Modellizzazione dei Sistemi Logistici

Elementi di teoria della complessità

Corso di Studi in Ingegneria Gestionale

a.a. 2020-2021

giuseppe.bruno@unina.it

Docente: Prof. Giuseppe Bruno

I contenuti

- **Alcune definizioni fondamentali**
- **Complessità di un algoritmo**
- **Problemi trattabili e problemi intrattabili**
- **Problemi decisionali**
- **Problemi P ed NP**
- **I problemi NP-hard**

Massimo globale e locale

In termini generali un problema di ottimizzazione può essere formulato come

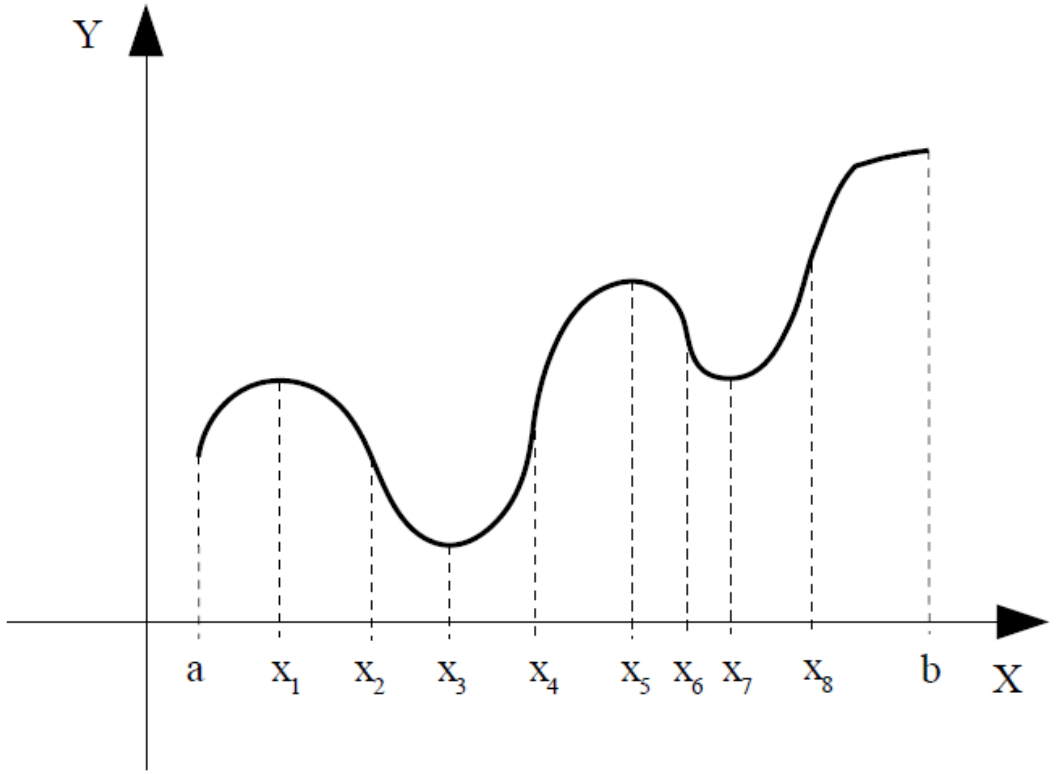
$$z = f(\underline{x}) \qquad Max\ !$$
$$\underline{x} \in X$$

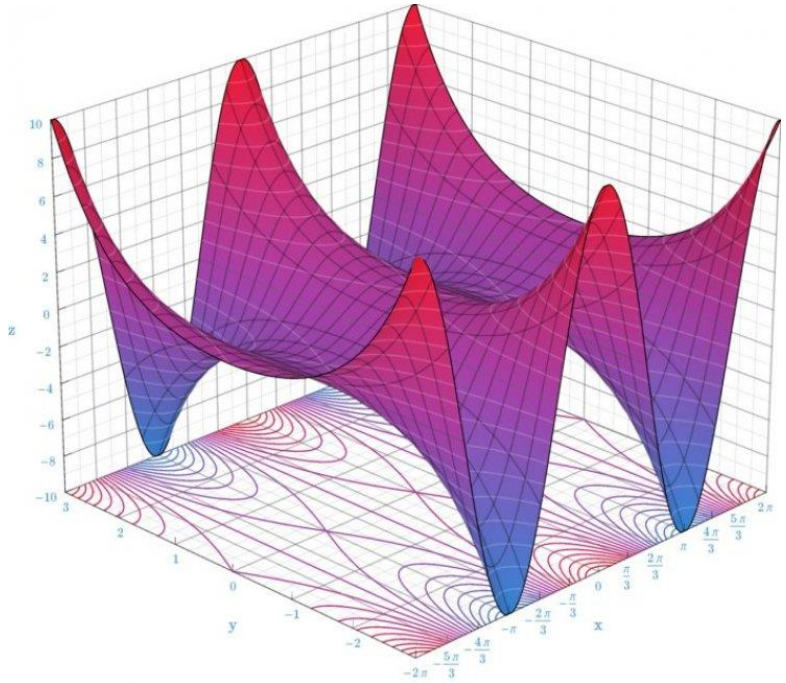
- X insieme delle ***soluzioni ammissibili***
- $z = f(\underline{x})$ ***funzione obiettivo***

Una soluzione $\underline{x}^* \in X$ è un **massimo locale** se $f(\underline{x}^*) \geq f(\underline{x})$ per ogni $\underline{x} \in I_\varepsilon(\underline{x})$

Una soluzione $\underline{x}^* \in X$ è un **massimo** globale se $f(\underline{x}^*) \geq f(\underline{x})$ per ogni $\underline{x} \in X$

Massimo globale e locale





Metodo di risoluzione

Un ***metodo di risoluzione*** è una tecnica che consente di trovare soluzioni del problema.

Un metodo di risoluzione può essere

esatto se individua un ottimo globale ammissibile

euristico se è orientato alla ricerca di una buona soluzione (ma non necessariamente ottima).

Istanza e dimensione di un problema

Un'**istanza** di un problema rappresenta un esempio del problema in esame

Dato un problema Π, l'**istanza** I di Π ha **dimensione** n, se n è il numero di informazioni che la caratterizzano e che devono essere elaborate per risolvere il problema

Istanza del problema dello zaino

Si consideri uno zaino di peso massimo 25 e volume massimo 40 e un insieme di oggetti per ciascuno dei quali si riporta in Tabella il valore c_i, il peso p_i, e il volume v_i. Si formuli il modello matematico per la individuazione degli oggetti di valore totale massimo da inserire nello zaino nel rispetto dei vincoli sul peso e la capacità.

