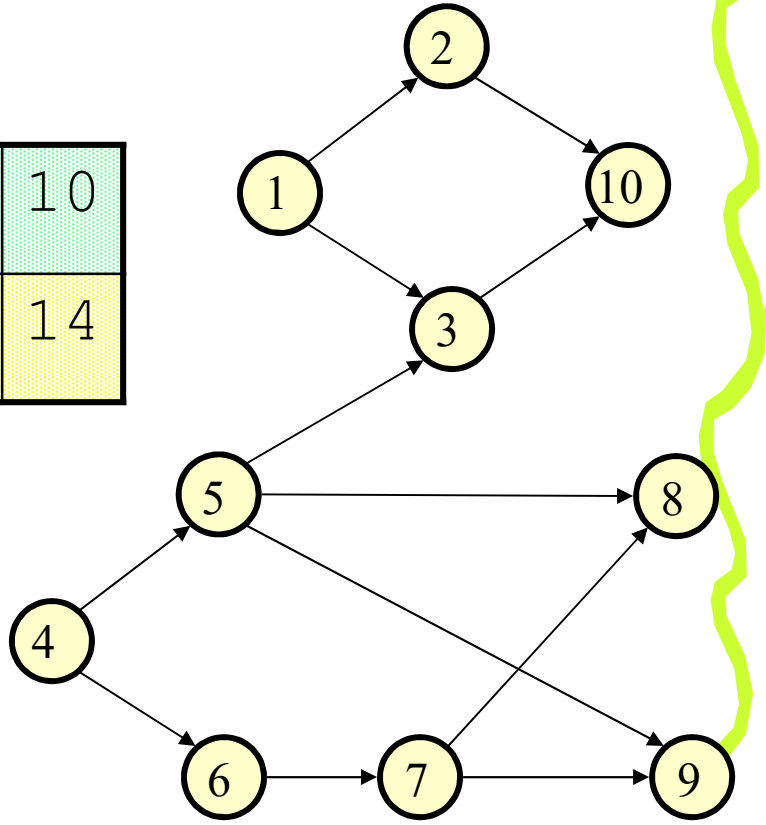
# Esempio: fermi macchina non forzati



• Richiedere che lo schedule sia privo di fermi macchina non forzati può portare a comportamenti controintuitivi.

