“Sapienza” Università di Roma - Dipartimento di Ingegneria Informatica, Automatica e Gestionale

# *Ottimizzazione dei Progetti*

*Renato Bruni*
**bruni@dis.uniroma1.it**

**Il materiale presentato è derivato da quello dei proff. A. Sassano e C. Mannino**

# I Progetti

• Un numero sempre crescente di attività è organizzato "*per progetto*" – tutte quelle per cui è determinato un *inizio* e una *fine*.

**Caratteristiche dei progetti**

Obiettivo: un prodotto finale, un risultato o un output, tipicamente definibili in termini di tempo, costo e qualità.

Unicità: un progetto (anche quelli ripetibili) è unico per risorse disponibili e ambiente di sviluppo

Temporaneità: i progetti hanno date di inizio e termine prestabilite e le organizzazioni vengono create appositamente per la durata del progetto

Multidisciplinarità: i progetti richiedono molteplici competenze che devono essere coordinate. Le relazioni fra i vari task del progetto possono essere molto complicate.

Incertezza: i progetti devono essere pianificati *prima* della loro realizzazione

Risorse Limitate: un progetto utilizza risorse costose e limitate

# Esempi di progetti

Esempi storici
- Piramidi egizie
- Muraglia cinese
- …

Esempi moderni
- Manhattan Project (bomba atomica)
- Progetto Apollo Moon
- Tunnel sotto la Manica
- Olimpiadi o Campionati mondiali
- Ricostruzione Torri dopo l'attacco terroristico
- Riconversione IBM a fornitore di servizi software
- …

# Progetti Tipici

Costruzione di aeroporti

Costruzione grandi impianti sportivi

Progetto di un nuovo aeromobile

Sviluppo di sistemi informativi

Costruzione di impianti produttivi

Introduzione di un nuovo prodotto sul mercato

Reingegnerizzazione di un'azienda

Sviluppo un nuovo prodotto software

Costruzione di una diga, un ponte, un'autostrada

Ricerche Scientifiche

Campagne Pubblicitarie

Project Management (*PM*)*:* processo di gestione, allocazione e temporizzazione delle risorse per ottenere obiettivi predefiniti in modo efficiente (*Badiru 1991*)

# Ciclo di Vita di un progetto 1/2

1. Disegno concettuale: l'organizzazione concepisce la necessità di un progetto o riceve richieste da un cliente, tipicamente in forma scritta (*request for proposal* – (RFP)). Selezione del progetto.

2. Definizione: in questa fase vengono stabiliti gli obiettivi e la strategia.

   a. Obiettivi: contenuti del progetto e dei suoi prodotti finali

   b. Strategia: definizione di massima di come il progetto realizzerà gli obiettivi e soddisferà le misure di qualità

3. Pianificazione: il progetto viene decomposto in unità controllabili (*work packages*) formati da specifiche attività *(Work Breakdown Structure*).

   Per ogni attività si stabilisce la richiesta di risorse e la loro disponibilità, la durata e i rapporti di precedenza con le altre attività. Si stimano i costi. Si costruisce così la *rete del progetto* (*project network*).

# Ciclo di Vita di un progetto 2/2

4. Sequenziamento (*Scheduling*): definizione del *project base plan*. Vengono determinati i tempi di inizio e fine delle singole attività in base alla disponibilità di risorse, alle durate, ai rapporti di precedenza, alla funzione obiettivo.

5. Esecuzione e controllo: implementazione del *project base plan*. Se si rivelano ritardi o sforamenti nel budget bisogna apportare delle correzioni. In genere è difficile trovare dei corretti indicatori dello stato di un progetto.

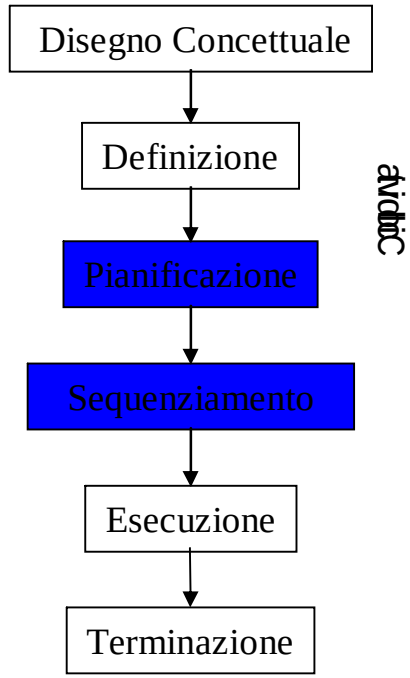
6. Terminazione: consegna del prodotto o del servizio

# Ciclo di vita / Pianificazione e Sequenziamento

- Costruzione della *Work Breakdown Structure* (WBS)
- Vengono definite risorse umane, tempi e risorse finanziarie
- La pianificazione dettagliata in genere riguarda poche settimane o mesi, a causa dell'incertezza sulla disponibilità futura delle risorse necessarie: questa fase deve essere rivista continuamente.

PASSI
FONDAMENTALI

1. *Strutturazione*: definizione delle attività e delle relazioni di precedenza; definizione dell'*organigramma aziendale* e del budget di progetto.
2. *Sequenziamento*: la definizione del piano temporale, ovvero gli istanti in cui le attività cominciano e terminano; contestualmente viene definita *l'allocazione delle risorse*: le risorse (limitate) devono essere assegnate alle singole attività e i conflitti devono essere risolti.



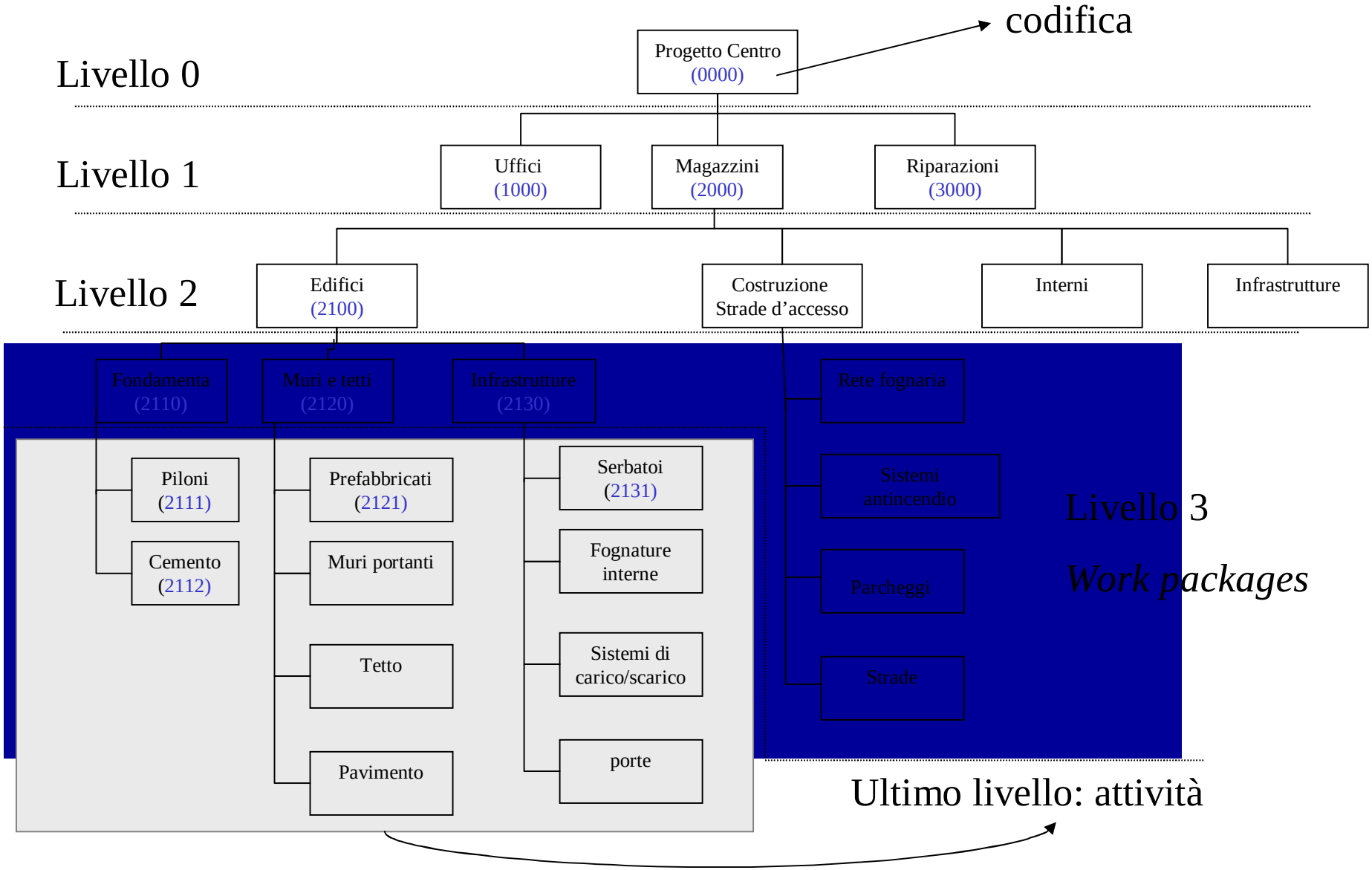
# Pianificazione: Work Breakdown Structure

Rappresentazione del *Work Breakdown*: la *Work Breakdown Structure* (*WBS*)

- La *WBS* è una decomposizione gerarchica delle operazioni del progetto.
- La *WBS* viene rappresentata attraverso un albero.
- La distanza di un nodo dalla radice viene detta *livello*.
- I nodi a uno stesso livello rappresentano uno stesso grado di decomposizione delle operazioni.
- A ogni nodo è associata univocamente un'etichetta che identifica il livello e la posizione nel livello

# Pianificazione: Work Breakdown Structure

Progetto di costruzione di un centro di stoccaggio.



# Reti di attività

# Definizioni di base

Il Progetto è costituito di *eventi* (o pietre miliari) e di *attività* (o *task*) che devono essere eseguite rispettando vincoli di precedenza.

Ogni attività ha una durata e richiede risorse (eventualmente nulle)

Un evento si riferisce a un insieme di attività che devono essere completate in un certo istante di tempo

Risorse sono, ad esempio, il credito, la forza lavoro, le macchine, l'equipaggiamento, l'energia, lo spazio, ecc.

Vincoli di precedenza fra attività: il più tipico è il vincolo *fine/inizio*: un'attività non può cominciare prima che un'altra sia finita.

Altri esempi di relazioni temporali: *inizio/inizio*, *fine/fine*, *inizio/fine*, con o senza intervalli di tempo (*time lag*).

Tempo: il tempo viene tipicamente misurato in unità prestabilite (ore, giorni, …).

Orizzonte temporale l'insieme dei periodi richiesti per l'esecuzione del progetto o disponibili a priori se in presenza di scadenze (*deadline*).

# Tecniche reticolari

Scopi:

1. Definire lo *schedule* delle attività, cioè associare a ogni attività un istante iniziale.
2. Permettere l'analisi dei ritardi, cioè determinare la durata complessiva del progetto e le attività che la determinano.

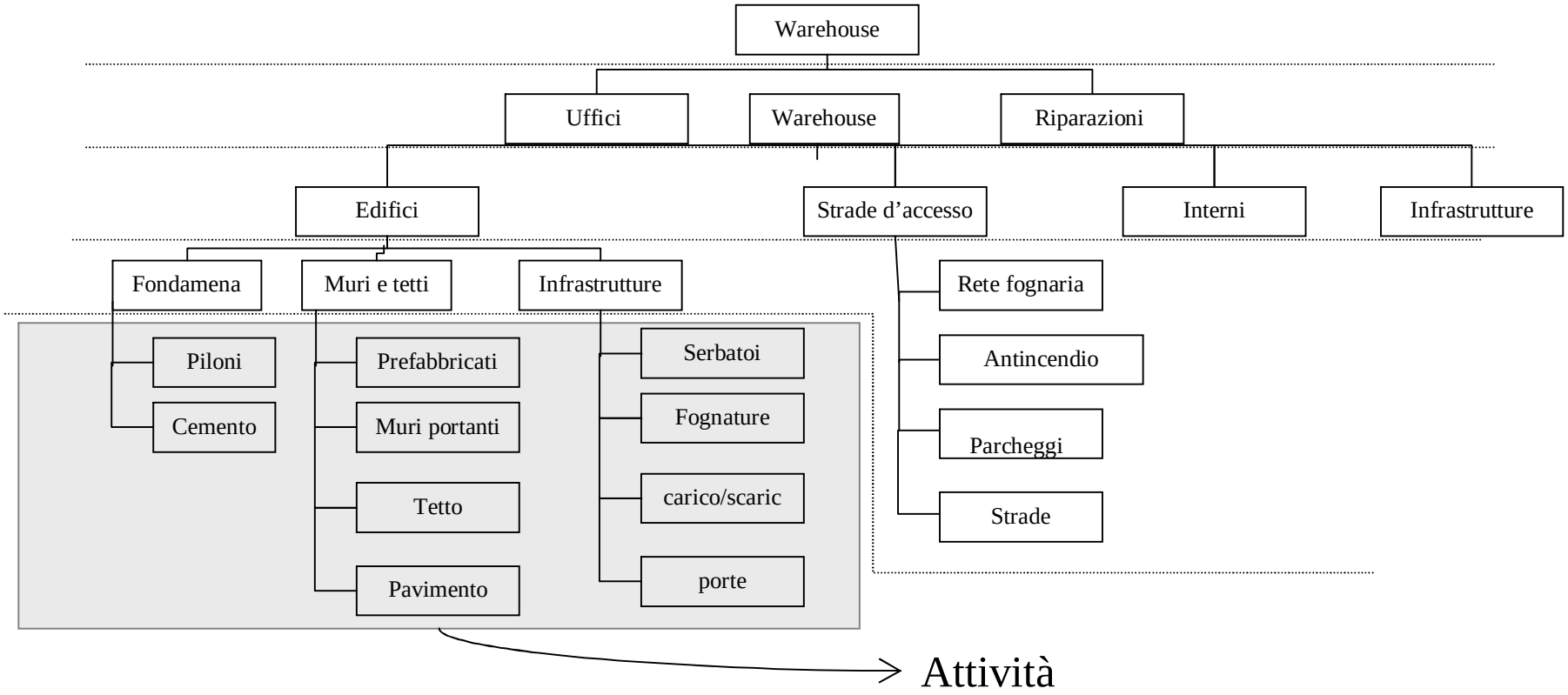
Input:

1. L'insieme delle attività del progetto
2. La durata di ciascuna attività (certa o incerta)
3. Le precedenze temporali.

Metodi classici:

1. CPM (*critical path method*): durate certe e relazioni temporali di fine/inizio (certe attività possono cominciare solo dopo che altre sono terminate).
2. MPM (*metra potential method*): come il CPM, ma con relazioni temporali più complicate (inizio/inizio, fine/fine, inizio/fine)
3. PERT (*program evaluation & review techique*): come CPM ma durate stocastiche.
4. .... GERT, VERT, ....)