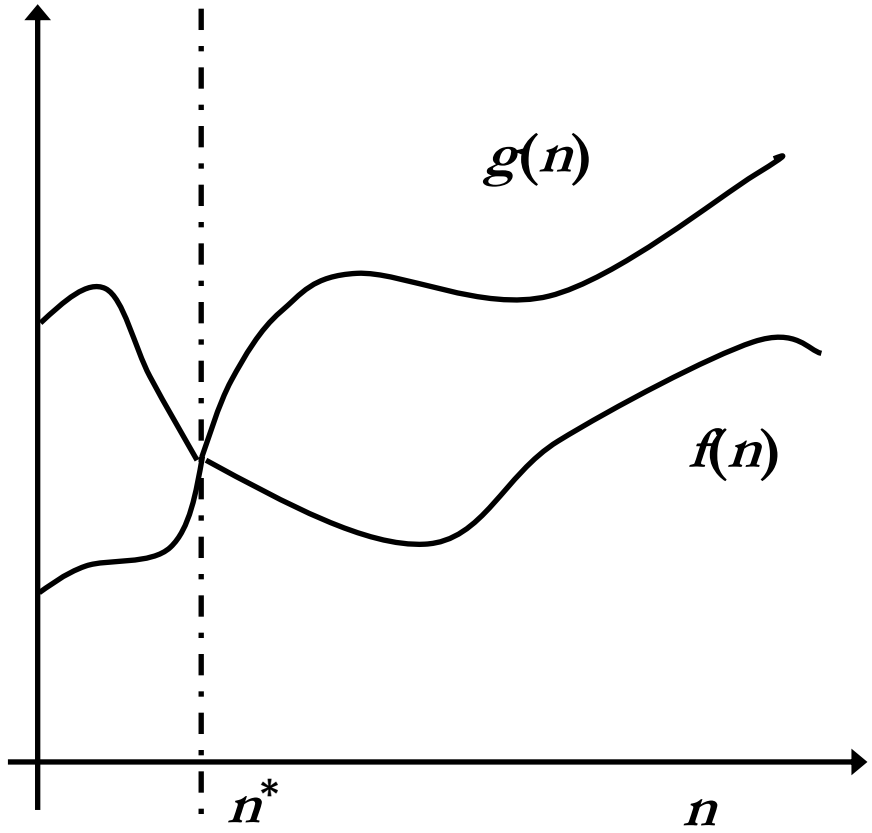
Oggetto	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***
Valore c_i	*10*	*8*	*12*	*15*	*20*	*10*	*10*	*18*
Peso p_i	*8*	*8*	*6*	*5*	*7*	*4*	*5*	*10*
Volume v_i	*16*	*10*	*15*	*13*	*8*	*9*	*11*	*15*

Istanza di dimensione 3x8=24

Notazione

Una funzione f(n) è detta O(g(n)) **($f(n)=O(g(n))$)** se esistono un $n^* \geq 0$ e un $c \geq 1$ tali che $f(n) \leq c \cdot g(n)$ per ogni $n \geq n^*$

La notazioni *O*(.) consente di definire, per una data funzione, un **limite superiore** rispetto ad un fattore moltiplicativo

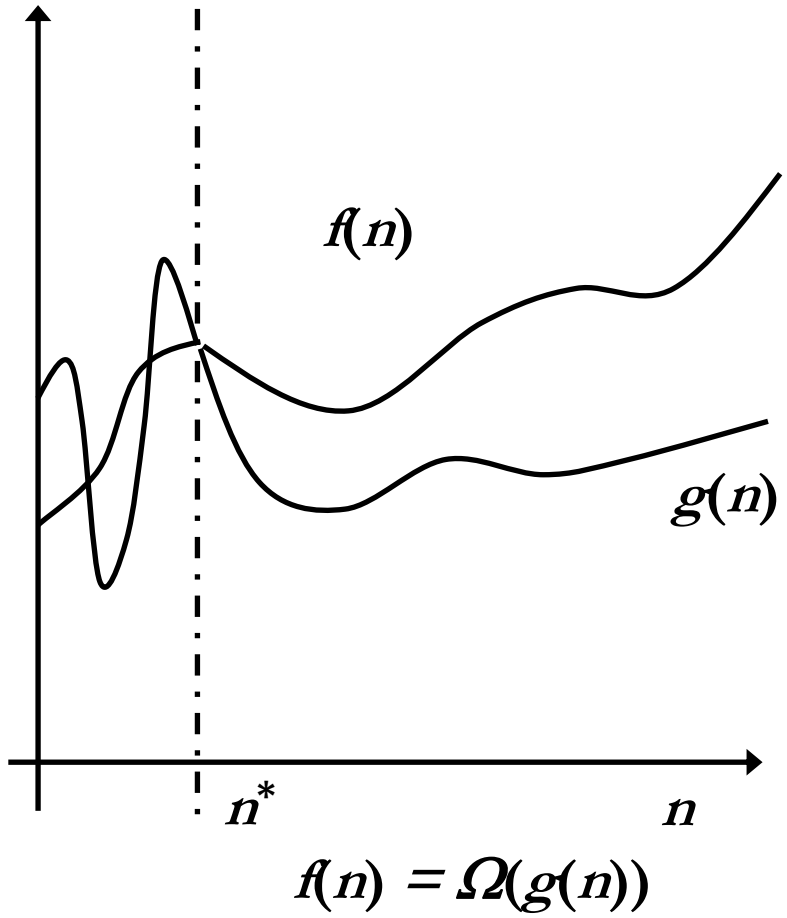


$f(n) = O(g(n))$

Notazione

Una funzione f(n) è detta $\Omega(g(n))$ $(f(n) = \Omega(g(n))$ se esistono un $n^* \geq 0$ e un $c \geq 1$ tali che $f(n) \geq c \cdot g(n)$ per ogni $n \geq n^*$

La notazioni Ω(.) consente di definire, per una data funzione, un **limite inferiore** rispetto ad un fattore moltiplicativo



