Parte 3: Gestione dei progetti, Shop scheduling

Rappresentazione reticolare di un progetto

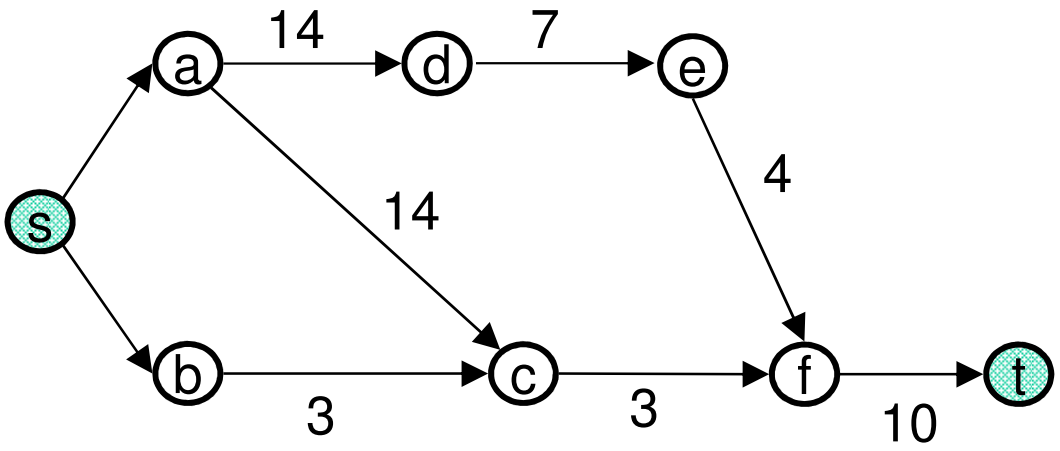
- Insieme di attività {1,...,*n*}
- p_i durata (nota e deterministica dell'attività i)
- relazione di precedenza fra attività: Grafo $G = (V, A)$ diretto (**Activity on Node**) **aciclico**

- aggiungiamo due nodi fittizi s e t a rappresentare l'inizio e la fine del progetto; inseriamo gli archi:
  - (s, j) per ogni nodo j che non ha archi entranti
  - (i, t) per ogni nodo i che non ha archi uscenti

- associamo una lunghezza $l_{ij} = p_i$ a ciascun arco $(i, j) \in A$

Durata del progetto

attività	durata	pred.
a	14	-
b	3	-
c	3	a,b
d	7	a
e	4	d
f	10	c,e



Rappresentiamo un **schedule** con un vettore $[y_1, ..., y_n]$ (istanti di inizio delle attività)

La durata del progetto è quindi pari a $y_t - y_s$

Minimizzare la durata del progetto (*makespan*)

Formulazione PL:

$$\min \quad y_t - y_s$$

$$y_j - y_i \geq l_{ij} \qquad (i,j) \in A$$

vincoli di precedenza

~~$y_i \geq 0$~~ $\qquad i \in V$

---

una sol. con qualche $y_i < 0$ può essere scalata, rimanendo ammissibile e di pari valore

Problema duale

$$\max \quad \sum_{(i,j)\in A} l_{ij} x_{ij}$$

$$\sum_{(j,i)\in \delta^-(i)} x_{ji} - \sum_{(i,j)\in \delta^+(i)} x_{ij} = 0 \qquad i \in V \setminus \{s,t\}$$

$$-\sum_{(s,j)\in \delta^+(s)} x_{sj} = -1$$

$$\sum_{(j,t)\in \delta^-(t)} x_{jt} = 1$$

$$x_{ij} \geq 0 \qquad (i,j) \in A$$

problema del **cammino di lunghezza massima da *s* a *t* in G**

Makespan e cammini massimi

- per il teorema della dualità forte il makespan equivale alla lunghezza del cammino massimo in G
- l'albero dei cammini massimi in un grafo diretto aciclico si calcola in tempo O(m)
- per ottenerlo si inizia con il calcolo di un **ordinamento topologico** dei nodi
- un vettore $e(.)$ di interi definisce un ordinamento topologico dei nodi di G se risulta $e(i) < e(j)$ per ogni arco $(i,j) \in A$

Ordinamento topologico

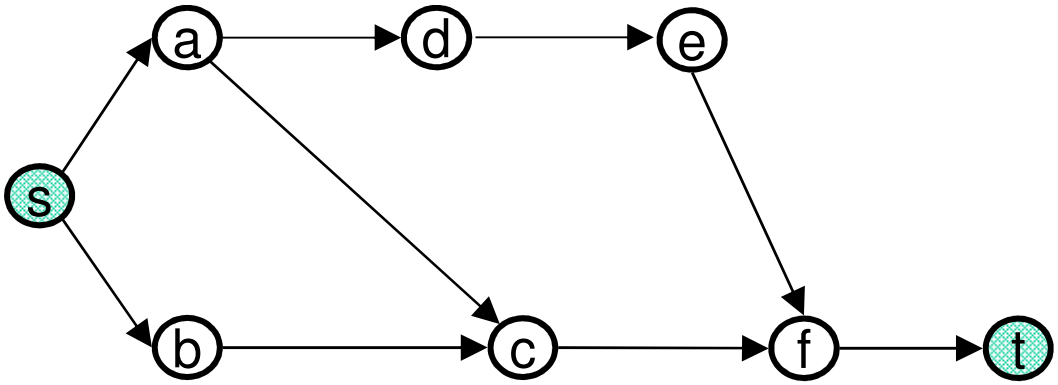
j:=0
repeat
- j := j + 1
- Sceglie un nodo v con grado entrante pari a zero
- Assegna a v l'etichetta $e(v) := j$
- Elimina v e i suoi archi uscenti

until (esiste qualche nodo con grado entrante nullo)

Se la rete rimanente contiene nodi ed archi allora G contiene un ciclo diretto

Altrimenti le etichette definiscono un ordinamento topologico

Esempio



v	e(v)
s	1
a	2
b	5
c	6
d	3
e	4
f	7
t	8

L'algoritmo ha complessità O(m) [mantenendo il grado dei nodi e la lista dei nodi a grado entrante nullo]

Albero dei cammini massimi

- dato un ordinamento topologico (etichette e(.))
- d(.) vettore delle distanze
- p(.) vettore dei predecessori

Init. $d(\mathrm{s}) = 0,\ d(\mathrm{v}) = -\infty,\ \mathrm{p_v} = \mathrm{NULL},\ v \in V.$

loop: **for** $(j = 1 \text{ to } |V|)\{$

u: $e(u) = j$

SCAN(u): **forall** v tale che $(u,v) \in \mathrm{A}$

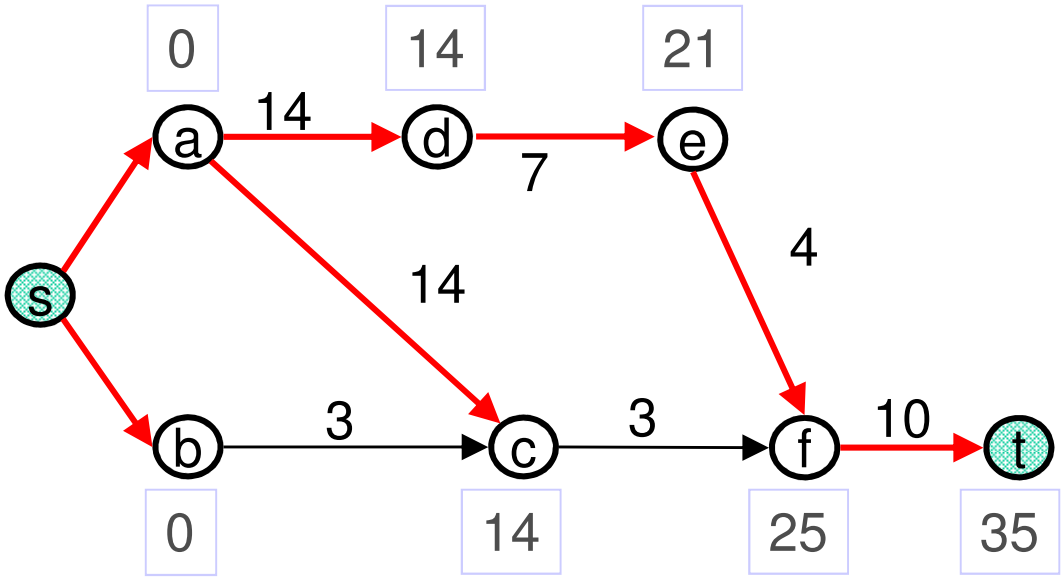
if $(d(v) < d(u) + l_{uv})$

$d_v := d_u + l_{uv};$

$p_v := u;$

$\}$

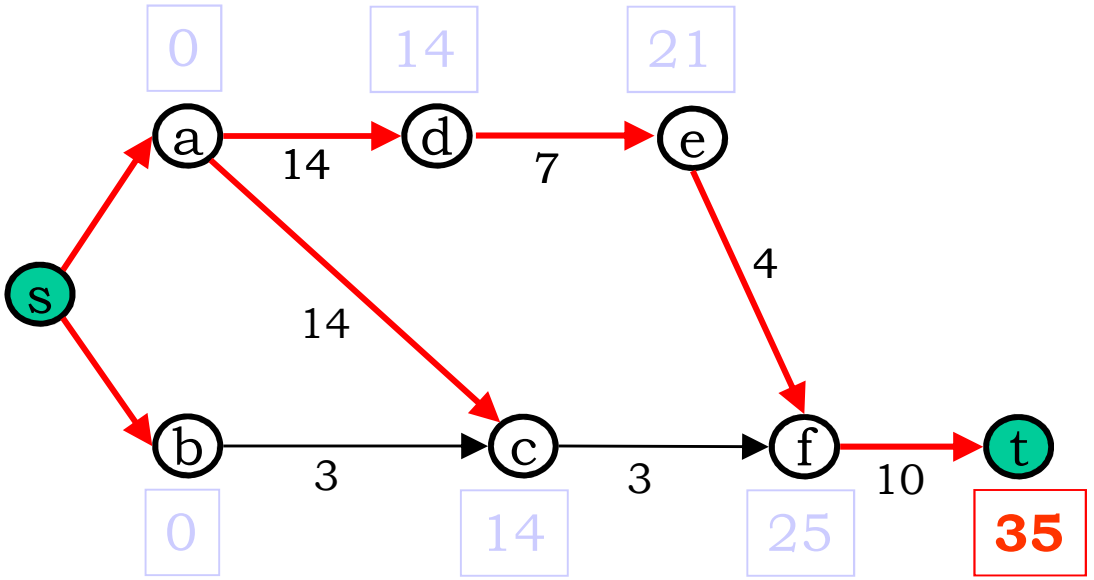
Esempio



	d	*p*
s	0	-
a	0	s
b	0	s
c	14	a
d	14	a
e	21	d
f	25	e
t	35	f

Earliest start time

- ponendo $y_v = d_v$, $v \in V$, si ha che y_v rappresenta il minimo istante di inizio per l'attività v tale da rispettare i vincoli di precedenza.
- Questo è detto **Earliest Start Time**, EST(v)
- EST(t) equivale quindi al minimo makespan



Cammino critico

- Ogni cammino di lunghezza massima da *s* a *t* è detto **cammino critico**
- le attività apparenenti ad un cammino critico sono dette **attività critiche**: un ritardo in una di esse provoca un ritardo dell'intero progetto

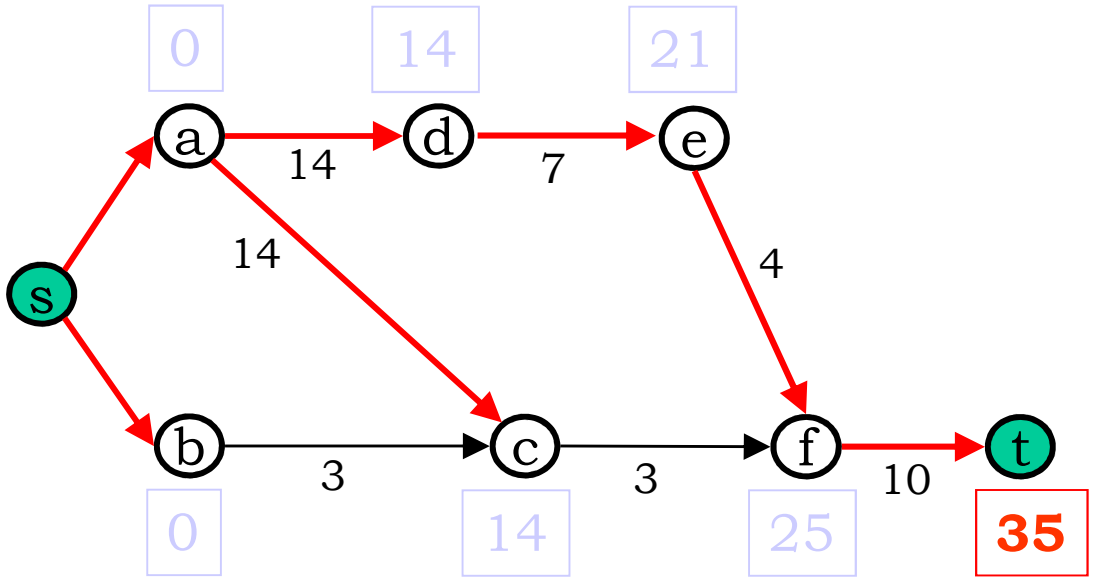
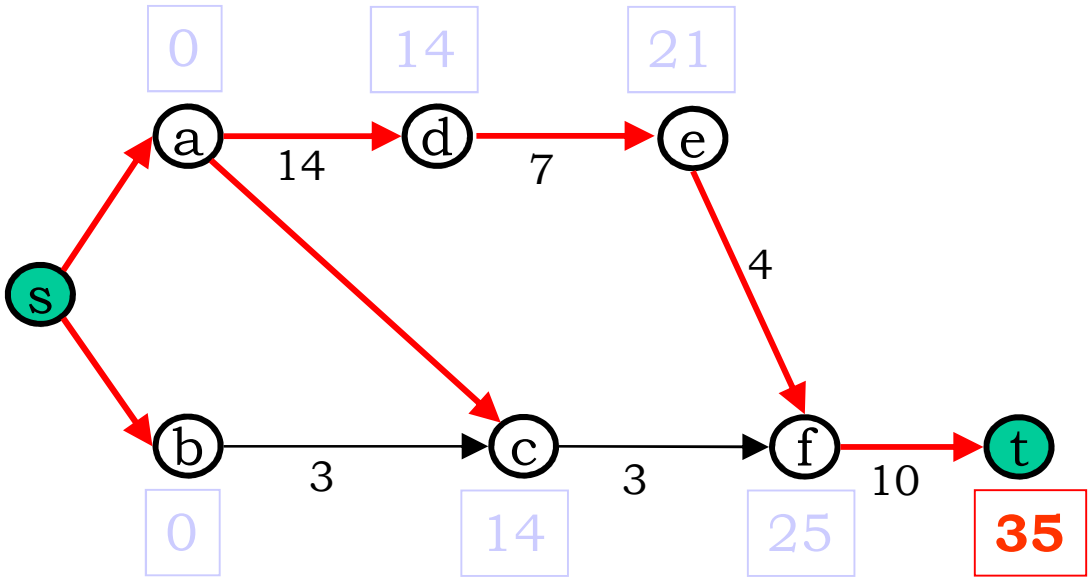
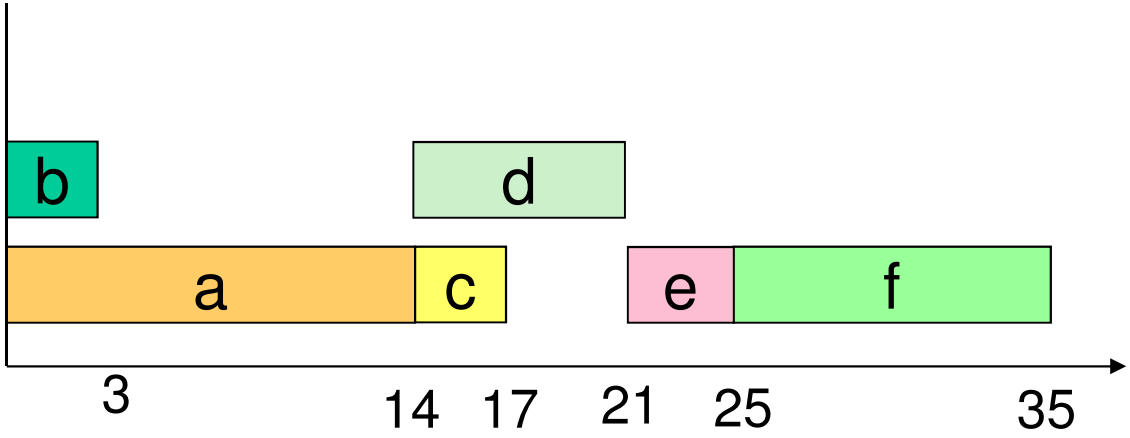


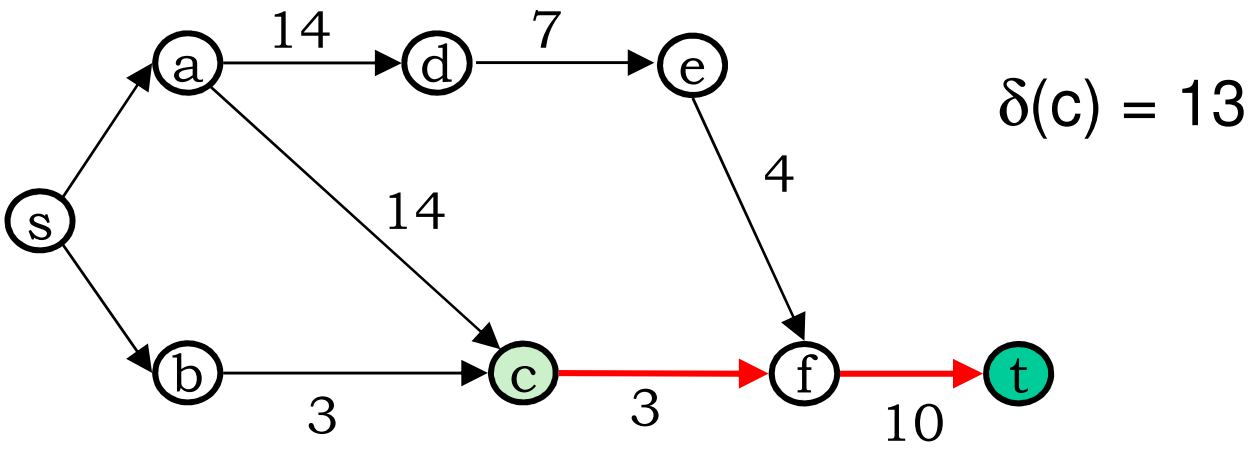
Diagramma di Gantt





Latest start time

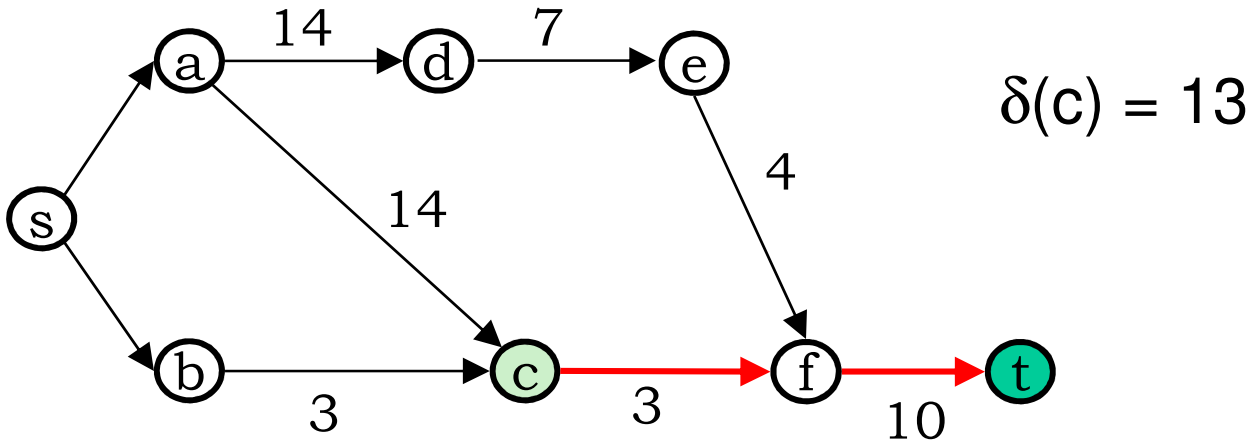
- Tipicamente, il progetto deve essere terminato entro una **data di consegna T**
- ciò vincola ogni attività v ad essere iniziata entro un certo istante, detto **Latest Start Time**, LST(v)
- Se indichiamo $\delta(v)$ con la lunghezza del cammino massimo da v a t, si ha che LST(v) = T – $\delta(v)$



$\delta(c) = 13$

Total float

- La quantità LST(*v*) – EST(*v*) è detta **total float** TF(v)



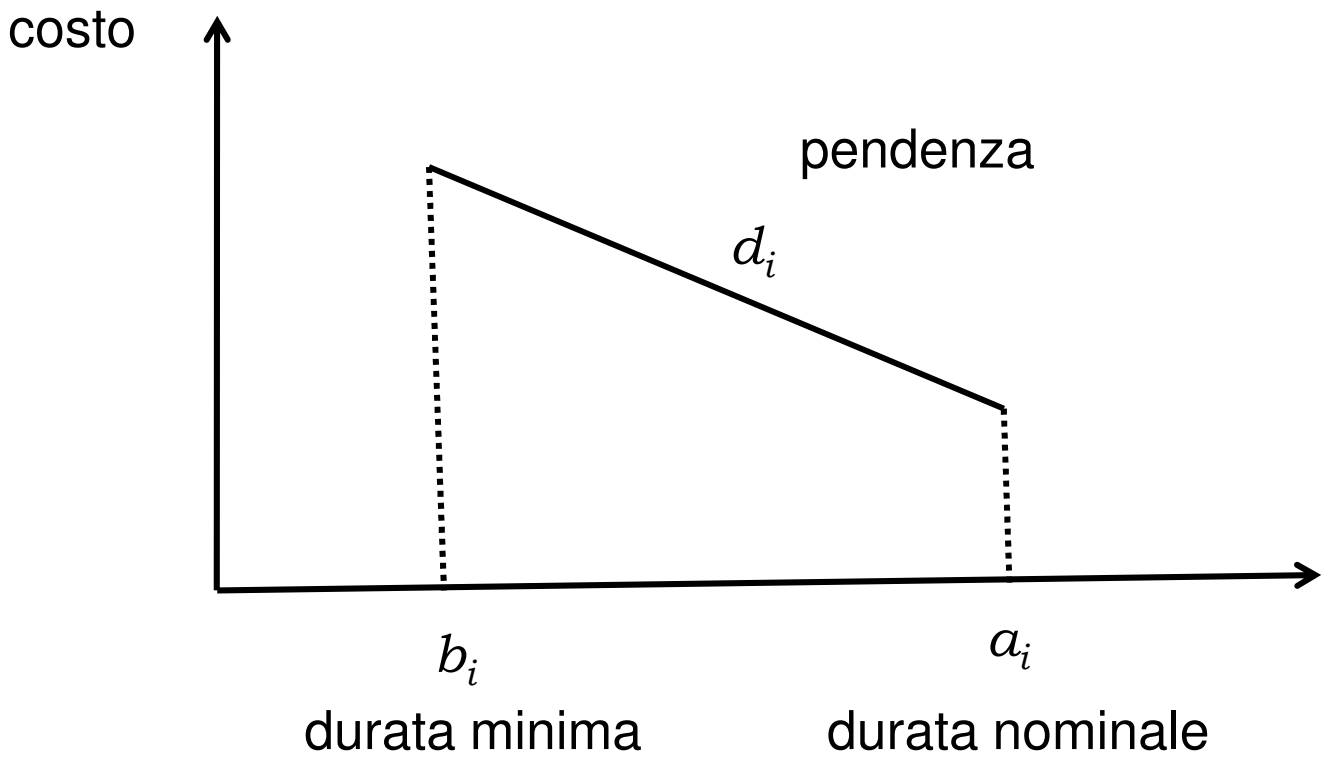
$\delta(c) = 13$

T = 29

EST(c) = 14 LST(c) = 16 TF(c) = 2

Trade-off tempi-costi

In molti casi, una maggiore allocazione di risorse su una certa attività permette di ridurne la durata.