# Input: Identificazione delle attività




- Le attività vengono identificate nella fase di pianificazione e rappresentate nella WBS (nelle foglie).
- Per ogni attività si devono determinare la durata e i rapporti di interdipendenza temporali con le altre attività, e le risorse richieste.
- Le attività e i rapporti di precedenza vengono rappresentati con un grafo orientato

# Un esempio

Per la produzione di un nuovo prodotto sono state individuate le seguenti attività con relativi tempi di completamento

| Attività | Descrizione | Tempo |
|---|---|---|
| 1 | Disegno del prodotto | (6 settimane) |
| 2 | Disegno della confezione | (2 settimane) |
| 3 | Ordine e ricezione dei materiali per prodotto | (3 settimane) |
| 4 | Ordine e ricezione dei materiali per confezione | (2 settimane) |
| 5 | Assemblaggio dei prodotti | (4 settimane) |
| 6 | Preparazione delle Confezioni | (1 settimana) |
| 7 | Impacchettamento dei prodotti | (1 settimana) |
| 8 | Test di mercato del prodotto | (6 settimane) |
| 9 | Revisione del prodotto | (3 settimane) |
| 10 | Revisione della confezione | (1 settimana) |
| 11 | Presentazione dei risultati al CdA | (1 settimana) |

Descriviamo le relazioni di precedenza (immediate). 1 deve finire prima che 3 cominci (1 < 3). Tutte le relazioni sono di tipo Finish/Start

$2 < 4,\ 3 < 5,\ 4 < 6,\ 5 < 7,\ 6 < 7,\ 7 < 8,\ 8 < 9,\ 8 < 10,\ 9 < 11,\ 10 < 11.$

# Rete di attività

- La *rete delle attività* (*rete delle precedenze*, *grafo dei vincoli*) è un grafo orientato che rappresenta le relazioni di precedenza fra attività.
- Esistono due possibili rappresentazioni:

Rete delle attività

$G = (V, A)$

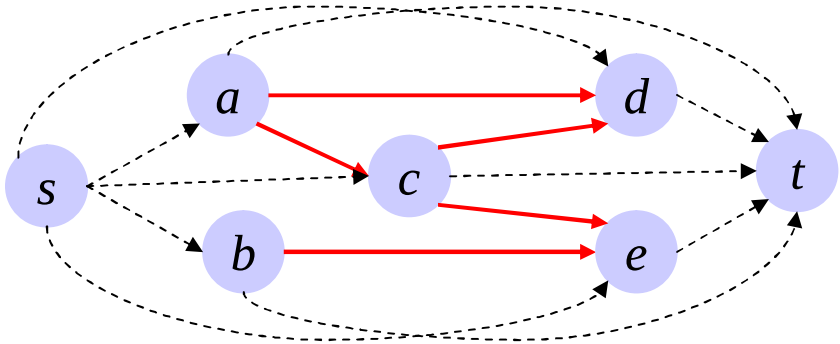
- *Activity-on-arc*: gli archi sono attività e i nodi eventi
- *Activity-on-node*: i nodi sono attività ed eventi, gli archi rappresentano vincoli di precedenza

# Activity on Nodes (*AoN*)

Nodi rappresentano attività + due nodi fittizi (*inizio* e *fine* progetto).

Archi rappresentano relazioni di precedenza.

Es. 5 attività {*a*, *b*, *c*, *d*, *e*}: precedenze (semplici) $a < c$ , $a < d$, $b < e$, $c < d$, $c < e$.




I due nodi fittizi (inizio e fine progetto) vanno sempre aggiunti e sono anche utilizzati per rappresentare complesse relazioni di precedenza.

• L'attività *inizio progetto* ha durata nulla e finisce prima dell'inizio di qualunque altra attività del progetto

• L'attività *fine progetto* ha durata nulla e comincia dopo la fine di qualunque altra attività del progetto

# Altre relazioni di precedenza

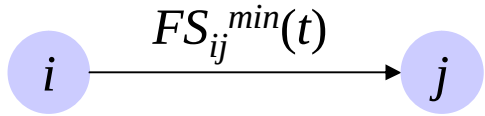
Finora si è vista solo una sola relazione di precedenza, la cosiddetta

*Finish/Start* (*b* comincia dopo che *a* è terminato).

Questa è l'unica facilmente rappresentabile nello standard *AoA*.

Ne esistono altre, più complesse, rappresentabili con *AoN*: ad esempio, un'attività può cominciare *un certo tempo* dopo che un'altra è terminata, oppure deve essere eseguita in parallelo.

• *Finish – Start*: $FS_{ij}$ l'attività *j* deve cominciare dopo che l'attività *i è finita*. Se è richiesto un tempo *t* di attesa (*time lag*) minimo prima che *j* cominci, ad esempio per il *lead time* si scrive: $FS_{ij}^{min}(t)$




Esempio: si può posare il parquet solo dopo un po' di tempo che il pavimento è stato completato (per farlo asciugare)

Se invece si può aspettare *al massimo* un certo tempo *t* si scrive $FS_{ij}^{max}(t)$

Esempio: Tra lancio pubblicitario di un prodotto e sua effettiva presenza nei negozi non può passare più di un certo tempo

# Altre relazioni di precedenza

• *Start – Start*: $SS_{ij}^{min}(t)$ l'attività *j* deve cominciare almeno *t* unità di tempo dopo che l'attività *i* è cominciata. Analogamente $SS_{ij}^{max}(t)$ significa che *j* deve cominciare al più *t* unità di tempo dopo che l'attività *i* è cominciata

Esempio: stendere l'asfalto e spianarlo: la spianatura deve cominciare un po' dopo che si è cominciato a stendere l'asfalto ma non troppo dopo (altrimenti l'asfalto si raffredda).

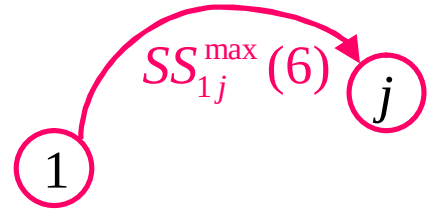
• *Finish – Finish*: $FF_{ij}^{min}(t)$, $FF_{ij}^{max}(t)$, l'attività *j* deve finire dopo che l'attività *i* è finita.

• *Start – Finish*: $SF_{ij}^{min}(t)$, $SF_{ij}^{max}(t)$, l'attività *j* deve finire dopo l'attività *i* è cominciata.

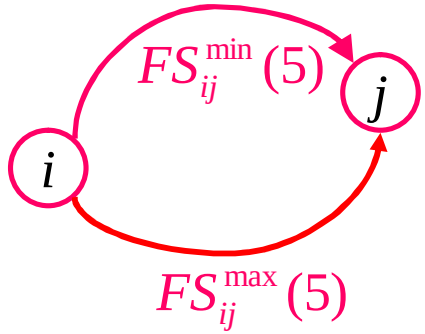
Esempio: un vecchio impianto produttivo può essere fermato solo dopo che quello nuovo ha cominciato a operare.

# Esempi di relazioni complesse

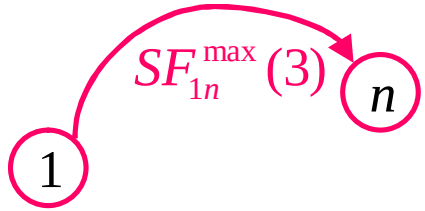
L'attività *j* deve cominciare al massimo dopo 6 settimane (unità di tempo) dopo l'inizio del progetto




L'attività *j* deve cominciare esattamente 5 settimane dopo che l'attività *i* è finita.




Il progetto deve terminare entro e non oltre tre settimane.

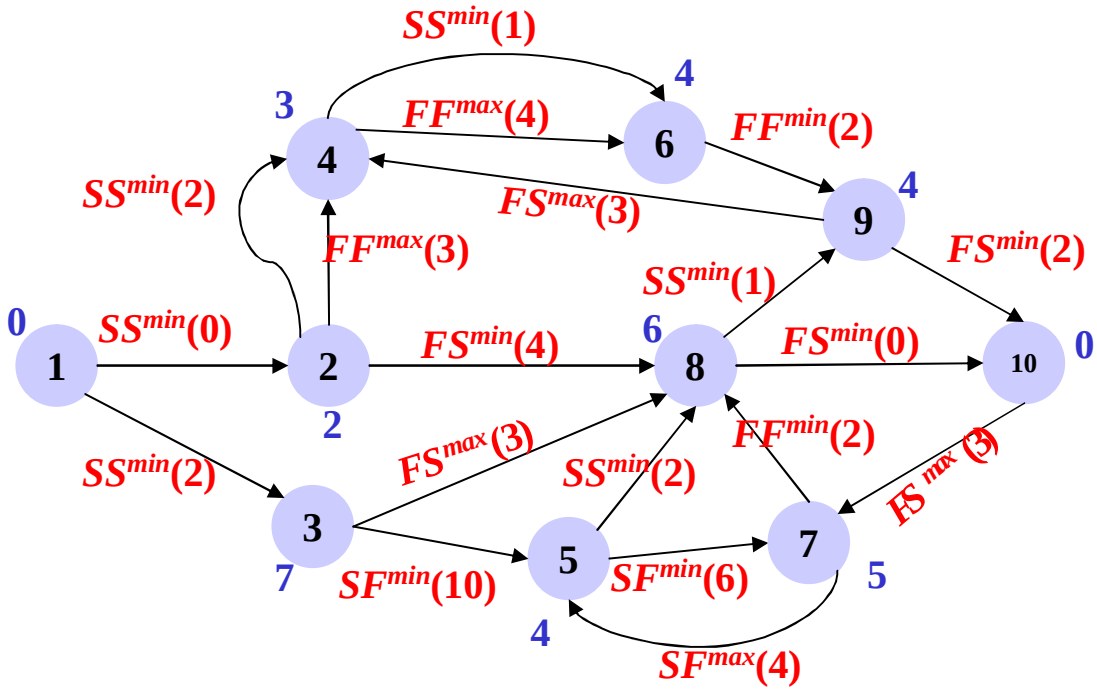


# Il grafo delle precedenze

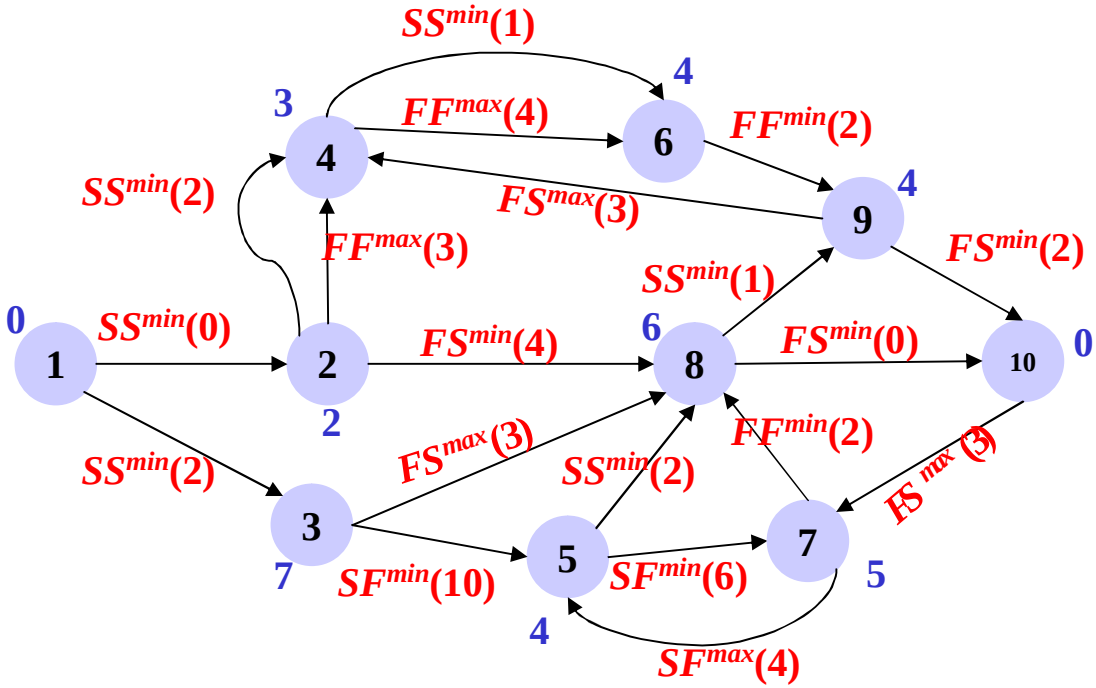
Le relazioni di precedenza generalizzate vengono rappresentate mediante il *grafo delle precedenze generalizzate* $H(V,F)$.

- $V = \{1,\ldots,n\}$ insieme delle attività, 1 inizio progetto, 10 fine progetto.

Esempio di grafo di precedenze (generalizzate) (De Reyck (1998))



# Nodi e precedenze fittizie




Le relazioni di precedenza fra i nodi fittizi e gli altri nodi del grafo vengono spesso omesse (tranne quelle “immediate ”) nelle rappresentazioni grafiche.

L’attività *inizio progetto* ha durata nulla e finisce prima dell’inizio di qualunque altra attività del progetto. Spesso vengono solo rappresentate le relazioni fra in nodo 1 e le attività (riconosciute come) iniziali.

L’attività *fine progetto* ha durata nulla e comincia dopo la fine di qualunque altra attività del progetto. Spesso vengono solo rappresentate le relazioni fra le attività riconosciute come *finali* e il nodo *n* di fine progetto.

In realtà, tutte le relazioni “fittizie” devono essere considerate negli algoritmi per la costruzione dei piani delle attività.

# Analisi temporale

L'analisi temporale serve a rispondere a una serie di quesiti quali:

- in quanto tempo il progetto (o una sua parte) terminerà?
- quando può (o deve) cominciare una determinata attività?
- quanto può essere ritardato l'inizio di un'attività senza rallentare l'intero progetto?

ASSUNZIONI DEL MODELLO

- Risorse: nessun vincolo
- Attività non interrompibili (*no preemption*)

OBIETTIVI:

- ➢ calcolare il minimo tempo di completamento del progetto (*makespan*)
- ➢ identificare le attività critiche e altre grandezze d'interesse

# Piano temporale delle attività

• L'input è il grafo delle precedenze generalizzate $H = (V, F)$ con insieme di attività $V$ e insieme di archi $F$ (precedenze generalizzate), unitamente al vettore delle durate delle attività $d \in R_+^{|V|}$

• Principale prodotto dell'ottimizzazione è il *piano temporale* delle attività (*schedule*) che soddisfi tutti i vincoli di precedenza e ottimizzi una specifica funzione obiettivo.

• Il piano può essere rappresentato associando a ogni attività $i \in V$ una variabile reale $s_i$ che indica *l'istante iniziale* (start) dell'attività.

• Il piano è quindi un vettore $s \in R^{|V|}$

• L'obiettivo è minimizzare la durata dell'intero progetto, ovvero la quantità:

$s_n - s_1$ (inizio attività fine progetto – inizio attività inizio progetto)

# Problema del makespan

*Def.* *Makespan*: durata minima del progetto

Problema del calcolo del *makespan*: trovare un piano $s$ che soddisfi tutti i vincoli di precedenza e minimizzi la durata del progetto $s_n - s_1$ .

• Questo problema può essere formulato come problema di PL, costruito a partire dal grafo delle precedenze generalizzate.

• Introduciamo anche la variabile $f_i$ che rappresenta *l'istante finale* (finish) dell'attività $i \in V$

• Valendo l'ipotesi di *no-preemption* si ha $f_i = s_i + d_i$ *per ogni* $i \in V$.