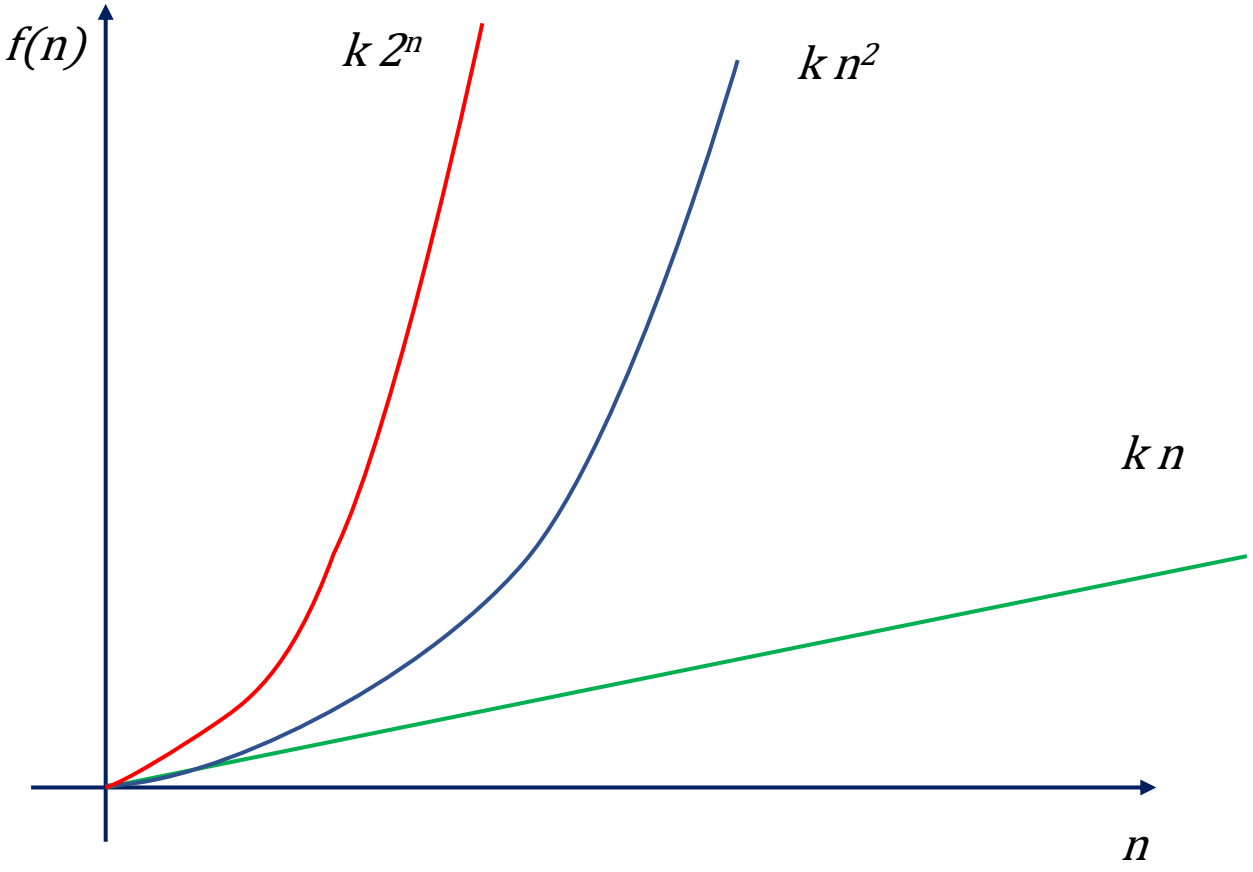
Funzione di complessità di un algoritmo

Rappresenta, per ogni possibile valore *n* della dimensione del problema, il **massimo numero di operazioni** necessario all'algorimo a risolvere un problema di dimensione *n*.

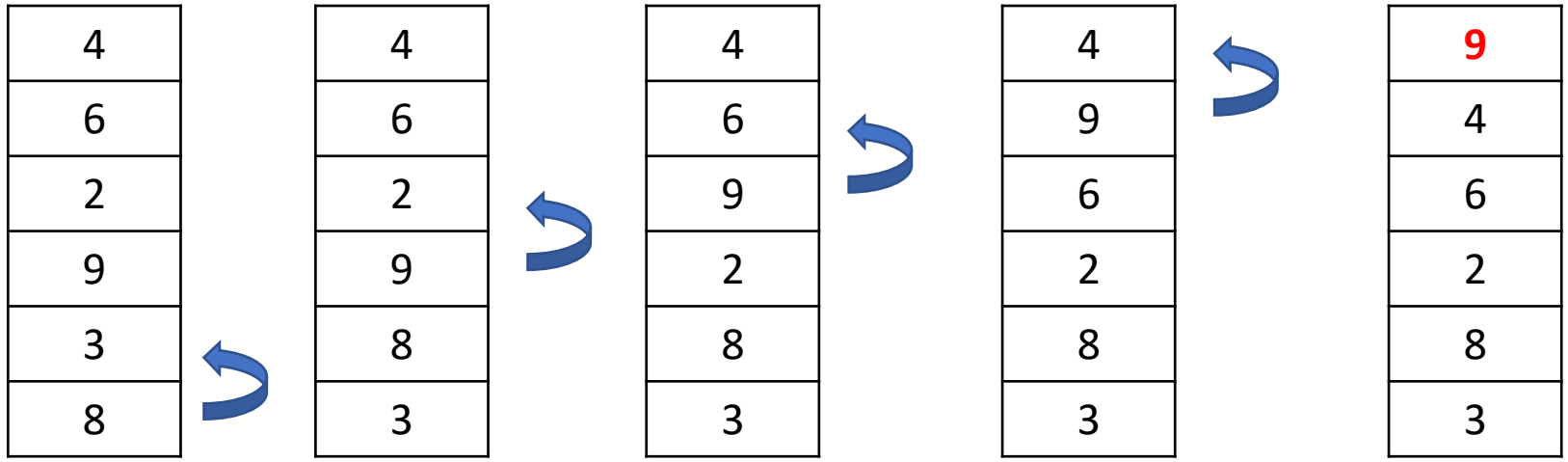
Un algoritmo è di tipo **polinomiale** se la sua funzione di complessità
$\mathbf{f(n)=O(n^k)}$

Ogni algoritmo la cui funzione di complessità non è polinomiale è detto di tipo **esponenziale,** ovvero $\mathbf{f(n)= \Omega(g(n))}$ con *g(n) funzione esponenziale*

Funzione di complessità di un algoritmo

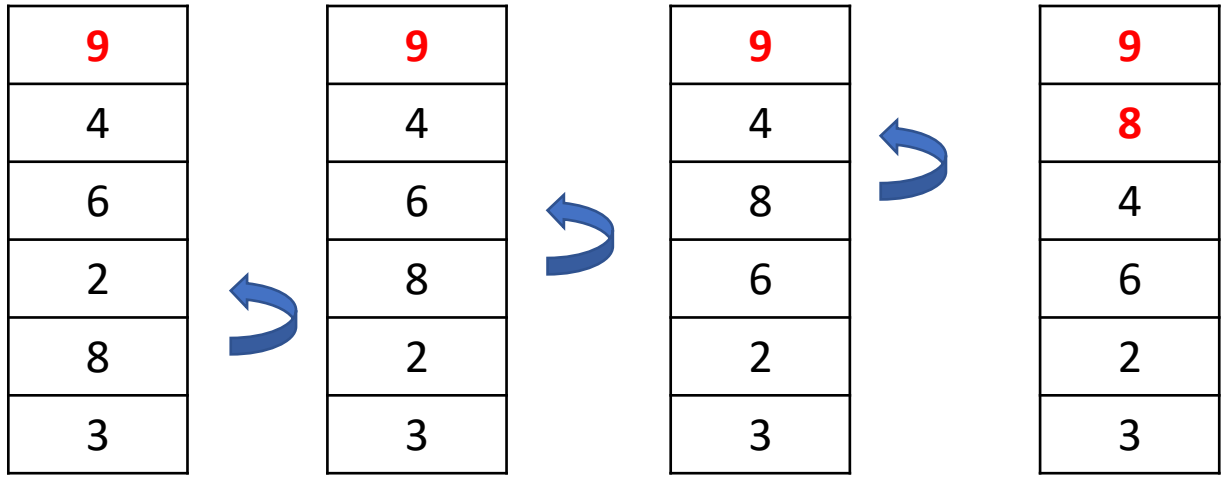


Funzione di complessità di un algoritmo

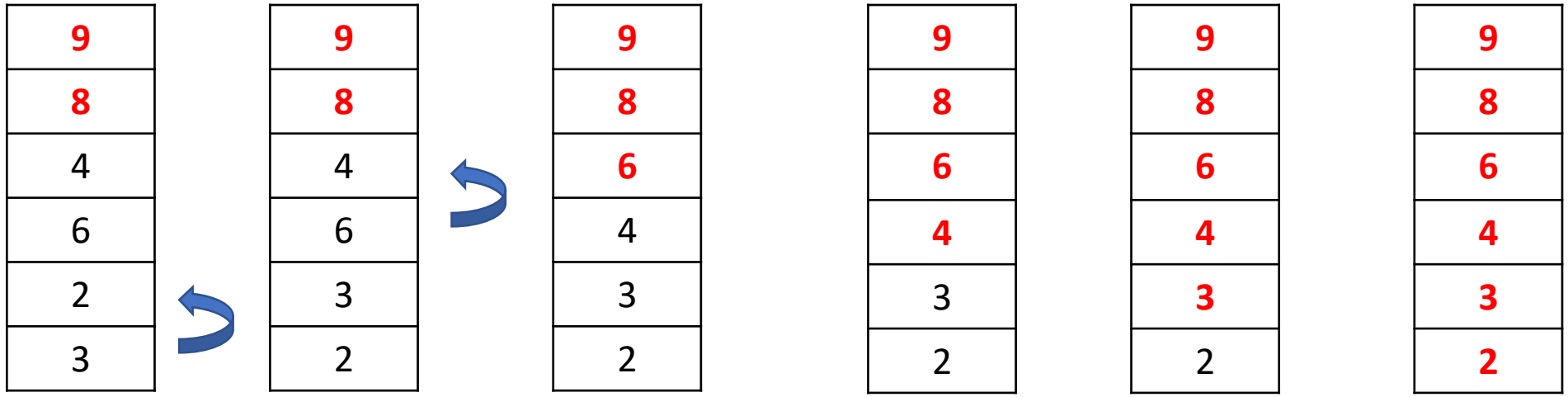


Si ordini il vettore utilizzando un algoritmo di bubble sort
Istanza dimensione 6