Formulazione

variabili decisionali:

y_i istante di inizio dell'attività i,

β_i riduzione del tempo di processamento dell'attività i

Problema: determinare uno schedule tale che l'intero progetto abbia durata non superiore a λ

$$\min \quad \sum_{i \in N} d_i \beta_i$$

$$y_t - y_s \leq \lambda$$

$$y_j - y_i \geq a_i - \beta_i \qquad \forall (i,j) \in A$$

$$0 \leq \beta_i \leq a_i - b_i \qquad i \in V$$

Formulazione

$$\min \quad \sum_{i \in N} d_i y_i - \sum_{i \in N} d_i \beta'_i$$

$$y_t - y_s \leq \lambda$$

$$y_j - \beta'_i \geq a_i \qquad \forall (i,j) \in A$$

$$y_i - \beta'_i \geq 0 \qquad \forall i \in N$$

$$y_i - \beta'_i \leq a_i - b_i \qquad i \in N$$

y_i, β'_i non vincolate

sostituzione di variabili:

$\beta'_i = y_i - \beta_i$

ogni riga ha al più un +1 ed un –1, quindi è il duale di un problema di **flusso a costo minimo**

Job Shop Scheduling

- insiemi $J = \{1,...,n\}$ di job, $M=\{1,...,m\}$ di macchine
- Ogni job richiede una fissata sequenza di operazioni, ciascuna eseguita su una specifica macchina (con capacità unitaria e buffer di ingresso); cioè, visita le macchine secondo un dato ***instradamento***
- Un job visita una macchina al più una volta (no-ricircolo)

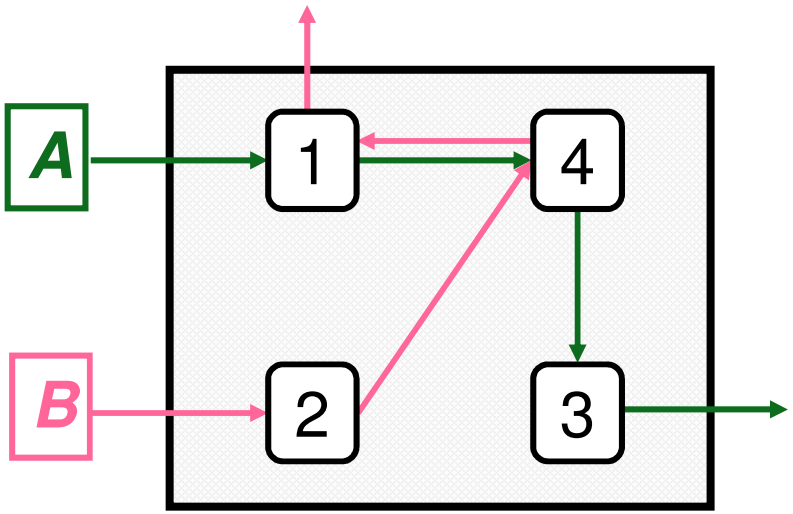
Problema: decidere la sequenza di processamento delle operazioni su ciascuna macchina in modo da **minimizzare il makespan**

Notazione

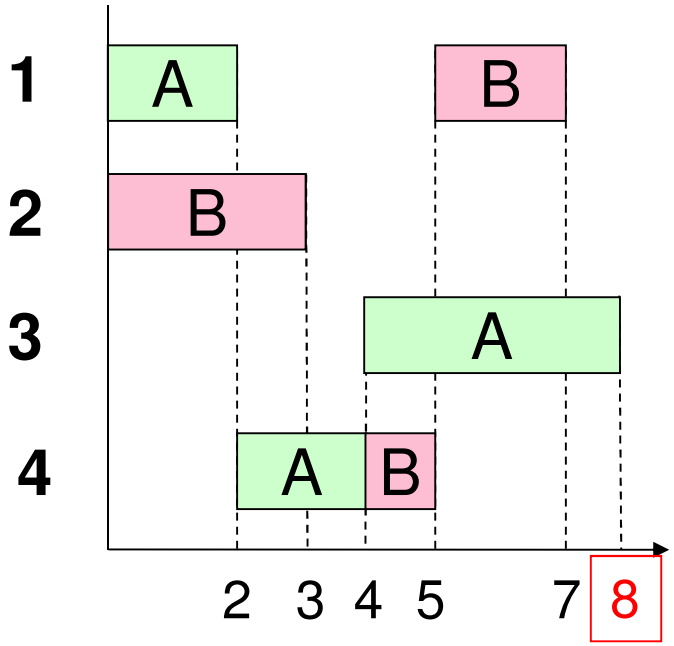
- (i, j), $i \in M$, $j \in J$ operazione
- p_{ij} durata dell'operazione (i, j)
- N insieme di tutte le operazioni (se ogni job visita tutte le macchine esattamente una volta $|N| = nm$)
- m_j numero di macchine visitate dal job j
- $(i(j,1),\ldots,i(j,m_j))$ instradamento di j (sequenza delle macchine che visita)
- $J(i)$ insieme dei job che visitano la macchina i

Esempio

job	i(j,1) $[p_{i(j,1),j}]$	i(j,2) $[p_{i(j,2),j}]$	i(j,3) $[p_{i(j,3),j}]$
A	1 [2]	4 [2]	3 [4]
B	2 [3]	4 [1]	1 [2]



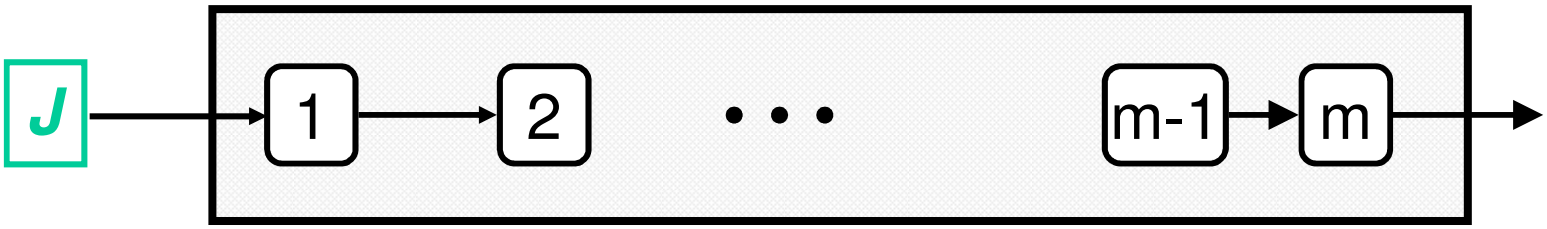
Una soluzione:
- A precede B sulla macchina 1
- A precede B sulla macchina 4



se B precedesse A sulla macchina 1 ?

Caso particolare: Flow Shop

- Tutti i job hanno la stessa sequenza (1, 2,…, m) di visita delle macchine



- Il buffer di ingresso consente i sorpassi: la sequenza di processamento della macchina $i + 1$ può essere diversa da quella sulla macchina i. Il numero di soluzioni è pari a $(n!)^m$
- Nel caso senza buffer (*no-wait*) ci sono $n!$ soluzioni

Formulazione disgiuntiva

A insieme delle coppie
$(i(j,h), j),(i(j,h+1), j), \quad h = 1,\ldots,m_j-1,\ j=1,\ldots,n$

$t_{ij} \in R^n$ istante di inizio dell'operazione ij

$$\min C_{\max}$$

$$C_{\max} \geq t_{ij} + p_{ij} \quad (i,j) \in N$$

$$t_{rj} - t_{ij} \geq p_{ij} \quad (i,j),(r,j) \in A$$

$$t_{ij} - t_{ik} \geq p_{ik} \quad or \quad t_{ik} - t_{ij} \geq p_{ij} \quad (i,j),(i,k) \in N$$

$$t_{ij} \geq 0 \quad (i,j) \in N$$

Formulazione PLM

$$y_{jk}^{i} = \begin{cases} 1 & \text{Se il job } j \text{ precede } k \text{ sulla macchina } i \\ 0 & \text{altrimenti} \end{cases}$$

$$\min C_{\max}$$

$$C_{\max} \geq t_{ij} + p_{ij} \quad (i,j),(r,j) \in N$$

$$t_{rj} - t_{ij} \geq p_{ij} \quad (i,j),(r,j) \in A$$

$$t_{ik} - t_{ij} + M(1 - y_{jk}^{i}) \geq p_{ij} \quad i \in M,(i,j),(i,k) \in N$$

$$t_{ij} - t_{ik} + M y_{jk}^{i} \geq p_{ik} \quad i \in M,(i,j),(i,k) \in N$$

$$t_{ij} \geq 0 \quad (i,j) \in N$$

Esempio

job	i(1) [$p_{i(1),j}$]	i(2) [$p_{i(2),j}$]	i(3) [$p_{i(3),j}$]	i(4) [$p_{i(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [6]
3	1[8]	2 [7]	4 [3]	

Grafo disgiuntivo

$G = (N \cup \{s,z\},\ \boldsymbol{A},\ D)$ grafo diretto
s (sorgente) e z (pozzo)

***A* archi congiuntivi**

rappresentano la relazione di precedenza fra le operazioni di ciascun job *j*

$$(i(j,h),\ j) \rightarrow (i(j,h+1),\ j), \qquad h = 1,\ldots,m_j - 1$$

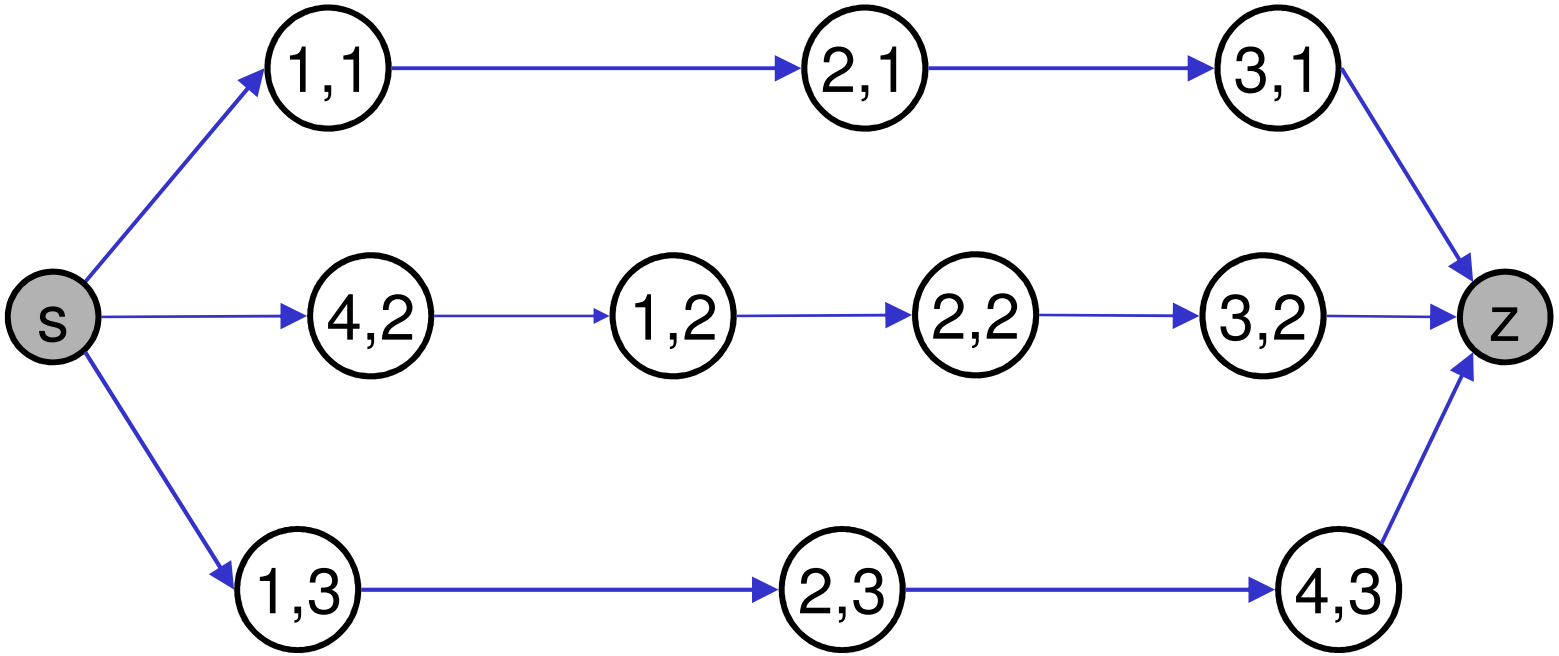
in aggiunta, per ogni $j \in J$, gli archi

$s \rightarrow (i(j,\ 1),\ j)$ dalla sorgente alle prime operazioni

$(i(j,\ m_j),\ j) \rightarrow z$ dalle ultime operazioni al pozzo

Esempio (archi congiuntivi)

job	$i(j,1)$ $[p_{i(j,1),j}]$	$i(j,2)$ $[p_{i(j,2),j}]$	$i(j,3)$ $[p_{i(j,3),j}]$	$i(j,4)$ $[p_{i(j,4),j}]$
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [6]
3	1[8]	2 [7]	4 [3]	



Archi disgiuntivi

$G = (N \cup \{s, z\}, A, D)$

Rappresentano la sequenza dei job su ciascuna macchina

per ogni coppia di operazioni da eseguirsi sulla macchina i, l' arco $(i,j) \rightarrow (i,k)$ indica che l'operazione (i,j) precede (i,k)

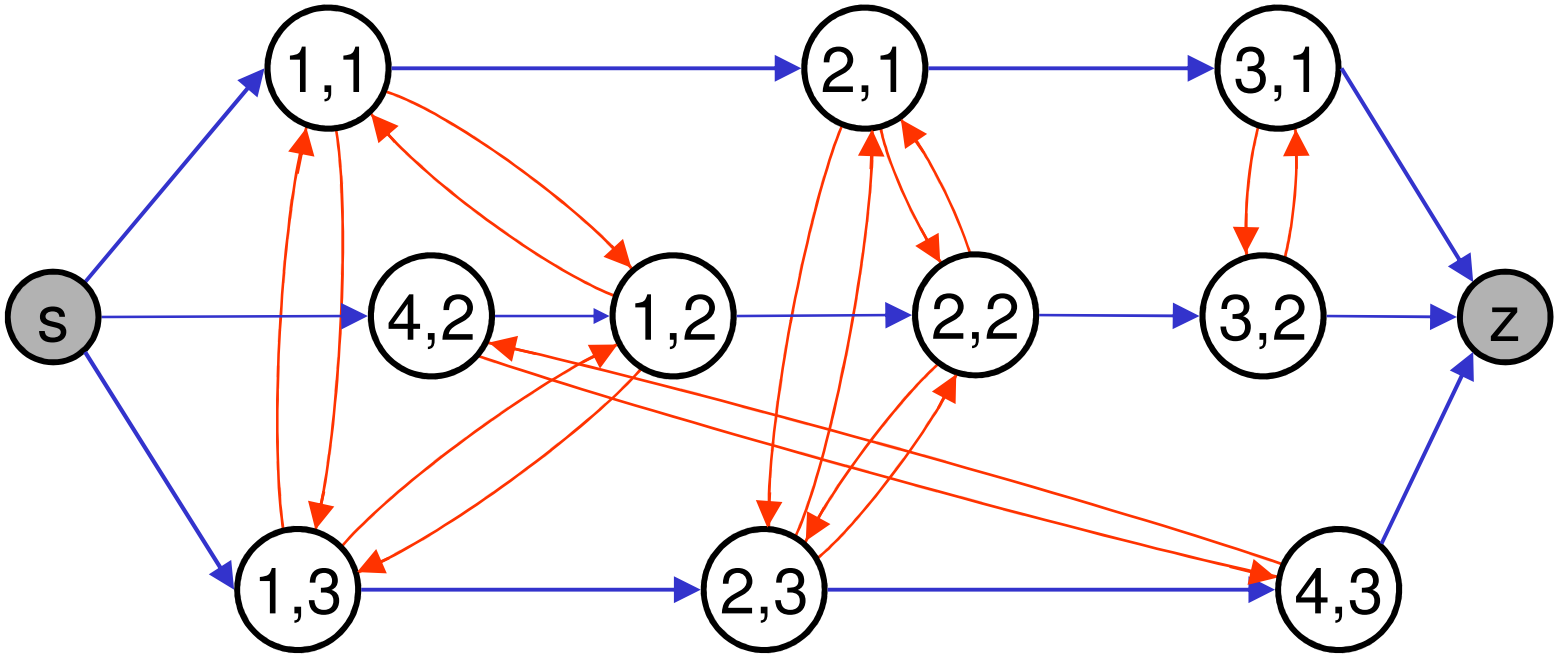
Per ogni macchina $i \in M$, e ciascuna coppia j,k di job in $J(i)$, D contiene una coppia (disgiuntiva) di archi

$$(i, j) \rightarrow (i, k) \qquad (i, k) \rightarrow (i, j)$$

$D(i)$ = {archi disgiuntivi associati alla macchina i}, inducono una clique in G

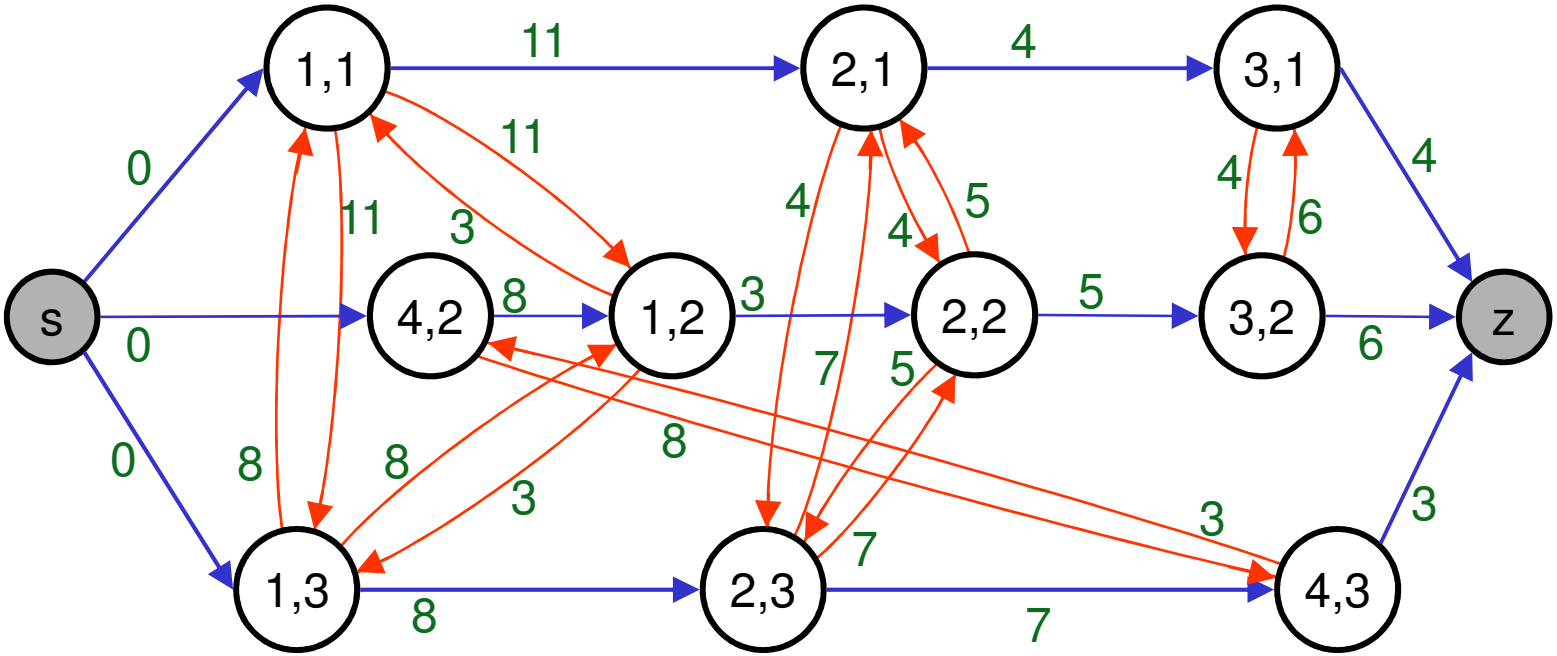
Esempio (archi disgiuntivi)

job	$i(j,1)$ $[p_{i(j,1),j}]$	$i(j,2)$ $[p_{i(j,2),j}]$	$i(j,3)$ $[p_{i(j,3),j}]$	$i(j,4)$ $[p_{i(j,4),j}]$
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [6]
3	1[8]	2 [7]	4 [3]	



Lunghezza degli archi

job	$i(j,1)$ $[p_{i(j,1),j}]$	$i(j,2)$ $[p_{i(j,2),j}]$	$i(j,3)$ $[p_{i(j,3),j}]$	$i(j,4)$ $[p_{i(j,4),j}]$
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [6]
3	1[8]	2 [7]	4 [3]	

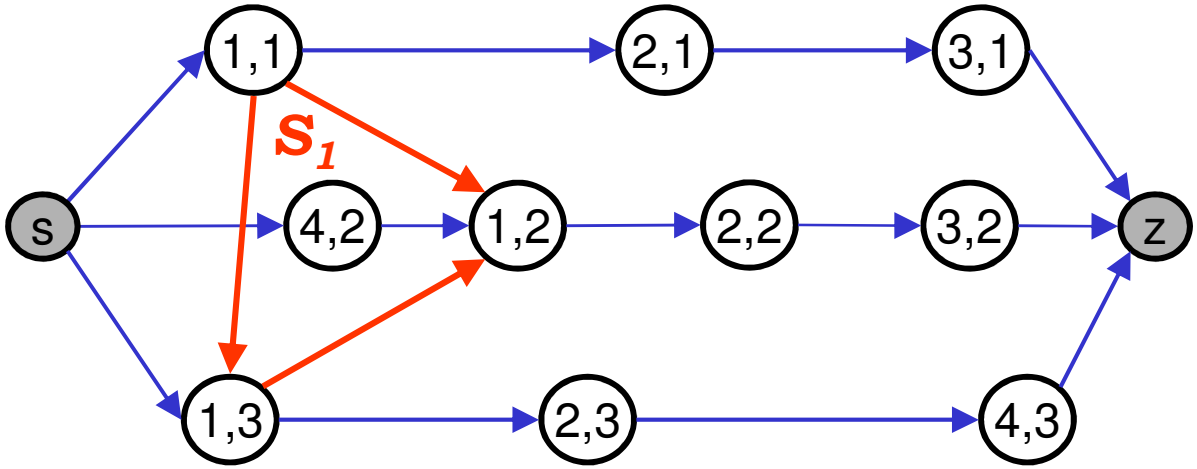


Ad ogni arco $(i,j) \rightarrow (h,k)$ si associa una lunghezza pari a p_{ij}

Selezione

Definizione. Un insieme $S_i \subseteq D_i$ che contiene esattamente un elemento per ciascuna coppia disgiuntiva è detto ***selezione*** della macchina i.

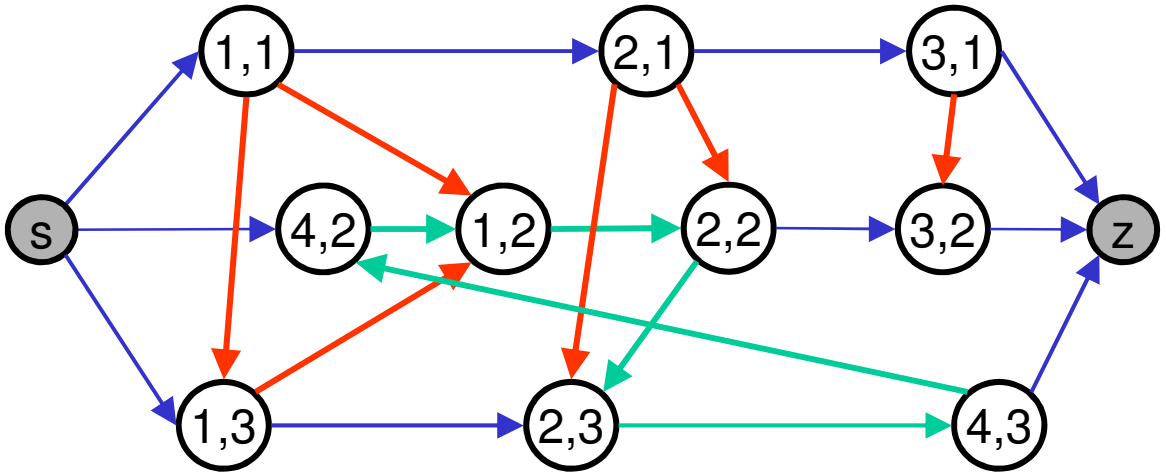
- S_i è aciclica se (il grafo parziale) non contiene cicli orientati.
- Una selezione aciclica S_i corrisponde (biunivocamente) ad un sequenziamento delle operazioni sulla macchina i.



Schedule ammissibili

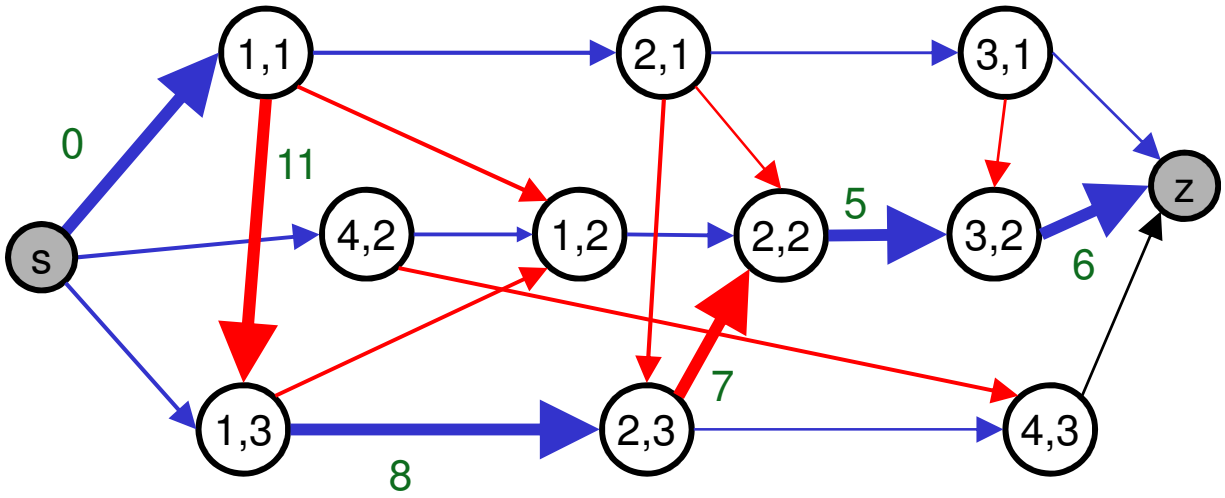
Definizione. L'unione degli archi congiuntivi e delle selezioni S_i, $i \in M$, è detta selezione completa.