# Un modello di PL

• Il grafo delle precedenze generalizzate è la base di partenza per costruire un modello di Programmazione lineare

• Le variabili del modello sono associate ai nodi del grafo

• I vincoli del modello sono associati agli archi del grafo

• Le relazioni di precedenza sono tradotte in vincoli lineari sulle variabili $s$ e $f$

• Inoltre, essendo $f_i = s_i + d_i$ *per ogni* $i \in V$, le variabili $f$ possono essere eliminate

• Obiettivo: rappresentare i vincoli temporali come vincoli lineari della forma $Ms \geq l$ dove $l$ è il vettore dei termini noti e $M$ la matrice dei vincoli

• I vincoli avranno tutti forma $s_j - s_i \geq l_{ij}$ corrispondenti alla relazione $SS_{ij}^{min}(l_{ij})$

• I coefficienti $l_{ij}$ possono assumere valori positivi, negativi o nulli

# Determinazione vincoli I

1. **Relazione FINISH/START** $FS_{ij}^{min}$ , $FS_{ij}^{max}$

   a. *j* deve cominciare almeno $FS_{ij}^{min}$ istanti dopo che *i* finisce:

$$s_j \geq f_i + FS_{ij}^{min} \quad \rightarrow \quad s_j - s_i \geq d_i + FS_{ij}^{min}$$

   b. *j* deve cominciare al massimo $FS_{ij}^{max}$ istanti dopo che *i* finisce

$$s_j \leq f_i + FS_{ij}^{max} \quad \rightarrow \quad s_i - s_j \geq - d_i - FS_{ij}^{max}$$

2. **Relazione START/START** $SS_{ij}^{min}$ , $SS_{ij}^{max}$

   a. *j* deve cominciare almeno $SS_{ij}^{min}$ istanti dopo che *i* è cominciata:

$$s_j \geq s_i + SS_{ij}^{min} \quad \rightarrow \quad s_j - s_i \geq SS_{ij}^{min}$$

   b. *j* deve cominciare al massimo $SS_{ij}^{max}$ istanti dopo che *i* è cominciata

$$s_j \leq s_i + SS_{ij}^{max} \quad \rightarrow \quad s_i - s_j \geq - SS_{ij}^{max}$$

# Determinazione vincoli II

3. **Relazione START/FINISH** $SF_{ij}^{min}$ , $SF_{ij}^{max}$

   a. *j* deve finire almeno $SF_{ij}^{min}$ istanti dopo che *i* è cominciata:

$$f_j \geq s_i + SF_{ij}^{min} \quad \rightarrow \quad s_j - s_i \geq SF_{ij}^{min} - d_j$$

   b. *j* deve finire al massimo $SF_{ij}^{max}$ istanti dopo che *i* è cominciata

$$f_j \leq s_i + SF_{ij}^{max} \quad \rightarrow \quad s_i - s_j \geq d_j - SF_{ij}^{max}$$

4. **Relazione FINISH/FINISH** $FF_{ij}^{min}$ , $FF_{ij}^{max}$

   a. *j* deve finire almeno $FF_{ij}^{min}$ istanti dopo che *i* è finita:

$$f_j \geq f_i + FF_{ij}^{min} \quad \rightarrow \quad s_j - s_i \geq d_i - d_j + FF_{ij}^{min}$$

   b. *j* deve finire al massimo $FF_{ij}^{max}$ istanti dopo che *i* è finita

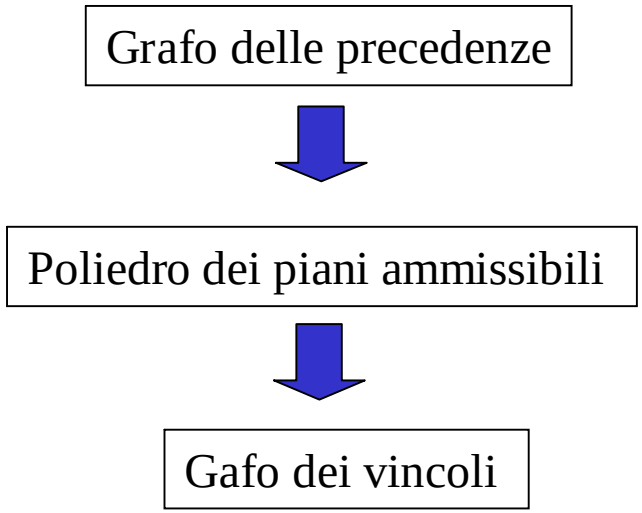
$$f_j \leq f_i + FF_{ij}^{max} \quad \rightarrow \quad s_i - s_j \geq d_j - d_i - FF_{ij}^{max}$$

# Dal grafo precedenze a quello dei vincoli

• Questi vincoli lineari di precedenza definiscono un poliedro: il *poliedro dei piani ammissimibili*

• Al poliedro dei piani ammissibili è possibile associare un grafo orientato: *il grafo dei vincoli*.

• Le proprietà del grafo dei vincoli permettono di studiare il poliedro dei piani ammissibili e quindi la struttura delle soluzioni.



# Poliedro dei piani ammissibili e grafo dei vincoli

Obiettivo: definire il poliedro $\{s \in R^{|V|}: Ms \geq l\}$ dei piani ammissibili

Le precedenze si traducono in vincoli asssociati a coppie ordinate di attività del tipo

$$s_v - s_u \geq l_{uv} \qquad uv \in A \qquad \text{(Vincoli di precedenza)}$$

Se esistono due vincoli distinti sulla stessa coppia (ordinata) $uv$ di variabili

(1) $s_v - s_u \geq l_{uv}$ , (2) $s_v - s_u \geq l'_{uv}$

con $l'_{uv} \leq l_{uv}$ , il vincolo (2) può essere eliminato perché dominato.

La matrice $M$ ha esattamente un +1 e un -1 in ogni riga (tutti gli altri elementi sono nulli)

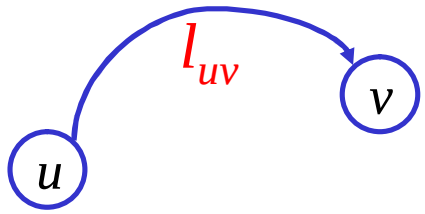
$M$ è la matrice d'incidenza archi-nodi di un grafo orientato, semplice:

grafo dei vincoli $G(V,A)$

Attività = Nodi $V$

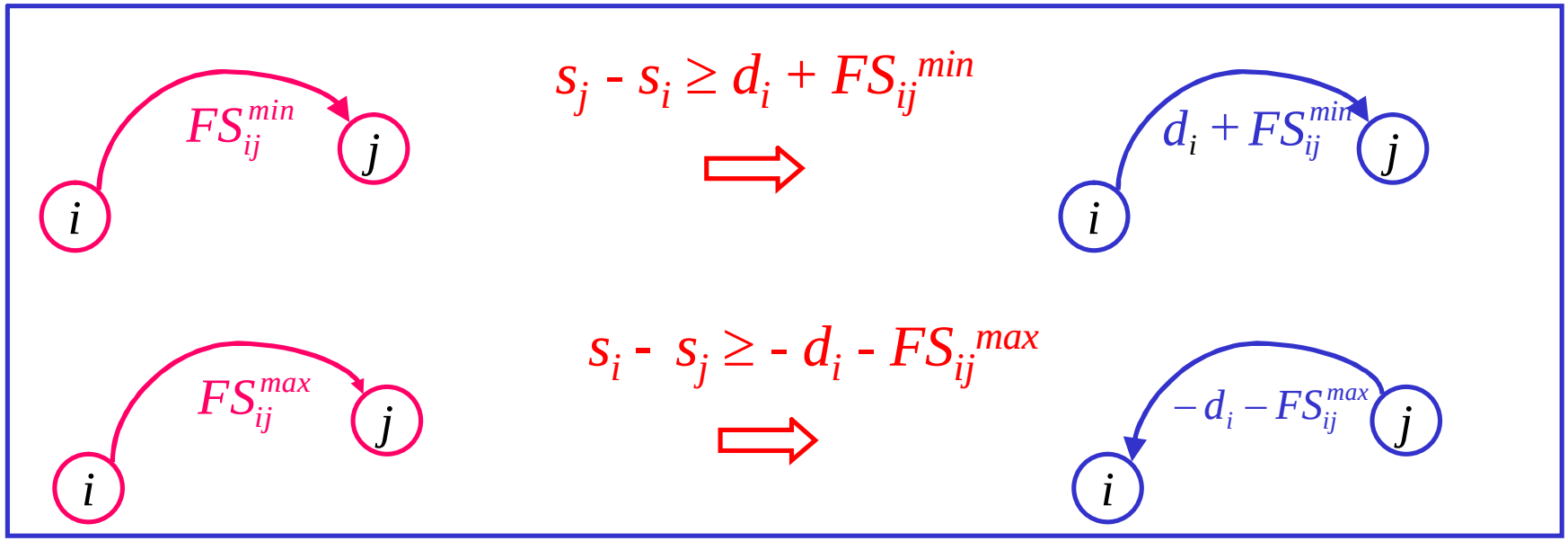
vincoli di precedenza = Archi $A$

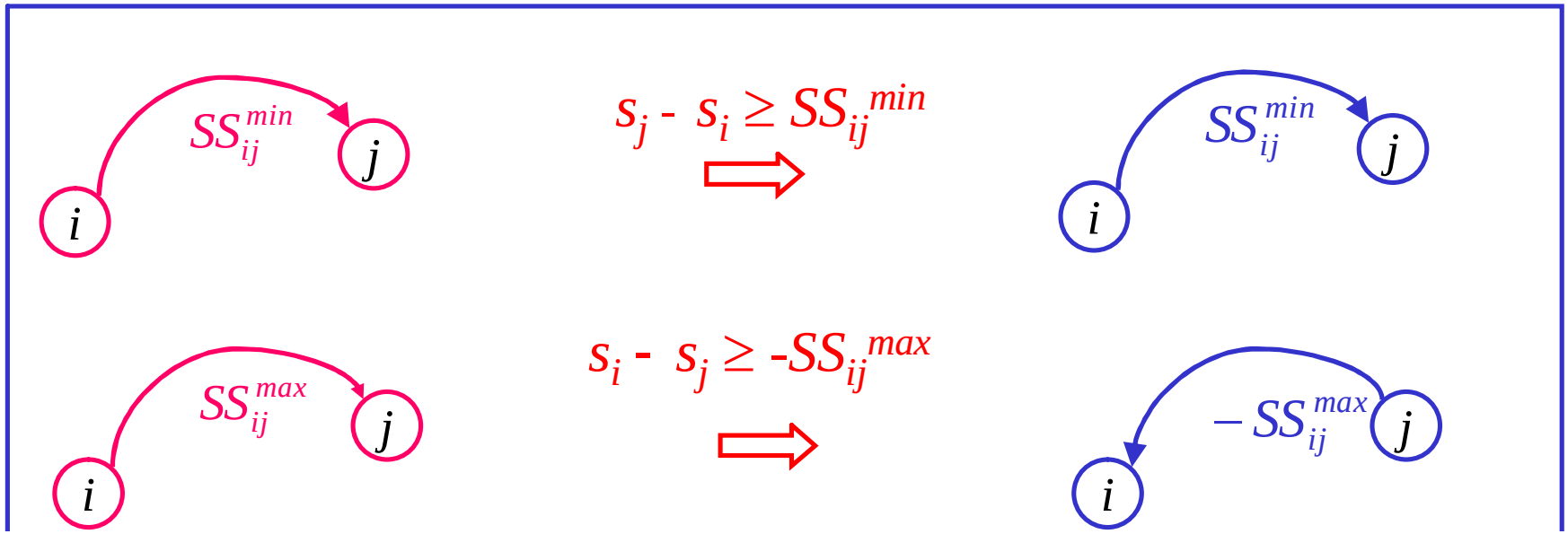
Temini noti: $l_{uv}$ peso dell'arco $uv \in A$



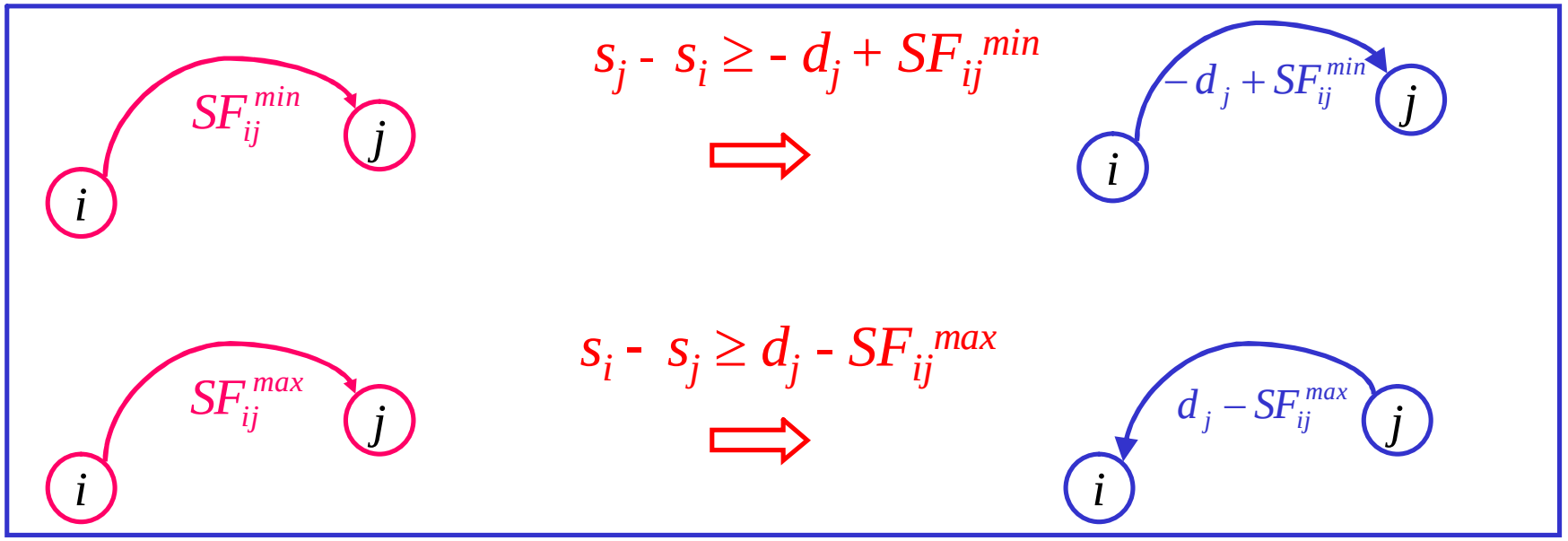
# Il grafo dei vincoli

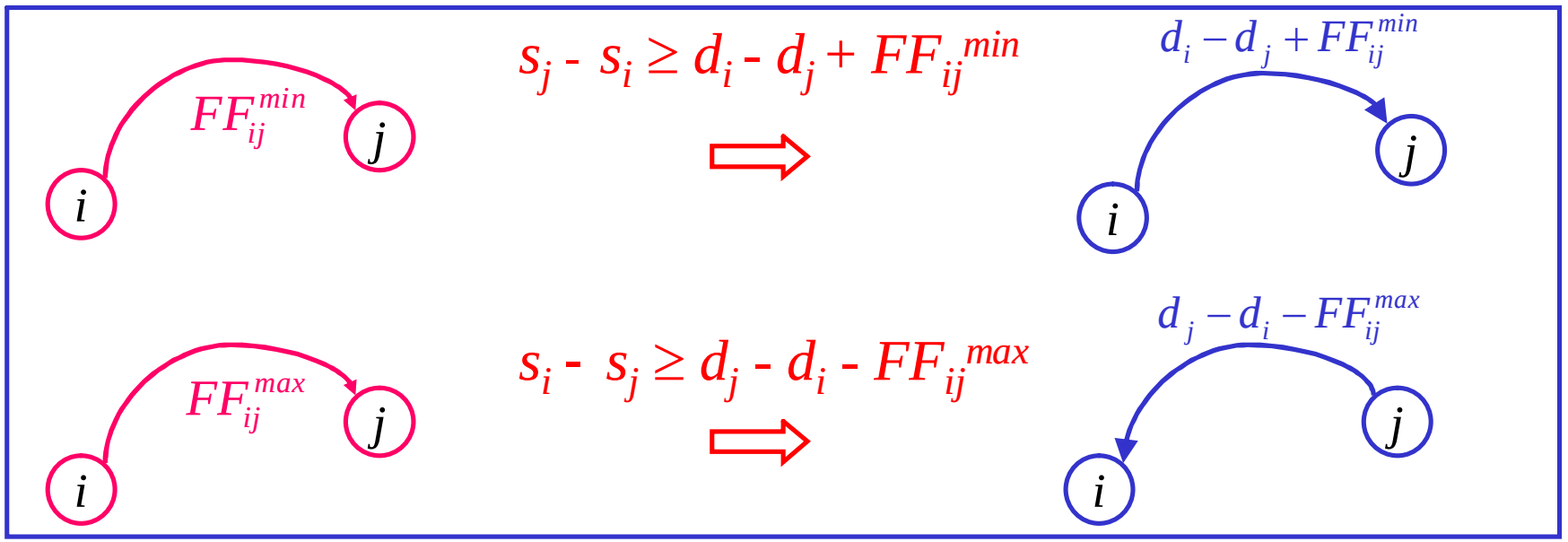
Costruzione del grafo dei vincoli a partire dal grafo delle precedenze generalizzate:



# Il grafo dei vincoli



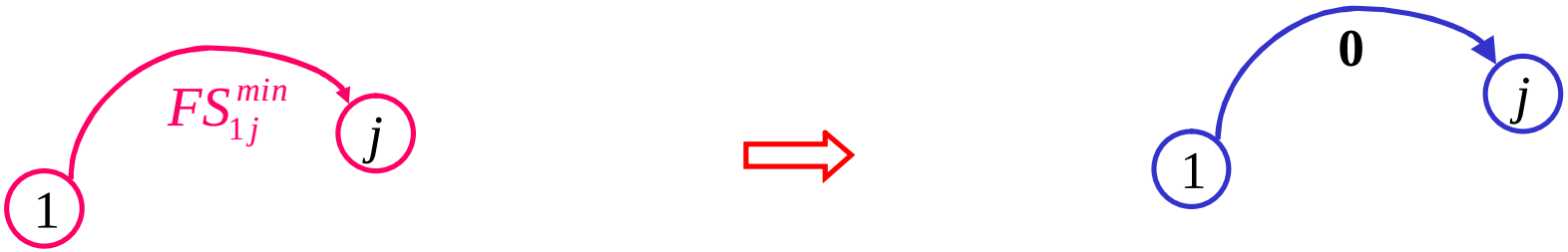



# I nodi inizio e fine progetto

Ogni nodo *j* è in relazione con i nodi 1 e *n*.