Funzione di complessità di un algoritmo



Funzione di complessità di un algoritmo



Funzione di complessità di un algoritmo

Fissata la dimensione n dell'istanza, il numero di operazioni da effettuare dipende dalla specifica istanza

4
6
2
9
3
8

9
8
6
4
3
2

2
3
4
6
8
8

La funzione di complessità fa riferimento al **caso peggiore**, ovvero, pe ogni dimensione, all'istanza che richiede il numero massimo di operazioni

Problemi trattabili e intrattabili

Un problema si dice ***trattabile*** se esiste un algoritmo polinomiale in grado di risolverlo.

In caso contrario, ovvero se non può esistere un algoritmo polinomiale in grado di risolverlo, si dice ***intrattabile*.**

Problemi trattabili e intrattabili

Dimensione del problema

Complessità algoritmo risolutivo

	10	20	30	40	50	60
n	10×10^{-6}	20×10^{-6}	30×10^{-6}	40×10^{-6}	50×10^{-6}	60×10^{-6}
n^2	1×10^{-4}	$2^2\times10^{-4}$	$3^2\times10^{-4}$	$4^2\times10^{-4}$	$5^2\times10^{-4}$	$6^2\times10^{-4}$
n^3	1×10^{-3}	$2^3\times10^{-3}$	$3^3\times10^{-3}$	$4^3\times10^{-3}$	$5^3\times10^{-3}$	$6^3\times10^{-3}$
n^5	1×10^{-1}	$2^5\times10^{-1}$ =3,2 s	$3^5\times10^{-1}$ = 24,3 s	$4^3\times10^{-3}$ = 1,7 m	$5^3\times10^{-3}$ = 5,2 m	$6^3\times10^{-3}$ = 13,0 m

Algoritmi polinomiali

Algoritmi esponenziali

2^n	10^{-3} s	1,0 s	17,9 min	12,7 giorni	35,7 anni	366 secoli
3^n	0,059 s	58 min	6,5 anni	3855 secoli	$2{,}0\times10^{8}$ secoli	$1{,}3\times10^{13}$ secoli

Tabella di Garey and Johnson (1979)

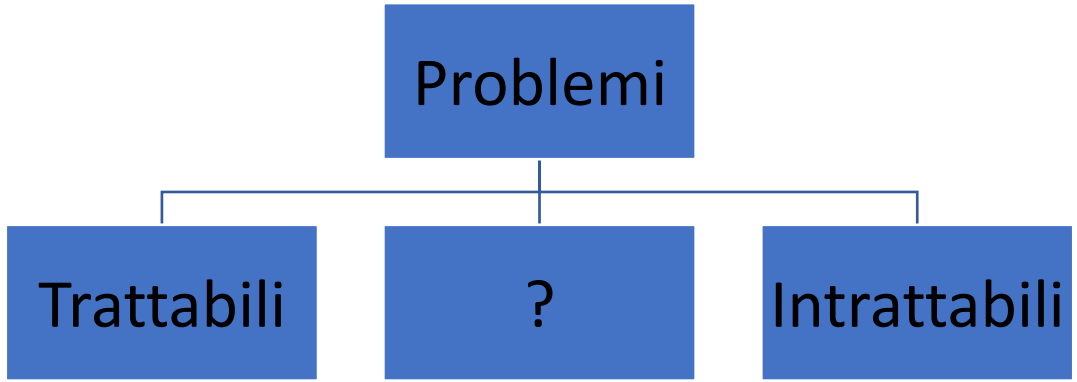
Si assume che la singola operazione viene eseguita in 10^{-6} secondi

Problemi trattabili e intrattabili

Per dimostrare che un problema è ***intrattabile*** bisogna dimostrare **che non può esistere** un algoritmo polinomiale in grado di risolverlo.

Come si classificano tutti quei problemi per i quali

- non esiste un algoritmo polinomiale in grado di risolverli
- non è stato dimostrato che **tale algoritmo non può esistere**



Problemi di ottimizzazione combinatoria

Il problema o modello dello zaino (Knapsack Problem)

Definizione

Si consideri uno **zaino** di **capacità massima *C*** ed ***n* oggetti** a ciascuno dei quali è associato un **valore** v_j e un **peso** p_j. Bisogna individuare la combinazione di oggetti di valore totale massimo da inserire nello zaino nel rispetto del vincolo sulla capacità.

Formulazione

Se si associa a ciascun oggetto *j* la variabile binaria x_j=0/1 che, se pari a 1, indica la presenza dell'oggetto nello zaino (0 altrimenti), il modello matematico assume la forma

$$z = \sum_j v_j x_j \qquad Max!$$

sottoposto a

$$\sum_j p_j x_j \leq C$$

$$x_j = 0/1 \quad j=1,..,n$$

Problemi di ottimizzazione combinatoria

Il problema o modello dello zaino (Knapsack Problem)

Esempio numerico

$$z = 3x_1 + 5x_2 + 8x_3 + 5x_4 + 9x_5 + 12x_6 + 6x_7 + 15x_8 \qquad Max!$$

sottoposto a

$$10x_1 + 12x_2 + 8x_3 + 7x_4 + 5x_5 + 8x_6 + 4x_7 + 15x_8 \leq 28$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 = 0/1$$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	**0**	**1**	**1**	**1**	**0**	**1**	**0**

Numero massimo soluzioni ammissibili: 2^n

(soluzione a combinazione)

Problemi di ottimizzazione combinatoria

Il problema del commesso viaggiatore (Travelling Salesman Problem)

Definizione

Date n città, si individui il circuito di lunghezza minima che un commesso viaggiatore deve effettuare per visitare ciascuna città una sola volta