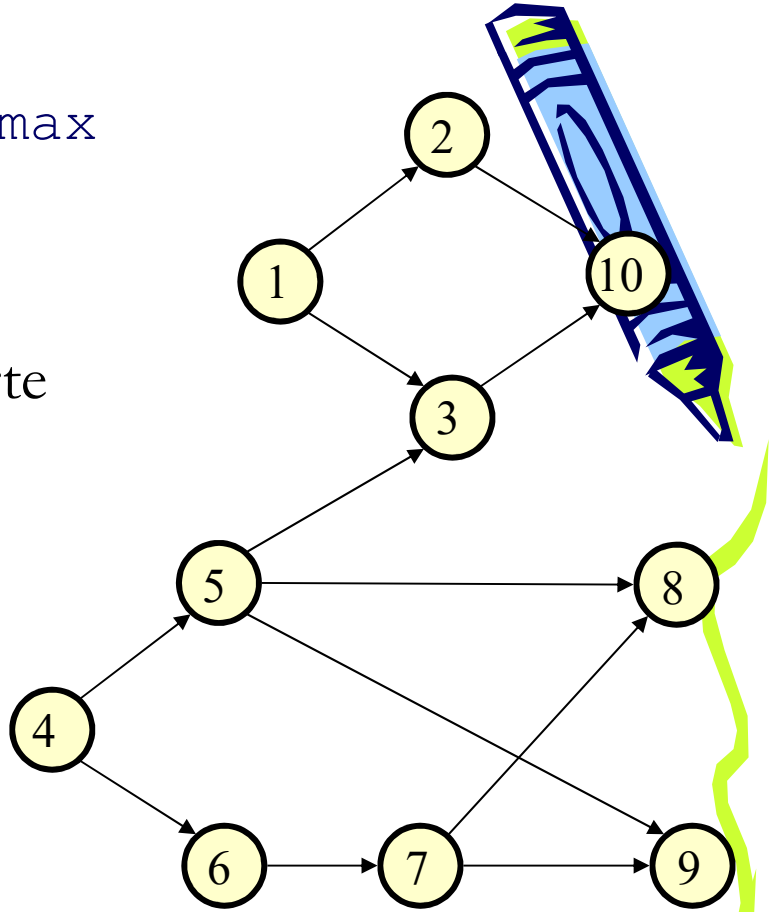
Esempio: $P_2/prec/C_{max}$

| job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_j$ | 7 | 6 | 6 | 1 | 2 | 1 | 1 | 7 | 7 | 14 |

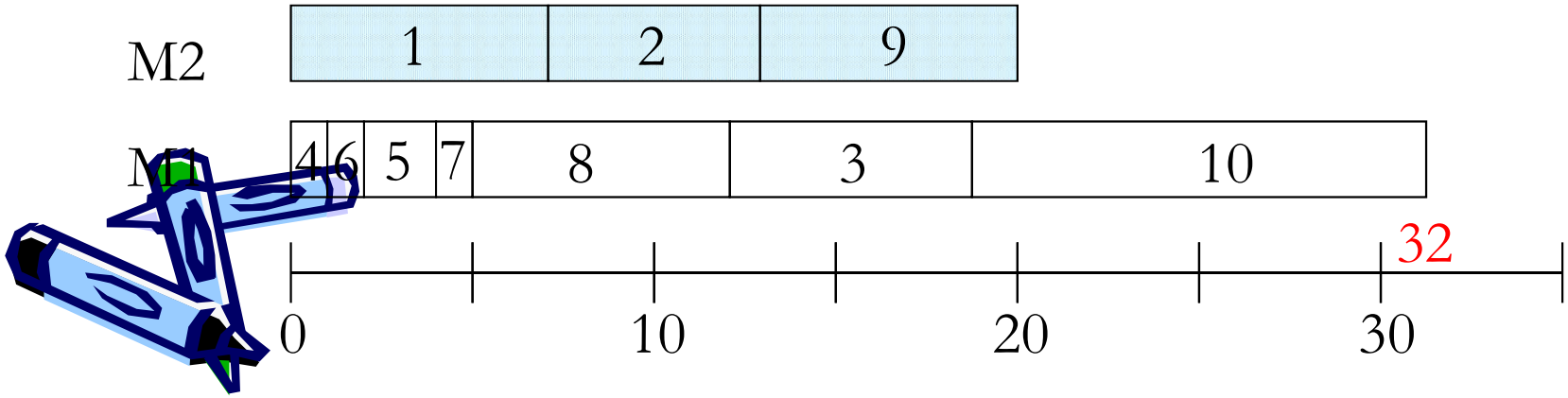


# $P_2/prec/C_{max}$

Regola di buon senso: mandare una parte processabile sulla prima macchina libera.