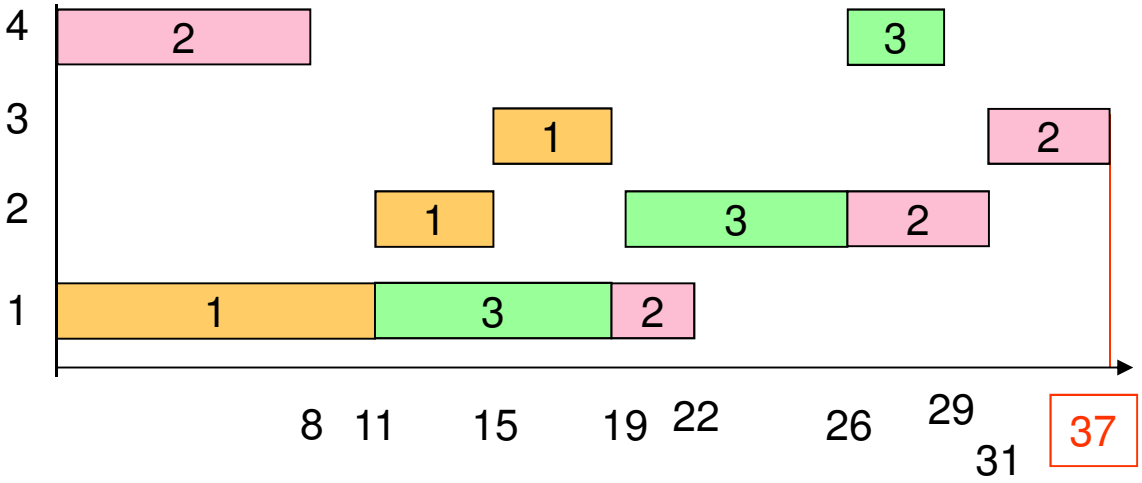
Gli schedule ammissibili corrispondono biunivocamente alle selezioni complete **acicliche**



schedule **non** ammissibile

Selezioni complete e schedule ammissibili





lunghezza del cammino critico

=

makespan

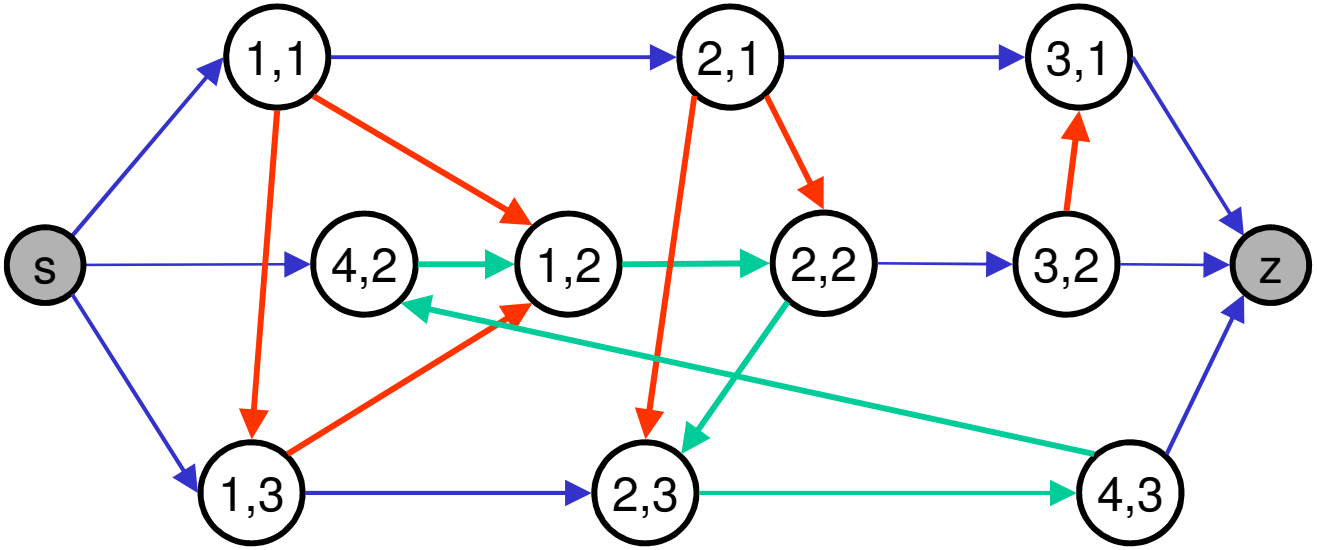
Riformulazione

Job shop scheduling: trovare una selezione completa aciclica che minimizzi la lunghezza del cammino critico

Oss. Valutare una soluzione (selezione completa) richiede la soluzione di un problema di ottimizzazione (calcolo del cammino critico)

Cicli

Una selezione completa può formare cicli anche se le selezioni S_i sono tutte acicliche



Quindi, non è immediato progettare algoritmi che decompongono il problema per macchine

Lower bound I

Supponiamo di aver scelto una selezione aciclica S_i, per $i \in M' \subset M$. Poniamo $S(M') = \cup_{i \in M'} S_i$

Definiamo $G(M')$ il grafo parziale formato dagli archi congiuntivi e dagli archi in $S(M')$

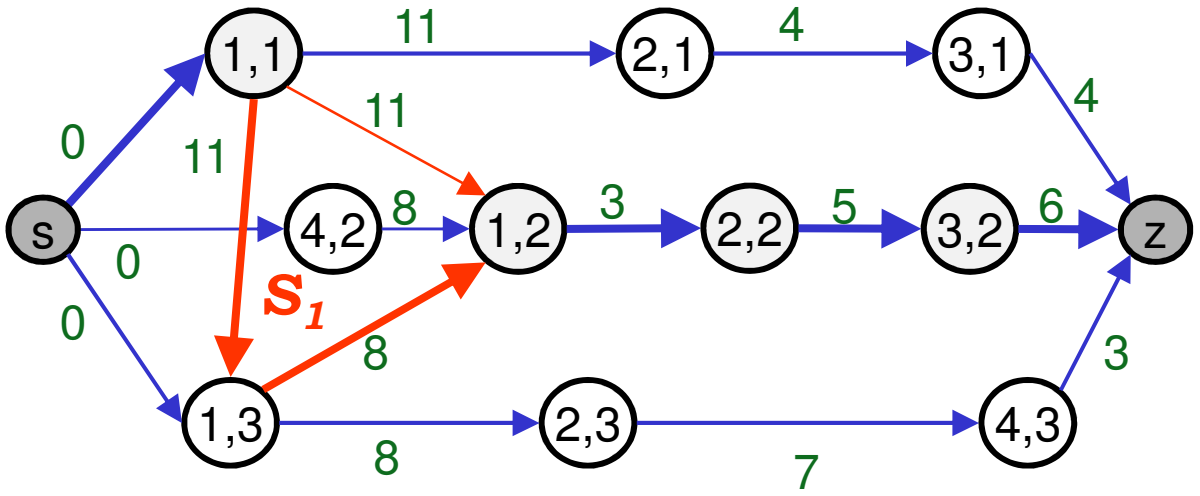
Supponiamo che $G(M')$ sia aciclico e sia $C_{max}(M')$ la lunghezza del cammino critico in $G(M')$

Proprietà. $C_{max}(M')$ è un lower bound di C_{max} quando le operazioni su ciascuna macchina $i \in M'$ sono sequenziate secondo S_i

Infatti, ciò equivale a considerare a capacità infinita le macchine in $M \setminus M'$

Esempio

$M' = \{1\}$



$C_{max}(M') = 33$

lower bound per C_{max} nel caso in cui la macchina 1 processa le operazioni secondo S_1

Lower bound II

Supponiamo di aver scelto una selezione aciclica S_i, per $i \in M' \subset M$. Poniamo $S = \cup_{i \in M'} S_i$

Teniamo adesso conto dell'effetto di sequenziare un'ulteriore macchina

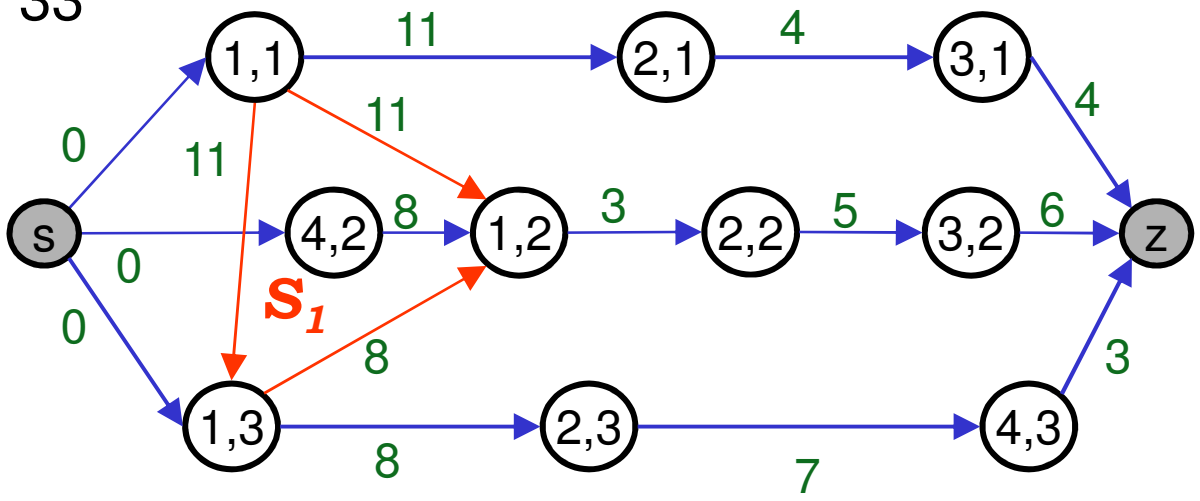
Per ogni operazione (i,j), $i \notin M'$, poniamo

- $r_{ij} = EST_{M'}(i,j)$ primo istante ammissibile per l'operazione (i,j) fissate S_i, $i \in M'$
- $d_{ij} = LST_{M'}(i,j) + p_{ij}$ due date per l'operazione (i,j) affinché $C_{max} \leq C_{max}(M')$

Esempio

$M' = \{1\}$

$C_{max}(M') = 33$



operazione	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(4,2)	(4,3)
p_{ij}	4	5	7	4	6	8	3
r_{ij}	11	22	19	15	27	0	26
d_{ij}	29	27	30	33	33	19	33

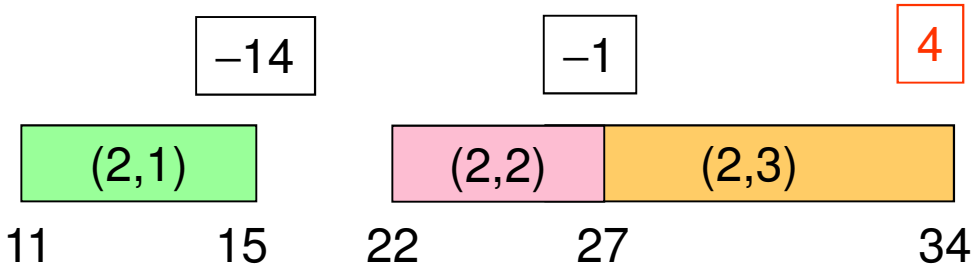
$1 / r_j / L_{max}$

- le operazioni (k,j), $j \in J(k)$ associate ad una macchina k, con release date $r_{(k,j)}$ e due date $d_{(k,j)}$, identificano un'**istanza del problema $1/r_j/L_{max}$**

- Definiamo $L_{max}(k)$ il suo valore ottimo

Esempio. $k = 2$

Op.	(2,1)	(2,2)	(2,3)
p_{ij}	4	5	7
r_{ij}	11	22	19
d_{ij}	29	27	30



$L_{max}(k) = 4$

Lower bound II

Proprietà. $C_{max}(M')$ + $L_{max}(k)$ è un lower bound di C_{max} quando le operazioni su ciascuna macchina $i \in M'$ sono sequenziate secondo S_i

Infatti, la selezione S_k è scelta in modo da minimizzare il ritardo sul makespan, mentre tutte le altre macchine ($M \setminus M' \setminus \{k\}$) sono assunte a capacità infinita

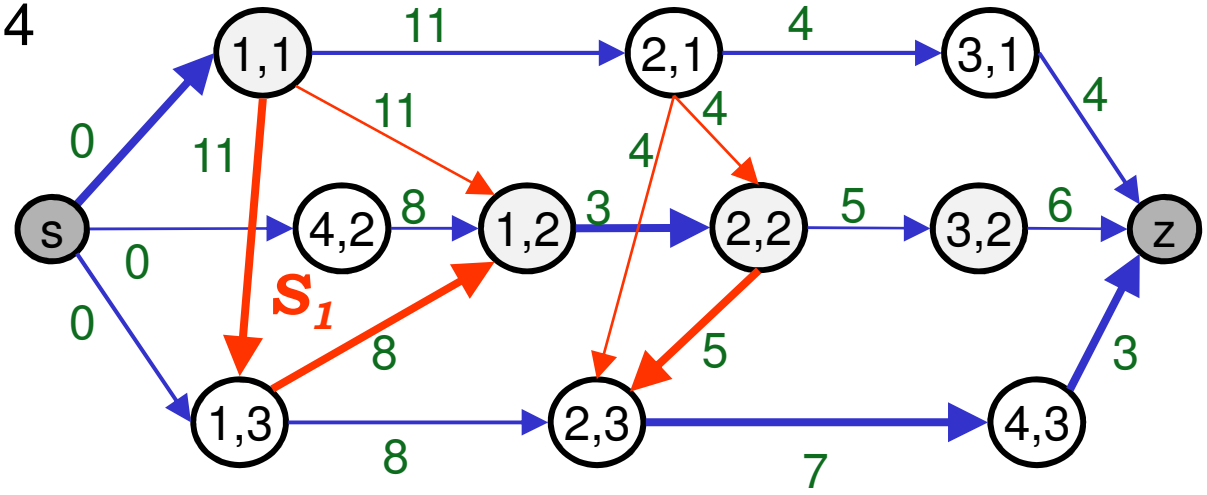
• il miglior lower bound si ottiene calcolando $L_{max}(k)$, per $k \notin M'$ e scegliendo il massimo fra questi valori:

$$\boldsymbol{L_{max}(k^*) = \max_{k \notin M'}\{L_{max}(k)\}}$$

Lower Bound II

- indichiamo con $S^*(k)$ la selezione associata alla soluzione ottima
- Aggiungendo gli archi $S^*(k)$ al grafo la lunghezza LB2 del cammino critico aumenta di $L_{max}(k)$

$L_{max}(k=2) = 4$



$C_{max}(M') = 33$ ⇨ $LB2 = 37$

Euristica *Shifting Bottleneck*

• Costruisce una selezione completa aciclica costruendo iterativamente selezioni parziali acicliche

M_0 insieme delle macchine già sequenziate
inizializzazione $M_0 = \emptyset$

Passo generico:

- Costruisce il grafo $G(M_0)$ e calcola $C_{max}(M_0)$
- sceglie la **macchina "critica"** (*bottleneck*)

$$k^* = \text{argmax}_{k \notin M_0}\{L_{max}(k)\}$$

- fissa (definitivamente) la selezione $S^*(k^*)$
- $M_0 := M_0 \cup \{k^*\}$

Validità

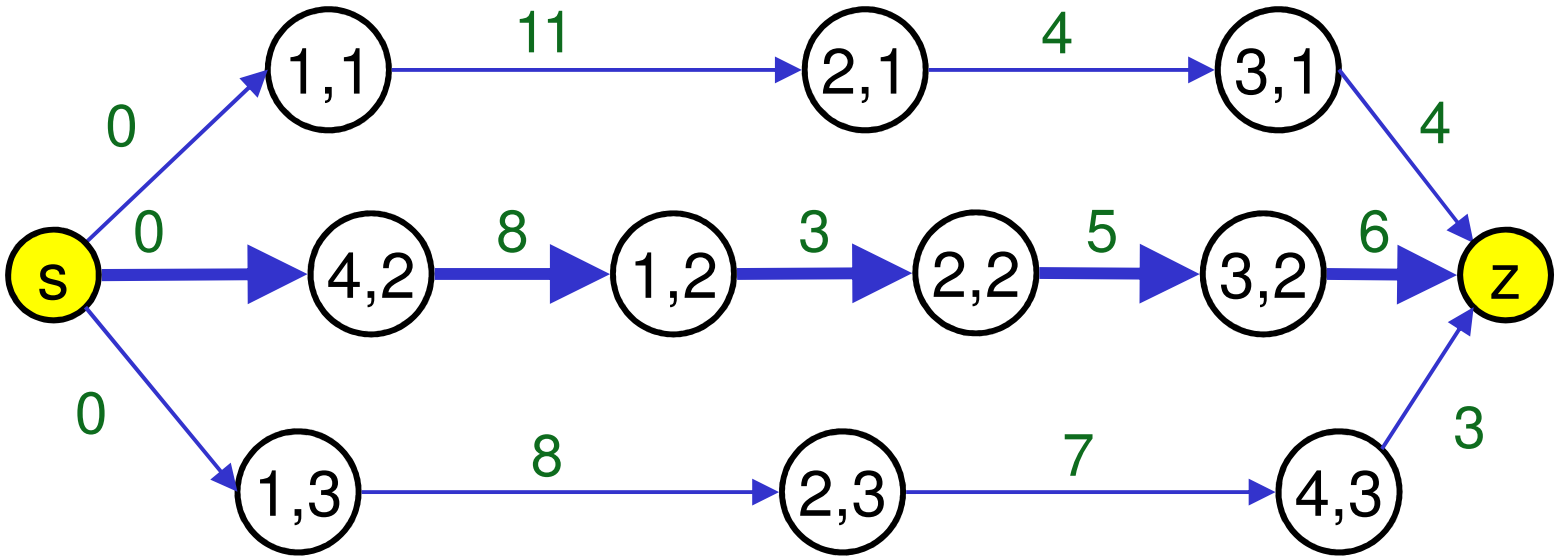
Teorema. Gli archi orientati associati alla soluzione ottima di un problema a macchina singola non creano cicli nel grafo $G(M_0)$.

Conseguenza: la selezione completa ottenuta dalle selezioni acicliche generate nelle singole iterazioni è aciclica.

Esempio

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [6]
3	1[8]	2 [7]	4 [3]	

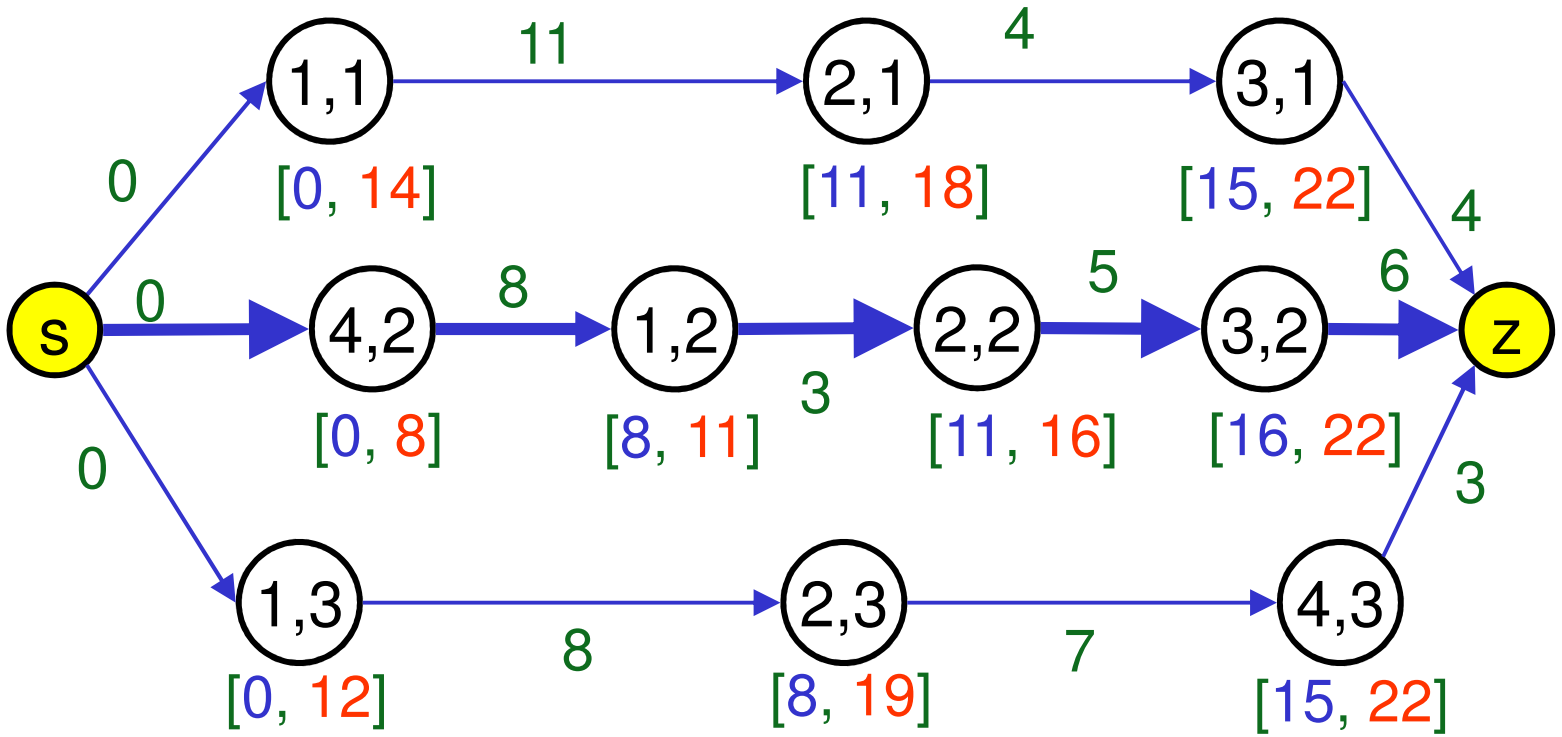
init. $M_0 = \varnothing$ $G(M_0)$



Esempio

Iter 1. Calcola C_{max}, r_{ij}, d_{ij}

$C_{max}(M_0) = 22$



Iter 1: problemi $1/r_j/L_{max}$

1

Job	1	2	3
p_j	11	3	8
r_j	0	8	0
d_j	14	11	12

2

Job	1	2	3
p_j	4	5	7
r_j	11	11	8
d_j	18	16	19

3

Job	1	2
p_j	4	6
r_j	15	16
d_j	22	22

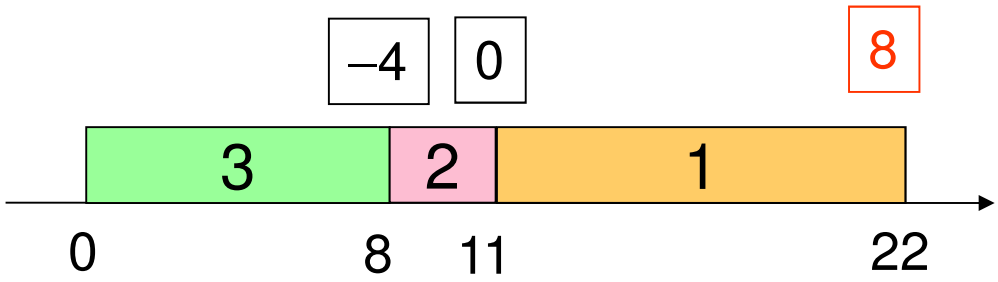
4

Job	2	3
p_j	8	3
r_j	0	15
d_j	8	22

$1/r_j/L_{max}$ macchina 1

Job	1	2	3
p_j	11	3	8
r_j	0	8	0
d_j	14	11	12

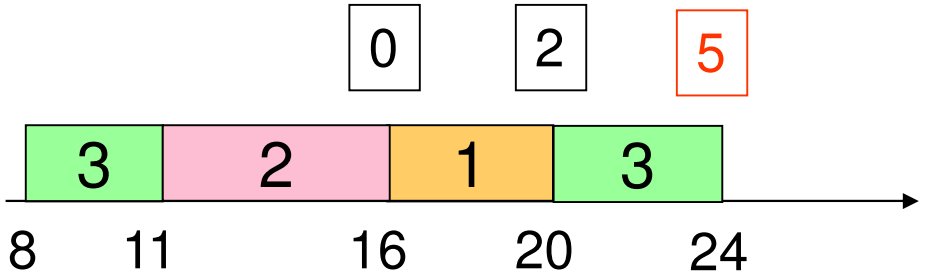
PEDD restituisce la soluzione ammissibile:



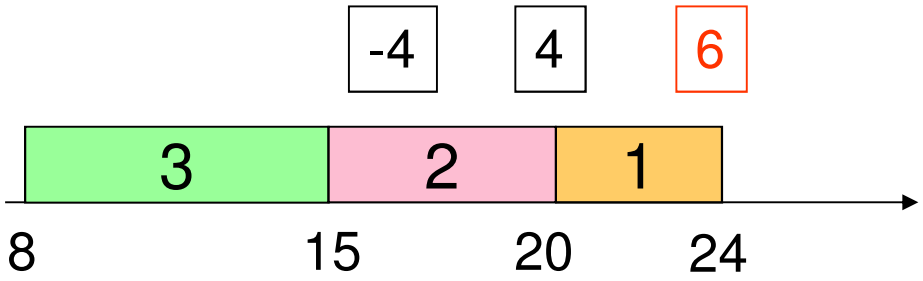
$1/r_j/L_{max}$ macchina 2

Job	1	2	3
p_j	4	5	7
r_j	11	11	8
d_j	18	16	19

PEDD restituisce la soluzione non ammissibile:



Soluzione ottima:

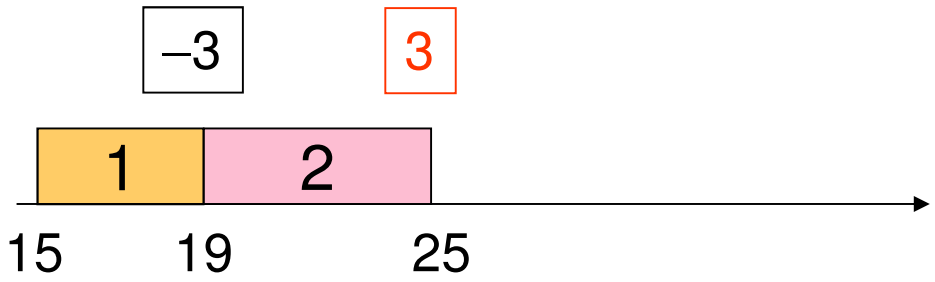


$L_{max} = 6$

$1/r_j/L_{max}$ macchina 3

Job	1	2
p_j	4	6
r_j	15	16
d_j	22	22

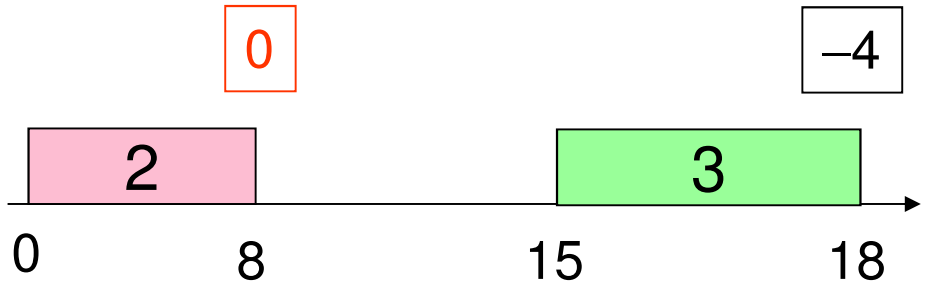
Soluzione ottima:



$1/r_j/L_{max}$ macchina 4

Job	2	3
p_j	8	3
r_j	0	15
d_j	8	22

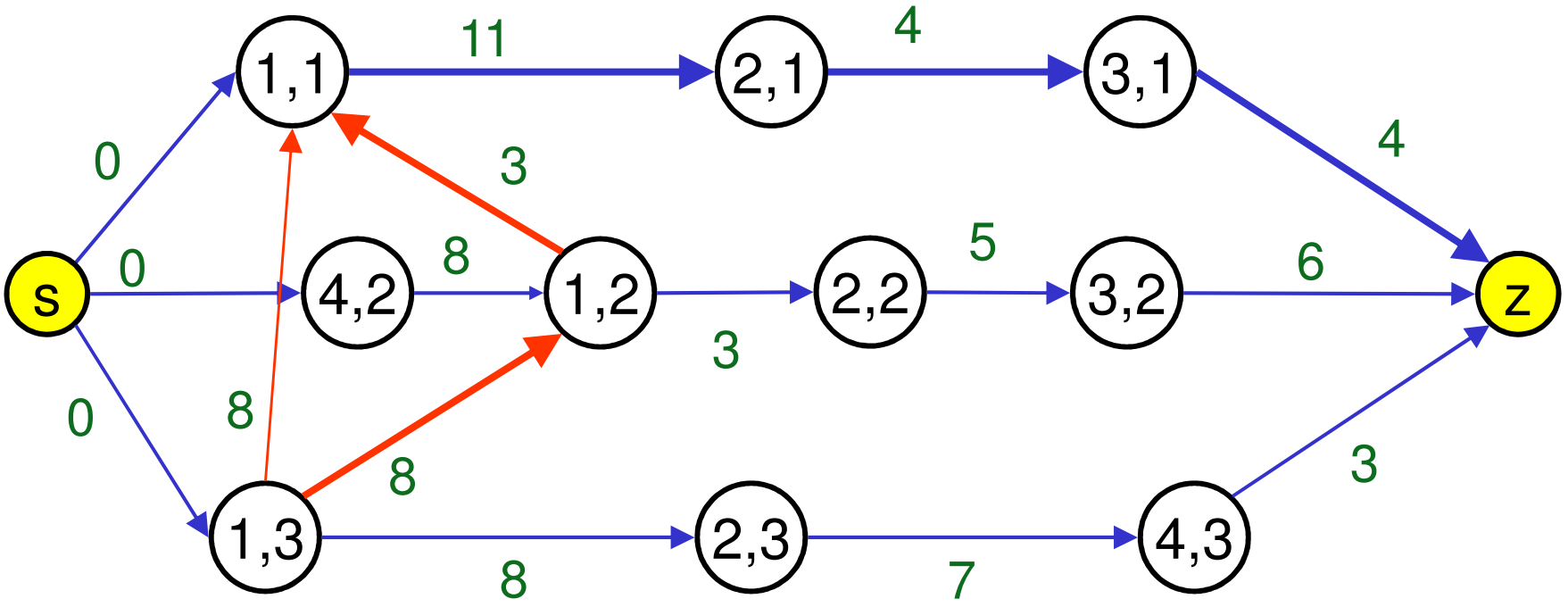
Soluzione ottima



iter. 2

Iter 2. Calcola C_{max}

$M_0 = \{1\}$

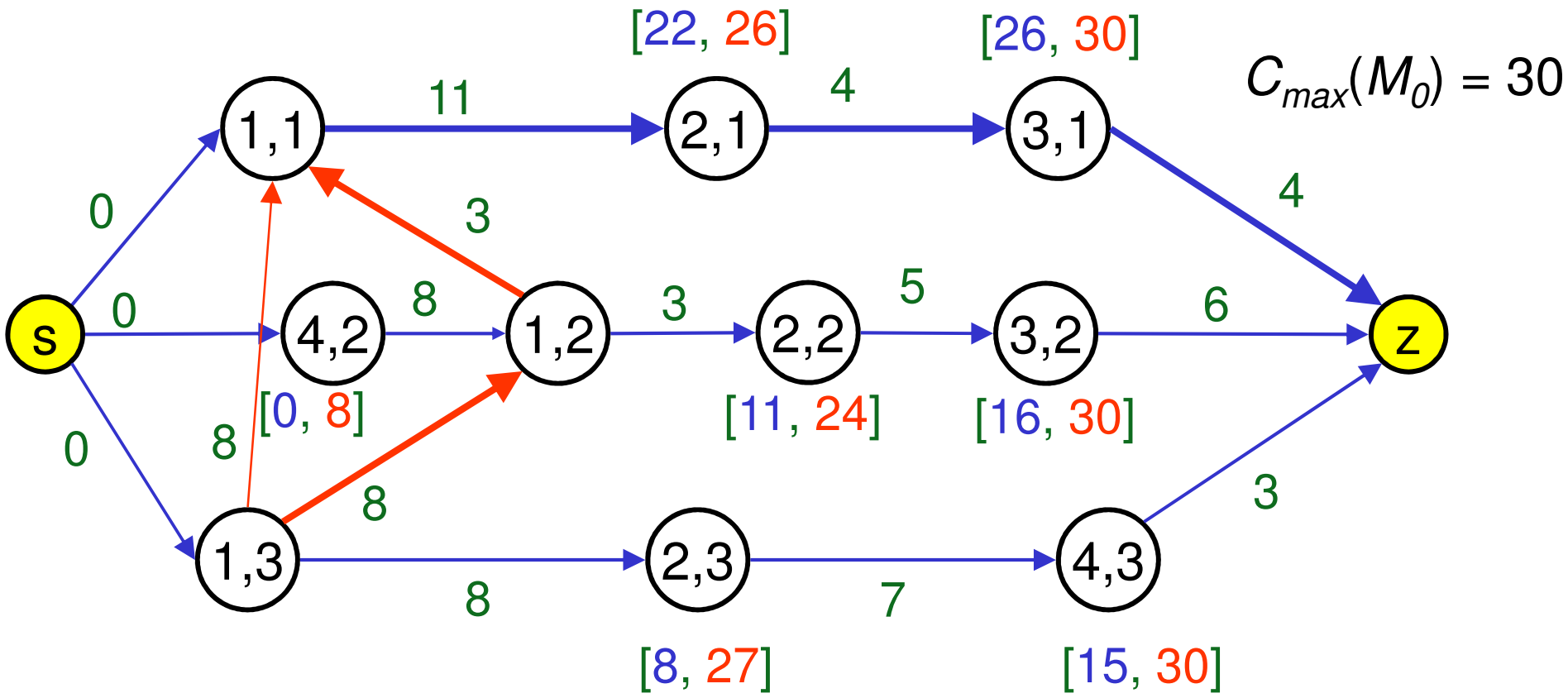


$C_{max}(M_0) = 30$

iter. 2

Iter 2. Calcola r_{ij}, d_{ij}

$M_0 = \{1\}$



Iter 2: problemi $1/r_j/L_{max}$

Job	1	2	3
p_j	4	5	7
r_j	22	11	8
d_j	26	24	27

2

Job	1	2
p_j	4	6
r_j	26	16
d_j	30	30

3

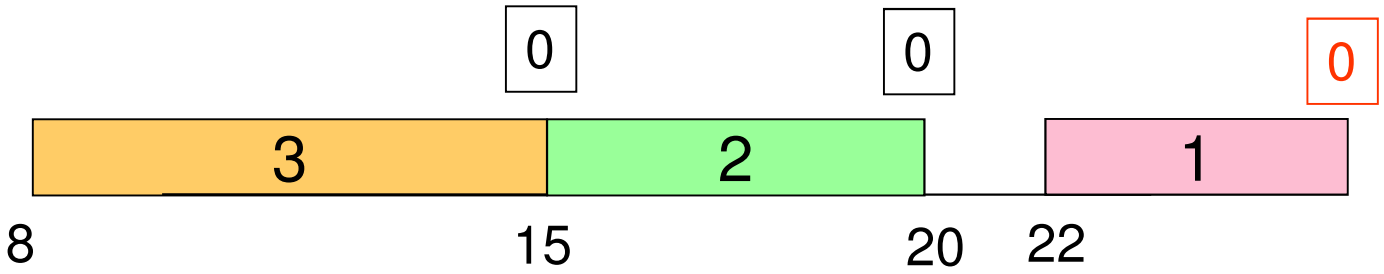
Job	2	3
p_j	8	3
r_j	0	15
d_j	8	30

4

$1/r_j/L_{max}$ macchina 2

Job	1	2	3
p_j	4	5	7
r_j	22	11	8
d_j	26	24	27

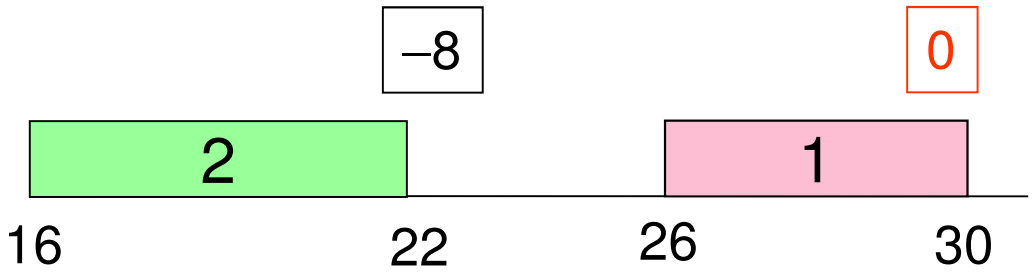
Soluzione ottima:



$1/r_j/L_{max}$ macchina 3

Job	1	2
p_j	4	6
r_j	26	16
d_j	30	30

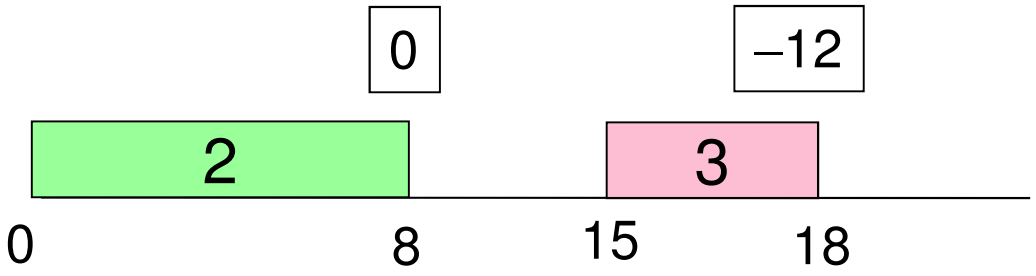
Soluzione ottima:



$1/r_j/L_{max}$ macchina 4

Job	2	3
p_j	8	3
r_j	0	15
d_j	8	30

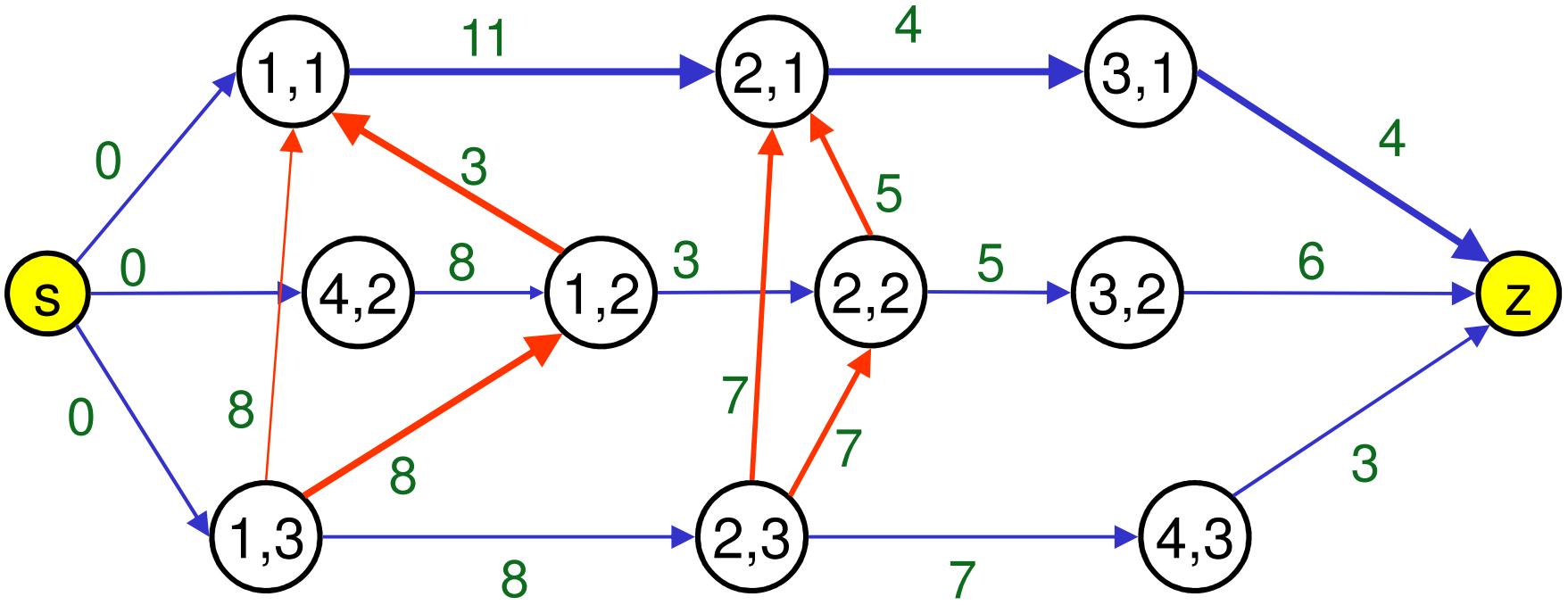
Soluzione ottima:



iter. 3

Iter 2. Calcola C_{max}

$M_0 = \{1,2\}$



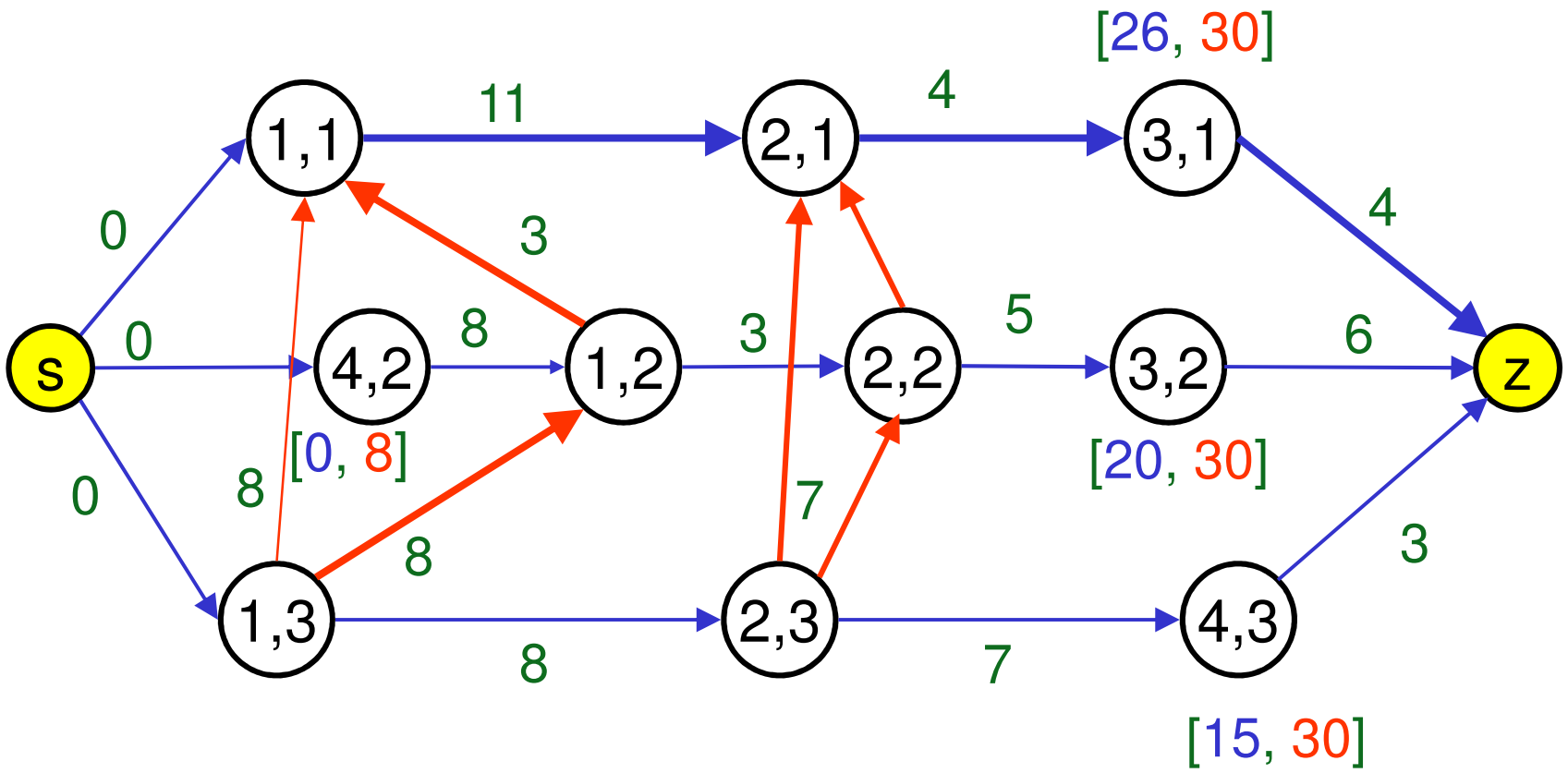
$C_{max}(M_0) = 30$

iter. 3

Iter 2. Calcola r_{ij}, d_{ij}

$M_0 = \{1,2\}$

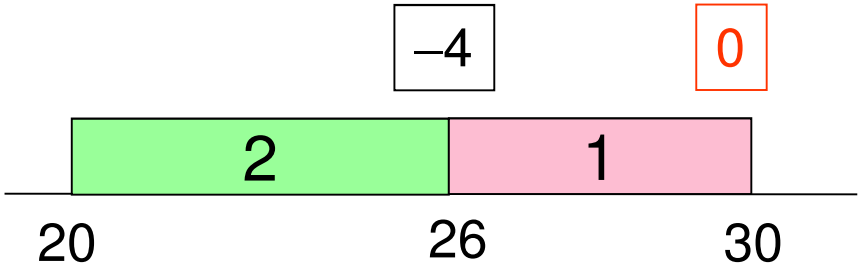
$C_{max}(M_0) = 30$



Iter 3: problemi $1/r_j/L_{max}$

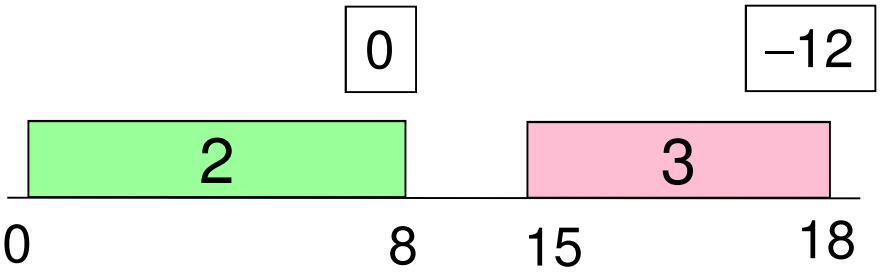
3

Job	1	2
p_j	4	6
r_j	26	20
d_j	30	30



4

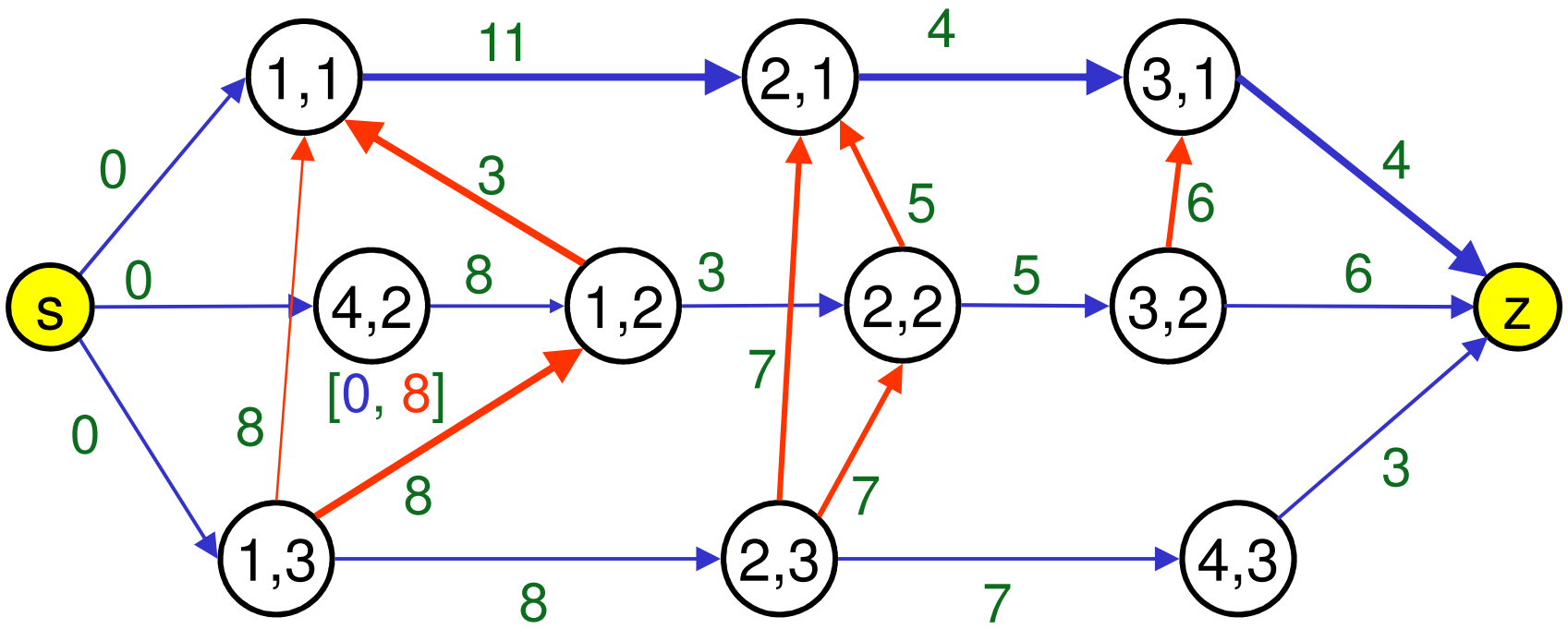
Job	2	3
p_j	8	3
r_j	0	15
d_j	8	30



Iter. 4

Iter 4. Calcola r_{ij}, d_{ij}

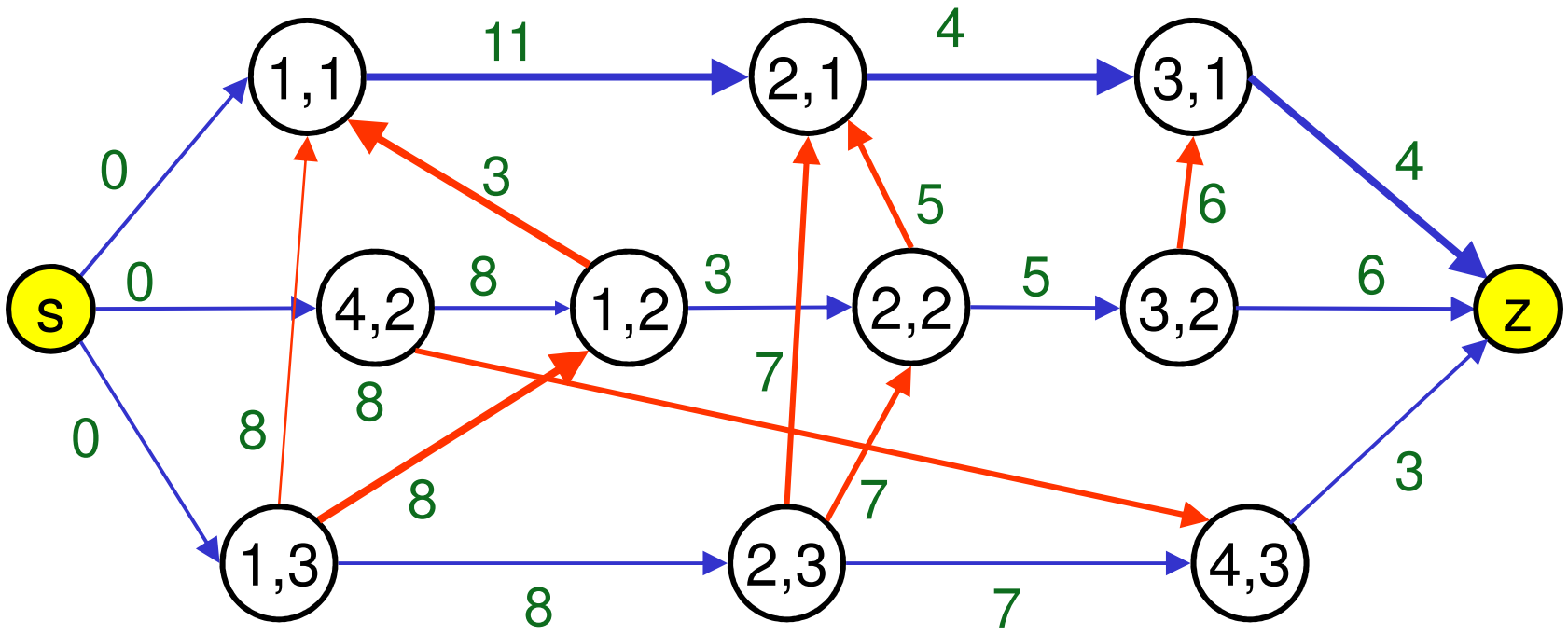
$M_0 = \{1,2,3\}$



$C_{max}(M_0) = 30$

STOP

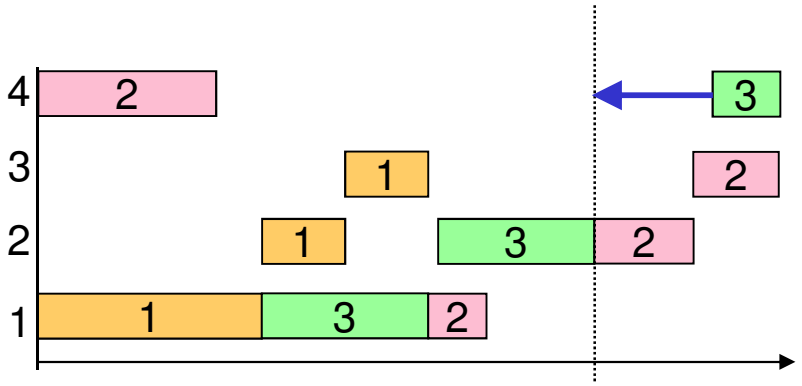
$M_0 = \{1,2,3, 4\}$ La soluzione calcolata è:



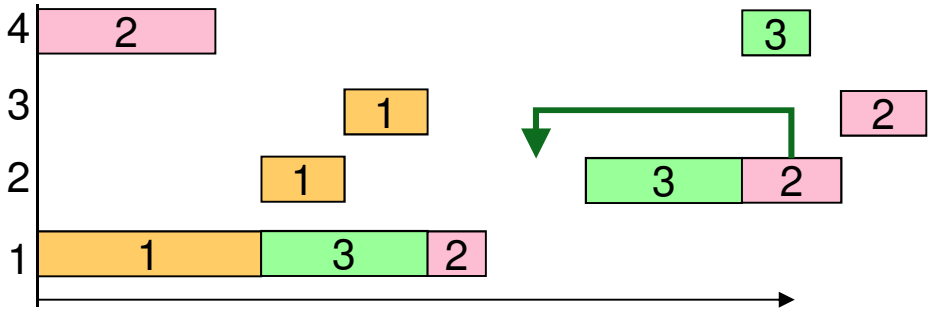
$C_{max}(M_0) = 30$

Operazioni su schedule

Due operazioni trasformano uno schedule ammissibile in un altro schedule ammissibile:



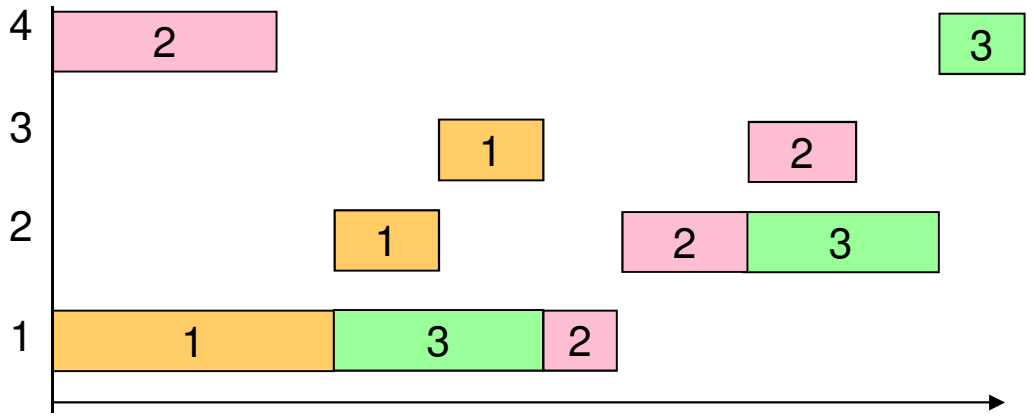
shift: anticipa un'operazione



jump: anticipa un'operazione in un intervallo in cui la macchina è ferma modificando la sequenza

Schedule attivi

Definizione. Uno schedule ammissibile si dice **attivo** se nessuna operazione può essere anticipata mediante uno shift o un jump senza ritardare un'altra operazione



Branch-and-bound

- Esiste uno schedule ottimo che è attivo, quindi restringiamo la ricerca agli schedule attivi
- ad ogni livello dell'albero di enumerazione si fissa un'operazione
- le operazioni fissate ad un dato sottoproblema formano uno schedule parziale
- un'operazione si dice disponibile se tutte le operazioni che la precedono sono state schedulate. AO = insieme delle operazioni disponibili

Branching

- step 1 (inizializzazione)
  calcola l'insieme AO e r_{ij} per ogni $(i,j) \in$ AO

- step 2 (selezione della macchina)
  calcola $t(\mathrm{AO}) = \min_{(i,j)\in \mathrm{AO}}\{r_{ij} + p_{ij}\} = r_{i^*j} + p_{i^*j}$

- step 3 (branching)
  BO = {operazioni $(i^*,j) \in$ AO tali che $\boldsymbol{r_{i^*j} < t(\mathrm{AO})}$}

