COMPLESSITÀ DEGLI ALGORITMI

Martina Gusai

23.04.2021

Argomenti trattati

- Introduzione
- Complessità computazionale
  - La classe $\mathcal{P}$
  - Le classi $\mathcal{NP}$ e co $-\mathcal{NP}$
  - Congettura di Cook-Edmonds-Levin
  - Congettura di Edmond
- Riduzione Polinomiale
- Problemi $\mathcal{NP} - completi$
  - La classe $\mathcal{NPC}$
  - Formule booleane
  - Soddisfacibilità delle formule booleane

Introduzione

In questo seminario parleremo in modo più formale del concetto di efficienza algoritmica utilizzato in precedenza.

Sottolineiamo nuovamente la differenza tra quegli algoritmi i cui tempi di esecuzione sono delimitati da un polinomio nella dimensione dell'input e quelli che non lo sono. Inoltre, introduciamo un'ampia classe di problemi, i cosiddetti problemi $\mathcal{NP} - completi$, che sono ritenuti intrattabili. Questa convinzione è in gran parte basata sul fatto che, nonostante i grandi sforzi nella ricerca di algoritmi efficienti, non vene sia noto nessuno. Inoltre, se un tale algoritmo fosse noto per qualcuno di questi problemi, allora esisterebbe per chiunque degli algoritmi.

Vedremo che ogni problema, per il quale non è stato possibile fornire una soluzione efficiente in precedenza, è $\mathcal{NP} - completo$:

Complessità Computazionale

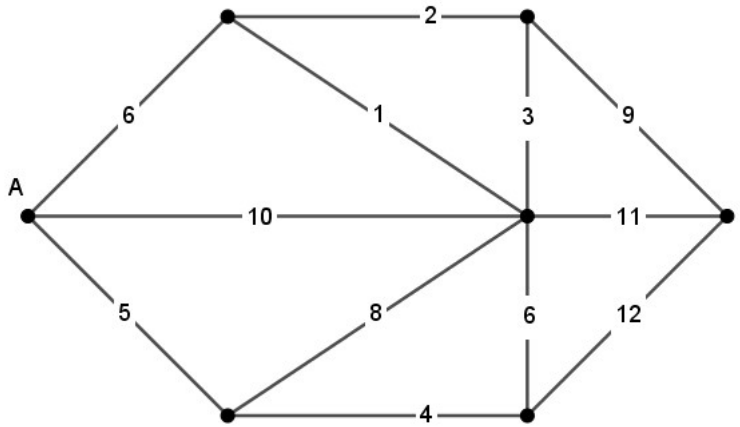
Per istanza di un problema, intendiamo il problema applicato a un membro specifico della famiglia.

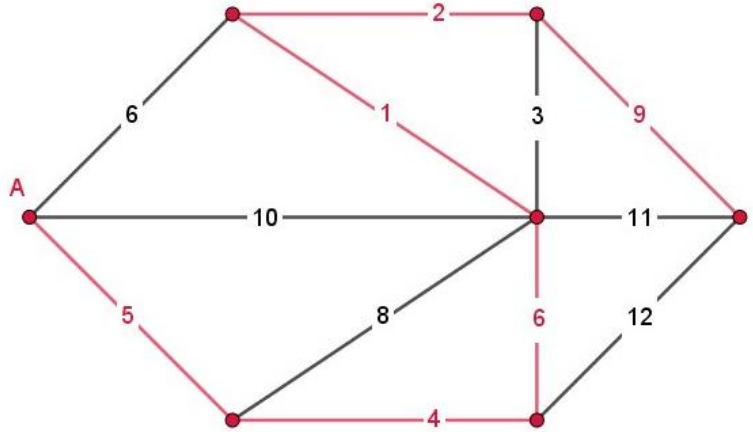
Un algoritmo per la risoluzione di un problema è una procedura di calcolo ben definita che accetta qualsiasi istanza del problema come input e restituisce una soluzione al problema come output.

Esempio:Algoritmo di Jarnik-Prim

Input: grafo pesato e connesso (G,w)

Output: albero ottimale T di G





I due aspetti di interesse teorico degli algoritmi sono:

- Verificare che un algoritmo proposto funzioni effettivamente in modo corretto;
- Analizzare quanto sia efficiente una procedura.