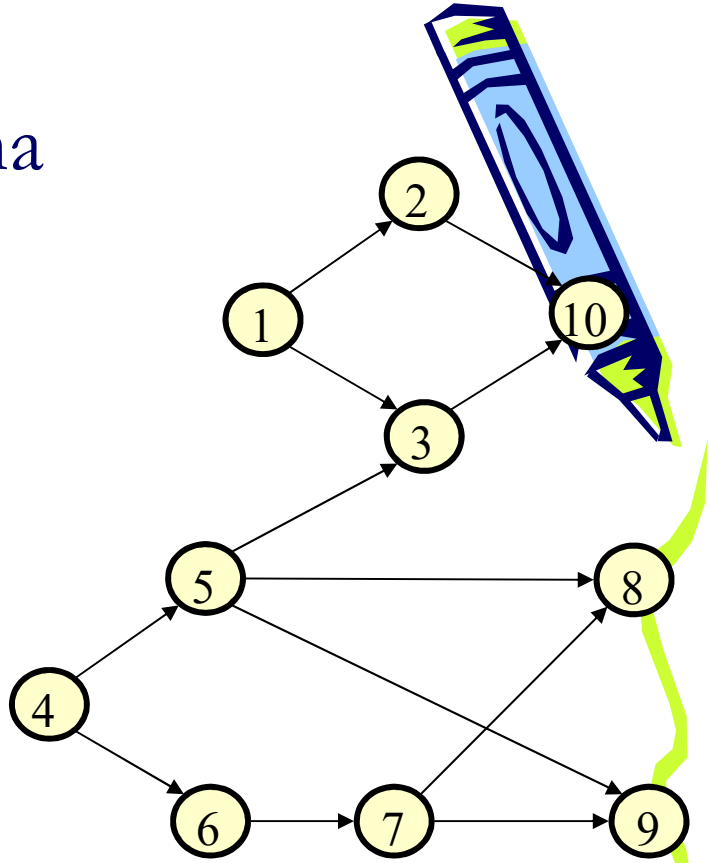


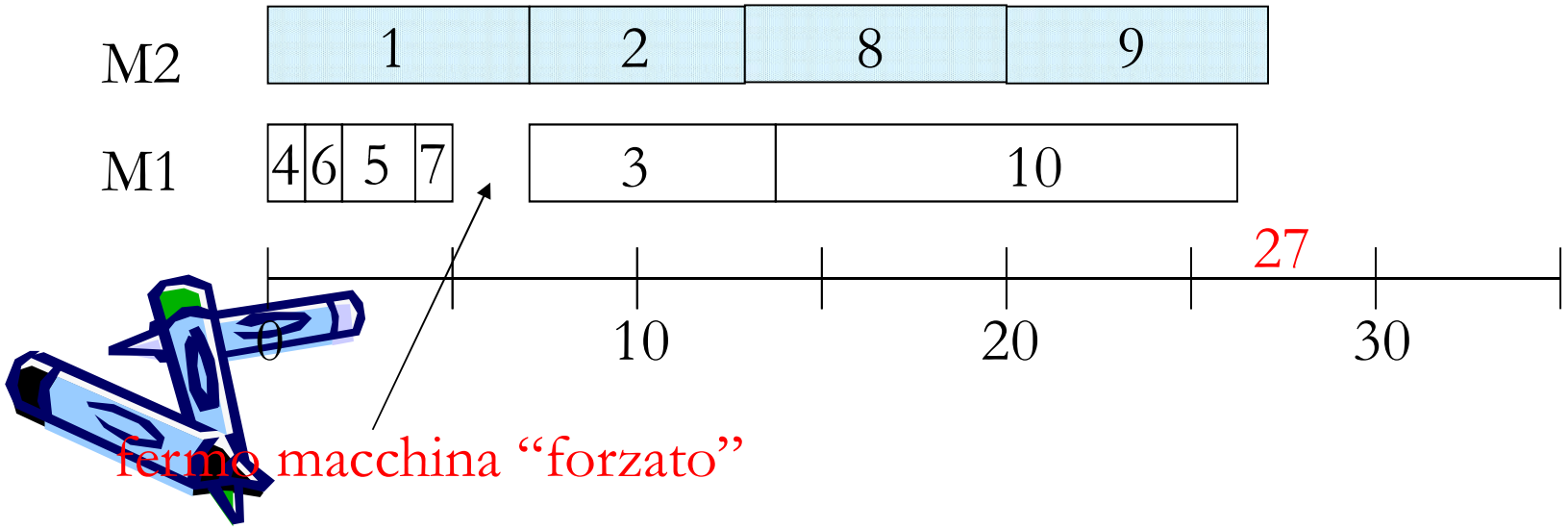

Soluzione:



# Soluzione ottima

• Nonostante l'idea di non fermare le macchine se non necessario sia ragionevole, la seguente soluzione migliora quella ottenuta dalla regola precedente:



# Scheduling su singola macchina

*Descrizione del problema*

Un insieme di *n* operazioni deve essere eseguito su una macchina

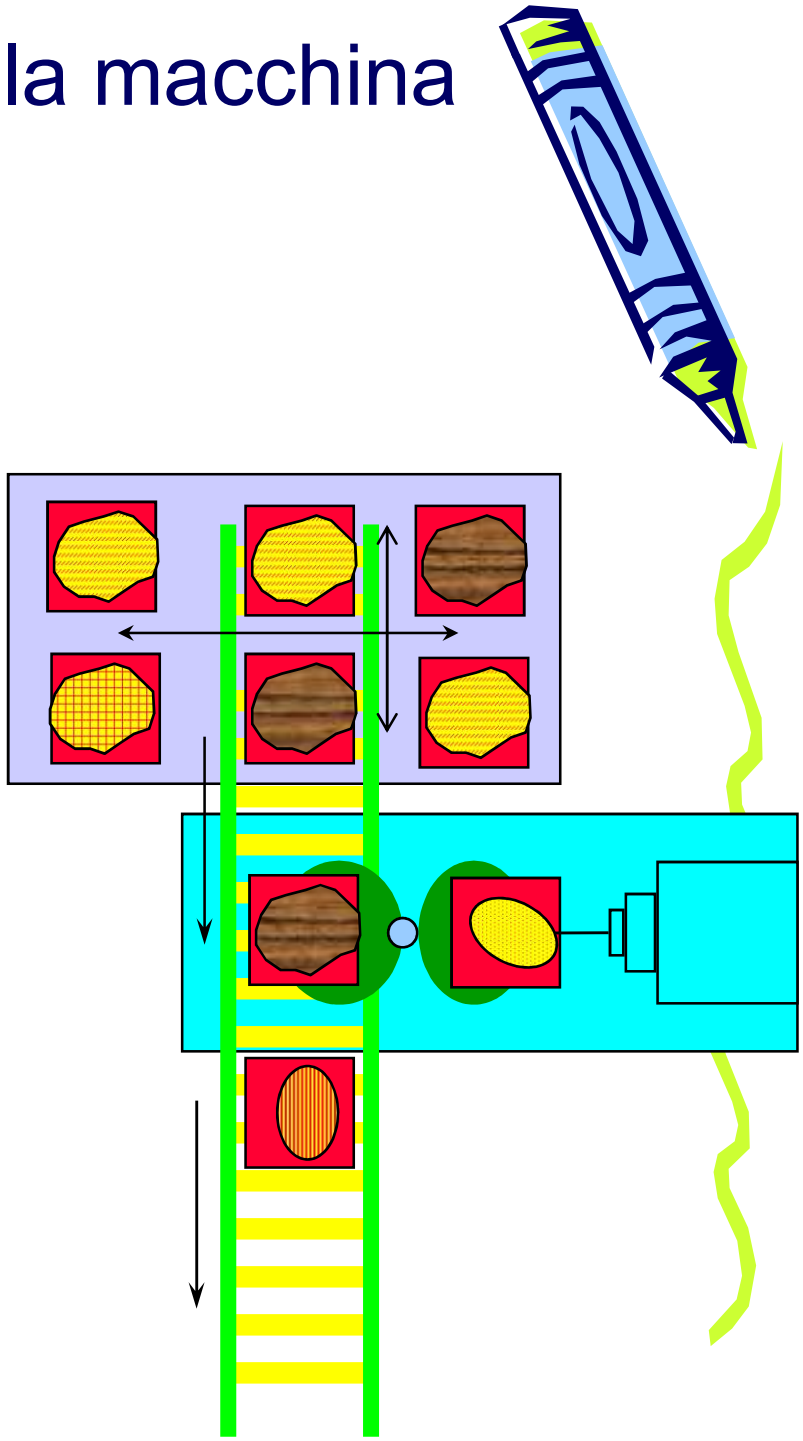
*Dati*

I tempi di *processamento* $p_i$, $i$=1,…,$n$, del lavoro $i$ sulla macchina sono noti.