Formulazione

Se si introducono le variabili binarie x_{ij}=0/1 che, se pari a 1, indicano che la città j segue la città i (presenza di un arco di collegamento tra *i* e *j*) e indicato con c_{ij} la distanza dalla città i alla città j, il modello matematico assume la forma

$$z = \sum_{ij} c_{ij} x_{ij} \quad Max!$$

$$\sum_{i} x_{ij} = 1 \quad j = 1..n$$

Da ogni città *i* deve uscire un solo arco di collegamento

$$\sum_{j} x_{ji} = 1 \quad i = 1..n$$

In ogni città *i* deve entrare un solo arco di collegamento

$$x_{ij} = 0/1 \qquad i, j = 1..n$$

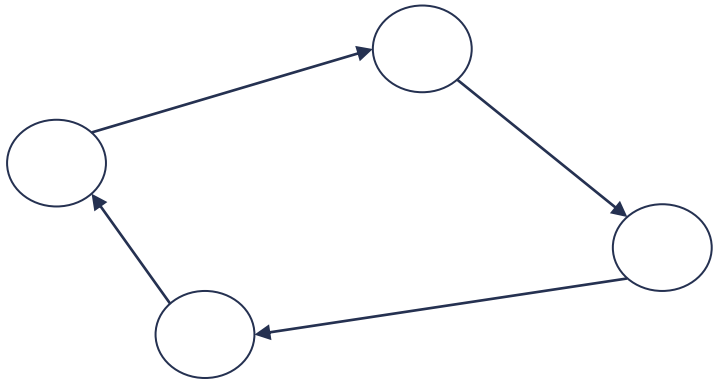
Problemi di ottimizzazione combinatoria

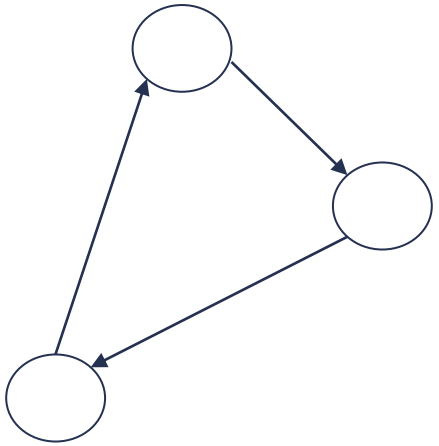
$$z = \sum_{ij} c_{ij} x_{ij} \quad Max!$$

$$\sum_{i} x_{ij} = 1 \quad j = 1..n$$

$$\sum_{j} x_{ji} = 1 \quad i = 1..n$$

$$x_{ij} = 0/1 \qquad i, j = 1..n$$





Questa formulazione renderebbe possibile la formazione di sottocircuiti e, quindi, di soluzioni non ammissibili

Problemi di ottimizzazione combinatoria

Il problema del commesso viaggiatore (Travelling Salesman Problem)

Definizione

Date n città, si individui il circuito di lunghezza minima che un commesso viaggiatore deve effettuare per visitare ciascuna città una sola volta

Formulazione

Se si introducono le variabili binarie x_{ij}=0/1 che, se pari a 1, indicano che la città j segue la città i (presenza di un arco di collegamento tra *i* e *j*) e indicato con c_{ij} la distanza dalla città i alla città j, il modello matematico assume la forma

$$z = \sum_{ij} c_{ij} x_{ij} \quad Max!$$

$$\sum_{i} x_{ij} = 1 \quad j = 1..n$$

Da ogni città *i* deve uscire un solo arco di collegamento

$$\sum_{j} x_{ji} = 1 \quad i = 1..n$$

In ogni città *i* deve entrare un solo arco di collegamento

$$x_{ij} \in S$$

Vincoli eliminazione sottocircuiti

$$x_{ij} = 0/1 \quad i,j = 1..n$$

Problemi di ottimizzazione combinatoria

Il problema o modello del commesso viaggiatore

Esempio numerico relativo ad un problema di n=6 città

Possibile rappresentazione soluzione ammissibile attraverso la sequenza di visita delle città

3	6	1	2	5	4

Valore funzione obiettivo associato alla soluzione

$$z = c_{36} + c_{61} + c_{12} + c_{25} + c_{54} + c_{46} =$$
$$10+13+8+9+10+9=59$$

	1	2	3	4	5	6
1	0	8	12	9	7	13
2	8	0	11	12	9	10
3	12	11	0	12	12	10
4	9	12	12	0	10	9
5	7	9	12	10	0	12
6	13	10	10	9	12	0

Matrice delle distanze tra le città

Numero massimo soluzioni ammissibili: **n!**

(soluzione a permutazione)

Problemi decisionali

La teoria della complessità computazionale o della “NP completezza” (NP-Completeness) punta ad un’ulteriore classificazione dei problemi facendo riferimento ai cosiddetti problemi decisionali

Un problema decisionale è un problema formulato in modo da ammettere due sole possibili risposte: SI o NO

Esempio:

Dato un numero intero K, esiste una coppia di numeri interi m ed n (con $m,n>1$) tali che $K=mn$?

Problemi decisionali

Ogni problema di ottimizzazione può essere formulato come problema decisionale

Esempio del problema del commesso viaggiatore (TSP)

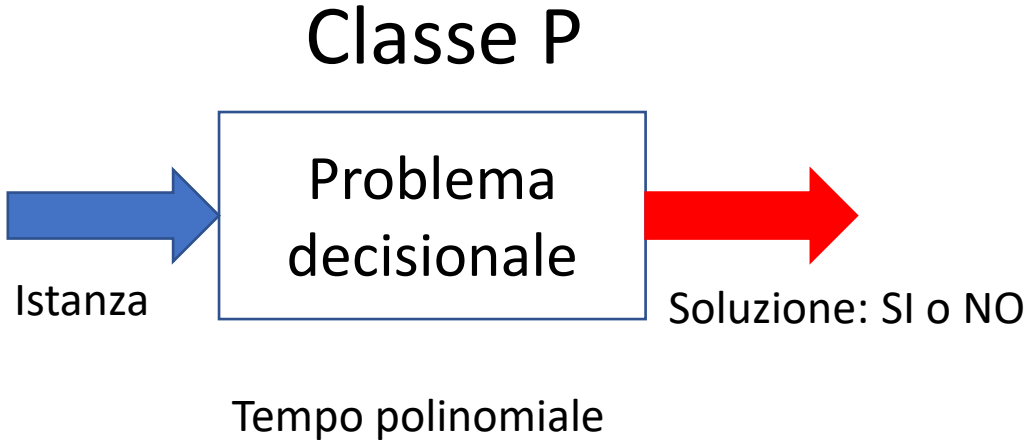
Problema di **ottimizzazione** del TSP
Date n città, si individui il circuito di lunghezza minima che un commesso viaggiatore deve effettuare per visitare ciascuna città una sola volta

Problema **decisionale** del TSP
Date n città, esiste un circuito che un commesso viaggiatore può effettuare per visitare ciascuna città una sola volta che risulti di lunghezza non superiore ad un valore B (≤B)?

Poiché per risolvere un problema di ottimizzazione si devono risolvere più problemi di decisione ad esso associati, si può concludere che
la complessità di un problema di ottimizzazione sarà almeno pari a quello del problema di decisione ad esso associato

Problemi P ed NP

Un problema decisionale è di **tipo P** se esiste un algoritmo polinomiale in grado di fornire, per ogni possibile istanza, una risposta di tipo SI o NO, ossia se è **trattabile**



Date 5 città (A,B,C,D,E) tra le quali è nota la mutua distanza, esiste un circuito che un commesso viaggiatore può effettuare per visitare ciascuna città una sola volta che risulti di lunghezza non superiore a 100?
Posso dare a questo quesito una risposta SI o NO in tempo polinomiale

Il problema del commesso viaggiatore non è un problema della classe P

Problemi P ed NP

Un problema decisionale è di **tipo NP** se esiste un algoritmo polinomiale in grado di **verificare** una risposta di tipo SI

Qualcuno ha trovato che visitando le 5 città nell'ordine B-C-D-A-E si ottiene un circuito di lunghezza 95 (Risposta SI).
E' possibile verificare in tempo polinomiale
- che la successione B-C-D-A-E sia associata ad un circuito di lunghezza 95;
- che la soluzione sia ammissibile per il TSP, ovvero è un circuito hamiltoniano