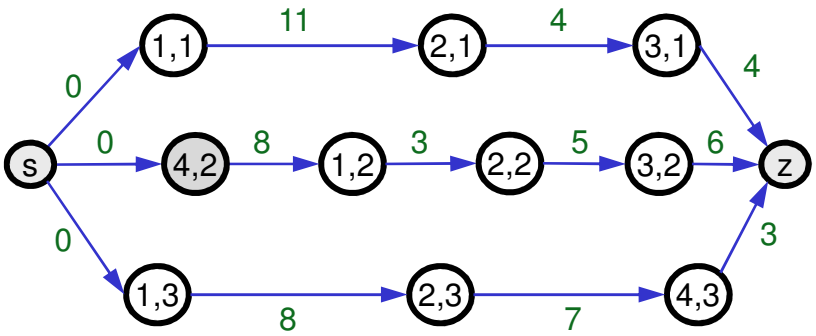
  per ogni operazione in BO:

  - genera un sottoproblema fissando tale operazione come successiva

Esempio

job	$i(j,1)$ $[p_{i(j,1),j}]$	$i(j,2)$ $[p_{i(j,2),j}]$	$i(j,3)$ $[p_{i(j,3),j}]$	$i(j,4)$ $[p_{i(j,4),j}]$
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [6]
3	1[8]	2 [7]	4 [3]	

nodo radice:



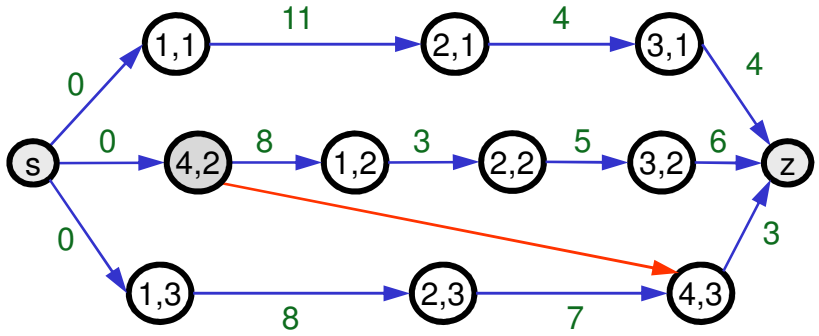
AO = {(1,1),(4,2),(1,3)}

t(AO) = min{0+11, 0+8, 0+8} = 8, i* = 4

$\mathbf{r_{4,2}=0} \Rightarrow$ BO = {(4,2)}

branching: genera un singolo sottoproblema A fissando l'operazione (4,2)

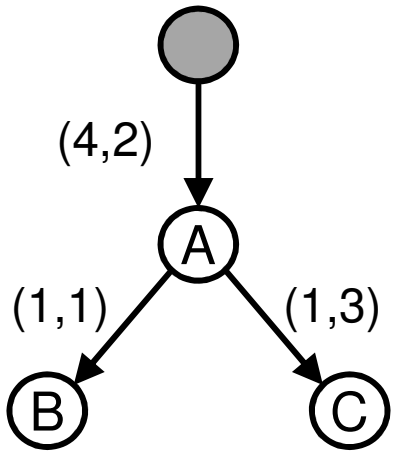
nodo A:



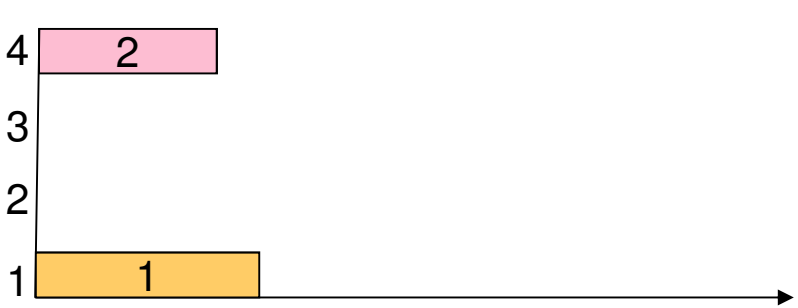
AO = {(1,1),(1,3),(1,2)}

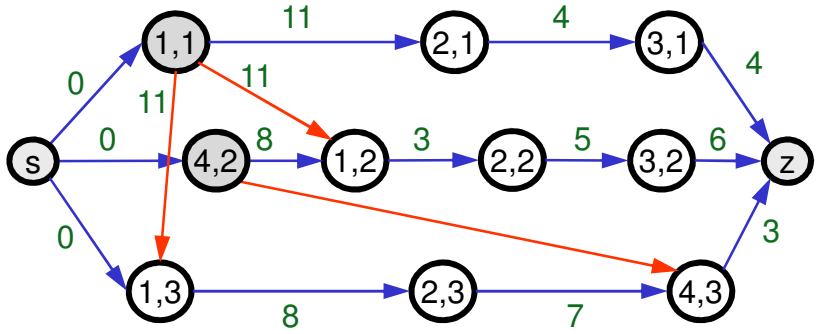
t(AO) = min{0+11, 0+8, 8+3} = 8, i* = 1

$r_{1,1}=0$, $r_{1,2}=8$, **$r_{1,3}=0$** $\Rightarrow$ BO={(1,1), (1,3)}



nodo B:

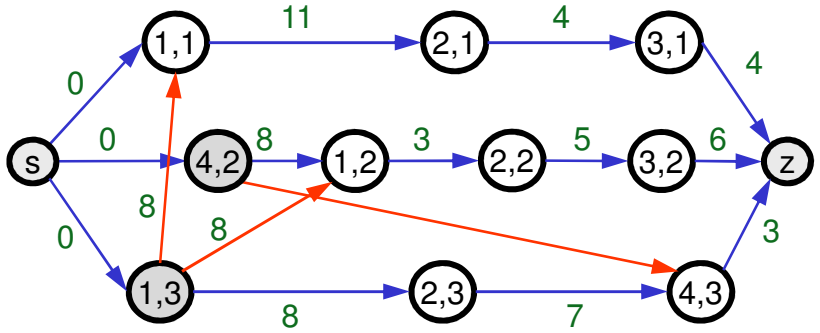




AO = {(1,3),(1,2),(2,1)} t(AO)=min{11+8,11+3,11+4} = 14, i* = 1

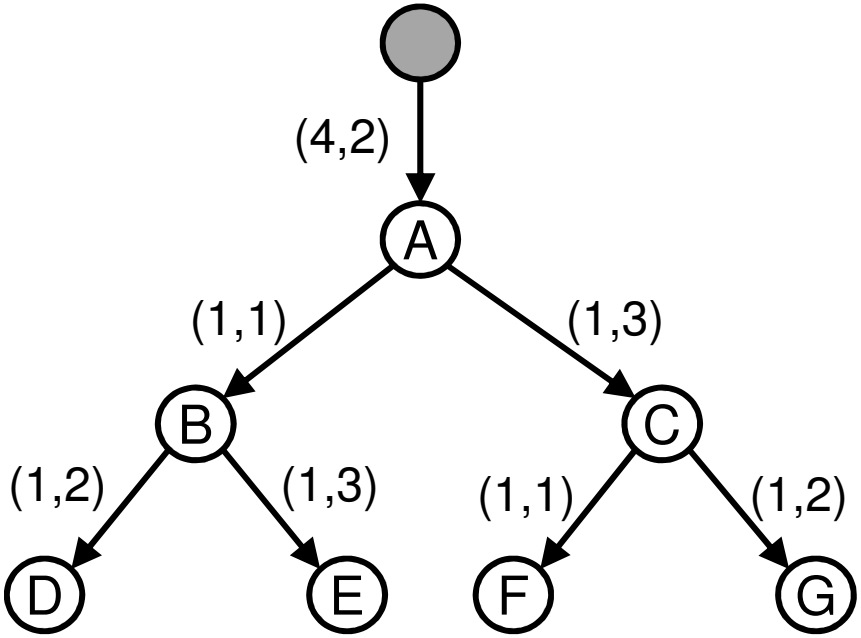
$r_{1,3}$=11, $r_{1,2}$=11 ⇒ BO = {(1,2),(1,3)} ⇒ sottoproblemi D,E

nodo C:

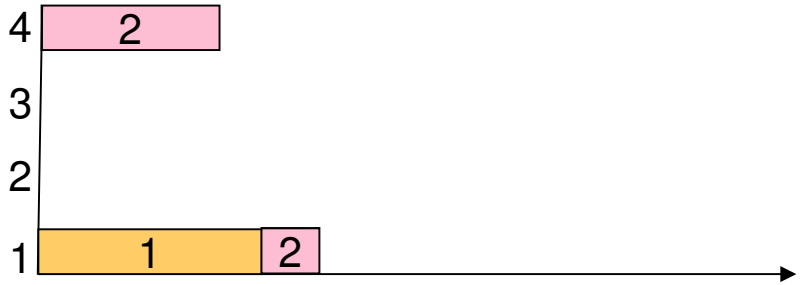


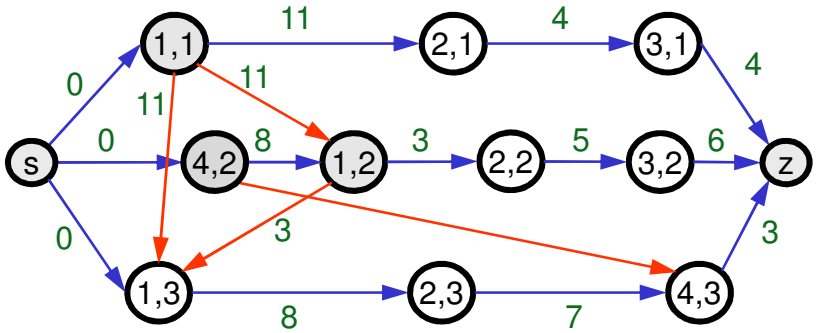
AO = {(1,1),(1,2),(2,3)} t(AO) =min{8+11, 8+3, 8+7} = 11, i* = 1

$r_{1,1}$=8, $r_{1,2}$=8 ⇒ BO = {(1,1),(1,2)} ⇒ sottoproblemi F,G


(4,2)
A
(1,1)
(1,3)
B
C
(1,2)
(1,3)
(1,1)
(1,2)
D
E
F
G

nodo D

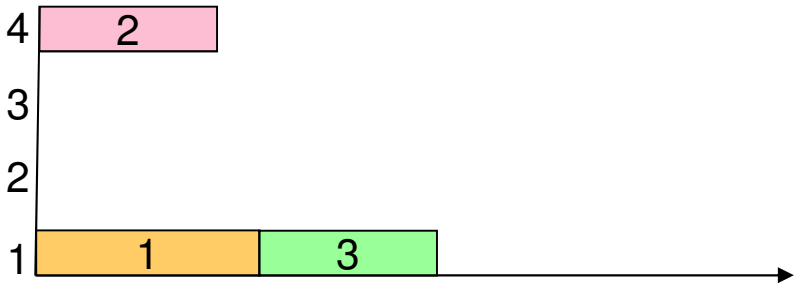


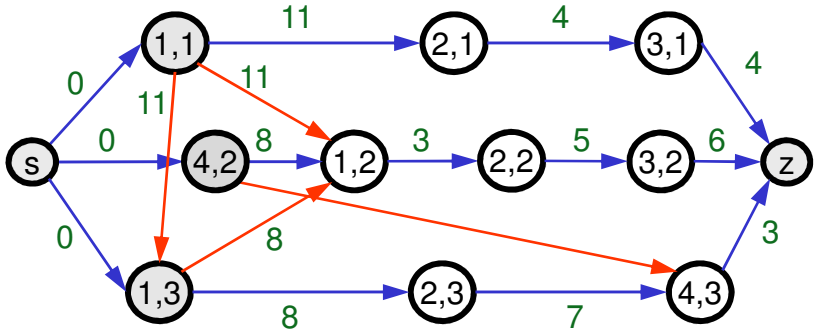


AO = {(1,3),(2,1),(2,2)} t(AO)=min{14+8, 11+4, 11+3} = 14, i* = 2

$r_{2,1}$=11, **$r_{2,2}$=11** ⇒ BO = {(2,1),(2,2)} ⇒ sottoproblemi H,I

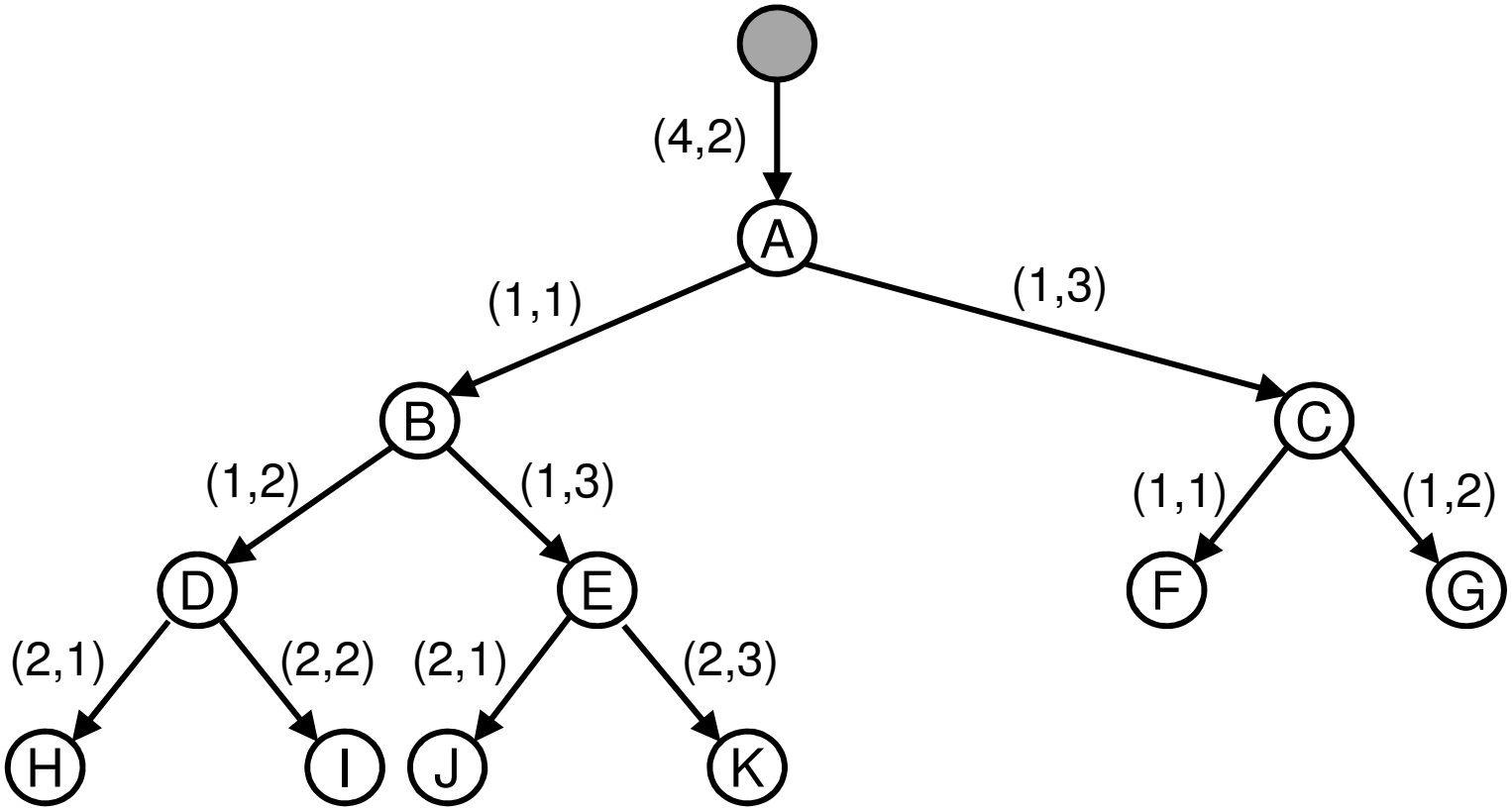
nodo E



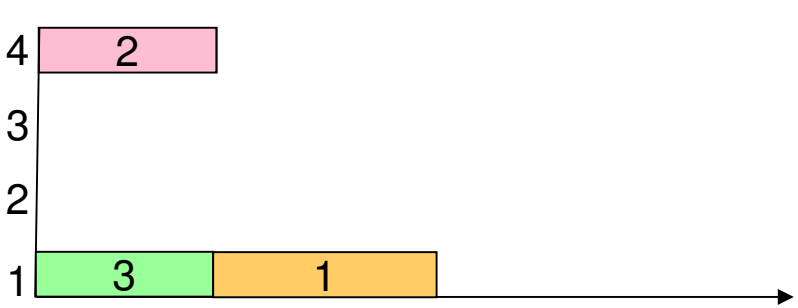


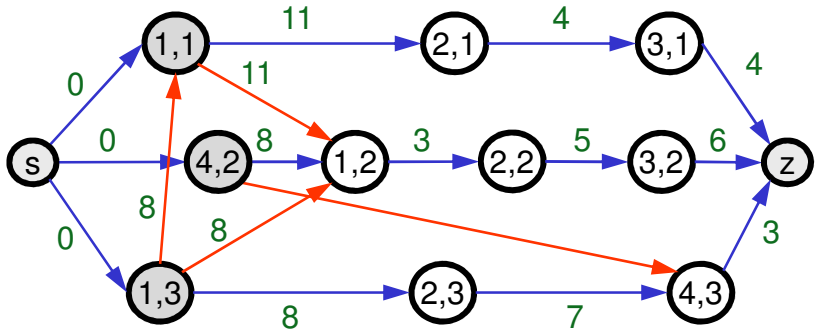
AO = {(1,2),(2,1),(2,3)} t(AO)=min{19+3,11+4, 11+8} = 15, i* = 2

$r_{2,1}$=11, **$r_{2,3}$=11** ⇒ BO = {(2,1),(2,3)} ⇒ sottoproblemi J,K


(4,2)
A
(1,1)
(1,3)
B
C
(1,2)
(1,3)
(1,1)
(1,2)
D
E
F
G
(2,1)
(2,2)
(2,1)
(2,3)
H
I
J
K

nodo F

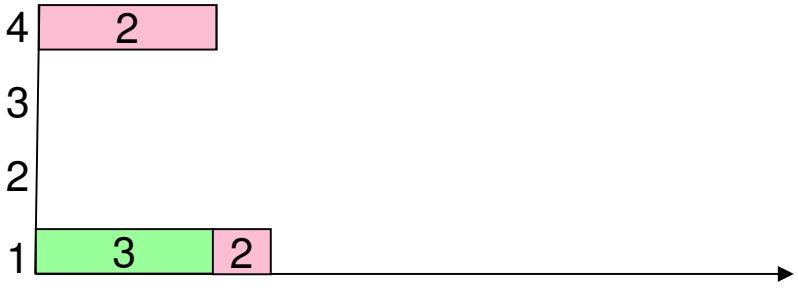


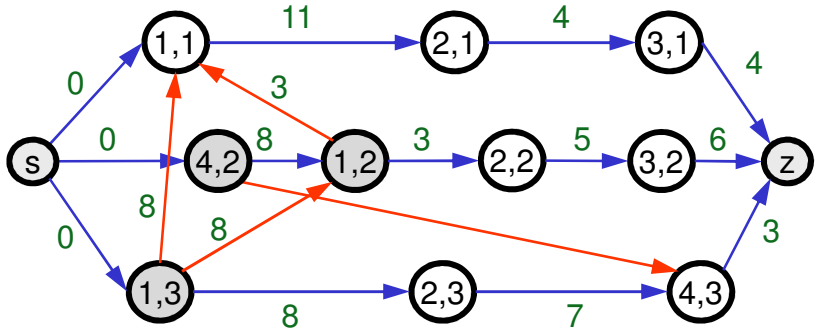


AO = {(1,2),(2,3),(2,1)} t(AO) =min{19+3, 8+7, 19+4} = 15, i* = 2

$r_{2,3}$=8, $r_{2,1}$=19 ⇒ BO = {(2,3)} ⇒ sottoproblema L

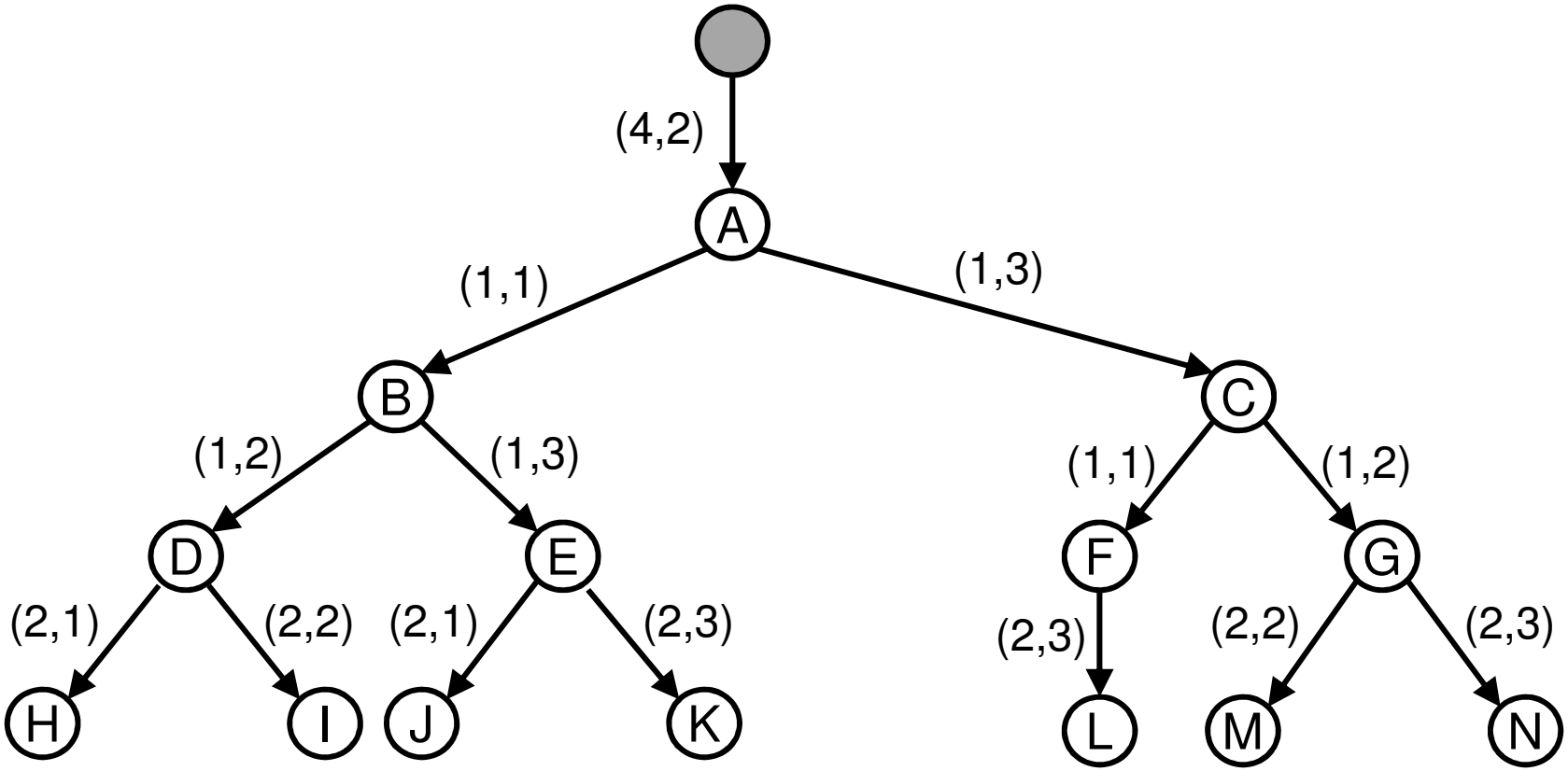
nodo G



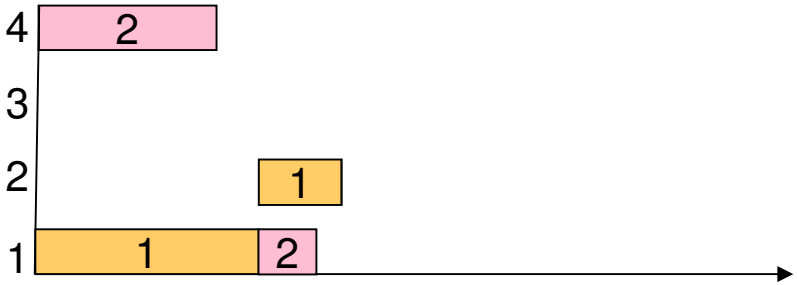


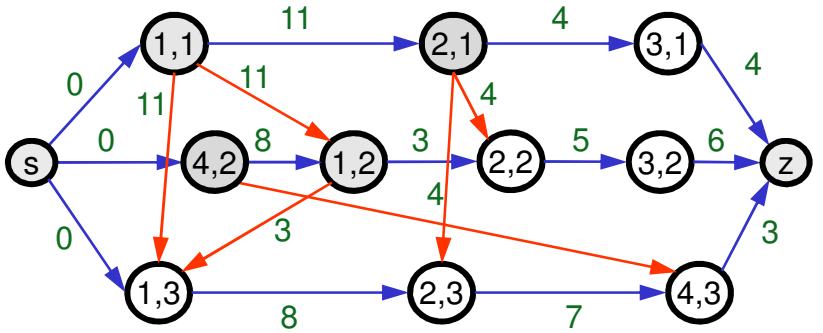
AO = {(1,1),(2,2),(2,3)} t(AO) =min{11+11, 11+5, 8+7} = 15, i* = 2