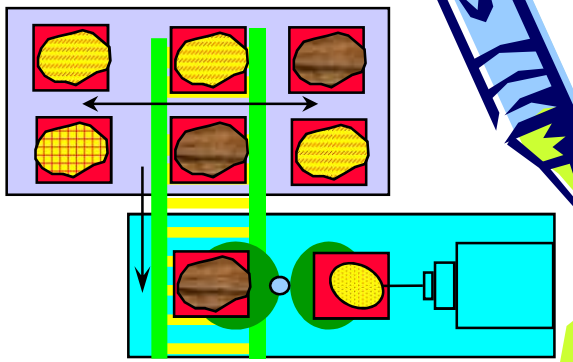
*Obiettivo*

Sequenziare le operazioni sulla macchina in modo da minimizzare la somma dei tempi di completamento.

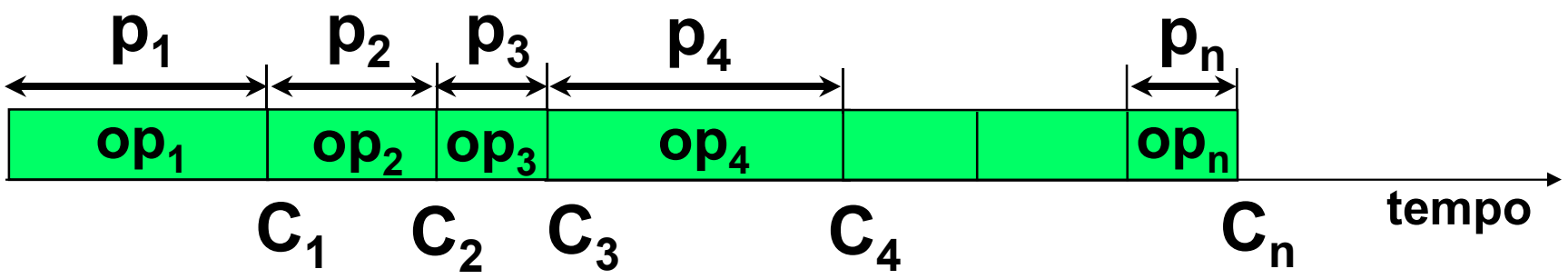
$$\min \Sigma_i C_i$$



# Gantt del Sequenziamento



Sequenza S



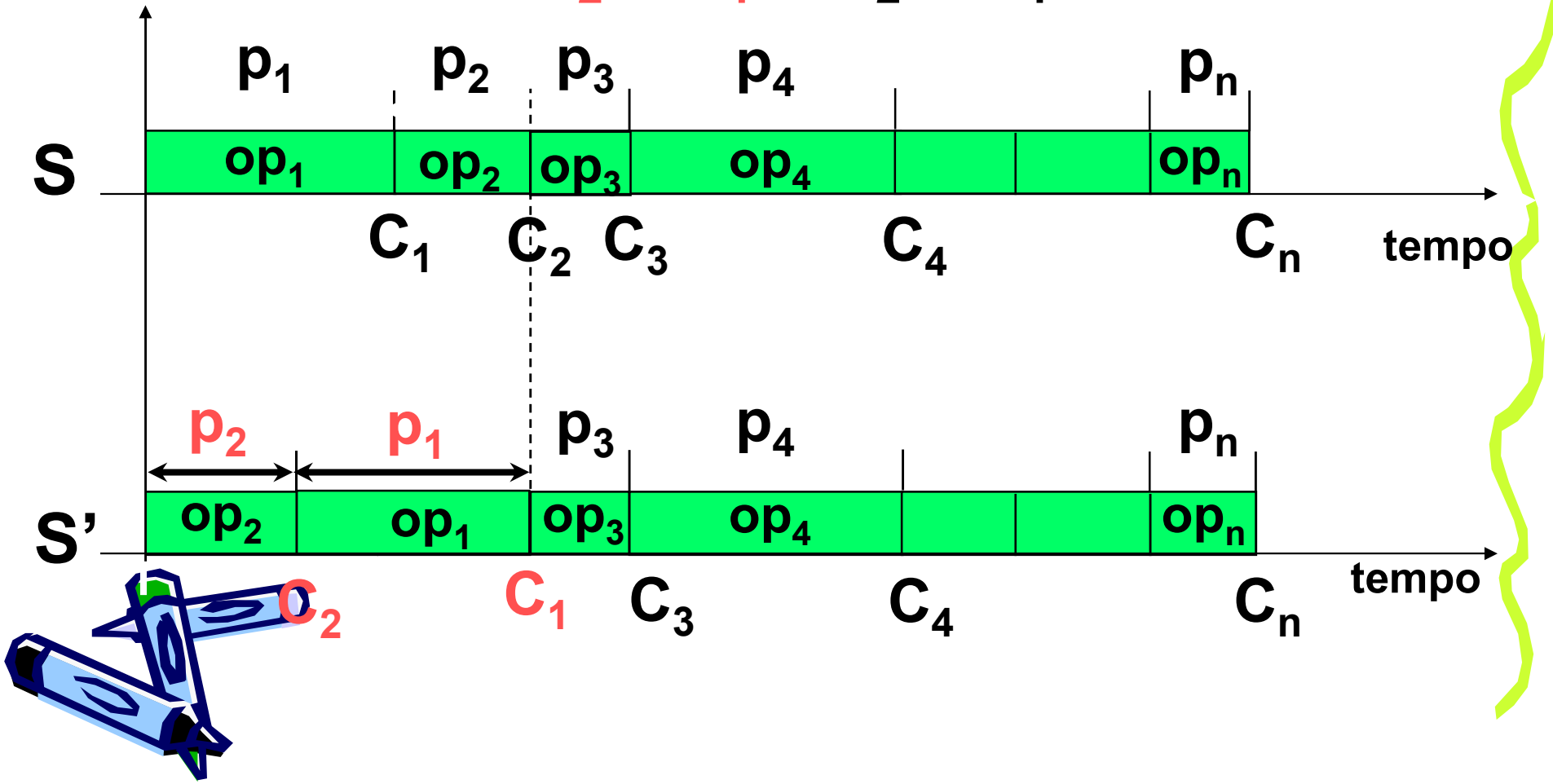

$C_n=\Sigma_i p_i$: tempo di completamento totale (*makespan*)

**Obiettivo: min $\Sigma_i C_i$**

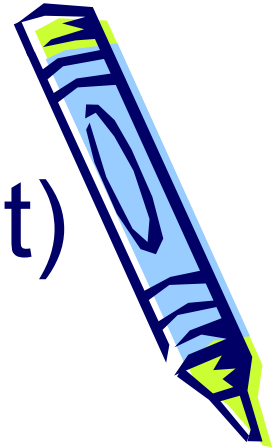
se $p_2 < p_1$ allora scambiando le op. *1* e *2* si ha

$C_2 < C_1$

$C_2 + C_1 < C_2 + C_1$



# Regola *SPT* (shortest processing time first)



*SPT*: sequenzia prima le operazioni che hanno tempo di esecuzione più piccolo

Consente di minimizzare la somma dei tempi di completamento $\Sigma_i \boldsymbol{C_i}$ di $\boldsymbol{n}$ operazioni (lavori) su una macchina