Il problema del commesso viaggiatore è un problema della classe NP

Problemi P ed NP

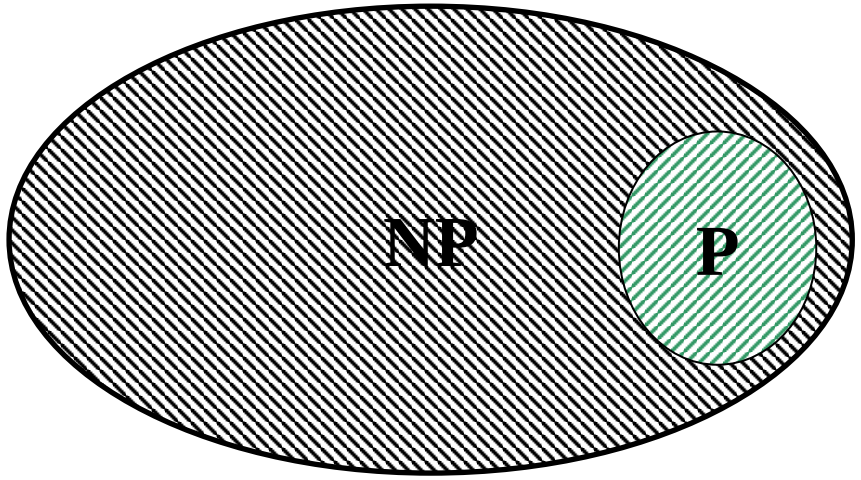
Differenza tra P ed NP
Mentre in un problema P bisogna **trovare** la risposta in tempo polinomiale, in un problema NP, **nel caso ci venga fornita una risposta di tipo SI**, bisogna **verificare che la risposta fornita sia effettivamente di tipo SI** in tempo polinomale.

Problemi P ed NP

Un problema di tipo P è certamente di tipo NP ($P \subseteq NP$) dal momento che l'algoritmo utilizzato per fornire una risposta in tempo polinomiale può essere utilizzato anche per verificare una risposta di tipo SI.

Non è stato ancora dimostrato che $P \subset NP$, ovvero che esistono problemi NP che non possano certamente essere risolti in tempo polinomiale.

Anche se non dimostrato formalmente si ritiene (congettura) che $P \subset NP$ ovvero



Riducibilità tra problemi

Un problema Π è **riducibile** ad un problema Π^* se ogni istanza di Π può essere trasformato in tempo polinomiale in un'istanza di Π^*

$$\Pi \propto \Pi^*$$

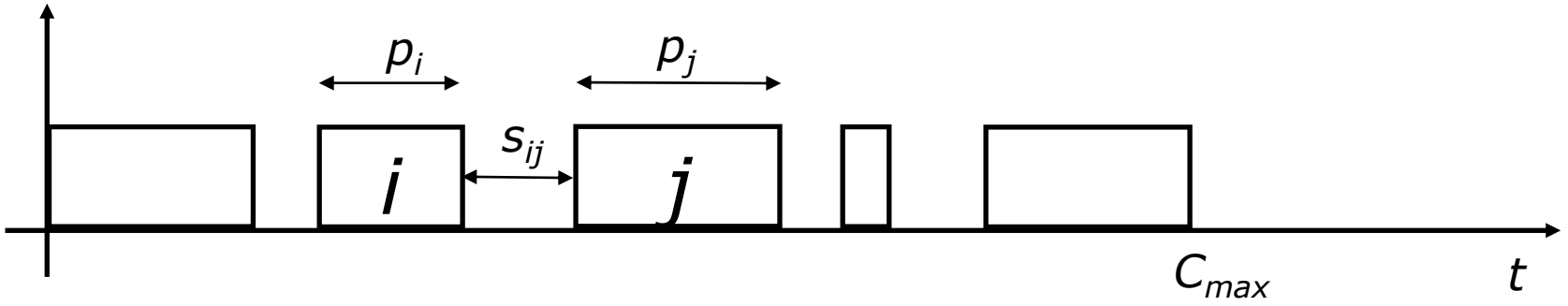
Riducibilità tra problemi

Problema del sequenziamento di operazioni su una linea di produzione

Date n operazioni caratterizzate da

- p_j *il tempo di esecuzione dell'operazione j*
- s_{ij} *il tempo che bisogna attendere per iniziare l'operazione j dopo il completamento dell'operazione i (tempo di setup).*

Esiste un sequenziamento delle operazioni tale che la somma dei tempi di setup non superi il valore B ($C_{max} \leq$ B)?)



Riducibilità tra problemi

Esempio:

Si consideri un problema di sequenziamento di cinque operazioni (A,B,C,D,E) con i tempi di setup indicati

	A	***B***	***C***	***D***	***E***
A	0	3	6	4	8
B	2	0	5	7	6
C	9	5	0	8	9
D	6	4	4	0	7
E	3	5	2	8	0

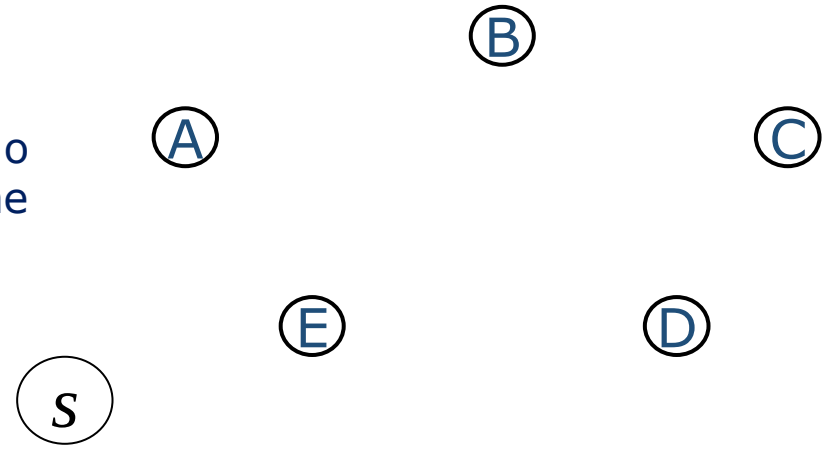
Riducibilità tra problemi

Esempio:

Si consideri un problema di scheduling di cinque operazioni (A,B,C,D,E) con tempi di setup indicati

	A	***B***	***C***	***D***	***E***
A	0	3	6	4	8
B	2	0	5	7	6
C	9	5	0	8	9
D	6	4	4	0	7
E	3	5	2	8	0

Si costruisca un grafo costituito da 6 nodi: un nodo associato ad ogni operazione (A,B,C,D,E) più un nodo fittizio s (start)

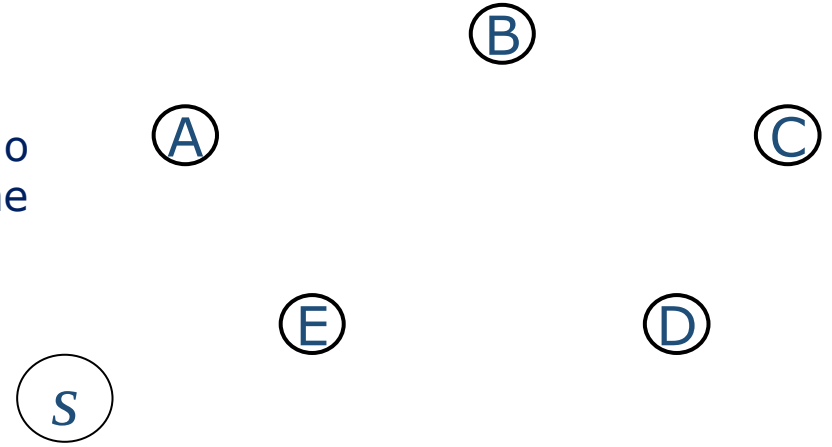


Riducibilità tra problemi

Esempio:
Si consideri un problema di scheduling di cinque operazioni (A,B,C,D,E) con tempi di setup indicati