$r_{2,2}$=11, **$r_{2,3}$=8** ⇒ BO = {(2,2),(2,3)} ⇒ sottoproblemi M,N


(4,2)
A
(1,1)
(1,3)
B
C
(1,2)
(1,3)
(1,1)
(1,2)
D
E
F
G
(2,1)
(2,2)
(2,1)
(2,3)
(2,3)
(2,2)
(2,3)
H
I
J
K
L
M
N

nodo H

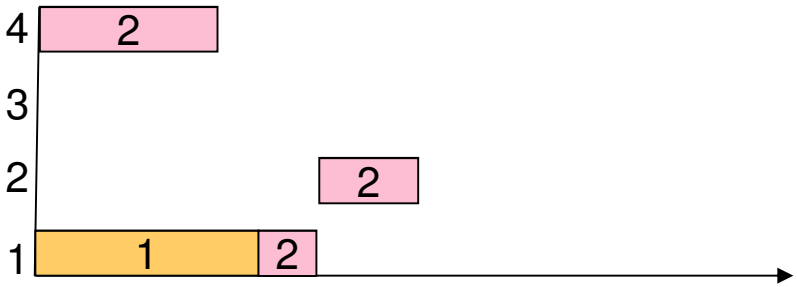


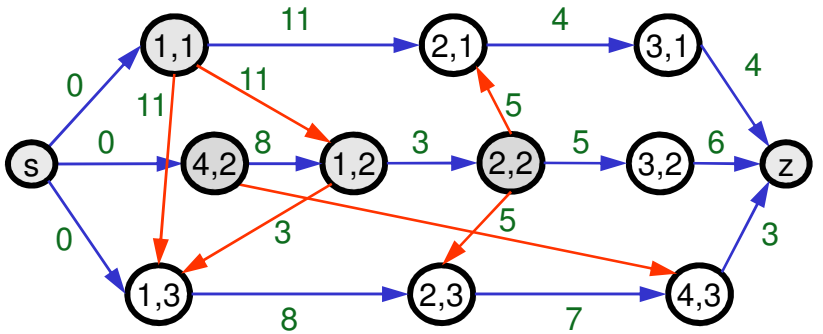


AO = {(1,3),(2,2),(3,1)} t(AO)=min{14+8, 15+5, 15+4} = 19, i* = 3

$r_{3,1}$=15 ⇒ BO = {(3,1)} ⇒ sottoproblema O

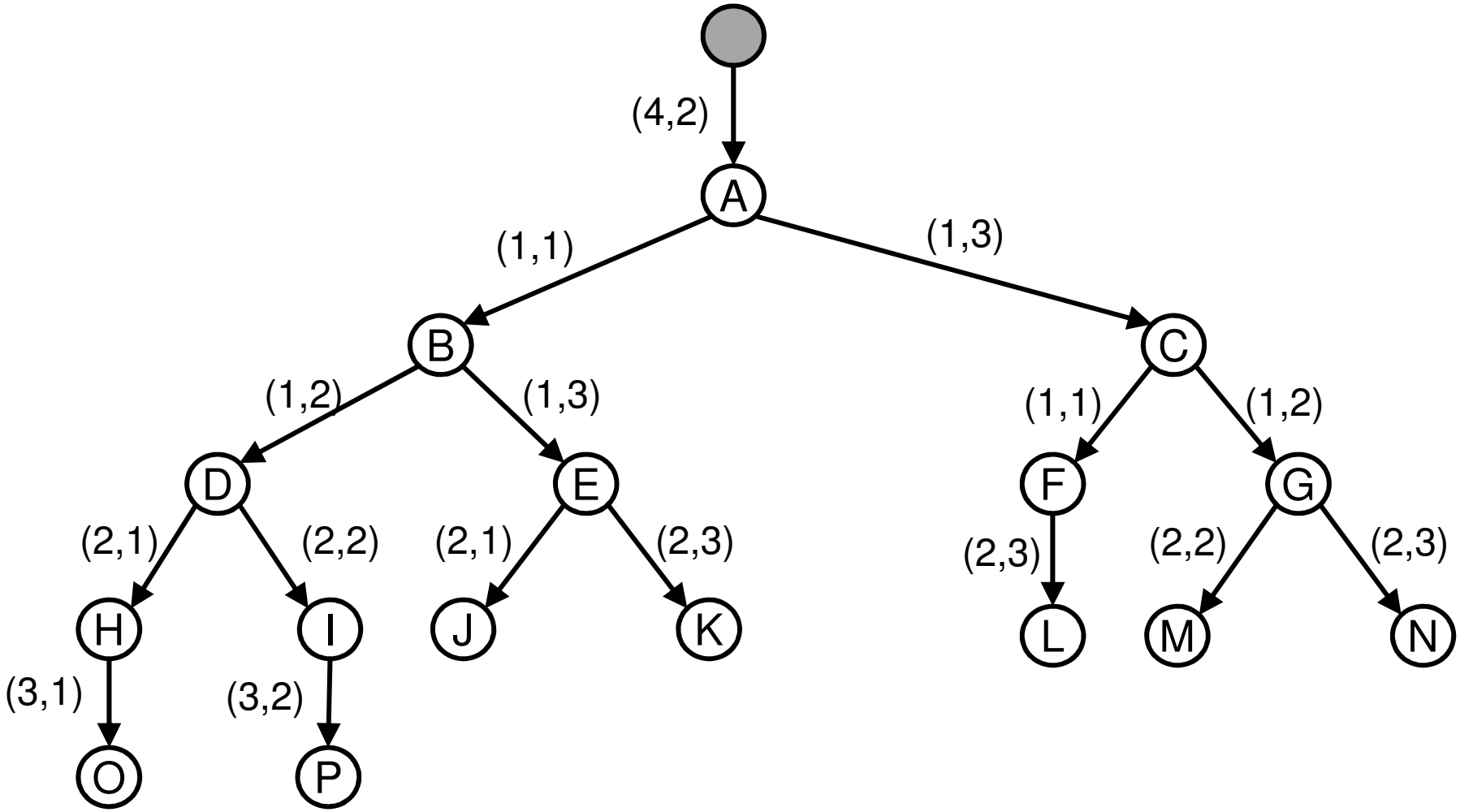
nodo I



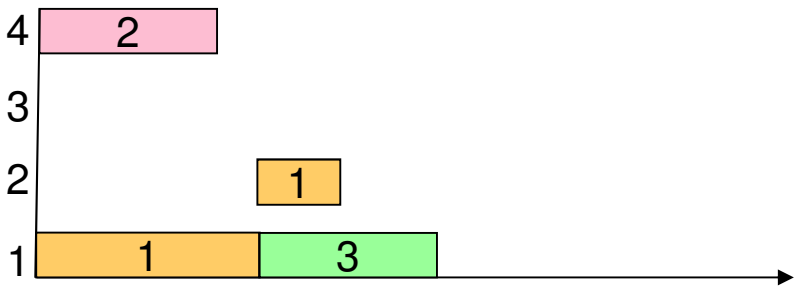


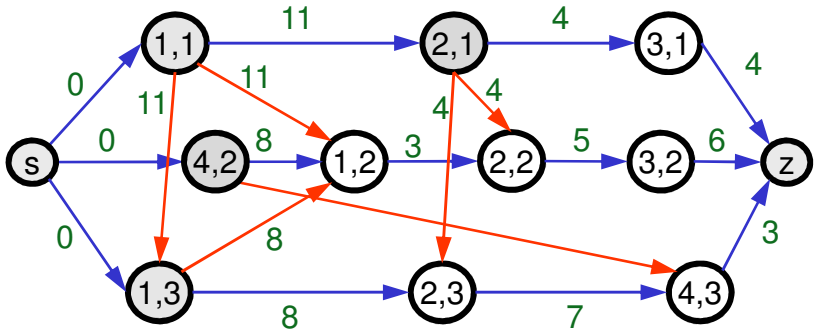
AO = {(1,3),(2,1),(3,2)} t(AO)=min{14+8, 19+4, 14+5} = 19, i* = 3

$r_{3,2}$=11 ⇒ BO = {(3,2)} ⇒ sottoproblema P


(4,2)
A
(1,1)
(1,3)
B
C
(1,2)
(1,3)
(1,1)
(1,2)
D
E
F
G
(2,1)
(2,2)
(2,1)
(2,3)
(2,3)
(2,2)
(2,3)
H
I
J
K
L
M
N
(3,1)
(3,2)
O
P

nodo J

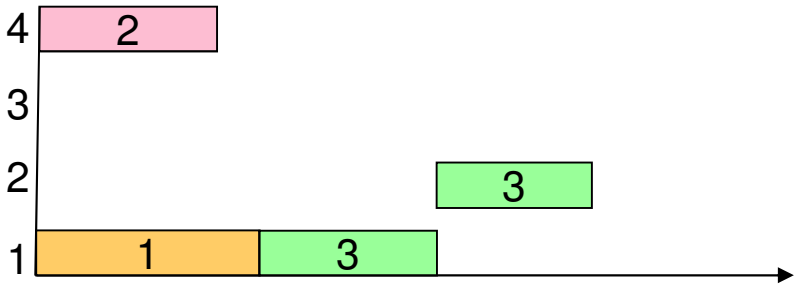


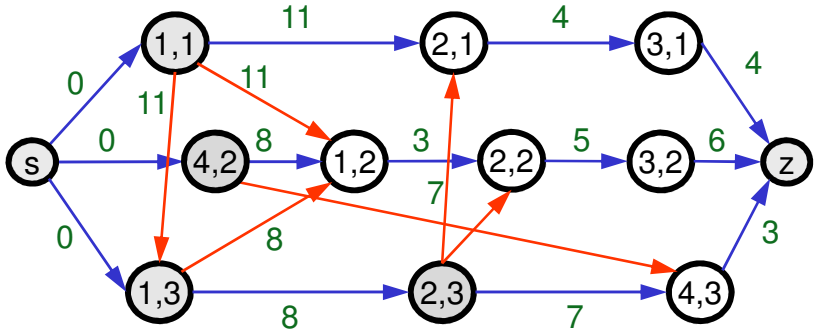


AO = {(1,2),(3,1),(2,3)} t(AO)=min{19+3,15+4, 15+7} = 19, i* = 3

$r_{3,1}=15 \Rightarrow$ BO = {(3,1)} $\Rightarrow$ sottoproblema R

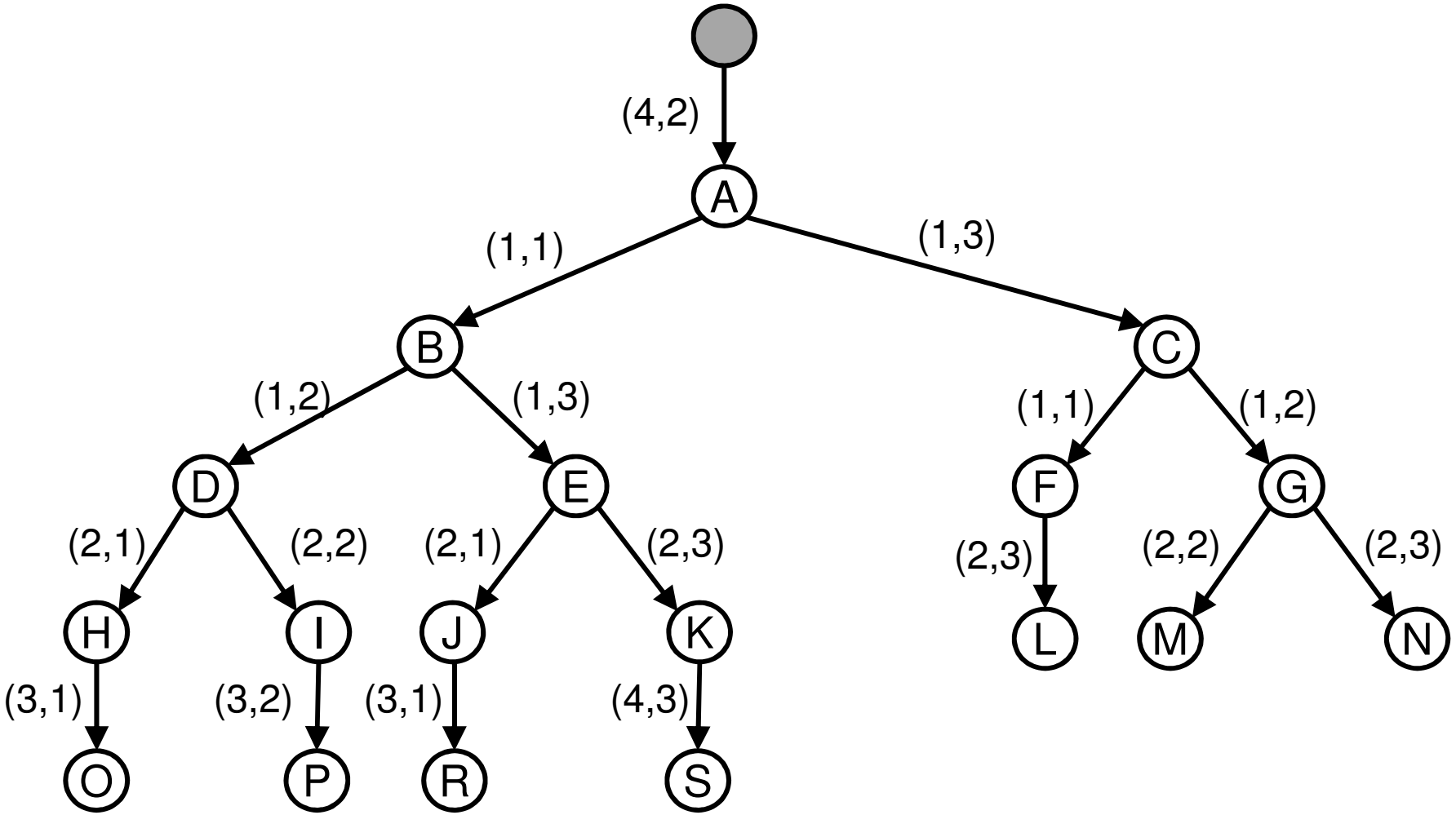
nodo K



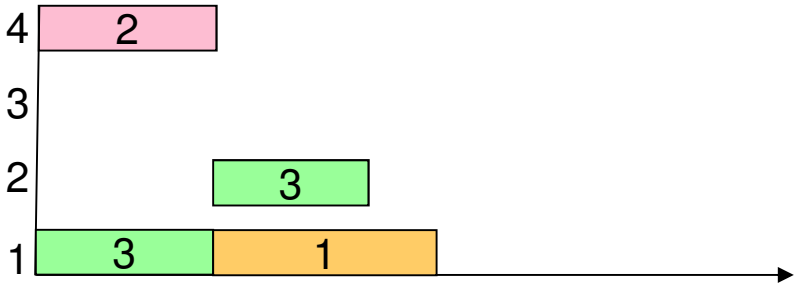


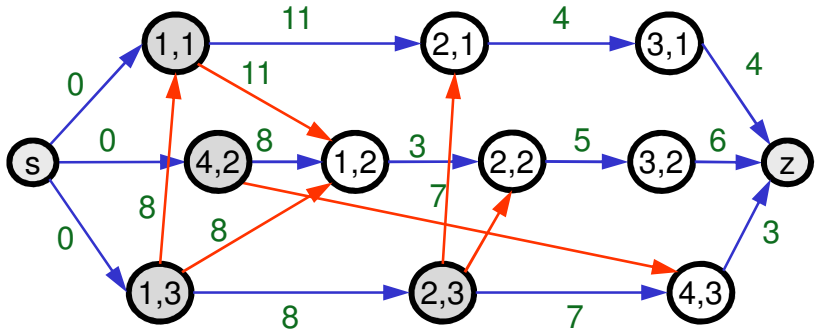
AO = {(1,2),(2,1),(4,3)} t(AO)=min{19+3,15+4, 15+3} = 18, i* = 4

$r_{2,3}=15 \Rightarrow$ BO = {(4,3)} $\Rightarrow$ sottoproblema S


(4,2)
A
(1,1)
(1,3)
B
C
(1,2)
(1,3)
(1,1)
(1,2)
D
E
F
G
(2,1)
(2,2)
(2,1)
(2,3)
(2,3)
(2,2)
(2,3)
H
I
J
K
L
M
N
(3,1)
(3,2)
(3,1)
(4,3)
O
P
R
S

nodo L

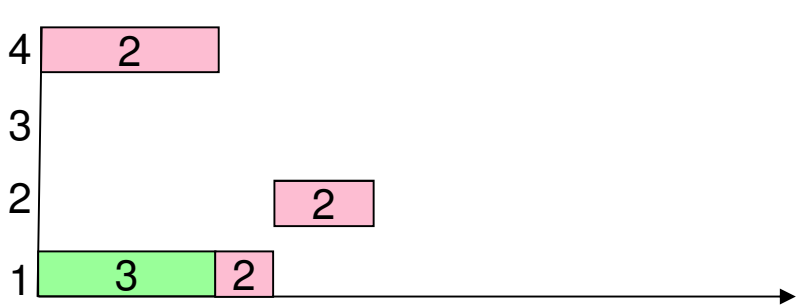


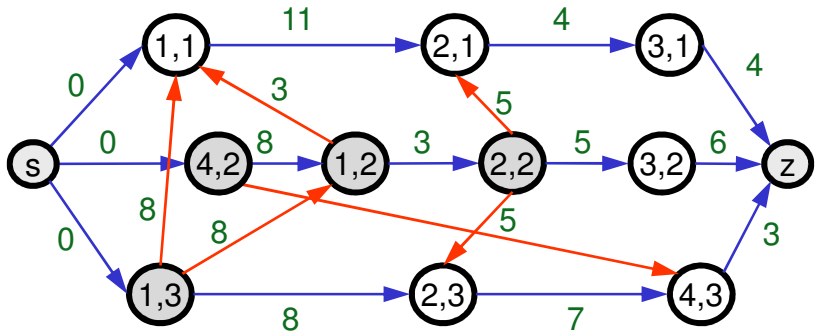


AO = {(1,2),(2,1),(4,3)} t(AO) =min{19+3, 19+4, 15+3} = 18, i* = 4

$r_{4,3}$**=15** $\Rightarrow$ BO = {(4,3)} $\Rightarrow$ sottoproblema T

nodo M

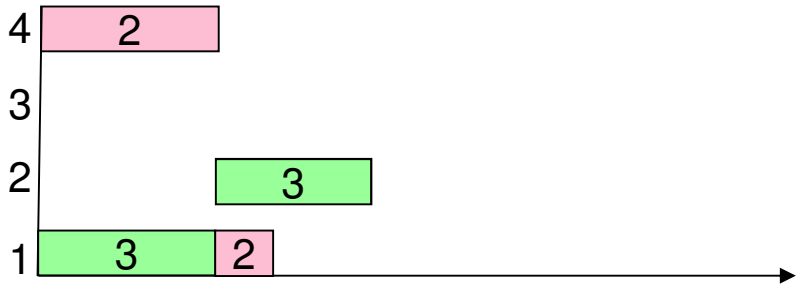


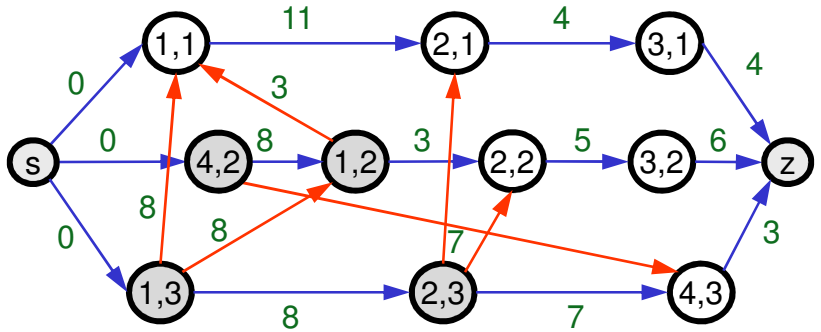


AO = {(1,1),(3,2),(2,3)} t(AO) =min{11+11, 16+6, 8+7} = 15, i* = 2

$r_{2,3}=8 \Rightarrow$ BO = {(2,3)} $\Rightarrow$ sottoproblema U

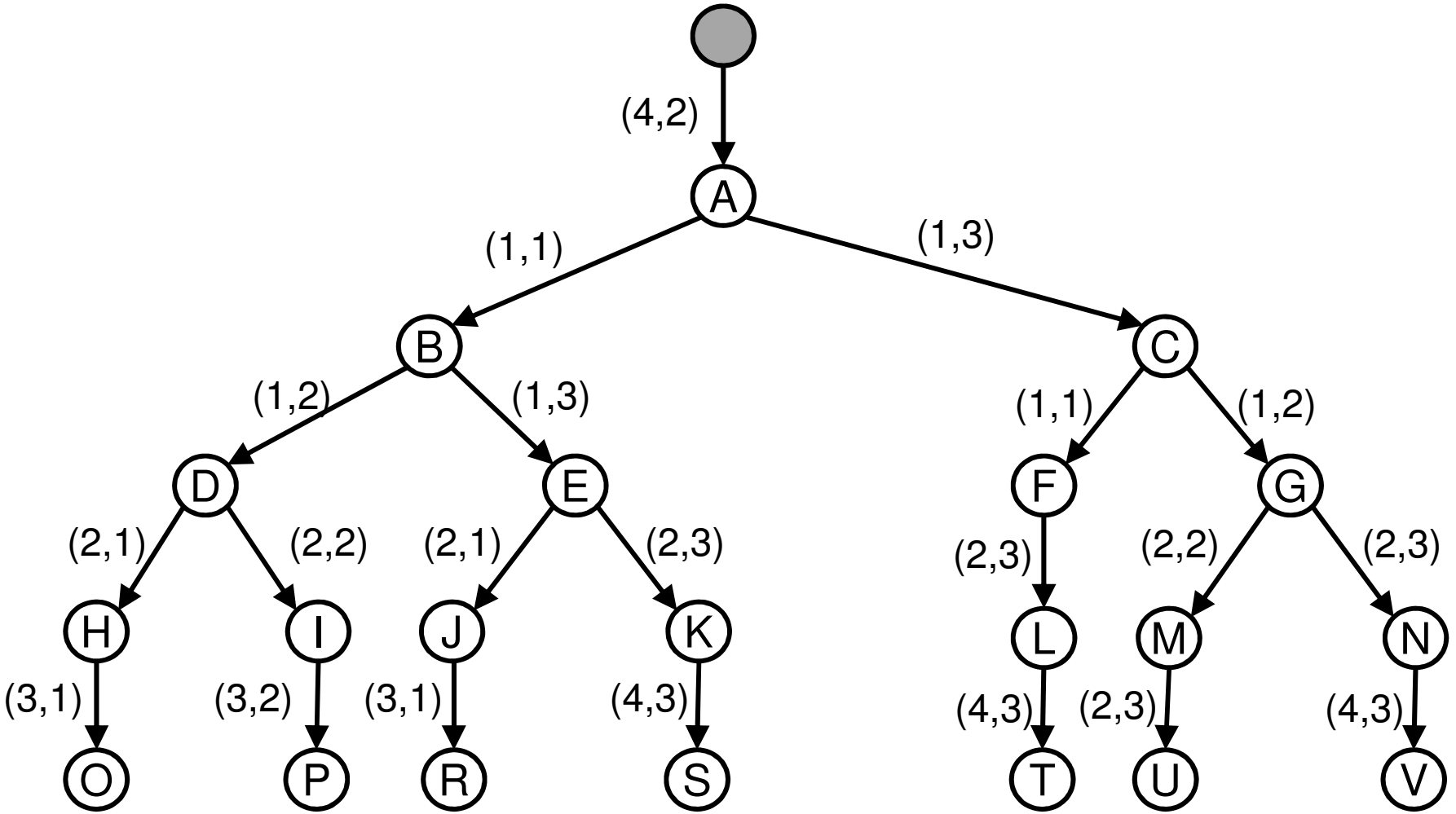
nodo N



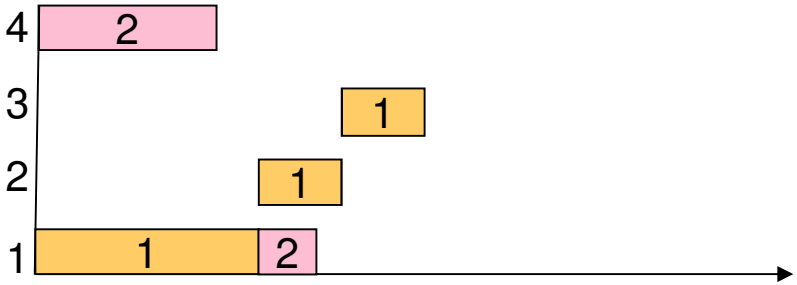


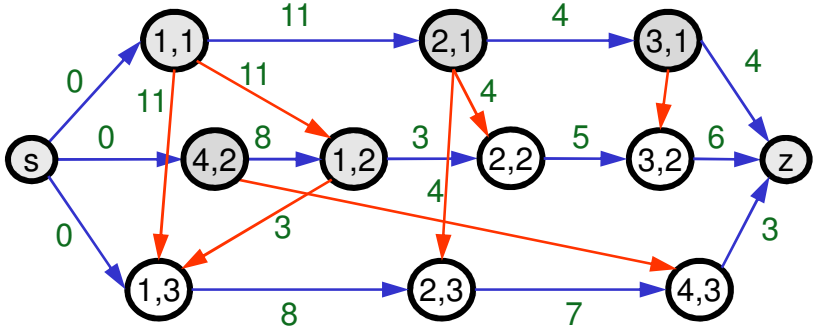
AO = {(1,1),(2,2),(4,3)} t(AO) =min{11+11, 15+5, 15+3} = 18, i* = 4