Per complessità computazionale (o complessità) di un algoritmo, si intende il numero di passaggi base, operazioni aritmetiche e confronti, necessari per la sua esecuzione.

Osservazioni:

- Questo numero dipende dalla dimensione e dalla natura dell'input;
- Quando l'input include informazioni aggiuntive, come i pesi sui vertici o sui bordi del grafo, anche questo deve essere preso in considerazione nel calcolo della complessità;
- Il numero di operazioni considerate è valutato nel caso peggiore.

La classe $\mathcal{P}$

Un problema è risolubile in un tempo polinomiale se esiste un algoritmo che lo risolve nel tempo $O(n^k)$, per qualche k costante.

Quindi, possiamo definire la classe di complessità $\mathcal{P}$ come l'insieme dei problemi che sono risolubili in un tempo polinomiale.

Gli algoritmi risolubili in un tempo polinomiale di solito si trovano fattibili dal punto di vista computazionale, anche per grafi di input di grandi dimensioni.

Esempi:

- Nella ricerca in ampiezza

  Prima viene eseguita una fase di inizializzazione lineare nel numero di vertici, O(n)

  Dopo nei due cicli vengono presi in esame tutti gli spigoli del grafo per cercare un vertice ancora da visitare, O(m)

- Lo stesso vale per la ricerca in profondità

  Prima viene eseguita una fase di inizializzazione lineare nel numero di vertici, O(n)

  Dopo il ciclo e la ricorsione vengono presi in esame tutti gli spigoli del grafo per cercare un vertice ancora da visitare, O(m)

Pertanto, questi algoritmi sono lineari in n e m,O(n+m).

Al contrario, gli algoritmi la cui complessità è esponenziale nella dimensione dell'input hanno tempi di esecuzione che li rendono inutilizzabili, anche su input di dimensioni moderate.

Esempio: Un algoritmo che controlla se due grafi su n vertici sono isomorfi considerando tutti gli n! biiezioni tra i loro vertici, è possibile solo per piccoli valori di n (certamente non maggiori di 20).

Ci sono molti problemi base per i quali devono ancora essere trovati algoritmi in $\mathcal{P}$, e potrebbero non esistere proprio.

Determinare quali problemi sono risolubili in tempo polinomiale e quali no è una questione fondamentale.

La classe di problemi denotata con $\mathcal{NP}$ (tempo polinomiale non deterministico) gioca un ruolo importante.

Le classi $\mathcal{NP}$ e *co-* $\mathcal{NP}$

Un problema decisionale è una domanda la cui risposta è «si» o «no».

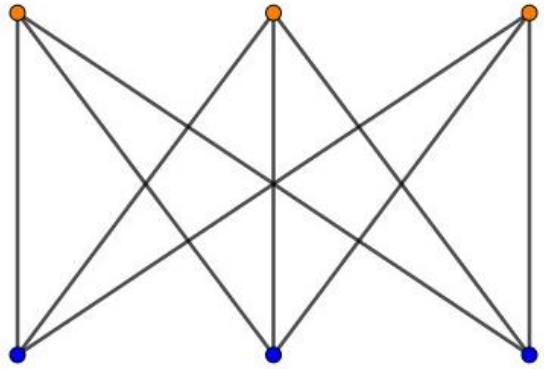
Tale problema appartiene alla classe $\mathcal{P}$ se esiste un algoritmo che risolve qualsiasi istanza del problema in tempo polinomiale.

Appartiene alla classe $\mathcal{NP}$ se, data una qualsiasi istanza del problema la cui risposta è «si», esiste un certificato che convalida questo fatto che può essere verificato in tempo polinomiale.

Tale certificato si dice succinto.

Esempio: Determinare se un grafo è bipartito

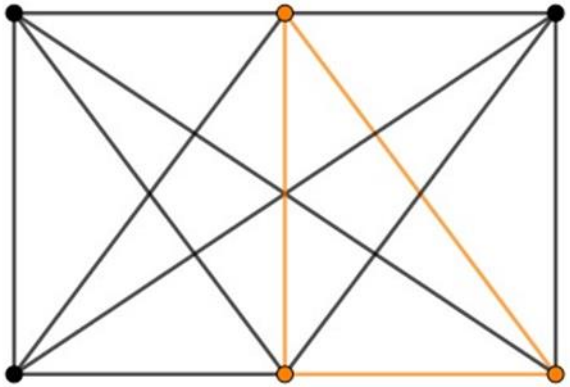
Questo problema è $\mathcal{NP}$, perché una bipartizione è un certificato succinto: data una bipartizione (X,Y) di un grafo bipartito G, è sufficiente verificare che ogni arco di G abbia un'estremità in X e un'estremità in Y.