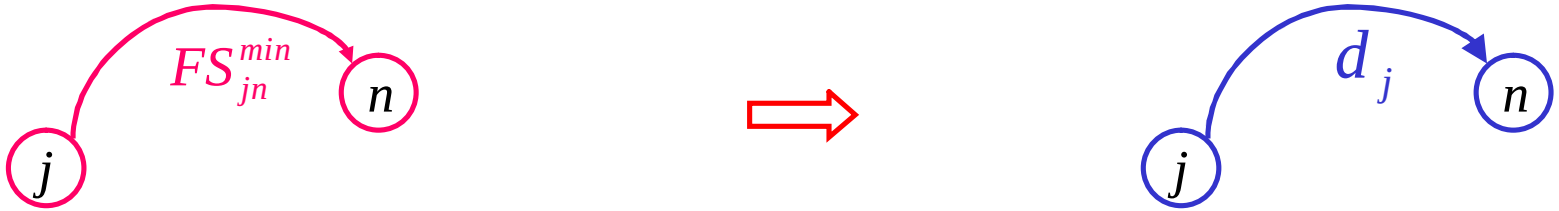
- Ogni nodo *j* *può cominciare solo* dopo che 1 è finito (time lag 0)

$$s_j \geq f_1 + FS_{1j}^{min} \rightarrow s_j - s_1 \geq d_1 + FS_{1j}^{min} \rightarrow s_j - s_1 \geq 0$$

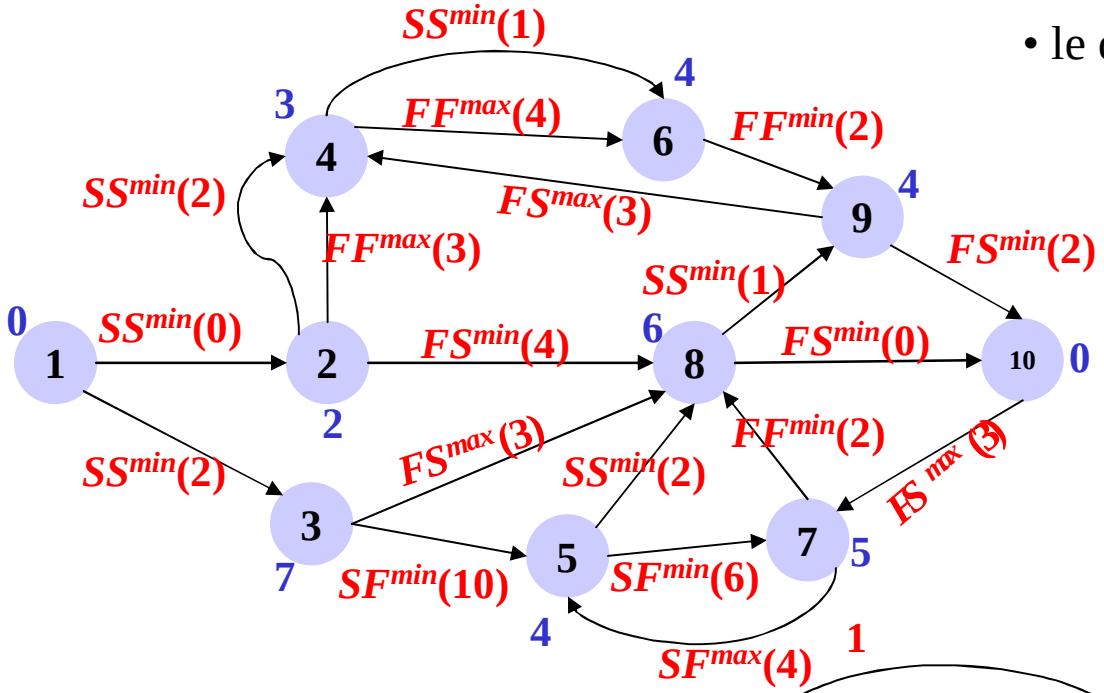



Il nodo *n* può cominciare solo dopo che ogni *j* è finito (time lag 0)

$$s_n \geq f_j + FS_{jn}^{min} \rightarrow s_n - s_j \geq d_j + FS_{jn}^{min} \rightarrow s_n - s_j \geq d_j$$



# Grafo dei vincoli




• le durate sono in blu accanto al nodo

$SS_{ij}^{min} \rightarrow \quad l_{ij} = SS_{ij}^{min}$

$SS_{ij}^{max} \rightarrow \quad l_{ji} = -SS_{ij}^{max}$

$SF_{ij}^{min} \rightarrow \quad l_{ij} = SF_{ij}^{min} - d_j$

$SF_{ij}^{max} \rightarrow \quad l_{ji} = d_j - SF_{ij}^{max}$

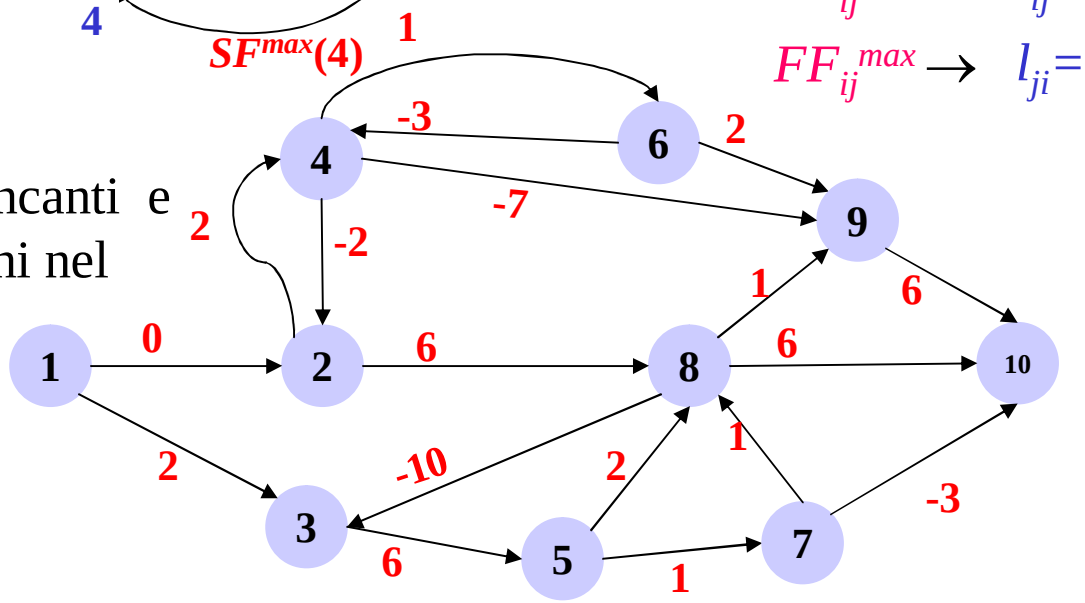
$FS_{ij}^{min} \rightarrow \quad l_{ij} = d_i + FS_{ij}^{min}$

$FS_{ij}^{max} \rightarrow \quad l_{ji} = -d_i - FS_{ij}^{max}$

$FF_{ij}^{min} \rightarrow \quad l_{ij} = d_i - d_j + FF_{ij}^{min}$

$FF_{ij}^{max} \rightarrow \quad l_{ji} = d_j - d_i - FF_{ij}^{max}$

• Vanno aggiunte le relazioni fittizie mancanti e i corrispondenti archi nel grafo dei vincoli




*Grafo dei vincoli*

# I nodi inizio e fine progetto

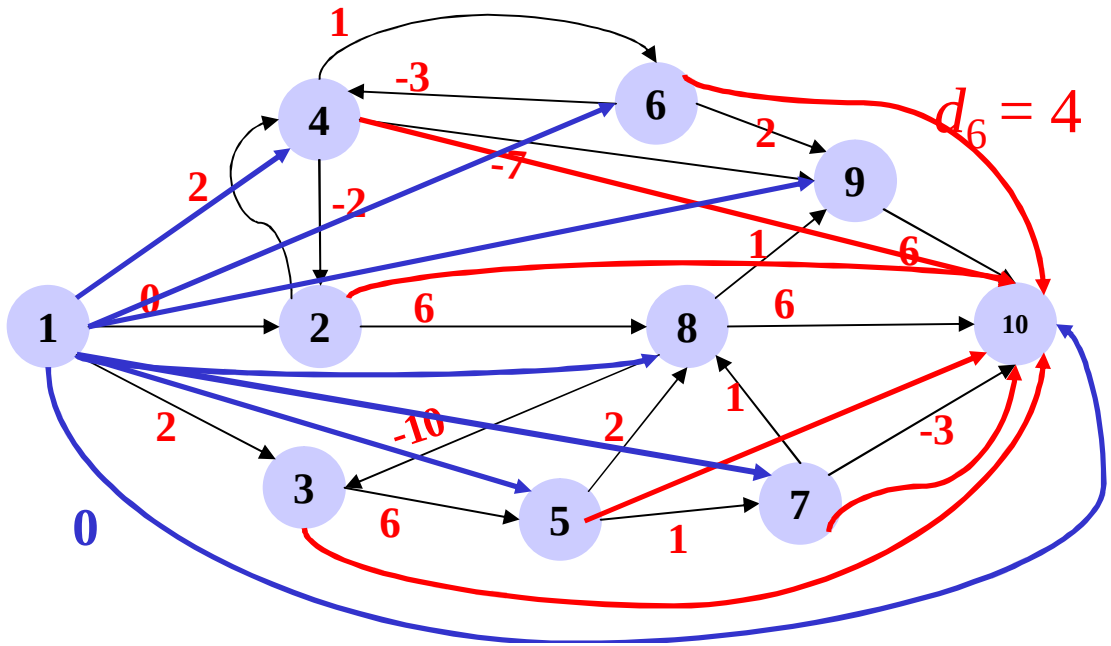
Ogni nodo *j* è in relazione con i nodi 1 e *n*.

- Ogni nodo *j può cominciare solo* dopo che 1 è finito (time lag 0)

$$s_j \geq f_1 + \mathrm{FS}_{1j}^{\min} \rightarrow s_j - s_1 \geq d_1 + \mathrm{FS}_{1j}^{min} \rightarrow s_j - s_1 \geq 0$$

Il nodo *n* può cominciare solo dopo che ogni *j* è finito (time lag 0)

$$s_n \geq f_j + FS_{jn}^{\min} \rightarrow s_n - s_j \geq d_j + FS_{jn}^{min} \rightarrow s_n - s_j \geq d_j$$



# Formulazione Makespan

Problema del calcolo del *makespan*: costruisci un piano (*schedule*) che soddisfi tutti i vincoli di precedenza e minimizzi la durata del progetto.

Il problema di calcolo del makespan può essere formualto come PL:

$$\begin{aligned} \min \quad & s_n - s_1 \\ & s_j - s_i \geq l_{ij} \quad ij \in A \\ & s \in R^{|V|} \end{aligned} \qquad \text{(PM)}$$

Ogni soluzione ammissibile di (PM) è detta *piano* o *schedule*

In forma compatta: $min\ \{s_n - s_1 : Ms \geq l,\ s \in R^{|V|}\},\ M \in R^{|A| \times |V|},\ l \in R^{|A|}$

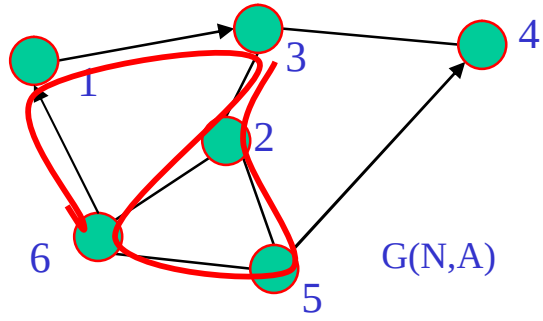
$M$ matrice d'incidenza archi-nodi del grafo dei vincoli

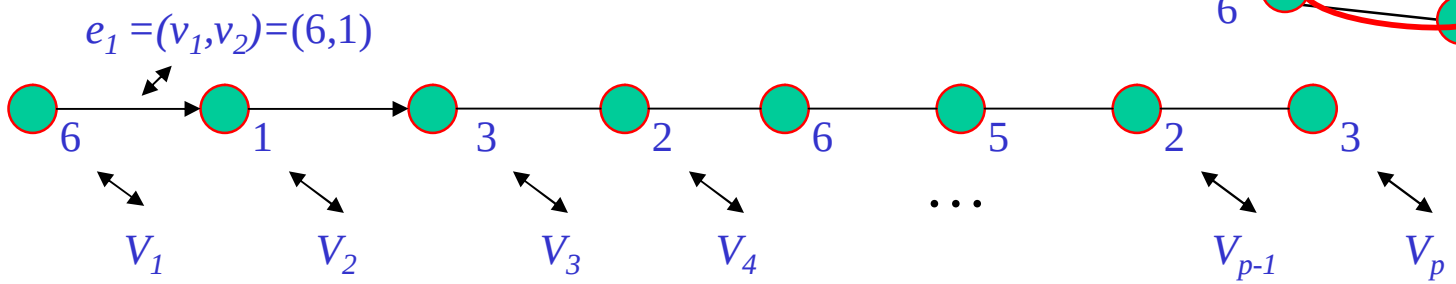
# “Walk”, Cammini e Cicli

*W “WALK” IN G(V,A)*

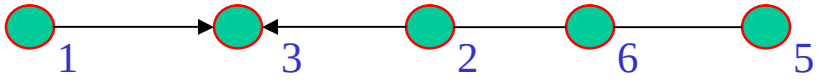
*SEQUENZA ALTERNANTE DI NODI e ARCHI*

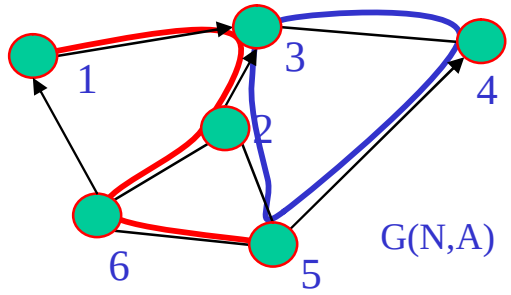
$W=(v_1, e_1, v_2, e_2, v_3, \ldots, e_p, v_p)$



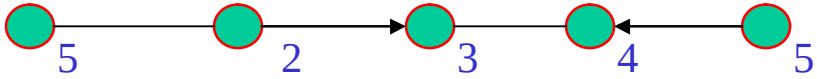




*CAMMINO ⇒ “WALK” in G(V,A) senza archi e nodi interni ripetuti*




*CICLO ⇒ CAMMINO CHIUSO (nodi estremi coincidenti)*

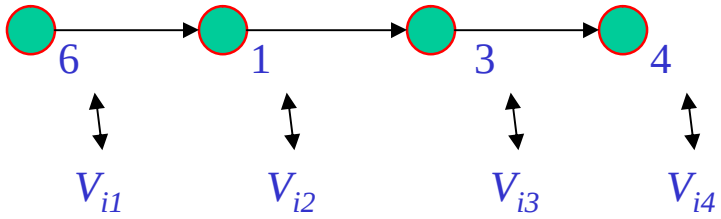


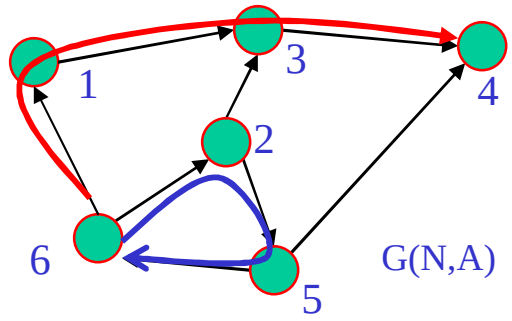
# Grafi Orientati: Cammini e Cicli Orientati

*CAMMINO ORIENTATO* $\Rightarrow$

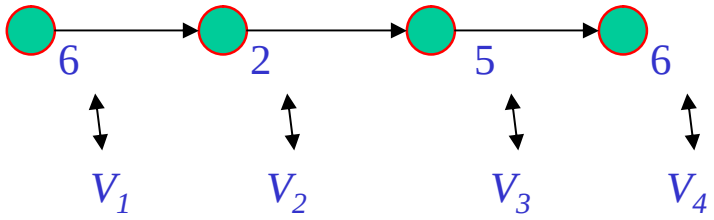
*CAMMINO* $P=(V_1, (V_1,V_2), V_2, (V_2,V_3),\ldots, (V_{p-1},V_p), V_p)$ *con:*

$V_k$ *coda di* $(V_k,V_{k+1})$ *per ogni* $k=1,\ldots,p-1$




*CICLO ORIENTATO* $\Rightarrow$ *CAMMINO ORIENTATO CHIUSO*




*Peso di un cammino P di G(V,A): somma dei pesi di suoi archi*

$$l(A(P))=l(P)=\sum_{uv\in A(P)} l_{uv}$$

# Problema del Cammino di Peso Massimo

OSS: Esiste (almeno) un cammino orientato dal nodo 1 a ogni nodo del grafo dei vincoli

**TEOREMA 3.1**: *Sia $P^*_u$ il cammino di peso massimo da 1 ad un generico nodo $u \in V$. Se il grafo dei vincoli $G(V,A)$ non ha cicli orientati di peso totale positivo allora $s'_u = l(P^*_u)$ per ogni $u \in V$ è una soluzione ammissibile per* (PM).