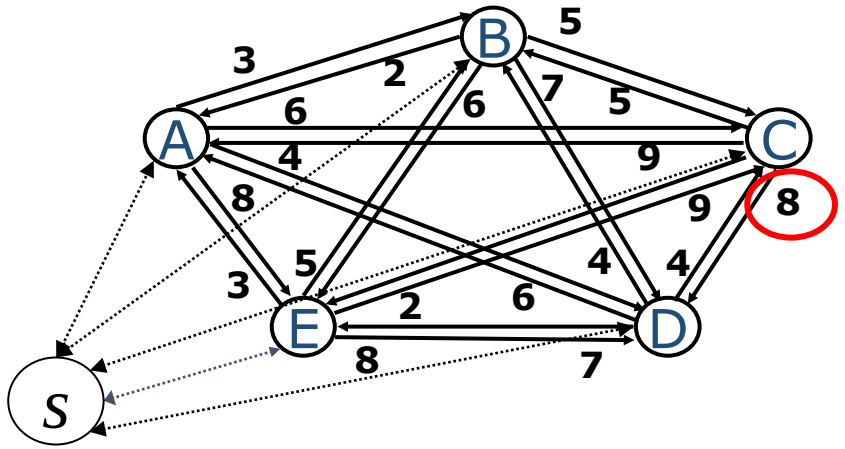
	A	B	C	D	E
A	0	3	6	4	8
B	2	0	5	7	6
C	9	5	0	8	9
D	6	4	4	0	7
E	3	5	2	8	0

Si costruisca un grafo costituito da 6 nodi: un nodo associato ad ogni operazione (A,B,C,D,E) più un nodo fittizio s



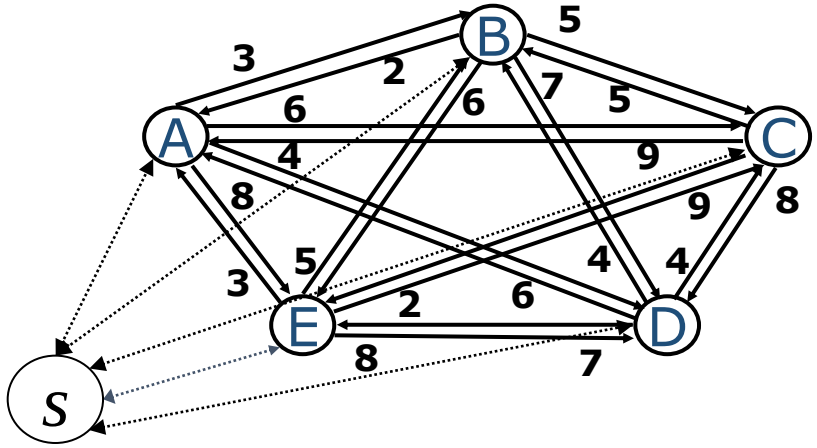
Ad ogni coppia di nodi associati ad un'operazione si associ un arco di costo $c_{ij} = s_{ij}$.

Il nodo s è collegato agli altri nodi con archi bidirezionali di costo nullo.



Riducibilità tra problemi

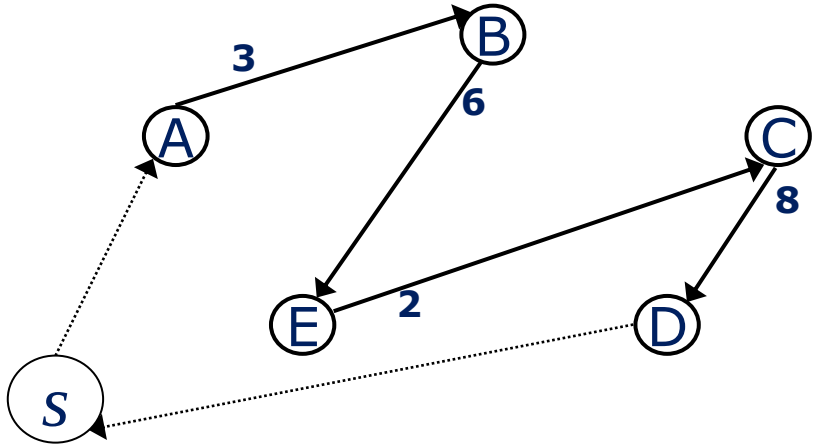
Ad ogni coppia di nodi associati ad un'operazione è associato un arco di costo $c_{ij} = s_{ij}$.
Il nodo s è collegato agli altri nodi con archi bidirezionali di costo nullo.



Si consideri una soluzione del TSP sul grafo costruito

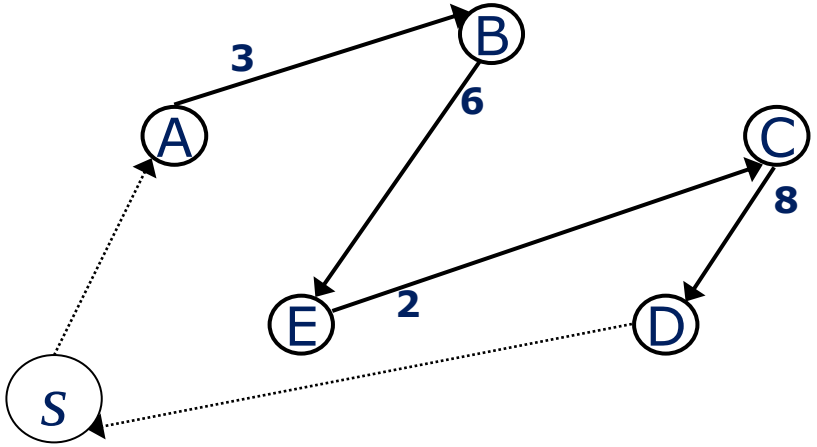
Per esempio
s,A,B,E,C,D,s

di lunghezza z=3+6+2+8=19)

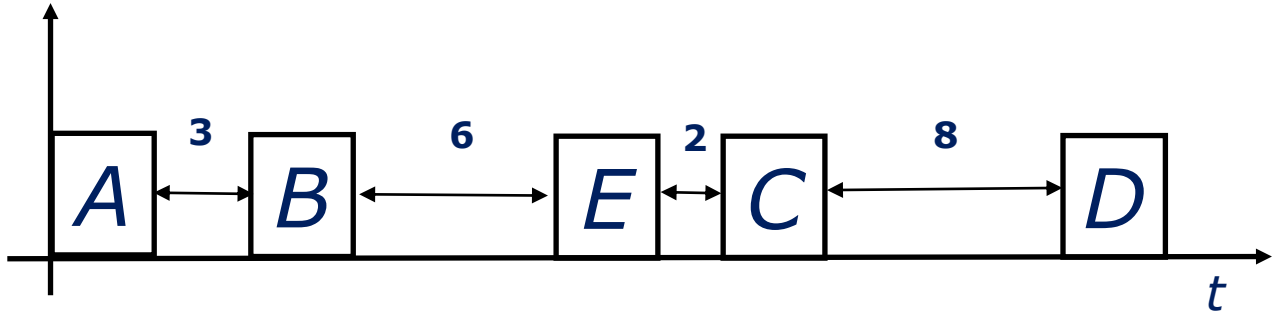


Riducibilità tra problemi

Una soluzione del TSP sul grafo costruito (es: s,A,B,E,C,D,s di costo z=3+6+2+8=19)