Analogamente, un problema appartiene alla classe *co-* $\mathcal{NP}$ se, data una qualsiasi istanza del problema la cui risposta è «no», esiste un certificato che convalida questo fatto che può essere verificato in tempo polinomiale.

Esempio: quello precedente del grafo bipartito

Il problema appartiene anche a *co-* $\mathcal{NP}$ perché, per il Teorema 4.7, ogni grafo non bipartito contiene un ciclo dispari, e ogni ciclo di questo tipo costituisce un breve certificato del carattere non bipartito del grafo.



Osservazioni:

- $\left.\begin{array}{l}\mathcal{P} \subseteq \mathcal{NP} \\ \mathcal{P} \subseteq \text{co} - \mathcal{NP}\end{array}\right\} \Rightarrow \mathcal{P} \subseteq \mathcal{NP} \cap \text{co} - \mathcal{NP}$
- Non è detto che se un grafo è $\mathcal{NP}$ allora è anche $\text{co} - \mathcal{NP}$

Consideriamo ora il problema di decidere se un grafo ha un ciclo di Hamilton:

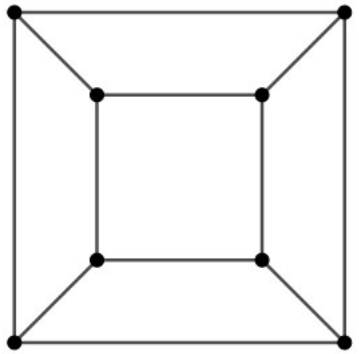
Dato: un grafo G

Decidi: G ha un ciclo di Hamilton?

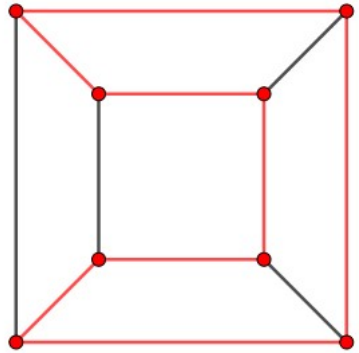
Osservazioni:

- Se la risposta è «si», qualsiasi ciclo di Hamilton servirebbe come un breve certificato

Esempio:



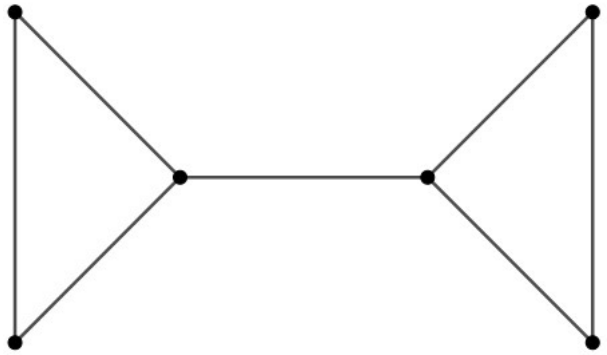
Grafo G



Ciclo di Hamilton

- Se la risposta fosse «no», non avremmo alcun certificato di questo tipo, per ora. In altre parole, nonostante il ciclo di Hamilton sia $\mathcal{NP}$, non è stato ancora dimostrato che appartenga a $\mathrm{co}-\mathcal{NP}$, e potrebbe benissimo non appartenergli.

Esempio:



Molti problemi che sorgono nella pratica, sono problemi di ottimizzazione piuttosto che problemi di decisione. Tuttavia, ciascuno di essi include implicitamente un'infinità di problemi decisionali.

Esempio: Il problema del percorso più breve

Include, per ogni numero reale, la seguente decisione del problema:

Dato un grafo orientato ponderato (D,w) con due vertici x e y, esiste un percorso diretto (x,y) in D di lunghezza l?