**DIMOSTRAZIONE:**

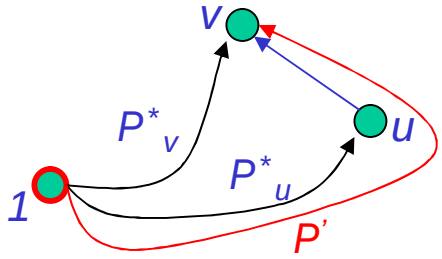
Se $s'_v - s'_u \geq l_{uv}$ per ogni $uv \in A$ **s'** è una soluzione ammissibile

Se invece ho **s'** che viola qualche vincolo, cioè $s'_v - s'_u < l_{uv}$ per qualche $uv \in A$

$\Rightarrow$ $l(P^*_v) - l(P^*_u) < l_{uv}$ $\Rightarrow$ $l(P^*_v) < l(P^*_u) + l_{uv}$

Se $v$ non appartiene a $P^*_u = (1,\dots,u)$

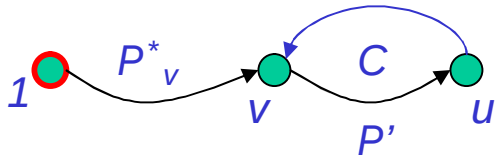
$\Rightarrow$ $P' = (1,\dots,u,uv,v)$ è un cammino con:
$l(P') = l(P^*_u) + l_{uv} > l(P^*_v)$ CONTRADDIZIONE




Quindi: $v$ appartiene a $P^*_u = (1,\dots,u)$

Detto $P' = (v,\dots,u)$ il sotto-cammino di $P^*_u$ da $v$ ad $u$

$\Rightarrow$ $C = P' \cup \{uv\}$ è un ciclo orientato




$l(P^*_v) < l(P^*_v) + l(P') + l_{uv}$ $\Rightarrow$ $0 < l(P') + l_{uv} = l(C)$

CICLO POSITIVO, **contraddizione**!

# Condizione di esistenza delle soluzioni

*Sappiamo che il prob. del cammino massimo ha soluzione ammissibile* ***solo se non ci sono cicli orientati di peso totale positivo***

***COROLLARIO 3.1:*** *Se il grafo dei vincoli $G(V,A)$ non ha cicli orientati di peso totale positivo allora* (PM) *ha soluzioni ammissibili.*

***TEOREMA 3.2:*** *Sia $P_{uv}$ un cammino dal nodo $u \in V$ al nodo $v \in V$ in $G$ e sia $s'$ una soluzione ammissibile per* (PM). Allora $s'_v \geq s'_u + l(P_{uv})$

***DIMOSTRAZIONE:***

*Sia $P_{uv}=(u=u_1, u_1u_2, u_2,\ldots, u_{k-1}u_k, u_k=v)$ un cammino orientato di $G(V,A)$*

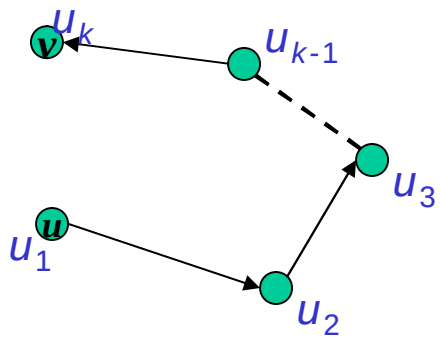
*Poichè* ***S'*** *è una soluzione ammissibile abbiamo:*

$s'_{u2} - s'_{u1} \geq l_{u1,u2}$

$s'_{u3} - s'_{u2} \geq l_{u2,u3}$

$\vdots \qquad +$

$s'_{uk} - s'_{uk-1} \geq l_{uk-1,uk}$




$$\underbrace{s'_{uk} - s'_{u1}}_{s'_v - s'_u} \geq l_{u1,u2} + l_{u2,u3} + \ldots + l_{uk,uk-1} = l(P_{uv})$$

■

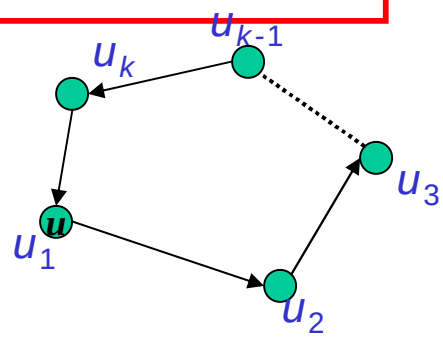
# Condizione di Limitatezza

***COROLLARIO 3.2:*** *Se* (PM) *ha soluzioni ammissibili allora il grafo dei vincoli* $G(V,A)$, *con pesi* $l$ *non ha cicli orientati di peso totale positivo*

***DIMOSTRAZIONE:***

*Sia* ***s'*** *una soluzione ammissibile*

*Per assurdo, sia* $C=(u=u_1,\ u_1u_2,\ u_2,\ldots,\ u_ku_1,\ u_1=u)$ *un ciclo orientato di* $G(V,A)$ *avente peso totale* $l(C)$ *positivo.*




*Applicando il Teorema 3.2 si ha* $0 = s'_u - s'_u \geq l(C) > 0$, contraddizione ■

***TEOREMA 3.3:*** (PM) *ha soluzioni ammissibili se e solo se il grafo dei vincoli* $G(V,A)$ *non ha cicli orientati di peso totale positivo .*

*Segue banalmente dai corollari 3.1 e 3.2* ■

*Def. Un progetto è realizzabile se e solo se il suo grafo dei vincoli* $G(V,A)$ *non ha cicli orientati di peso totale positivo*

Allora assumiamo che $G$ non contenga cicli orientati di peso positivo

# Condizione di ottimalità

***TEOREMA 3.4:*** *Sia $P^*_u$ il cammino di peso massimo da 1 ad un generico nodo $u \in V$. Allora $s^*_u = l(P^*_u)$ per ogni $u \in V$ è una soluzione ottima per* (PM)

***DIMOSTRAZIONE:***

$s^*$ è ammissibile (Teorema 3.1)

*Poiché G* non contiene cicli di peso positivo si ha $s^*_1 = 0$

*La soluzione vale* $s^*_n - s^*_1 = l(P^*_n) - 0 = l(P^*_n)$ (Peso del cammino massimo da 1 a n)

*Sia* ***s'*** *una qualunque soluzione ammissibile*

*Applicando il Teorema 3.2 si ha* $s'_n - s'_1 \geq l(P^*_n)$ *e allora* $s^*$ *è ottima* ■

Allora possiamo limitarci a soluzioni ammissibili con $s_1 = 0$.
(Il progetto inizia all'istante 0)

- Proprietà fondamentale di $s^*$: per ogni soluzione ammissibile $s'$ con $s'_1 = 0$, si ha

$$s'_u = s'_u - s'_1 \geq l(P^*_u) = s^*_u$$

- $s^*$: è chiamata *Earliest Start Schedule* (*es*) perché non esistono schedule ammissibili con $s'_u < s^*_u$ per qualche $u$

# Calcolo dell'es

L'*earliest start schedule* $s^*$ viene di solito denotato con ***es***

*Calcolo dell'es*

- *Calcola* il cammino (di peso) massimo $P^*_u$ da 1 a ogni nodo $u \in V$
- *Poni* $es_u = l(P^*_u)$ per ogni $u \in V$

- Gli algoritmi per il calcolo dell'*es* costruiscono un'arborescenza dei cammini massimi, e cioè un albero ricoprente con radice in 1 costituito da tutti i cammini orientati di peso massimo da 1 a ogni altro nodo $u$

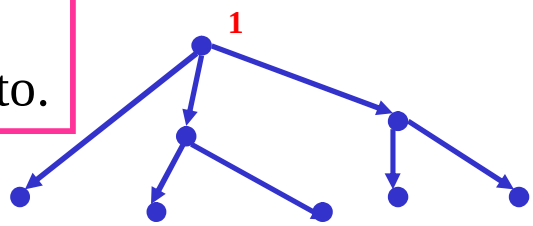
Si può ovviamente trasformarlo in un problema di cammino minimo:

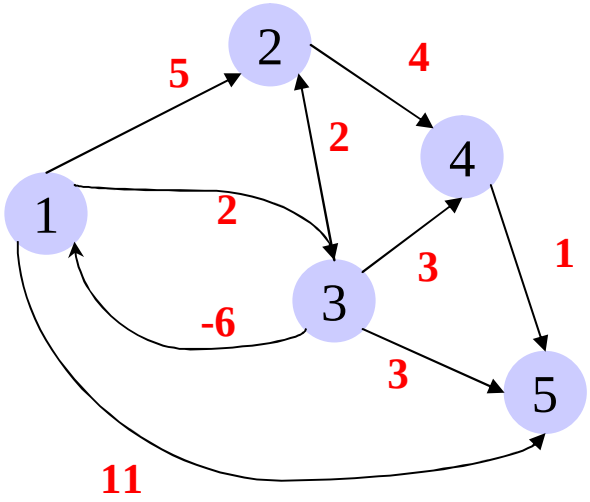
Se trasformiamo $l'_{ij} = - l_{ij}$ il problema è equivalente alla costruzione dell'arborescenza dei cammini minimi.

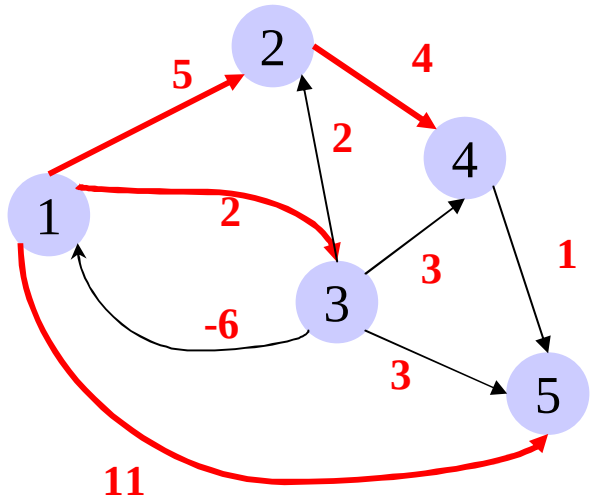
Se $G(V,A)$ non contiene cicli di peso totale positivo rispetto a $l$, non conterrà cicli orientati di peso totale negativo rispetto a $l' = - l$

# L’arborescenza dei cammini massimi

Arborescenza (di radice 1): albero in cui, per ogni $j \in V$, l’unico cammino dal nodo 1 al nodo $j$ è un cammino orientato.




OSS. Ogni nodo diverso dal nodo radice 1 ha un unico predecessore nell’arborescenza.

- Un’arborescenza può essere descritta mediante il vettore dei predecessori, *prec*

# Algoritmo iterativo di *Bellman Ford*

Per trovare questi cammini massimi, non possiamo usare l'algoritmo di *Dijkstra* perché i pesi possono avere valori sia positivi sia negativi

Alternativa: algoritmo di *Bellman-Ford* (*metodo del rilassamento*): applicabile quando il grafo non contiene cicli orientati di peso positivo

• Inizialmente vengono ordinati gli archi e definita una particolare soluzione iniziale $s^0$ non ammissibile.

• A ogni iterazione gli archi vengono visitati nell'ordine prefissato: se per un arco $(i,j)$ si ha $s_j < s_i + l_{ij}$ violando il corrispondente vincolo di precedenza (1), si pone

$$\boldsymbol{s_j = s_i + l_{ij}}$$

• Dimostreremo che dopo al più $n = |V|$ iterazioni tutti i vincoli saranno soddisfatti.

Inoltre $s$ è ottima e si avrà $s_i = es_i$ (*earliest start time* di $i$) per ogni $i \in V$

• Alla fine è possibile ricostruire l'arborescenza dei cammini massimi utilizzando il vettore dei predecessori *prec*

# Schema algoritmo *Bellman Ford*

*Calcolo dell'arborescenza dei cammini massimi dal nodo* 1 *a* $i$ per $i = 1, \ldots, n$

Inizializzazione:

$s_1 = 0$, $s_i = -\infty$ e $prec(i) = 0$ per $i = 2, \ldots, n$.

Ordina gli archi $A = \{e_1, \ldots, e_m\}$

Repeat (*finchè s non si modifica più*)

For $k = 1$ to $m$ (*per ogni arco*)

If $e_k = (i,j)$ è tale che $s_j < s_i + l_{ij}$

poni $s_j = s_i + l_{ij}$

poni $prec(j) = i$

Endfor

EndRepeat

Iterazioni "piccole"

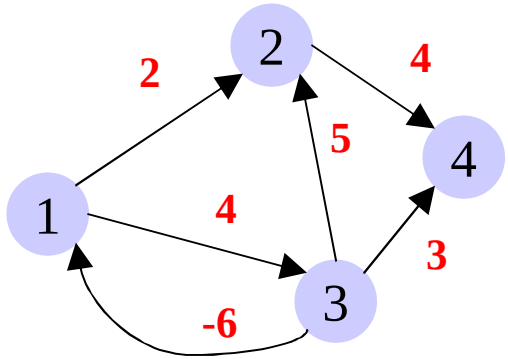
Iterazioni "grandi"

- L'algoritmo termina con $s = es$
- Il blocco {*Repeat … EndRepeat*} (iterazione *grande*) viene eseguito al più $n$ volte
- Ogni iter. grande sono $m$ iterazioni *piccole*, quindi la complessità è $O(mn)$ (numero totale di iterazioni piccole)

# Esempio di applicazione

Inizializzazione: $s(1) = 0$, $s(2) = s(3) = s(4) = -\infty$

Ordino gli archi: $e_1 = (1,2)$, $e_2 = (1,3)$, $e_3 = (3,1)$, $e_4 = (2,4)$, $e_5 = (3,4)$, $e_6 = (3,2)$




*Repeat* 1

Iter 1. $k = 1$. $s(2) = -\infty < s(1) + 2 \Rightarrow s(2) = s(1) + 2 = 2$ $prec(2) = 1$

Iter 2. $k = 2$. $s(3) = -\infty < s(1) + 4 \Rightarrow s(3) = s(1) + 4 = 4$ $prec(3) = 1$

Iter 3. $k = 3$. $s(1) = 0 > s(3) - 6$

Iter 4. $k = 4$. $s(4) = -\infty < s(2) + 4 \Rightarrow s(4) = s(2) + 4 = 6$ $prec(4) = 2$

Iter 5. $k = 5$. $s(4) = 6 < s(3) + 3 \Rightarrow s(4) = s(3) + 3 = 7$ $prec(4) = 3$

Iter 6. $k = 6$. $s(2) = 2 < s(3) + 5 \Rightarrow s(2) = s(3) + 5 = 9$ $prec(2) = 3$