Equivale ad una sequenza di operazioni in cui la somma dei tempi di setup coincide con la lunghezza del circuito (es: A,B,E, C,D di costo z=3+6+2+8=19)



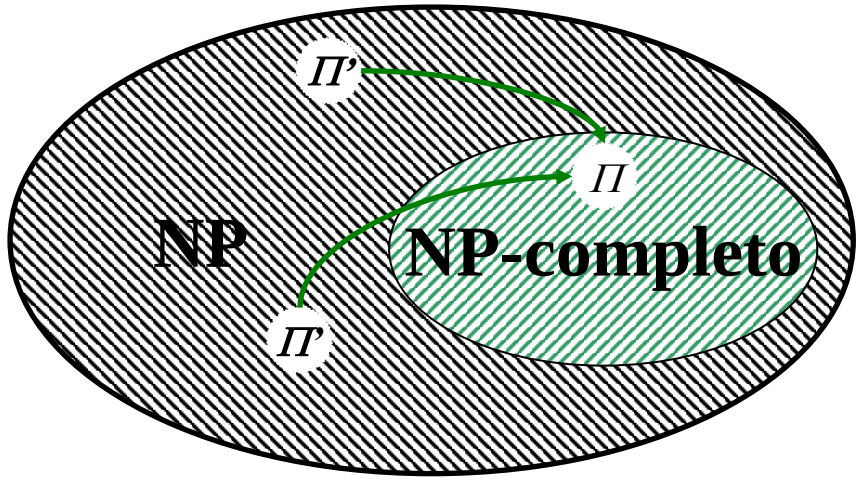
Il problema del sequenziamento può essere ridotto ad un problema di commesso viaggiatore su *n+1* città

NP-completezza

Un problema decisionale Π è di **tipo NP-completo** se:

- Π è di tipo NP

- qualsiasi altro problema Π' di tipo NP può essere ridotto a Π ($\Pi' \propto \Pi$) ovvero tutti i problemi NP possono essere ridotti a Π



NP-completezza

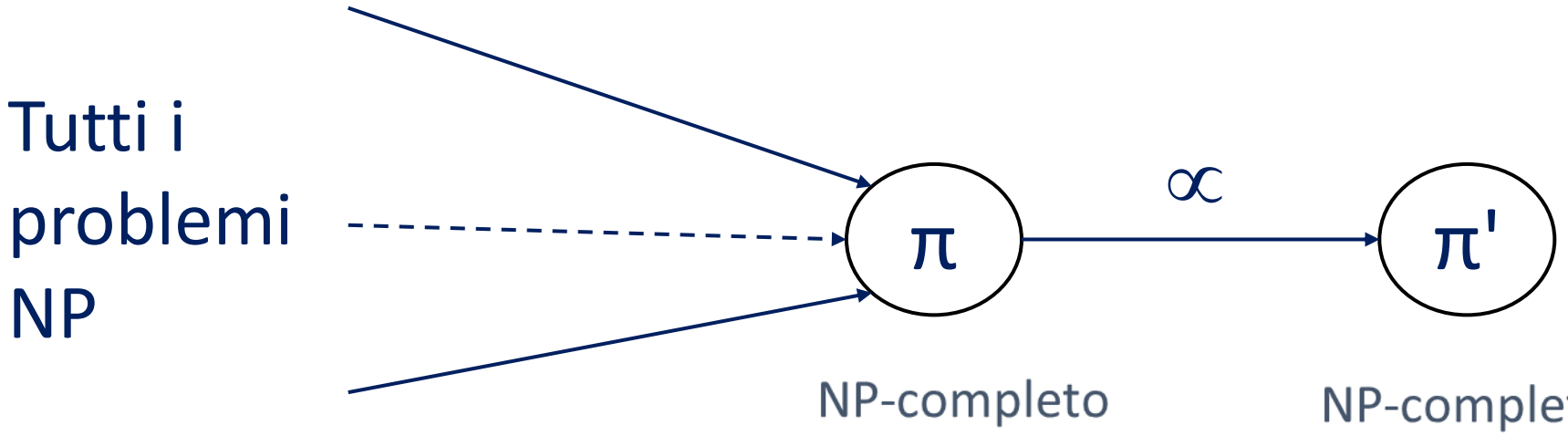
La classe dei problemi NP-completi è una classe di problemi **più difficili** all'interno della classe NP.

Se si individuasse un algoritmo polinomiale per risolvere un qualsiasi problema NP-completo allora tutti i problemi NP sarebbero trattabili ovvero P=NP

Un problema di ottimizzazione la cui versione decisionale appartiene alla classe dei problemi NP-completi è detto

NP-hard (NP-difficile)

NP-completezza



Per poter applicare questa procedura è necessario conoscere un primo problema di **tipo NP-completo**

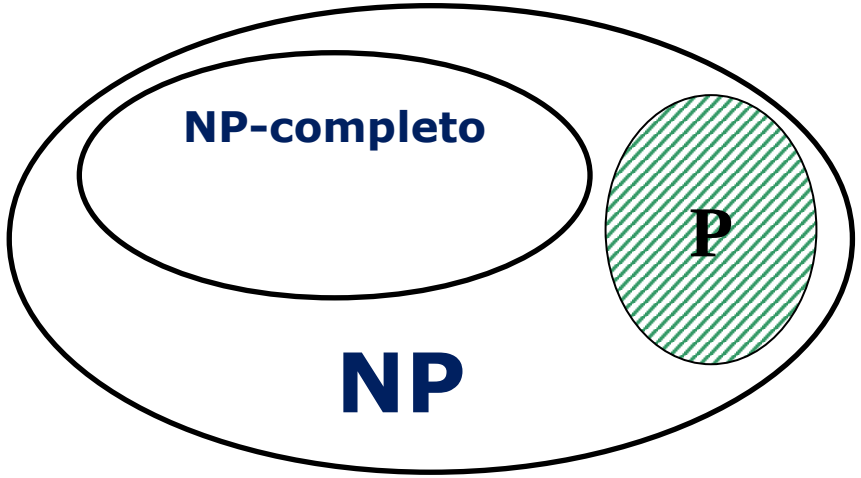
NP-completezza

Nel 1971 il matematico inglese Cook(-Levin) individuò un primo problema **tipo NP-completo**

Da allora, attraverso il procedimento descritto, sono stati individuati numerosi problemi NP-completi

Anche se non è stato dimostrato formalmente **si adotta la congettura (l'ipotesi) P≠NP, ovvero si ritiene che i problemi NP siano intrattabili**

https://en.wikipedia.org/wiki/List_of_NP-complete_problems



NP-completezza

La teoria della **NP-completezza** consente di classificare nella classe dei problemi NP e NP-completo, problemi che non risultano né trattabili, né intrattabili.

Anche se non è stato dimostrato formalmente si adotta la congettura (l'ipotesi) $P \neq NP$, ovvero si ritiene che i problemi NP siano intrattabili.

Né consegue che i problemi NP vanno trattati, in pratica, come se fossero intrattabili. Pertanto poiché i tempi di calcolo di algoritmi risolutivi esatti applicati a questi problemi risulyano esponenziali, nelle applicazioni pratiche bisogna ricorrere ad algoritmi di risoluzione euristici.