$r_{4,3}=15 \Rightarrow$ BO = {(4,3)} $\Rightarrow$ sottoproblema V


(4,2)
A
(1,1)
(1,3)
B
C
(1,2)
(1,3)
(1,1)
(1,2)
D
E
F
G
(2,1)
(2,2)
(2,1)
(2,3)
(2,3)
(2,2)
(2,3)
H
I
J
K
L
M
N
(3,1)
(3,2)
(3,1)
(4,3)
(4,3)
(2,3)
(4,3)
O
P
R
S
T
U
V

nodo O

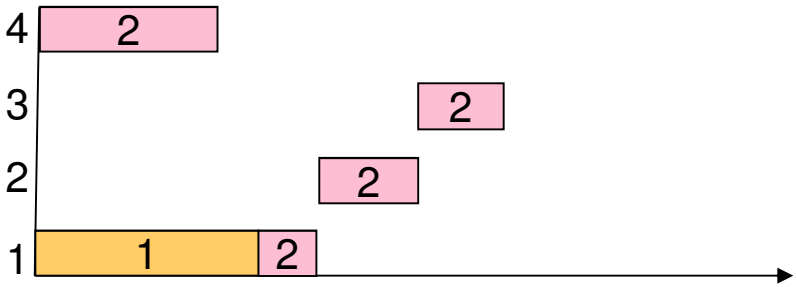


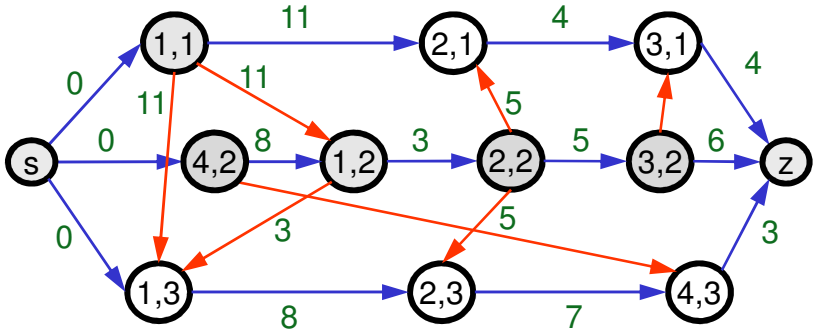


AO = {(1,3),(2,2)} t(AO)=min{14+8, 15+5} = 20, i* = 2

$r_{2,2}$=15 ⇒ BO = {(2,2)} ⇒ sottoproblema W

nodo P

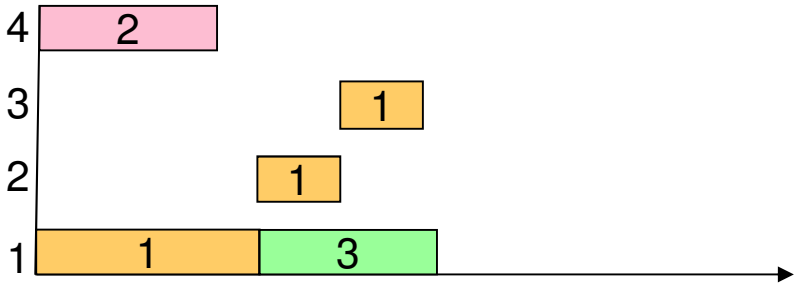


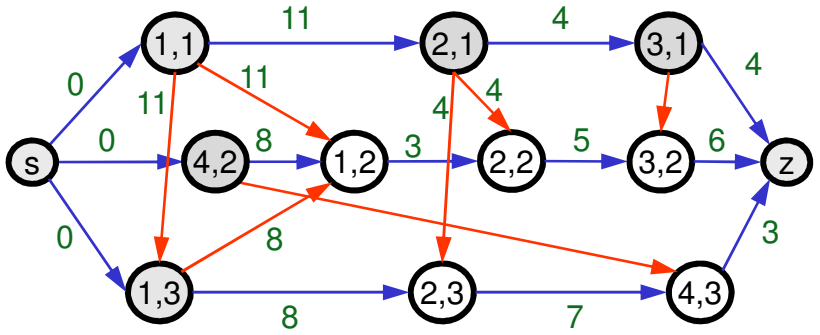


AO = {(1,3),(2,1)} t(AO)=min{14+8, 19+4} = 22, i* = 1

$r_{1,3}$=14 ⇒ BO = {(1,3)} ⇒ sottoproblema X

nodo R

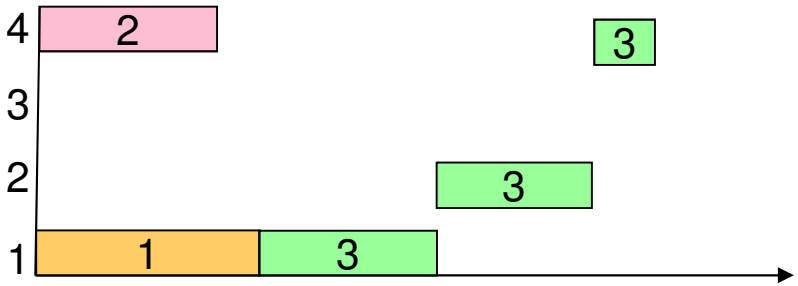


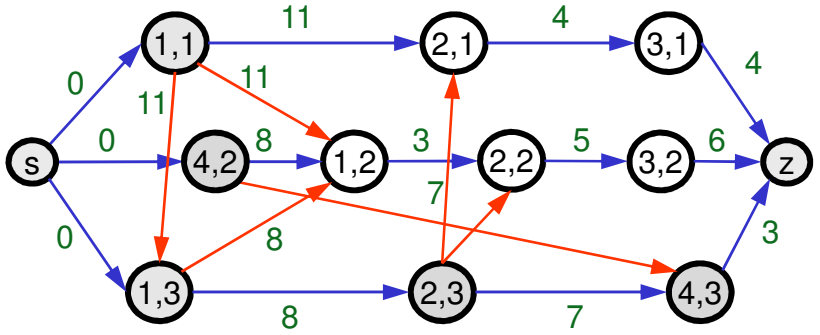


AO = {(1,2),(2,3)} t(AO)=min{19+3, 15+7} = 22, i* = 1

$r_{1,2}$=19 ⇒ BO = {(1,2)} ⇒ sottoproblema Y

nodo S

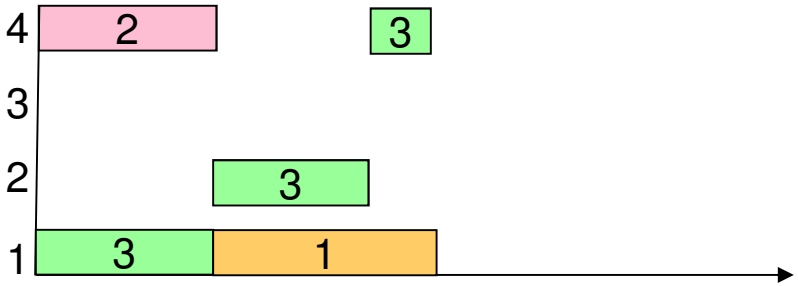


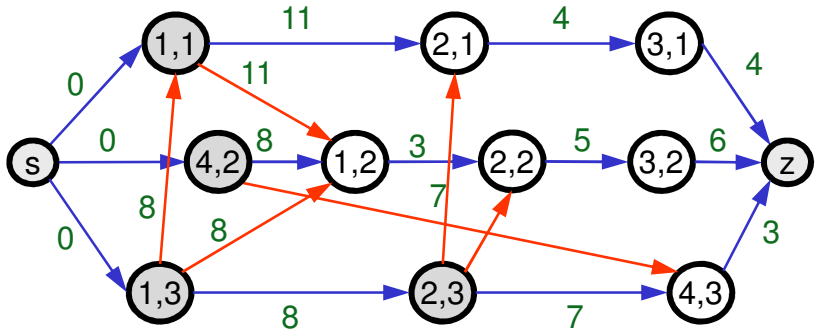


AO = {(1,2),(2,1)} t(AO)=min{19+3,15+4} = 19, i* = 2

$r_{2,1}$=15 ⇒ BO = {(2,1)} ⇒ sottoproblema Z

nodo T

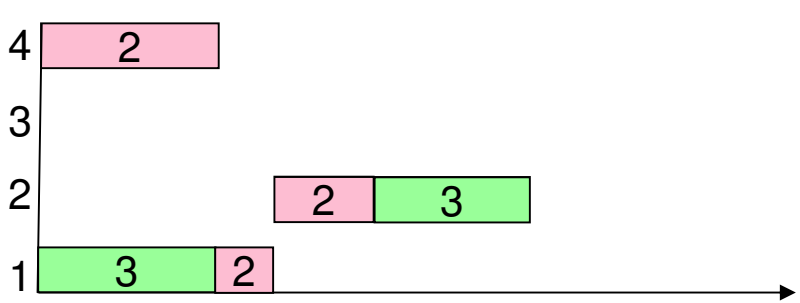


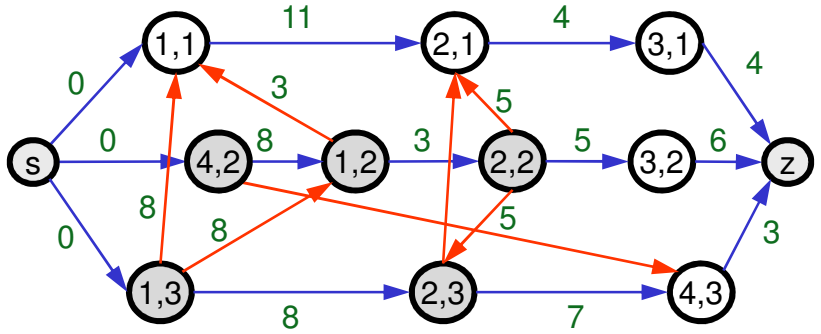


AO = {(1,2),(2,1)} t(AO) =min{19+3, 19+4} = 21, i* = 1

$r_{1,2}$=19 $\Rightarrow$ BO = {(1,2)} $\Rightarrow$ sottoproblema α

nodo U

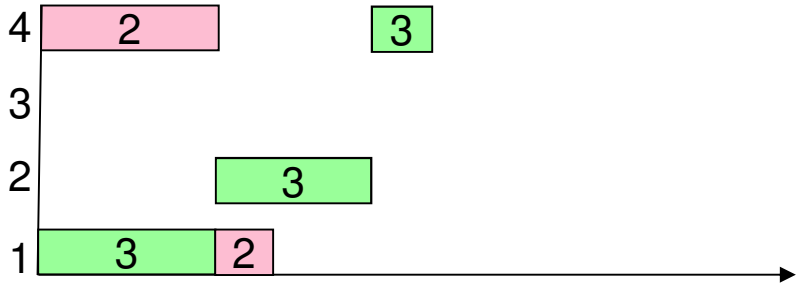


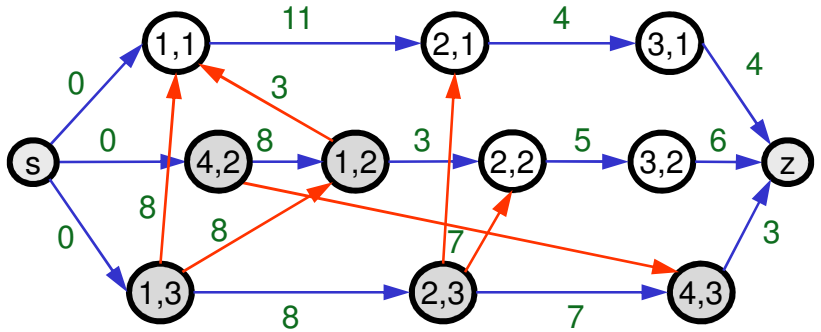


AO = {(1,1),(3,2)} t(AO) =min{11+11, 16+6} = 22, i* = 3

$r_{3,2}$=16 ⇒ BO = {(3,2)} ⇒ sottoproblema β

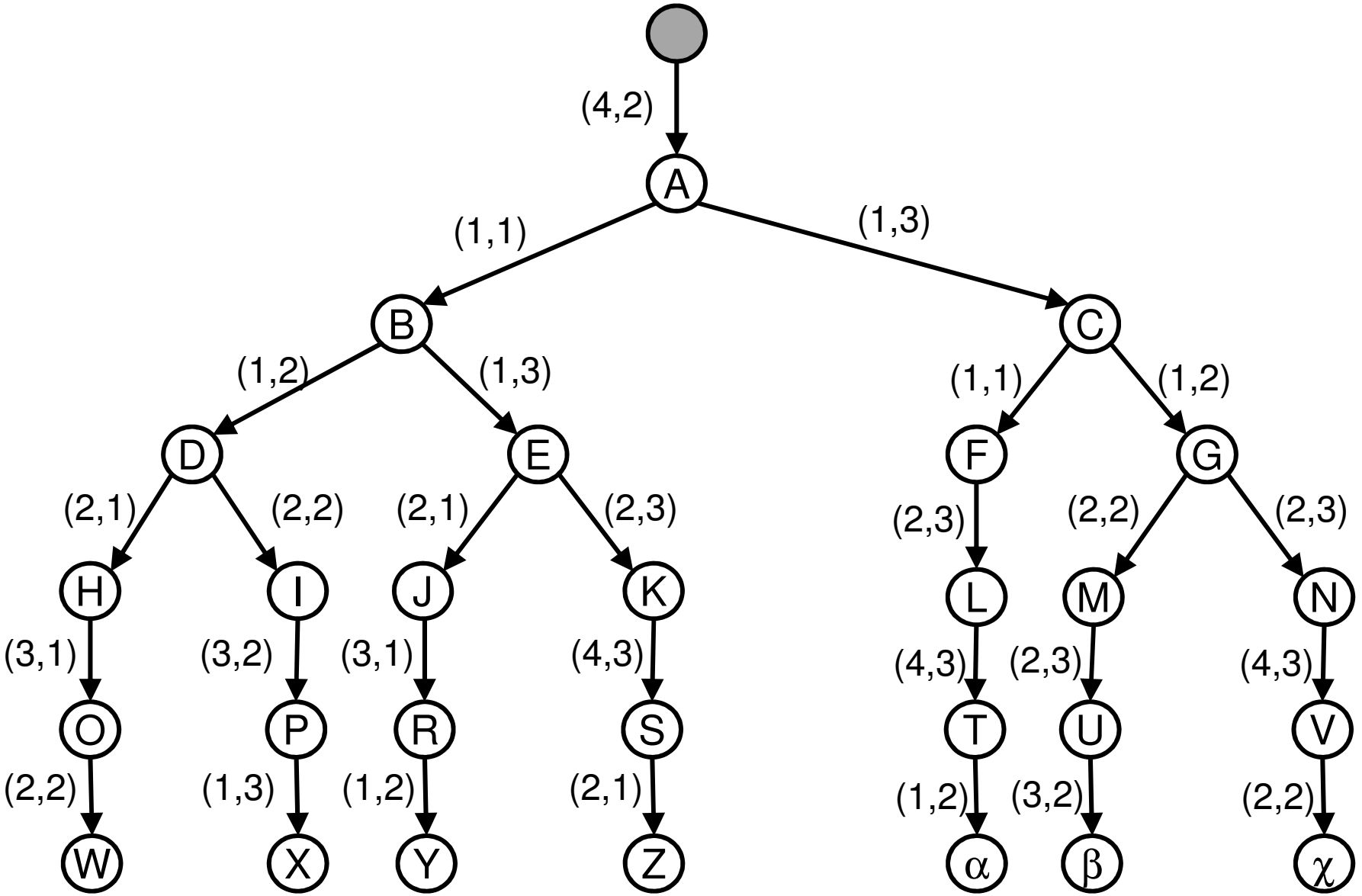
nodo V





AO = {(1,1),(2,2)} t(AO) =min{11+11, 15+5} = 20, i* = 2

$r_{2,2}$=15 ⇒ BO = {(2,2)} ⇒ sottoproblema χ


(4,2)
A
(1,1)
(1,3)
B
C
(1,2)
(1,3)
(1,1)
(1,2)
D
E
F
G
(2,1)
(2,2)
(2,1)
(2,3)
(2,3)
(2,2)
(2,3)
H
I
J
K
L
M
N
(3,1)
(3,2)
(3,1)
(4,3)
(4,3)
(2,3)
(4,3)
O
P
R
S
T
U
V
(2,2)
(1,3)
(1,2)
(2,1)
(1,2)
(3,2)
(2,2)
W
X
Y
Z
α
β
χ

Regole di *dispatching*

- FCFS (First Come First Served): ogni macchina esegue l'operazione processabile secondo l'ordine di arrivo

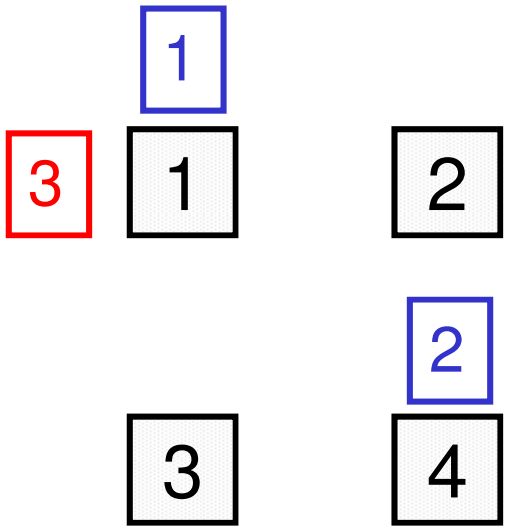
- MWR (Most Work Remaining): fra le operazioni processabili, ogni macchina processa quella con il maggior tempo di processamento residuo (di tutte le operazioni rimanenti del job)

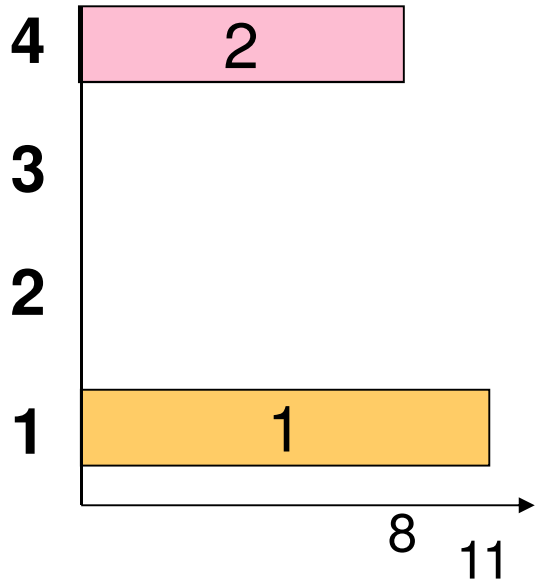
- SPT (Shortest Processing Time): ogni macchina processa l'operazione disponibile più breve

Esempio: FCFS

job	i(1) [$p_{i(1),j}$]	i(2) [$p_{i(2),j}$]	i(3) [$p_{i(3),j}$]	i(4) [$p_{i(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

t = 0:

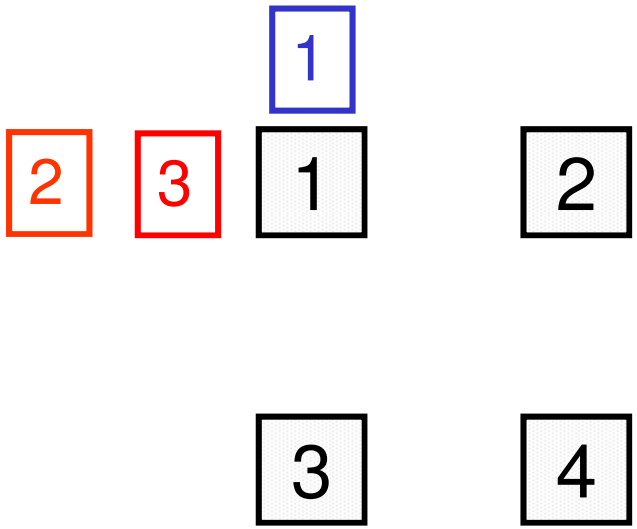


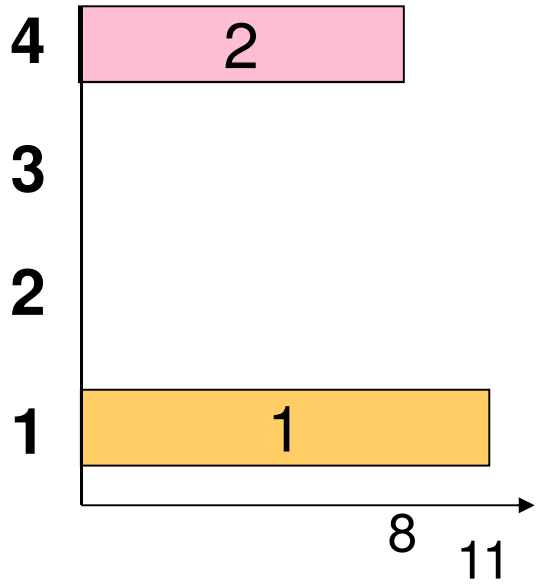


Esempio: FCFS

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

t = 8:

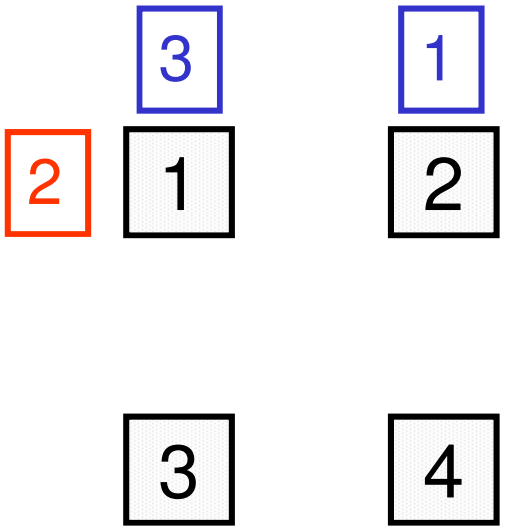


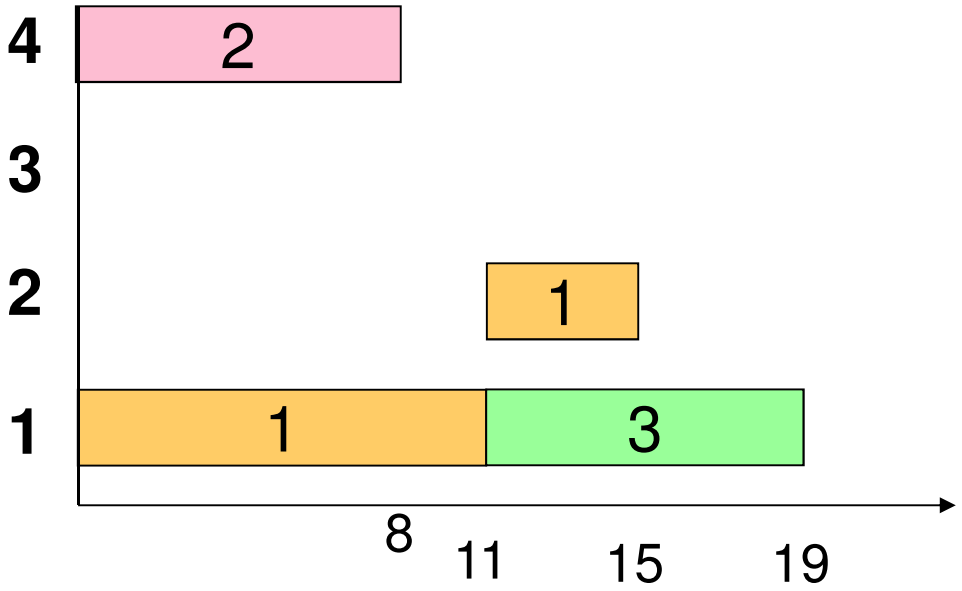


Esempio: FCFS

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

t = 11:

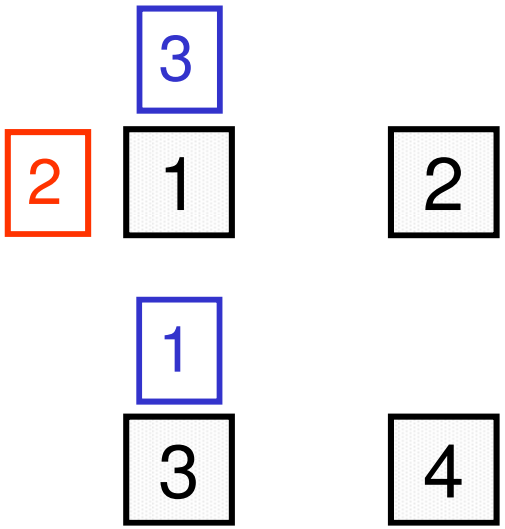


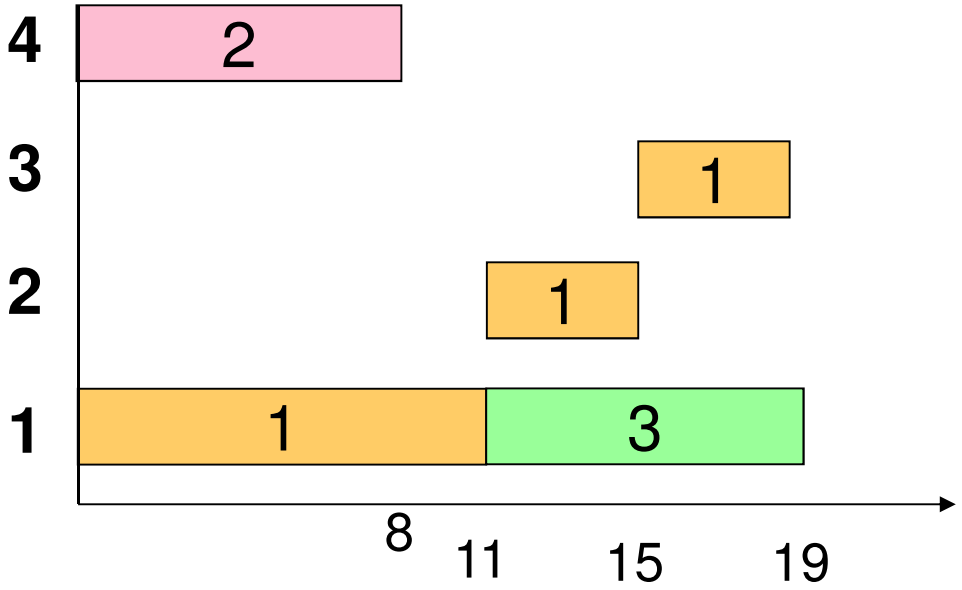


Esempio: FCFS

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

t = 15:

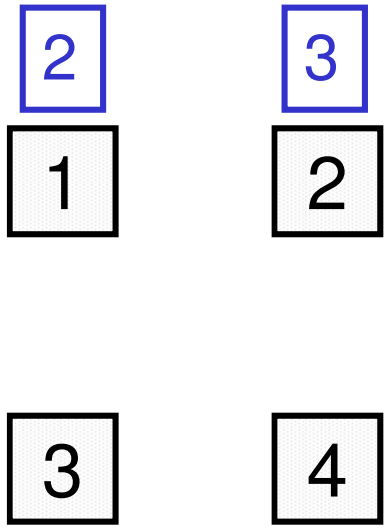


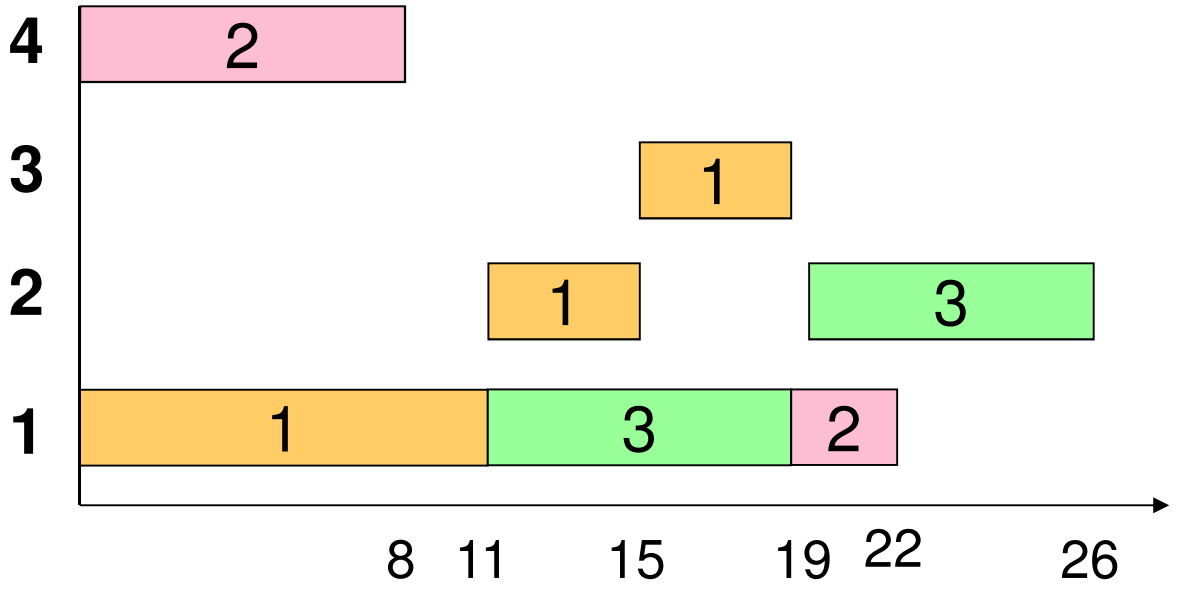


Esempio: FCFS

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

t = 19:

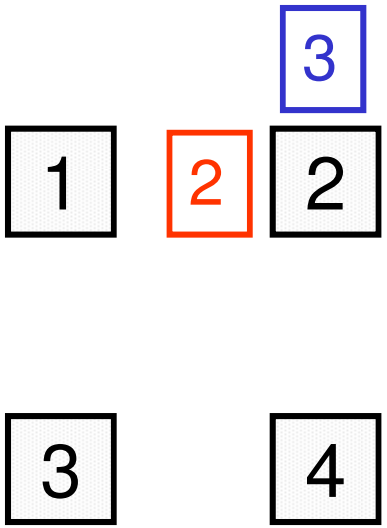


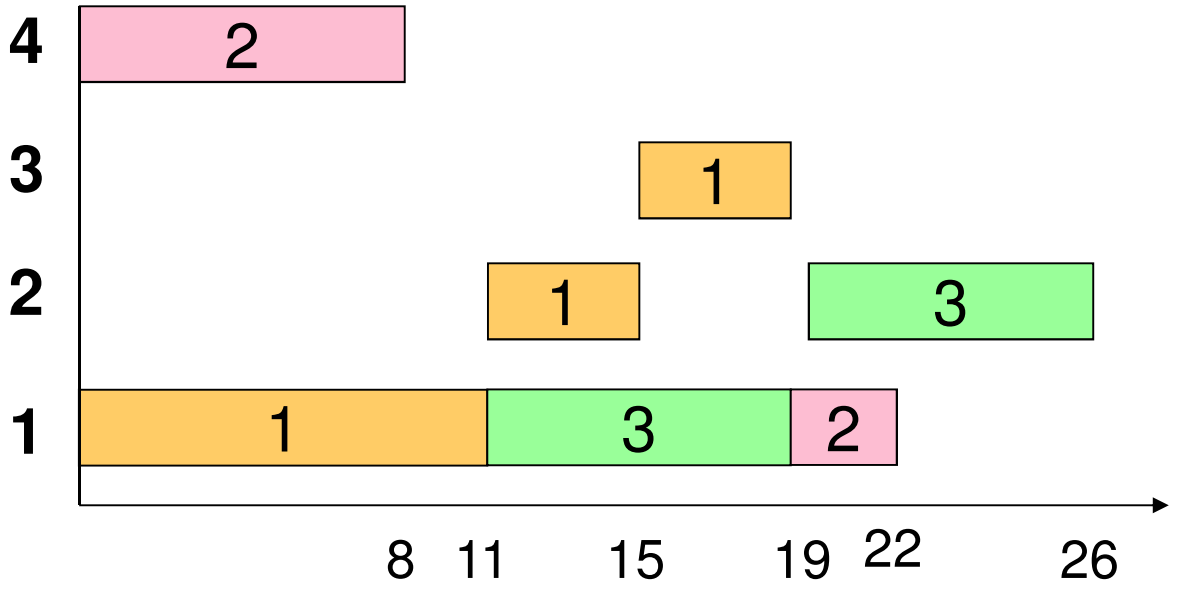


Esempio: FCFS

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

t = 22:

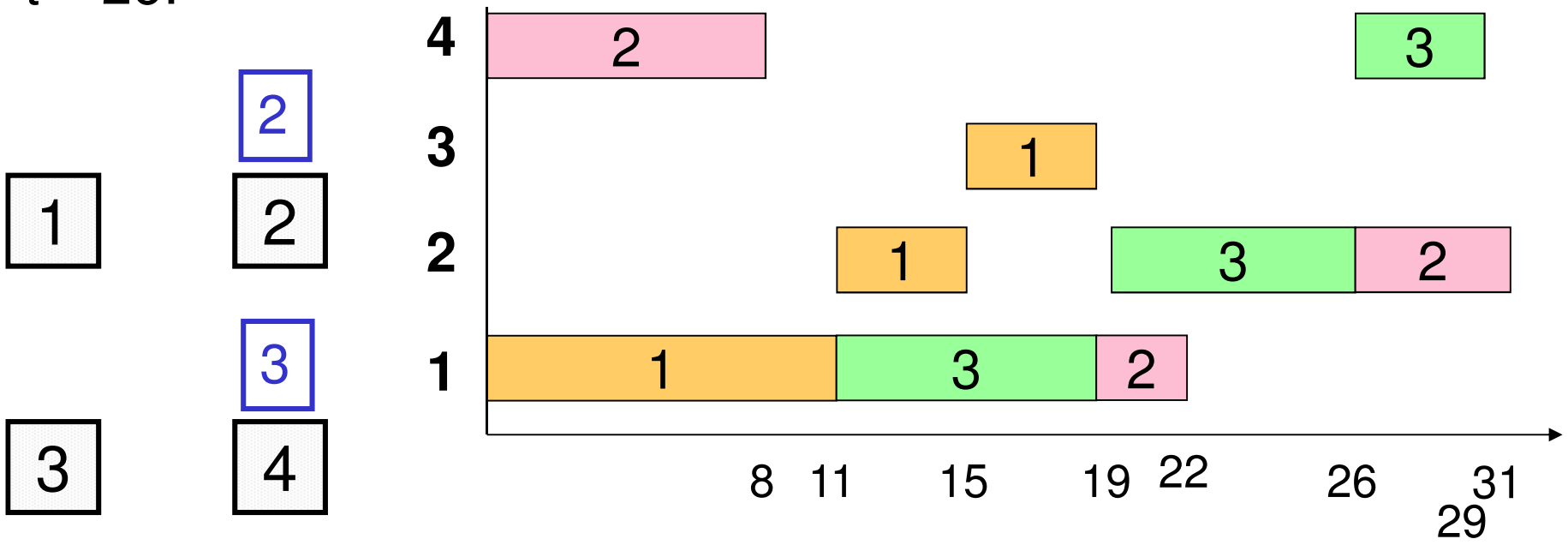




Esempio: FCFS

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

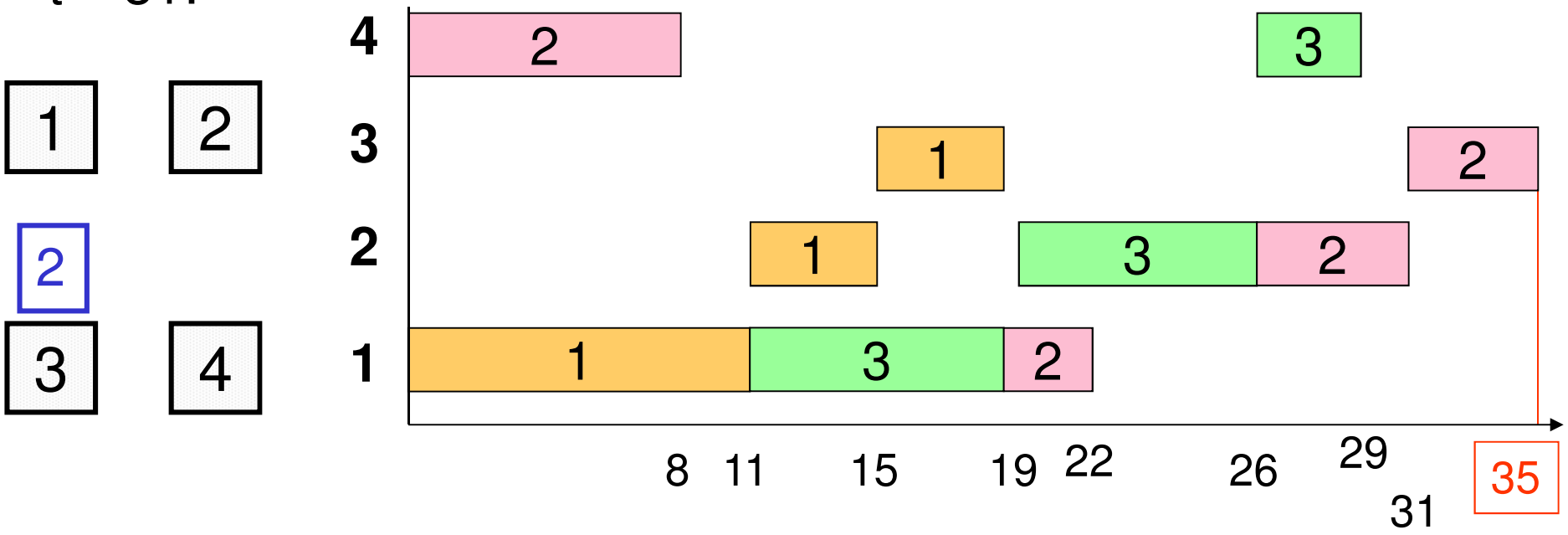
t = 26:



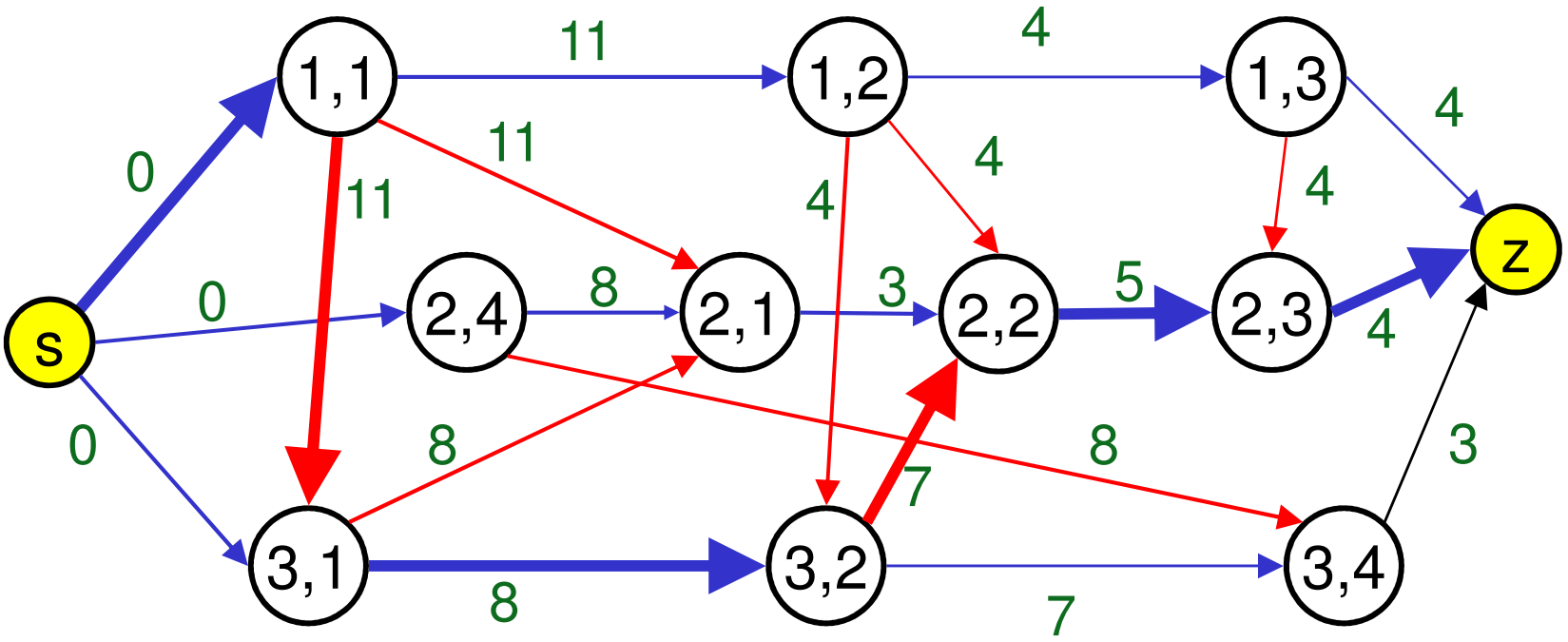
Esempio: FCFS

job	$m_j(1)$ [$p_{m(1),j}$]	$m_j(2)$ [$p_{m(2),j}$]	$m_j(3)$ [$p_{M(3),j}$]	$m_j(4)$ [$p_{M(4),j}$]
1	1 [11]	2 [4]	3 [4]	
2	4 [8]	1 [3]	2 [5]	3 [4]
3	1[8]	2 [7]	4 [3]	

t = 31:



Selezione completa FCFS



Cammino critico s-(1,1)-(3,1)-(3,2)-(2,2)-(2,3)-z di lunghezza 35

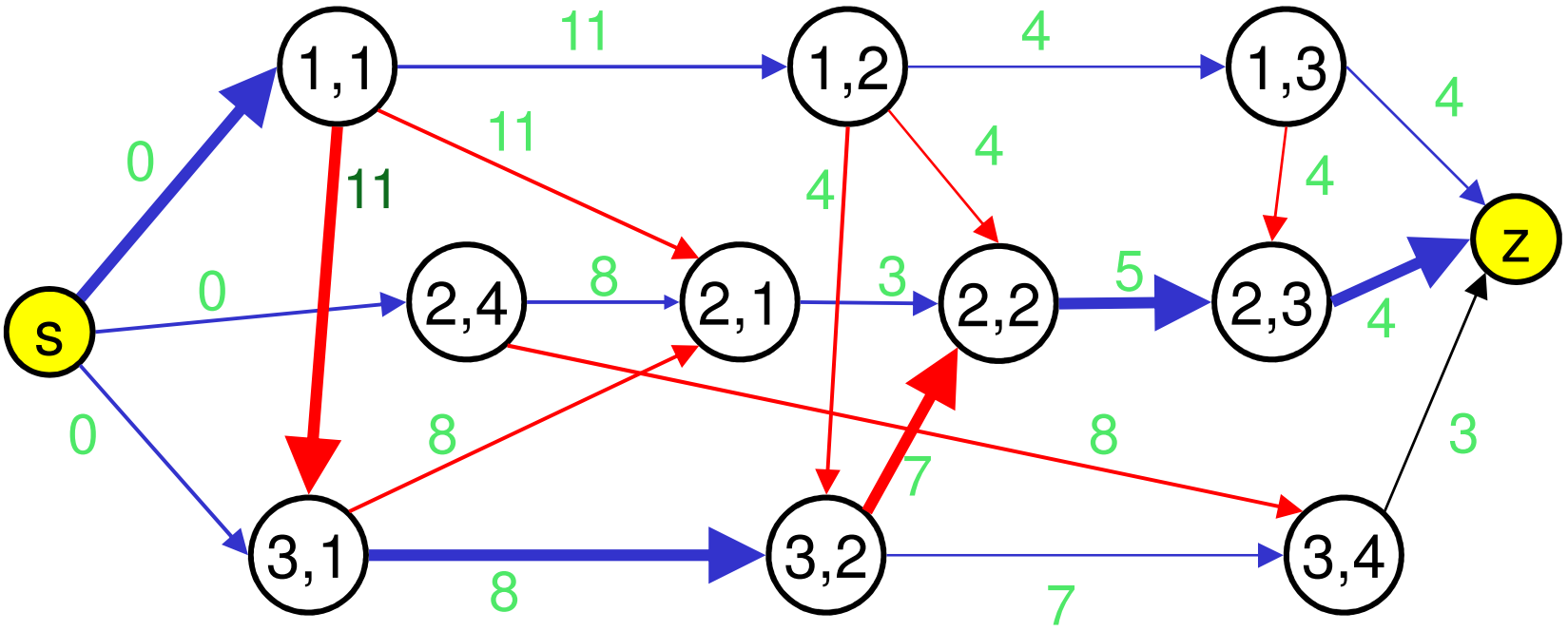
Ricerca Locale

Teorema. Sia S una soluzione ammissibile (selezione completa aciclica) e (i, j) un arco disgiuntivo di un cammino critico del grafo $G(S)$. Allora, sostituendo (i, j) con (j, i) si ottiene una nuova selezione completa aciclica (cioè una nuova soluzione ammissibile)

Il vicinato di una soluzione S è definito come l'insieme delle soluzioni ottenibili scambiando il verso di un arco disgiuntivo appartenente ad un cammino critico.

Teorema. Per ogni soluzione S esiste una sequenza finita di mosse che genera una soluzione ottima

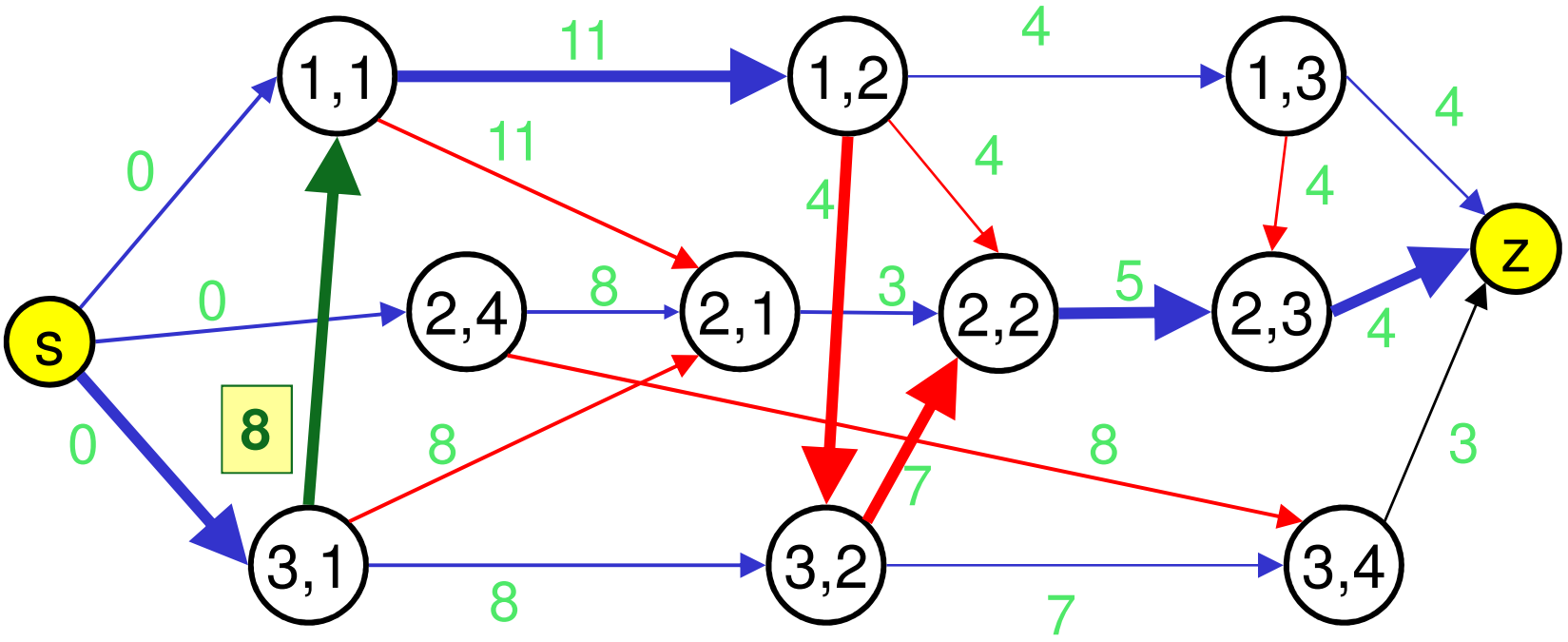
FCFS: soluzione di partenza



Cammino critico s-(1,1)-(3,1)-(3,2)-(2,2)-(2,3)-z di lunghezza 35

Due possibili mosse: (1,1)-(3,1) oppure (3,2)-(2,2). La seconda produce una soluzione di costo 37 (provare)

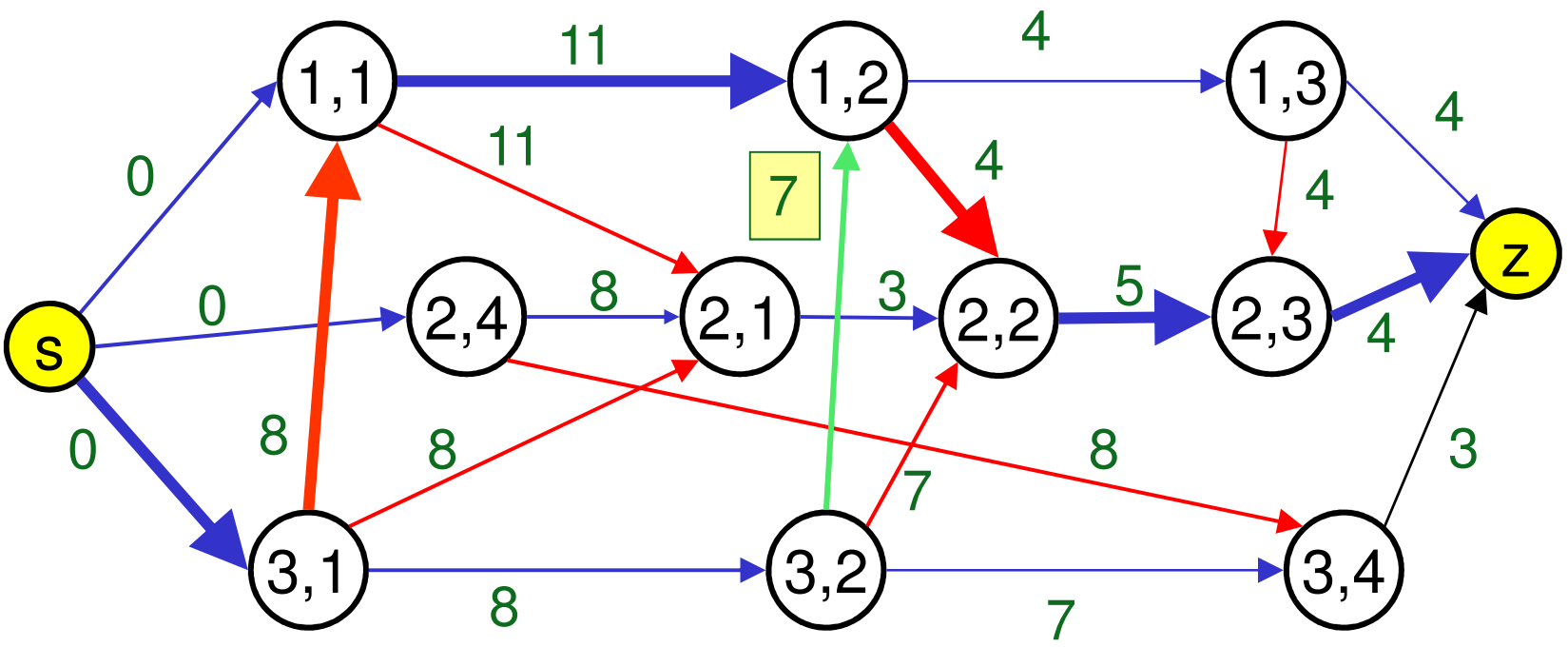
mossa 1: peggiorativa



Scambio arco (1,1)-(3,1) di lunghezza 11 con (3,1)-(1,1) di lunghezza 8

Cammino critico s-(3,1)-(1,1)-(1,2)-(3,2)-(2,2)-(2,3)-z di lunghezza 39

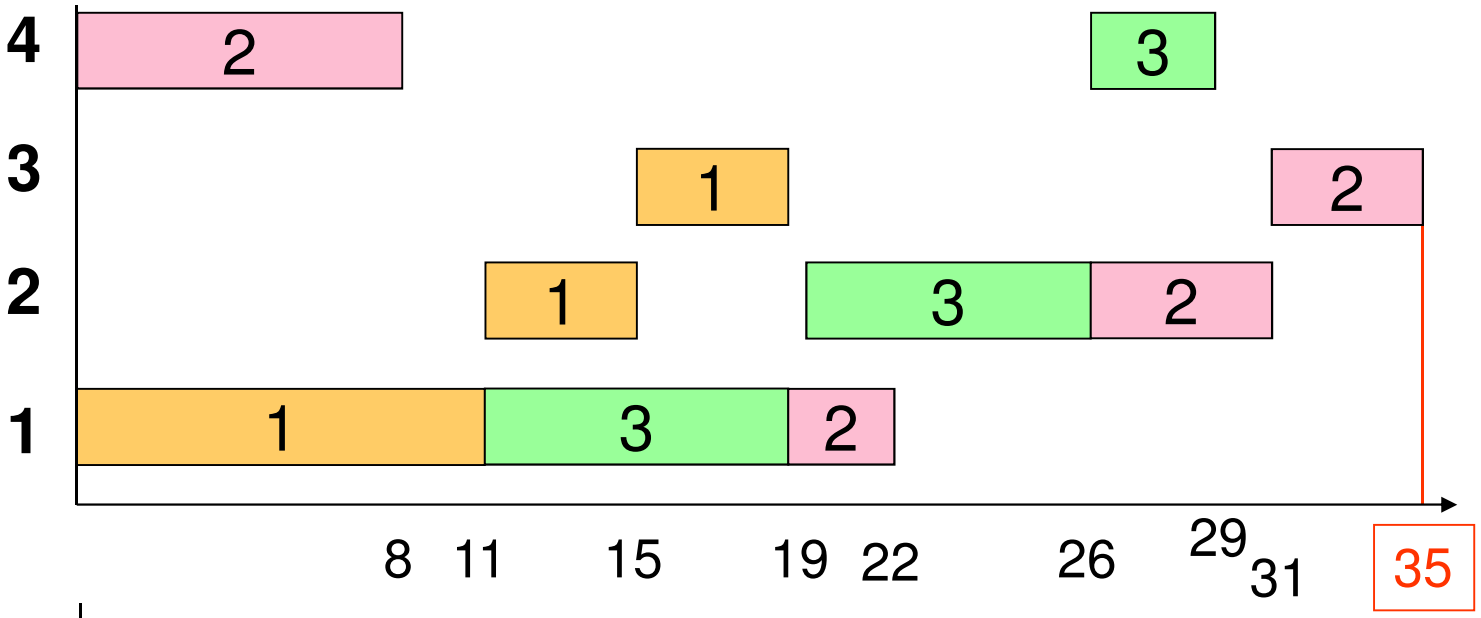
mossa 2



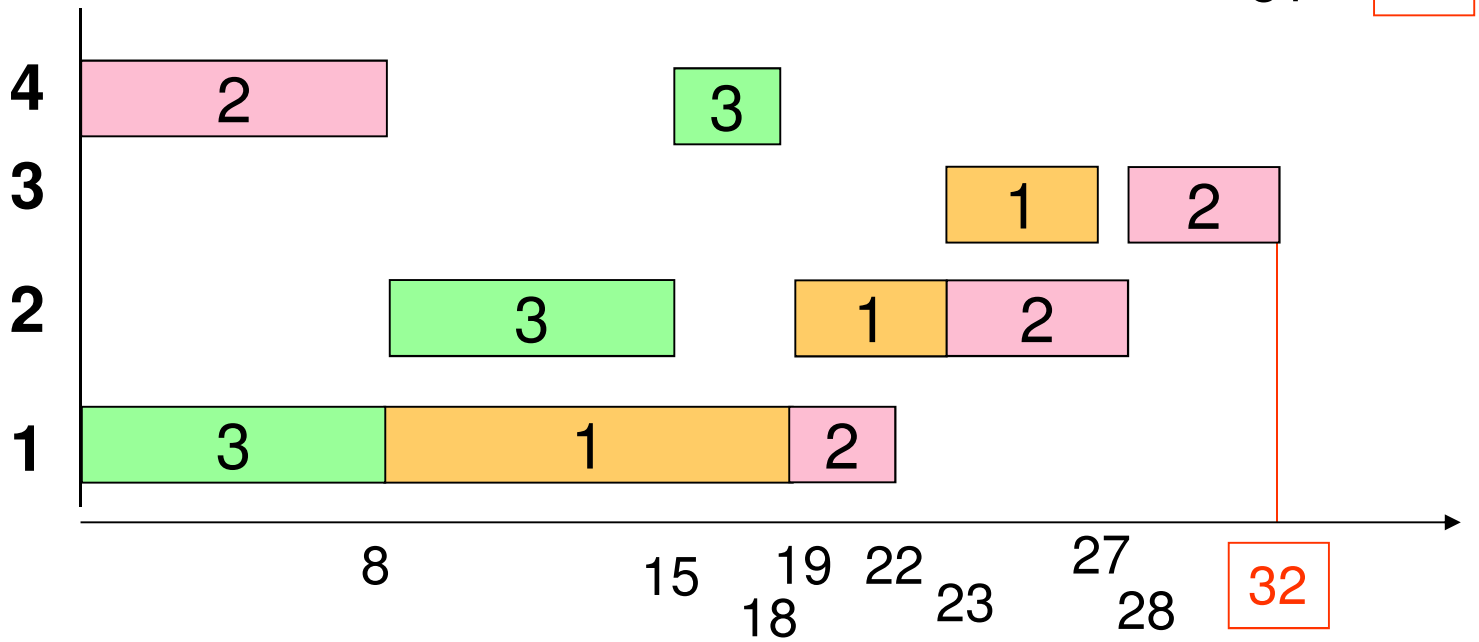
Scambio arco (1,2)-(3,2) di lunghezza 4 con (3,2)-(1,2) di lunghezza 7

Cammino critico s-(3,1)-(1,1)-(1,2)-(2,2)-(2,3)-z di lunghezza 32 (!!)

Soluzione dopo 2 mosse: migliora FCFS



FCFS



Ricerca Locale