*Repeat* 2

Iter 1. $k = 1$. $s(2) = 9 > s(1) + 2$

Iter 2. $k = 2$. $s(3) = 4 = s(1) + 4$

Iter 3. $k = 3$. $s(1) = 0 > s(3) - 6$

Iter 4. $k = 4$. $s(4) = 7 < s(2) + 4 \Rightarrow s(4) = s(2) + 4 = 13$ $prec(4) = 2$

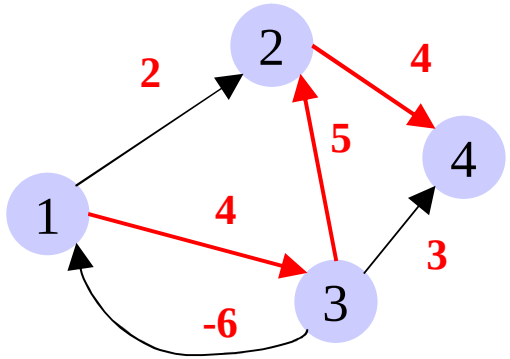
Iter 5. $k = 5$. $s(4) = 13 > s(3) + 3$

Iter 6. $k = 6$. $s(2) = s(3) + 5$

# L’arborescenza dei cammini massimi

Nella successiva iterazione grande non vengono aggiornate le variabili, quindi stop

• L’arborescenza dei cammini massimi può essere ricostruita a partire dal vettore dei predecessori, *prec*()




*prec*(4) = 2

*prec*(2) = 3

*prec*(3) = 1

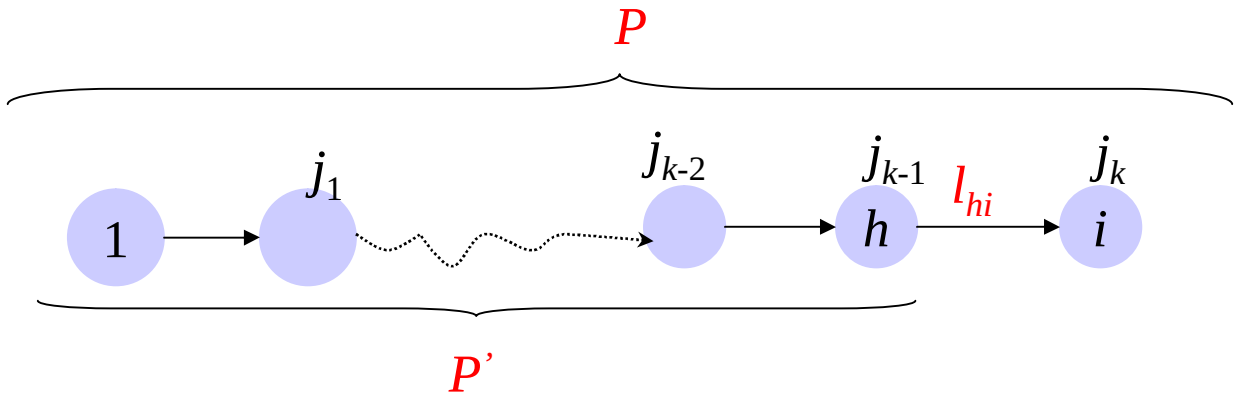
*prec*(1) non esiste

# Principio di Ottimalità per cammini massimi

*Def. Lunghezza* di un cammino = numero archi che lo compongono

Dato che non ci sono cicli orientati positivi, i cammini massimi contengono al più $n$-1 archi.

*Principio di ottimalità*. Se $P = \{1, j_1, j_2, \ldots, j_{k-1}, j_k\}$ è un cammino (di peso) massimo da 1 a $j_k$, allora $P' = \{1, j_1, j_2, \ldots, j_{k-1}\}$ è un cammino (di peso) massimo da 1 a $j_{k-1}$

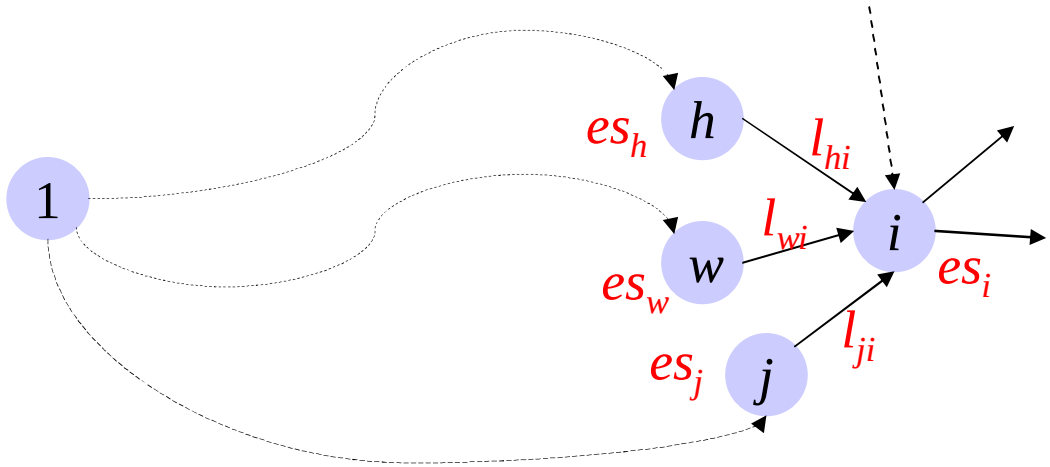


# Ottimalità dell'ultimo arco del cammino

Sia *es* il vettore dei pesi dei cammini massimi

Dal principio di ottimalità segue

$$es_i = \max_{ji \in \delta^-(i)} es_j + l_{ji} \quad \text{per ogni } i \in V - \{1\} \qquad \text{(A)}$$




***Proprietà di ottimalità*** $es_i \geq es_j + l_{ji}$, per ogni $ji \in \delta^-(i)$, $i \in V - \{1\}$

# Correttezza algoritmo Bellman-Ford

Per $i \in V$, sia $s_i^q$ il valore della variabile $s_i$ alla fine della $q$-*esima* iterazione *piccola*

OSS. A ogni iterazione piccola viene visitato un solo arco

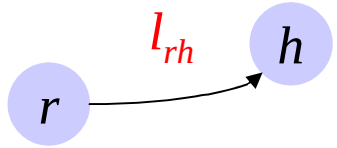
Se $(r,h)$ è l'arco visitato alla $q$-*esima* iterazione *piccola* allora $s_i^q = s_i^{q-1}$ per ogni $i \neq h$

*Lemma* 3.5. $s_i^q \leq es_i$ per ogni $q$ e per ogni $i \in V$

Per $q = 0$, si ha $\begin{cases} s_1^0 = 0 \leq 0 = es_1 \\ s_i^0 = -\infty \leq es_i \quad \forall i \in V - \{1\} \end{cases}$ e la condizione del lemma (*invariante*) è soddisfatta

Per assurdo, sia $q > 0$ la prima iterazione in cui l'invariante è violato.

Sia $(r,h)$ l'arco visitato alla $q$-esima iterazione.




Poiché alla $(q\text{-}1)$-esima iterazione l'invariante è soddisfatto sarà: $s_h^{q-1} \leq es_h$ e $s_r^{q-1} \leq es_r$ (1)

Poiché alla $q$-esima iterazione l'invariante è violato sarà: $s_h^q > es_h$

Quindi la variabile corrispondente ad $h$ è stata aggiornata e sarà $s_h^q = s_r^{q-1} + l_{rh}$ (2)

Da (1) e (2) $es_r + l_{rh} \geq s_r^{q-1} + l_{rh} = s_h^q$

Per la proprietà di ottimalità $es_h \geq es_r + l_{rh}$ ⇒ $es_h \geq s_h^q$ contraddizione

■

# Correttezza algoritmo Bellman-Ford

Indichiamo con $S_i^q = s_i^{qm}$ il valore della variabile $s_i$ alla fine della *q-esima* iterazione *grande*

*Lemma* 3.6. Sia $i \in V$ tale che esiste un cammino massimo da 1 a $i$ con un numero di archi minore o uguale $k$. Allora $S_i^q = es_i$ (cioè alla q-esima iterazione ho già raggiunto l'ottimo) per $q \geq k$

Per ogni $i \in V$, $q > t$ $S_i^t \leq S_i^q$ (*variabili non decrescenti*)

*Per induzione*. Per $k = 0$. L'unico cammino massimo di lunghezza 0 è il cammino da 1 a 1.

$S_1^0 = 0 = es_1$ Dal lemma 3.5 $S_1^q \leq es_1$ per $q \geq 0$. Per la non decrescenza $S_1^q = es_1$ per $q \geq 0$.

Per $k > 0$ sia $P = \{1, j_1, \ldots, j_{k-1}, j_k = i\}$ un cammino massimo fino a $i$. Poniamo $h = j_{k-1}$.

Il cammino $\{1, j_1, \ldots, j_{k-1} = h\}$ è un cammino massimo fino a $h$ di lunghezza $k$-1 (*principio di ottimalità*). Quindi, $es_i = es_h + l_{hi}$.

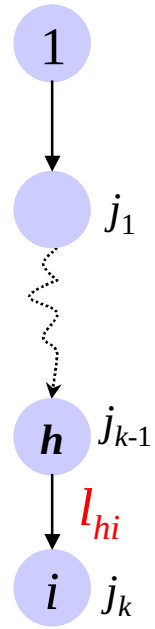
Per ipotesi induttiva, alla fine della $k$-1-esima iterazione *grande*, si ha $S_h^{k-1} = es_h$

Durante la $k$-esima iterazione *grande*, alla *r-esima* iterazione *piccola* si visita l'arco $(h,i)$

$s_h^{r-1} = es_h$ e per il *Lemma* 3.5 $s_i^{r-1} \leq es_i = es_h + l_{hi} = s_h^{r-1} + l_{hi}$

Dopo l'aggiornamento $s_i^r = \max(s_i^{r-1}, s_h^{r-1} + l_{hi}) = \max(s_i^{r-1}, es_i) = es_i$

Da cui $es_i = s_i^r \leq S_i^k \leq es_i$




■

# Terminazione algoritmo Bellman-Ford

*Corollario* 3.7. L'algoritmo di Bellman-Ford termina in al più $n$ iterazioni *grandi*

Il cammino massimo più lungo contiene al più $n$-1 archi

Alla fine della ($n$-1)-*esima* iterazione *grande* $S_i^{n-1} = es_i$ per ogni $i \in V$ (*Lemma* 3.6)

Per la non decrescenza, le variabili non vengono più aggiornate durante l'$n$-*esima* iterazione grande e l'algoritmo termina

# Ancora sulle relazioni fittizie

Precedenze fittizie:

Per ogni nodo $i \neq 1$, esiste l'arco $(1,i)$ di peso 0.

Per ogni nodo $i \neq n$, esiste l'arco $(i,n)$ di peso $d_i$.

• Nelle rappresentazioni grafiche del grafo delle precedenze generalizzate vengono spesso omesse.

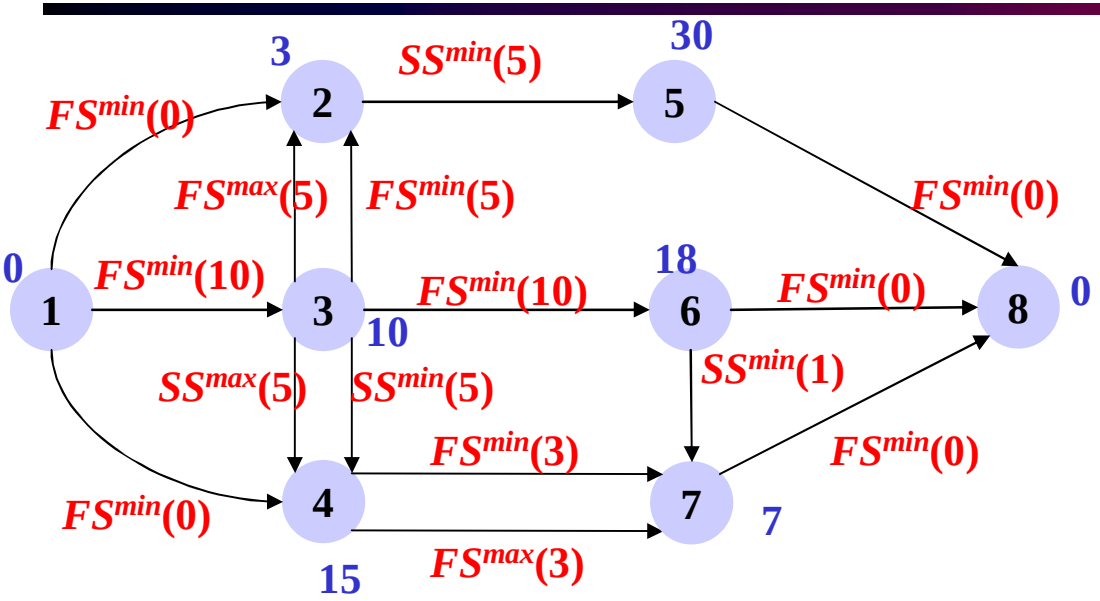
• Nella risoluzione degli esercizi possono essere omesse solo se certamente implicate dalle altre relazioni di precedenza.

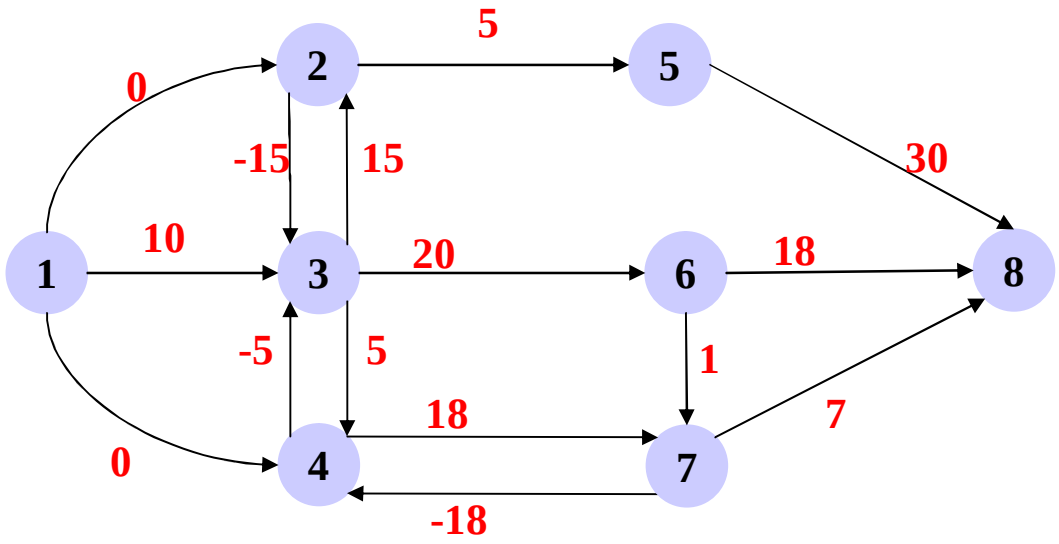
1. E' possibile omettere l'arco $(1,i)$ se esiste nel grafo un cammino orientato $P_{1i}$ da 1 a $i$ di peso $l(P_{1i})$ almeno pari a 0.

Infatti, per il Teorema 3.2 $s'_i \geq s'_1 + l(P_{1i})$ e il vincolo $s'_i \geq s'_1$ corrispondente all'arco $(1,i)$ è implicato

2. Analogamente, è possibile omettere l'arco $(i,n)$ se esiste nel grafo un cammino orientato da $i$ a $n$ di peso almeno pari a $d_i$

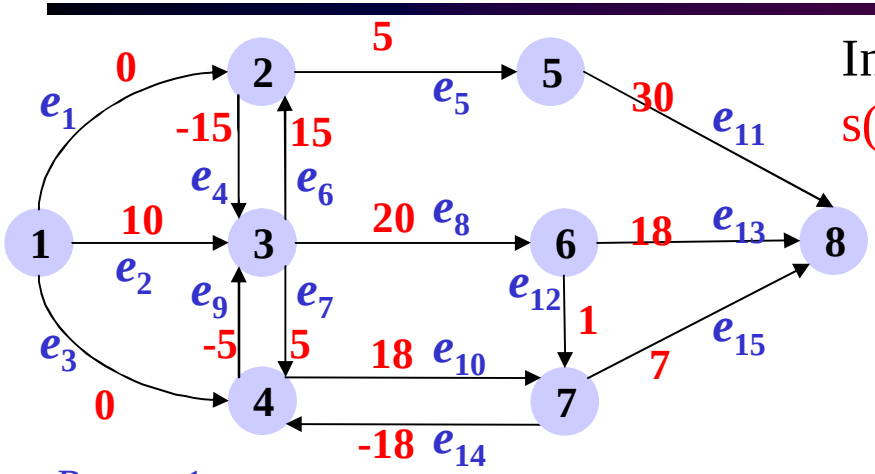
# Un esercizio completo




In questo esempio è facile vedere che le relazioni fittizie sono implicate dagli altri vincoli.