Abbiamo notato tre relazioni di inclusione tra le classi $\mathcal{P}$, $\mathcal{NP}$ e $\mathrm{co}-\mathcal{NP}$, ed è naturale chiedersi se queste inclusioni siano proprie. A questo punto vengono poste due congetture:

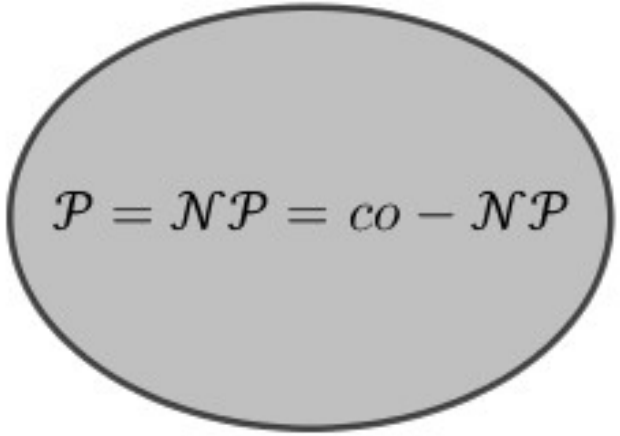
Congettura di Cook-Edmonds-Levin: $\mathcal{P} \neq \mathcal{NP}$

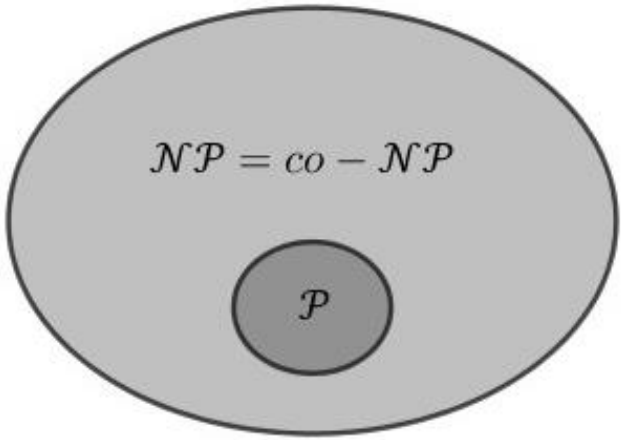
Congettura di Edmonds: $\mathcal{P} = \mathcal{NP} \cap \mathrm{co}-\mathcal{NP}$

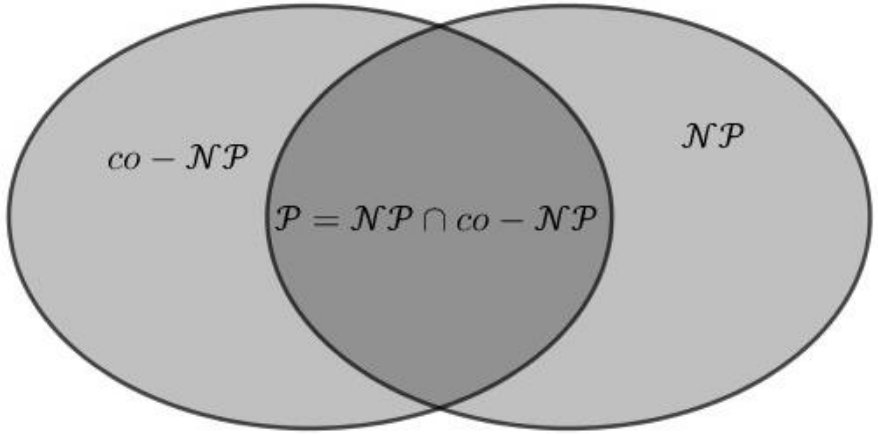
Riguardo la prima è opinione diffusa (ma non universale) che sia vera.