# 1.1 Tempo totale pesato di completamento

$$\Sigma w_j C_j$$

# Tempo totale pesato di completamento

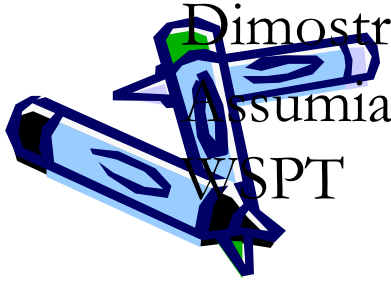
## $1//\ \Sigma w_j C_j$

Definiamo WSPT una regola che ordina i job per valori non crescenti del rapporto $w_j/p_j$

Sussiste il seguente

**Teorema 1.1** La regola WSPT calcola una soluzione ottima del problema $1//\ \Sigma w_j C_j$

Dimostrazione. (Per contraddizione)
Assumiamo che $S$ sia uno schedule ottimo e che non rispetti WSPT

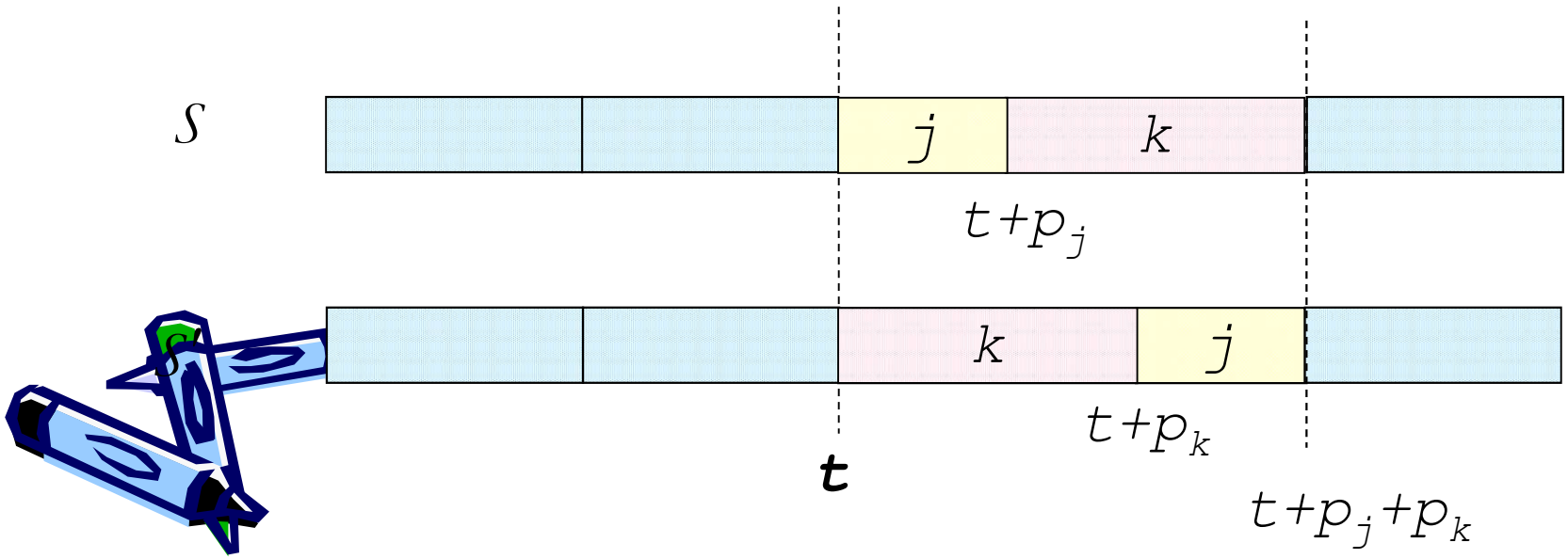


# Dimostrazione

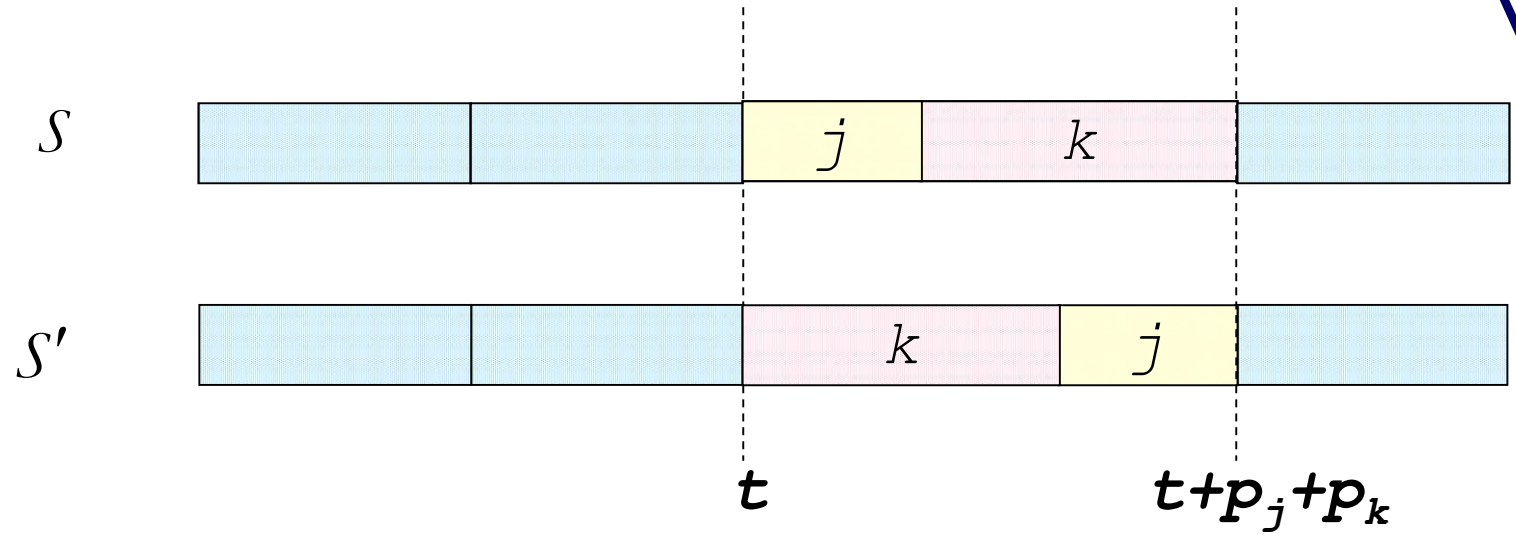
Allora devono esserci in $S$ due job adiacenti, diciamo $j$ seguito da $k$ tali che

$$w_j/p_j \; < \; w_k/p_k$$

Assumiamo che il job $j$ inizi all'istante $t$.
Eseguiamo uno scambio dei job $j$ e $k$ ottenendo un nuovo schedule $S'$



# Dimostrazione




Il tempo totale (pesato) di completamento dei job che precedono e seguono la coppia $(j,k)$ non è modificato dallo scambio. Il contributo dei job $j$ e $k$ in $S$ è:

$$(t+p_j)\,w_j \;+\; (t+p_j+p_k)\,w_k$$

mentre in $S'$ è:

$$(t+p_k)\,w_k \;+\; (t+p_k+p_j)\,w_j$$

# Dimostrazione

in $S$] $(t+p_j)w_j + (t+p_j+p_k)w_k$

in S'] $(t+p_k)w_k + (t+p_k+p_j)w_j$

Eliminando i termini uguali:

in $S$] $p_j w_k$

in S'] $p_k w_j$

Quindi, se $w_j/p_j < w_k/p_k$, risulta $p_k\ w_j < p_j\ w_k$, cioè il tempo totale pesato in $S'$ è strettamente inferiore a quello in $S$, contraddizione

□