# Un esercizio completo




Inizializzazione: $s(1) = 0$, $s(2) = s(3) = s(4) = s(6) = s(7) = s(8) = -\infty$, $prec(i) = *$, $i \in V$

Ordinamento archi: $e_1=(1,2)$, $e_2=(1,3)$, $e_3=(1,4)$, $e_4=(2,3)$, $e_5=(2,5)$, $e_6=(3,2)$, $e_7=(3,4)$, $e_8=(3,6)$, $e_9=(4,3)$, $e_{10}=(4,7)$, $e_{11}=(5,8)$, $e_{12}=(6,7)$, $e_{13}=(6,8)$, $e_{14}=(7,4)$, $e_{15}=(7,8)$.

*Repeat* 1

$k = 1$. $s(2) = -\infty < s(1) + 0 \Rightarrow s(2) = s(1) + 0 = 0$ — $prec(2) = 1$

$k = 2$. $s(3) = -\infty < s(1) + 10 \Rightarrow s(3) = s(1) + 10 = 10$ — $prec(3) = 1$

$k = 3$. $s(4) = -\infty < s(1) + 0 \Rightarrow s(4) = s(1) + 0 = 0$ — $prec(4) = 1$

$k = 4$. $s(3) = 10 > s(2) - 15$

$k = 5$. $s(5) = -\infty < s(2) + 5 \Rightarrow s(5) = s(2) + 5 = 5$ — $prec(5) = 2$

$k = 6$. $s(2) = 0 < s(3) + 15 \Rightarrow s(2) = s(3) + 15 = 25$ — $prec(2) = 3$

$k = 7$. $s(4) = 0 < s(3) + 5 \Rightarrow s(4) = s(3) + 5 = 15$ — $prec(4) = 3$

$k = 8$. $s(6) = -\infty < s(3) + 20 \Rightarrow s(6) = 30$ — $prec(6) = 3$

$k = 9$. $s(3) = s(4) - 5$

$k = 10$. $s(7) = -\infty < s(4) + 18 \Rightarrow s(7) = 33$ — $prec(7) = 4$

$k = 11$. $s(8) = -\infty < s(5) + 30 \Rightarrow s(8) = 35$ — $prec(8) = 5$

$k = 12$. $s(7) = 33 > s(6) + 1$

$k = 13$. $s(8) = 35 < s(6) + 18 \Rightarrow s(8) = 48$ — $prec(8) = 6$

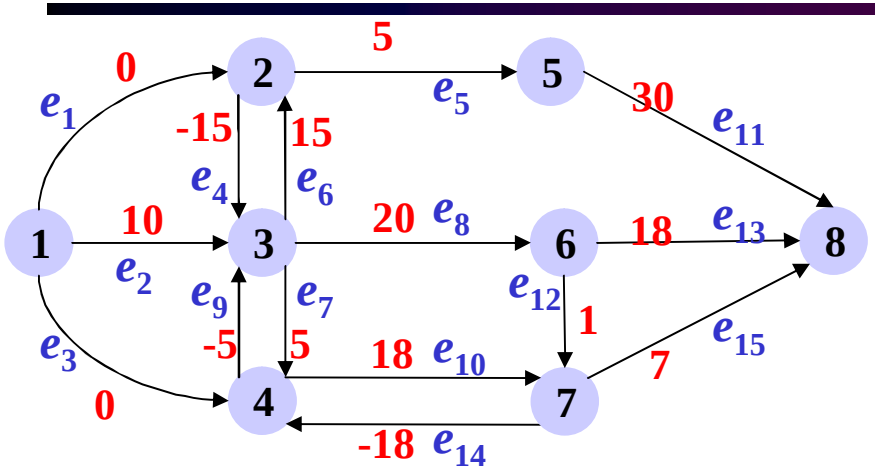
$k = 14$. $s(4) = 15 = s(7) - 18$

$k = 15$. $s(8) = 48 > s(7) + 7$

| | s |
|---|---|
| 1 | 0 |
| 2 | 25 |
| 3 | 10 |
| 4 | 15 |
| 5 | 5 |
| 6 | 30 |
| 7 | 33 |
| 8 | 48 |

| | prec |
|---|---|
| 1 | * |
| 2 | 3 |
| 3 | 1 |
| 4 | 3 |
| 5 | 2 |
| 6 | 3 |
| 7 | 4 |
| 8 | 6 |

# Un esercizio completo




Ordinamento archi: $e_1$=(1,2), $e_2$=(1,3), $e_3$=(1,4), $e_4$=(2,3), $e_5$=(2,5), $e_6$=(3,2), $e_7$=(3,4), $e_8$=(3,6), $e_9$=(4,3), $e_{10}$=(4,7), $e_{11}$=(5,8), $e_{12}$=(6,7), $e_{13}$=(6,8), $e_{14}$=(7,4), $e_{15}$=(7,8).

*Repeat 2*

$k$ = 1. $s(2) = 25 > s(1) + 0$

$k$ = 2. $s(3) = 10 = s(1) + 10$

$k$ = 3. $s(4) = 15 > s(1) + 0$

$k$ = 4. $s(3) = 10 > s(2) - 15$

$k$ = 5. $s(5) = 5 < s(2) + 5 \Rightarrow s(5) = s(2) + 5 = 30$ $\quad prec(5) = 2$

$k$ = 6. $s(2) = 25 = s(3) + 15$

$k$ = 7. $s(4) = 15 = s(3) + 5$

$k$ = 8. $s(6) = 30 = s(3) + 20$

$k$ = 9. $s(3) = s(4) - 5$

$k$ = 10. $s(7) = 33 = s(4) + 18$

$k$ = 11. $s(8) = 48 < s(5) + 30 \Rightarrow s(8) = 60$ $\quad prec(8) = 5$

$k$ = 12. $s(7) = 33 > s(6) + 1$

$k$ = 13. $s(8) = 50 > s(6) + 18$

$k$ = 14. $s(4) = 15 = s(7) - 18$

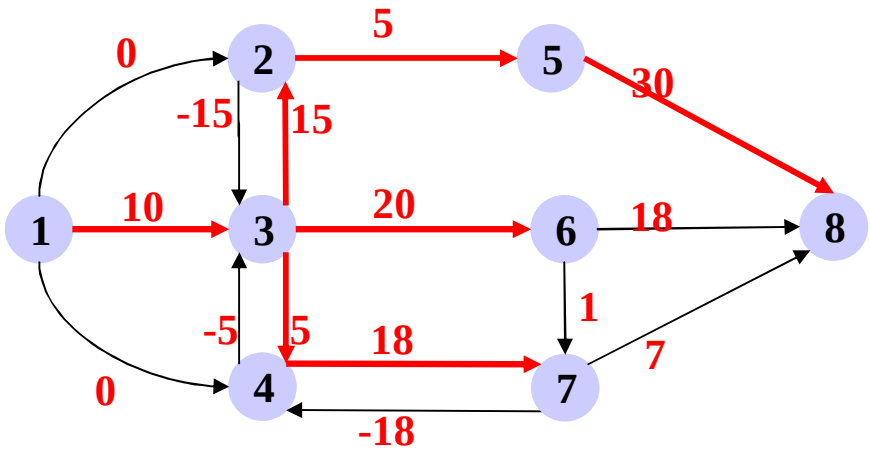
$k$ = 15. $s(8) = 50 > s(7) + 7$

| | s |
|---|---|
| 1 | 0 |
| 2 | 25 |
| 3 | 10 |
| 4 | 15 |
| 5 | 30 |
| 6 | 30 |
| 7 | 33 |
| 8 | 60 |

| | prec |
|---|---|
| 1 | * |
| 2 | 3 |
| 3 | 1 |
| 4 | 3 |
| 5 | 2 |
| 6 | 3 |
| 7 | 4 |
| 8 | 5 |

Nella prossima repeat le variabili non si modificano più e l'algoritmo termina

# Arborescenza dei cammini massimi



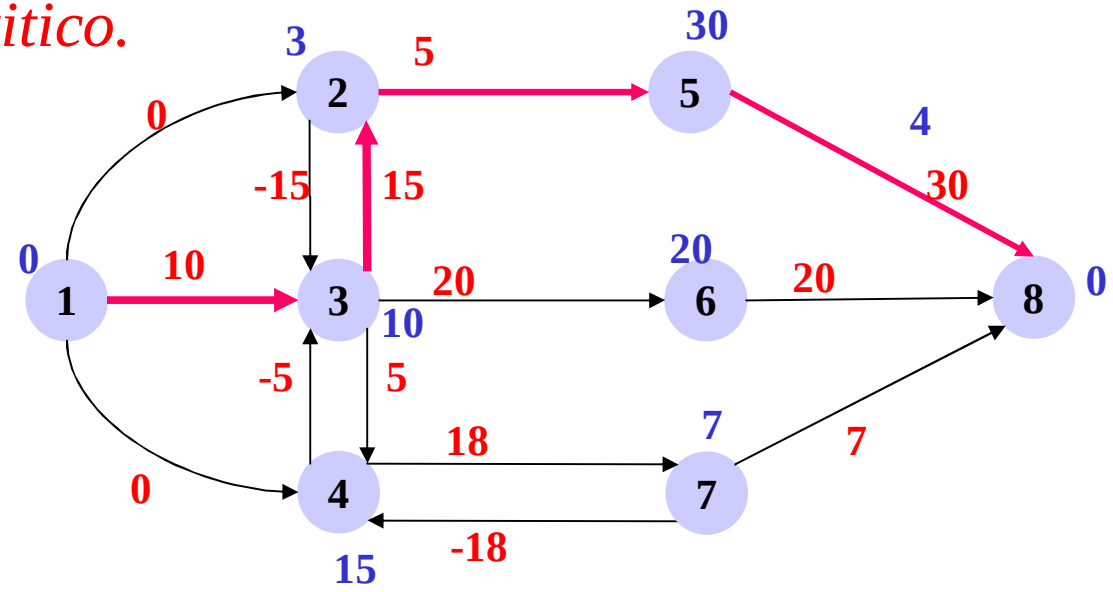

Arborescenza dei cammini massimi, costruita a partire dal vettore dei predecessori

| | prec |
|---|---|
| 1 | * |
| 2 | 3 |
| 3 | 1 |
| 4 | 3 |
| 5 | 2 |
| 6 | 3 |
| 7 | 4 |
| 8 | 5 |

| | s |
|---|---|
| 1 | 0 |
| 2 | 25 |
| 3 | 10 |
| 4 | 15 |
| 5 | 20 |
| 6 | 30 |
| 7 | 33 |
| 8 | 60 |

# Cammino critico

Il cammino massimo dal nodo 1 al nodo *n* (fine progetto) è detto *cammino critico.*




Il peso del cammino massimo dal nodo 1 al nodo *n* (fine progetto) corrisponde alla durata (minima) dell'intero progetto (*makespan*)

Se rallento attività *critiche* ho lo stesso *ritardo* sull'intero progetto!

Se rallento attività *non critiche*, *non ho ritardo* sul progetto fino a che, se rallentate troppo, il cammino massimo *cambia* e passa per esse, che quindi *diventano* critiche

# Riepilogando: dal grafo dei vincoli in poi

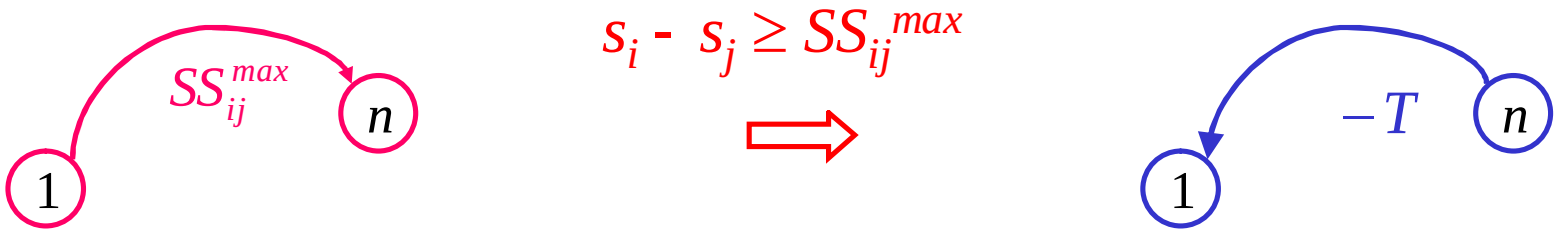
- Le relazioni di precedenza vengono rappresentate sul grafo dei vincoli $G(V,A)$ (orientato con pesi $l_{ij}$ associati agli archi)
- In assenza di cicli positivi (cioè progetti realizzabili) l'*es* viene calcolato utilizzando gli algoritmi per il cammino massimo dal nodo 1 (inizio progetto) a tutti gli altri nodi del grafo dei vincoli
- In particolare $es_i$ = *peso del cammino cammino di peso massimo* dal nodo 1 al nodo $i$, per ogni $i \in V$
- Il peso del cammino massimo dal nodo 1 al nodo $n$ (fine progetto) corrisponde alla durata minima dell'intero progetto e viene detto *makespan*
- Ogni cammino massimo dal nodo 1 al nodo $n$ (fine progetto) è detto *cammino critico*
- Esistono diversi algoritmi per il calcolo del cammino di peso massimo, con complessità distinte. Nel caso generale si può usare l'algoritmo di *Bellman-Ford*

# Calcolo del latest completion time

Supponiamo sia data una scadenza (*deadline*) $T$ per il progetto, cioè un vincolo:

$$s_n \leq T$$ *deadline*

Ciò corrisponde ad aggiungere l'arco $(n,1)$ di peso $-T$ al grafo dei vincoli




Vogliamo rispondere alla domanda: quando *al più tardi* possono cominciare le singole attività del progetto (determinando così il *Latest Start Schedule* e il *Latest Finish Schedule*) in modo da rispettare la deadline $T$ ?