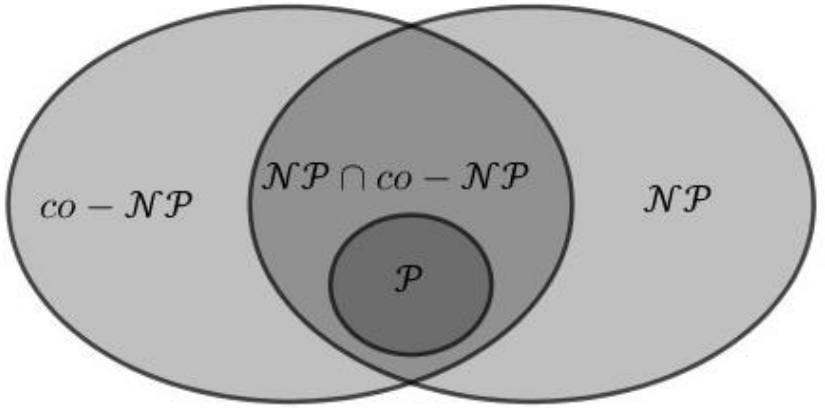
Riguardo la seconda, essa è fortemente supportata da prove empiriche. La maggior parte dei problemi decisionali noti per appartenere a $\mathcal{NP} \cap \mathrm{co}-\mathcal{NP}$ sono anche noti per appartenere a $\mathcal{P}$.

Abbiamo quindi 4 possibili scenari:









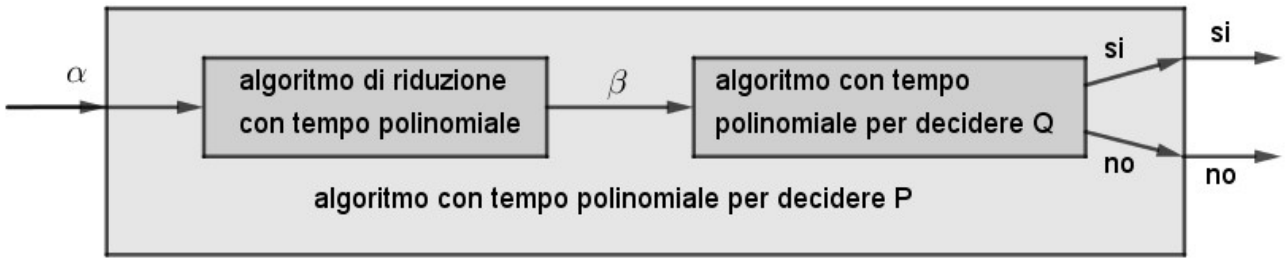
Riduzioni Polinomiali

Un approccio comune alla risoluzione dei problemi consiste nel trasformare il problema dato in uno la cui soluzione è già nota, e quindi convertire quella soluzione in una soluzione del problema originale.

Naturalmente questo approccio è possibile solo se la trasformazione può essere effettuata rapidamente. Il concetto di riduzione polinomiale cattura questo requisito.

Una riduzione polinomiale di un problema P ad un problema Q è una coppia di algoritmi polinomiali, uno che trasforma ogni istanza I di P in un'istanza J di Q, e l'altro che trasforma una soluzione per l'istanza J in una soluzione per l'istanza I.

Se esiste una tale riduzione, diciamo che P è riducibile polinomialmente a Q, e scriviamo $P \preccurlyeq Q$; questa relazione è chiaramente sia riflessiva che transitiva.



Osservazione: Il significato della riducibilità polinomiale è che se P$\preccurlyeq$Q, e se esiste un algoritmo polinomiale per risolvere Q, allora questo algoritmo può essere convertito in un algoritmo polinomiale per risolvere P. In simboli:

$$\mathrm{P} \preccurlyeq \mathrm{Q} \text{ e } \mathrm{Q} \in \mathcal{P} \Rightarrow \mathrm{P} \in \mathcal{P}$$

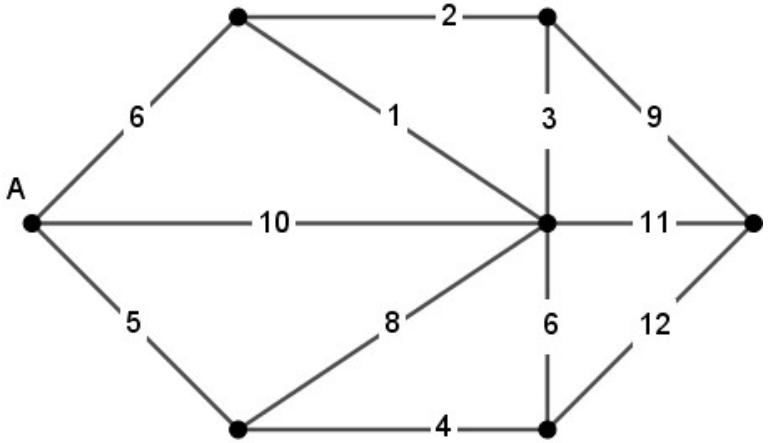
Esempio: La riduzione polinomiale del problema dello Spanning Tree di peso minimo nel seguente problema:

Spanning Tree con peso massimo:

Dato: un grafo connesso ponderato G
Trova: uno spanning tree di peso massimo in G

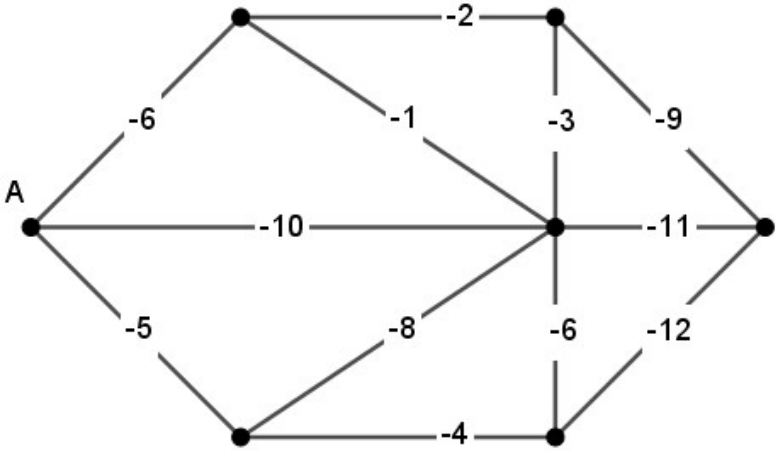
Per risolvere un'istanza di questo problema, è sufficiente sostituire ogni peso con il suo negativo ed applicare l'algoritmo di Jarnik-Prim per trovare un albero ottimale nel grafo ponderato risultante. Lo stesso albero sarà quello di peso massimo nel grafo ponderato originale.