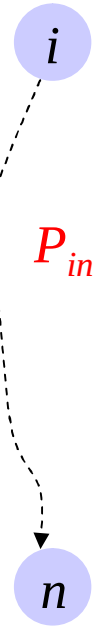
*Def. Per ogni* $j \in V$ si indica con $ls_j$ il *latest start time* dell'attività $j$, e cioè l'ultimo istante di tempo in cui l'attività $j$ può cominciare senza violare il vincolo di deadline $T$

# Calcolo del *Latest Start Schedule*

Sia $P_{in}$ un cammino da $i$ a $n$. La deadline $T$ implica $T \geq s_n$

- Per il Teorema 3.2 sarà necessariamente $s_n \geq s_i + l(P_{in})$.
- Quindi $T \geq s_i + l(P_{in}) \rightarrow s_i \leq T - l(P_{in})$ per ogni cammino $P_{in}$ da $i$ a $n$.
- Il vincolo più stringente si ottiene in corrispondenza del cammino massimo $P^*_{in}$, per cui vale

$$s_i \leq T - l(P^*_{in}) \quad = ls_i$$




*Algoritmo generico calcolo ls*

for $i = 1$ to $n$-1

  Calcola il peso $L_i$ del cammino massimo da $i$ a $n$

  Poni $ls_i = T - L_i$

Endfor

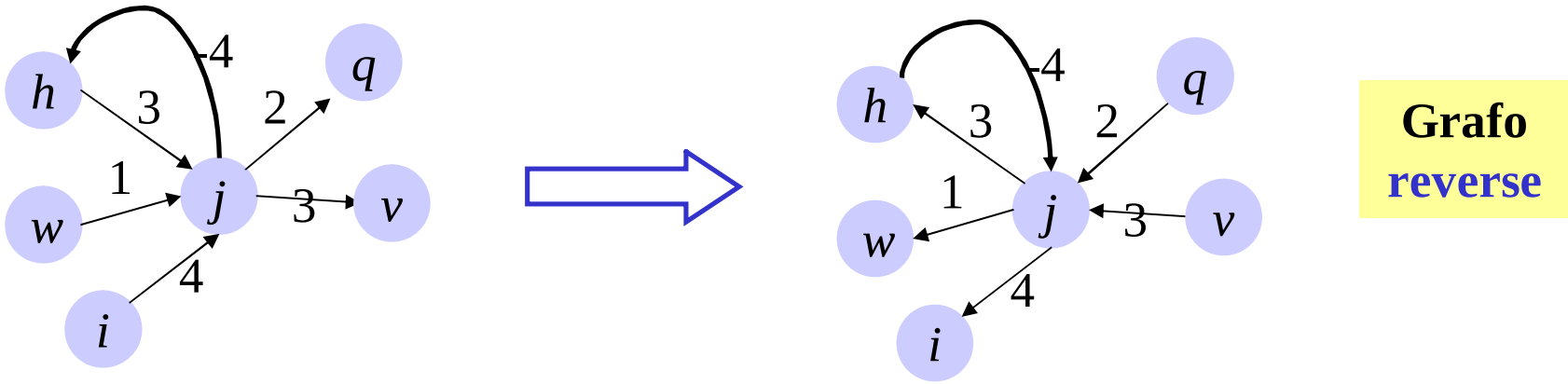
# Complessità del calcolo dell' ls

• Per ogni nodo $i \in V$ , l'algoritmo prevede il calcolo del cammino massimo da $i$ a $n$

• L'algoritmo del Bellman-Ford ha complessità $O(mn)$ e deve essere riapplicato partendo da ogni nodo $i$, quindi la complessità risulta $O(mn^2)$

• OSS. L'algoritmo di Bellman-Ford calcola il peso dei cammini massimi dal nodo iniziale a tutti gli altri nodi del grafo

• IDEA: costruisci un grafo $G'(V,A')$ ottenuto da $G(V,A)$ *invertendo* gli archi, ovvero sostituendo ogni arco $ij$ con un arco $ji$ di peso pari al peso dell'arco originario




Infatti ogni cammino da $n$ a $i$ in $G'$ corrisponde a un cammino da $i$ a $n$ in $G$ di uguale peso

# Algoritmo per calcolare *ls*

*Calcolo del latest start schedule*

1. Costruisci il grafo *G' reverse* di *G*

2. Applica *Bellman - Ford* per calcolare il peso $L_i$ del cammino massimo da $n$ a $i$ per ogni $i \in V - \{n\}$

3. Poni $ls_i = T - L_i$ per ogni $i \in V - \{n\}$

Poiché il grafo reverse può essere (facilmente) costruito in $O(m)$, la complessità dell'algoritmo e $O(mn)$.

# Total Float e Attività Critiche

*Def.* *Total float* $tf_i$ dell'attività $i$ misura quanto l'attività $i$ può essere ritardata senza violare la deadline del progetto

- L'attività $i$ non può cominciare prima del suo *earliest start time* $es_i$

- Se l'attività $i$ comincia dopo il suo *latest start time* $ls_i$ il progetto viene ritardato

$tf_i = ls_i - es_i$ per ogni $j$ in $V$

*Def.* Un'attività è detta *critica* se il suo total float è 0

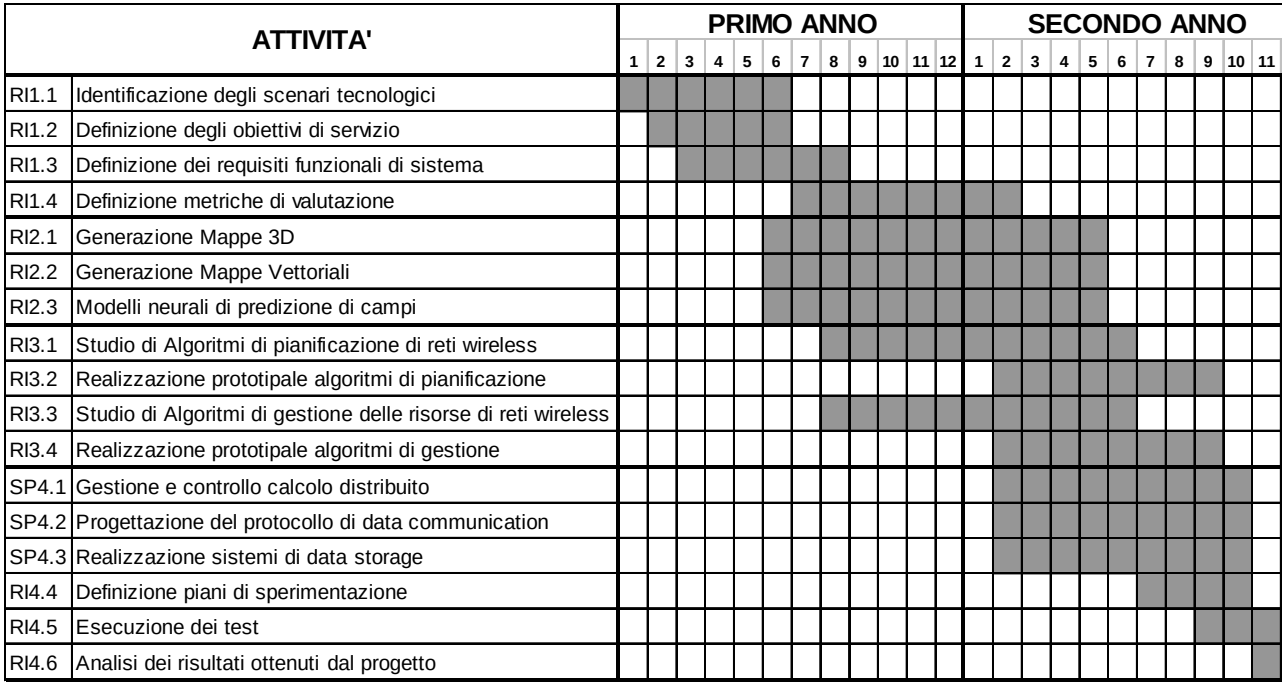
# Altre rappresentazioni (Gantt Charts)

Dato un piano di attività *s* ammissibile è possibile darne una rappresentazione grafica mediante *Gantt Charts* (introdotte dall'ingegner Gantt nel 1918)

Un *diagramma di Gantt* è una griglia bidimensionale: sulle righe ci sono le attività mentre sulle colonne gli istanti di tempo dell'orizzonte temporale.

Ogni attività *i* è rappresentata con una barra orizzontale di lunghezza pari alla sua durata $d_i$, partendo dalla casella nella colonna $s_i$

Vengono cioè riempite le caselle della griglia corrispondenti a istanti in cui l'attività è (prevista) in svolgimento.

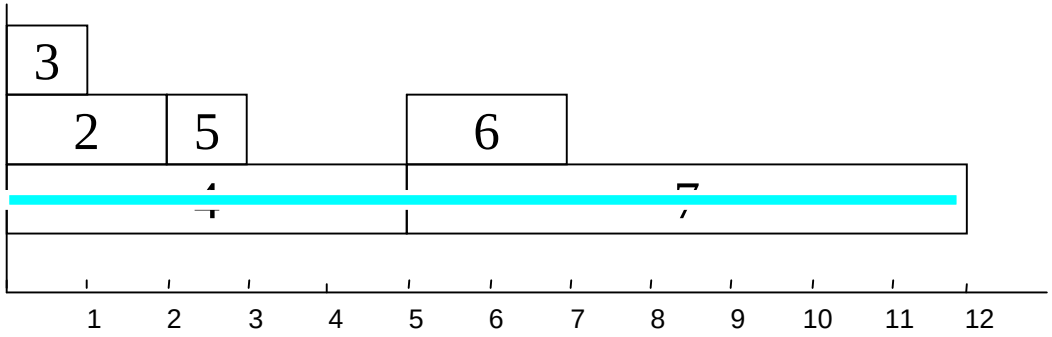


# Altre rappresentazioni (Gantt Charts)

Esistono “in natura” diversi formati per le Gantt Chart

Ad esempio, si possono rappresentare sulla stessa riga più attività che non si sovrappongono temporalmente

Di seguito riportiamo una rappresentazione per l’earliest start schedule

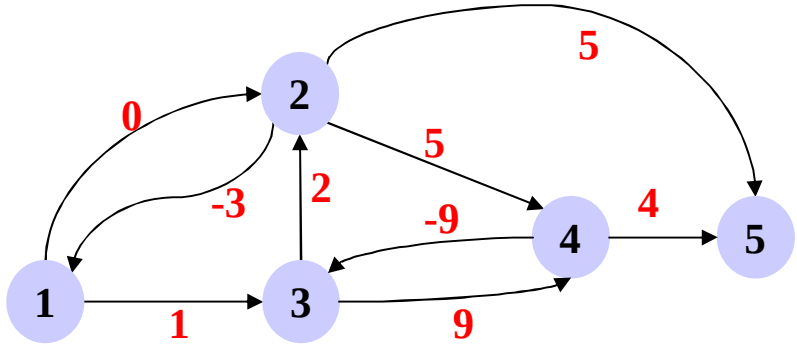



Gantt chart

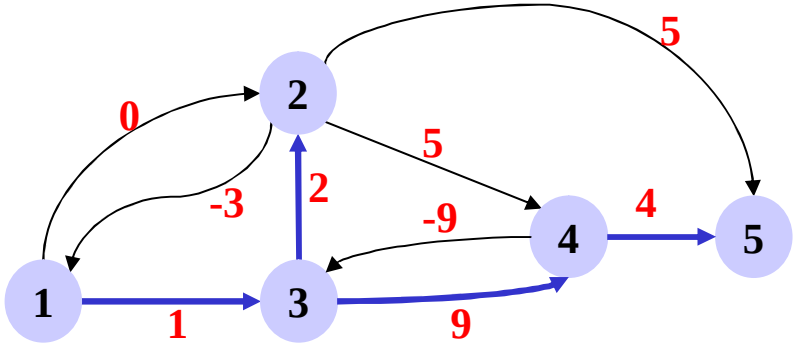
*Earliest start schedule*

Il cammino critico è evidenziato in rosso

# Esempio di calcolo *es* e *ls*

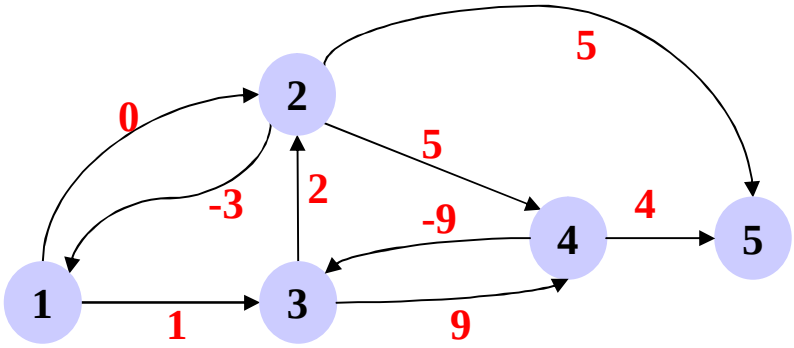



Applicando Bellman-Ford con nodo radice 1 calcolo l'arborescenza dei cammini massimi e quindi l'*earliest start schedule*




| | *es* |
|---|---|
| 1 | 0 |
| 2 | 3 |
| 3 | 1 |
| 4 | 10 |
| 5 | 14 |

# Calcolo dell'*ls*

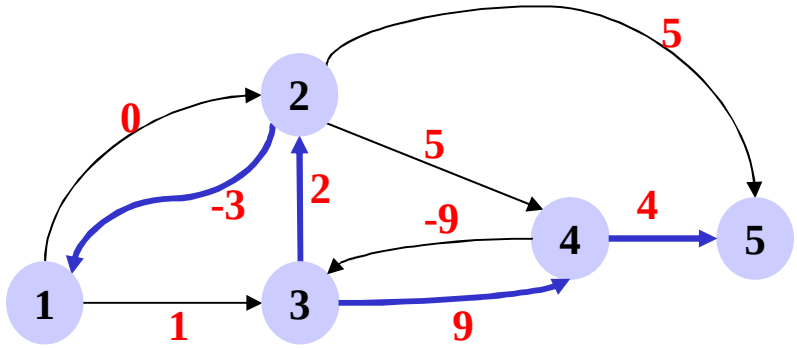


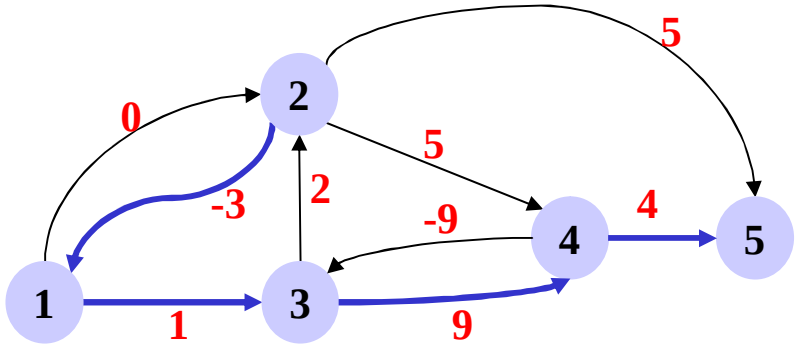

Per avere il *latest start time* corrispondente ad una deadline $T=17$ trovo il peso $L_i$ del cammino massimo da $i$ a $n$ e pongo $ls_i = T - L_i$

Applicando Bellman-Ford con nodo radice $i$ calcolo il cammino massimo da $i$ a $n$

Ad esempio, con radice 3 ho

$T = 17, i = 3 \Rightarrow L_3 = 13 \Rightarrow ls_3 = 17\text{-}13 = 4$

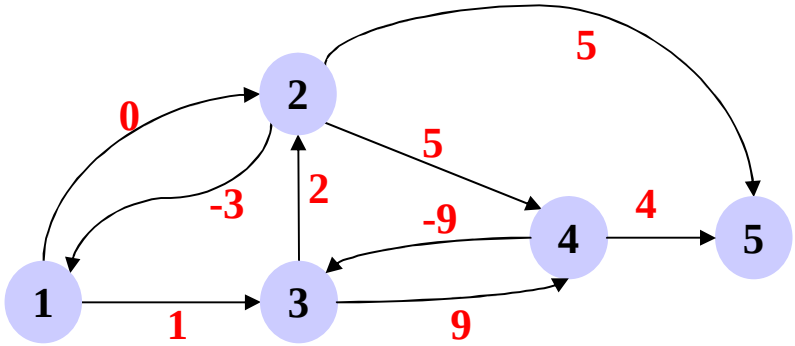


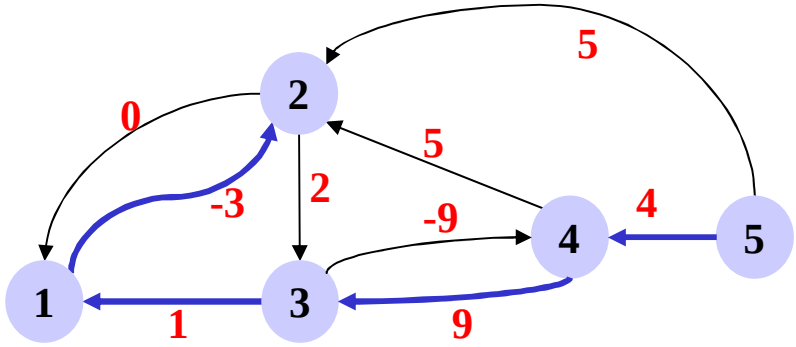

Ma non posso calcolare, ad esempio, $ls_2$ e sono quindi obbligato a riapplicare Bellman-Ford con radice 2

$T = 17, i = 2 \Rightarrow L_2 = 11 \Rightarrow ls_2 = 17\text{-}11 = 6$

# Calcolo dell'*ls*




Se invece applico Bellman-Ford al grafo reverse con radice *n* devo farlo *una sola volta* e ottengo l'arborescenza reverse con cui calcolo il *latest start time*




Es. $T = 17$

$L_4 = 4 \Rightarrow ls_4 = 17 - 4 = 13$

$L_3 = 13 \Rightarrow ls_3 = 17-13 = 4$

$L_1 = 14 \Rightarrow ls_1 = 17-14 = 3$

$L_2 = 11 \Rightarrow ls_2 = 17-11 = 6$