Spanning Tree peso massimo

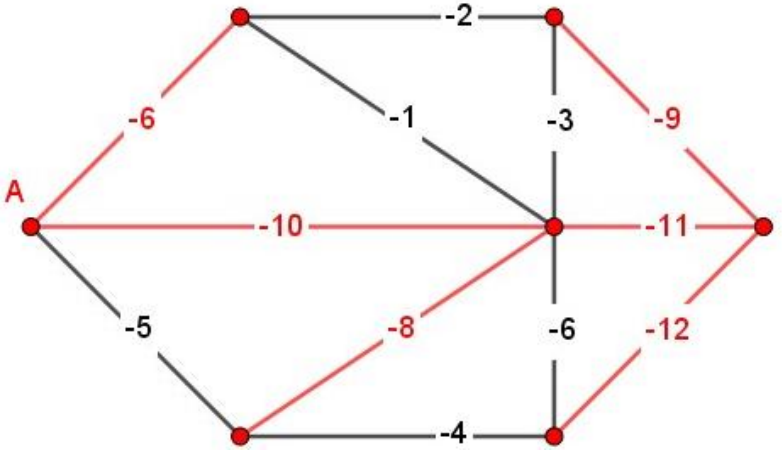


Algoritmo di riduzione
con tempo polinomiale

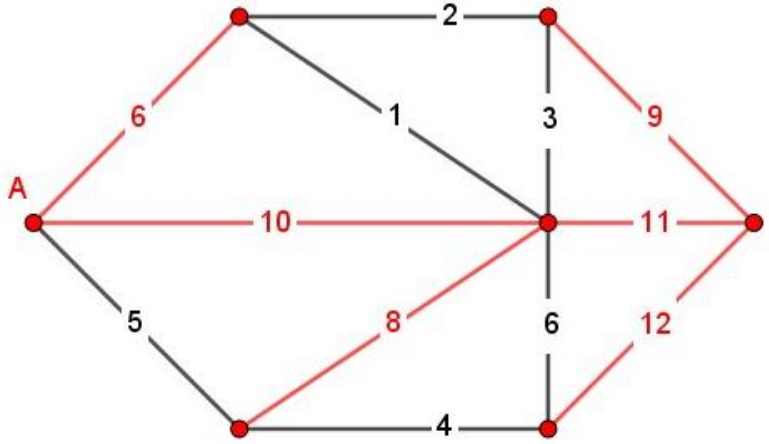
Spanning tree peso minimo



Algoritmo di Jarnik-Prim



Algoritmo per la soluzione



Problemi $\mathcal{NP} - completi$

La classe $\mathcal{NPC}$

Abbiamo appena visto come le riduzioni polinomiali possano essere utilizzate per produrre nuovi algoritmi da quelli esistenti. Allo stesso modo, le riduzioni polinomiali possono anche essere utilizzate per collegare problemi «difficili», quelli per i quali non esiste un algoritmo polinomiale, come si può vedere scrivendo $\mathrm{P} \preccurlyeq \mathrm{Q}$ e $\mathrm{Q} \in \mathrm{P} \Rightarrow \mathrm{P} \in \mathcal{P}$ in una forma diversa:

$$\mathrm{P} \preccurlyeq \mathrm{Q} \text{ e } \mathrm{P} \notin \mathcal{P} \Rightarrow \mathrm{Q} \notin \mathcal{P}$$

Questo punto di vista ha portato Cook e Levin a definire una classe speciale di problemi decisionali apparentemente intrattabili, la classe dei problemi $\mathcal{NP} - completi$. Informalmente, questi sono i problemi nella classe $\mathcal{NP}$ che sono «difficili da risolvere almeno» quanto qualsiasi problema in $\mathcal{NP}$.

Formalmente, un problema P in $\mathcal{NP}$ è $\mathcal{NP} - completo$ se P'≼P per ogni problema P' in $\mathcal{NP}$.

La classe di $\mathcal{NP}$-problemi completi è indicata con $\mathcal{NPC}$.

Per dimostrare che un problema Q in $\mathcal{NP}$ è $\mathcal{NP} - completo$, è sufficiente trovare una riduzione polinomiale di Q in qualche problema noto $\mathcal{NP} - completo$ P.

<u>Perché è così?</u>

Supponiamo che P sia $\mathcal{NP} - completo$ ⇒P'≼P,∀P'∈ $\mathcal{NP}$.
Se P≼Q⇒P'≼Q,∀P'∈ $\mathcal{NP}$, dalla transitività della relazione.

In altre parole, Q è $\mathcal{NP} - completo$. In simboli:

$$\mathrm{P} \preccurlyeq \mathrm{Q} \text{ e } \mathrm{P} \in \mathcal{NPC} \Rightarrow \mathrm{Q} \in \mathcal{NPC}$$

Cook e Levin fecero una svolta fondamentale dimostrando che esistono effettivamente problemi $\mathcal{NP} -$ completi. Più precisamente, hanno dimostrato che il problema di soddisfacibilità per le formule booleane è $\mathcal{NP} -$ completo.

Formule Booleane

Una variabile booleana è una variabile che assume uno dei due valori, 0 («falso») o 1 («vero»).

Le variabili booleane possono essere combinate in formule booleane, che possono essere definite ricorsivamente come segue:

- Ogni variabile booleana è una formula booleana;
- Se f è una formula booleana, allora lo è anche (¬f) la negazione di f;
- Se f e g sono formule booleane, allora lo sono anche:
  - (f∨g), la disgiunzione di f e g,
  - (f∧g), la congiunzione di f e g.

Osservazione: La negazione di una variabile booleana di x viene spesso scritta come $\bar{x}$.

Esempio:

$$(\neg(x_1 \vee \overline{x_2}) \vee x_3) \wedge (x_2 \vee \overline{x_3})$$

è una formula booleana nelle variabili x_1, x_2, x_3.

Un’assegnazione di valori alle variabili di una formula booleana è chiamata assegnazione di verità.

Dato un assegnamento di verità, il valore della formula può essere calcolato secondo le seguenti regole:

$\neg$	
0	1
1	0

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

Esempio: Riprendo la formula di prima con x_1=1, x_2 =0, x_3 =1

$$(\neg(1 \vee \bar{0}) \vee 1) \wedge (0 \vee \bar{1}) = (\neg(1 \vee 1) \vee 1) \wedge (0 \vee 0) = (\bar{1} \vee 1) \wedge 0 = (0 \vee 1) \wedge 0$$
$$= 1 \wedge 0 = 0$$

Due formule booleane sono equivalenti ($\equiv$) se assumono lo stesso valore per ogni assegnazione di verità delle variabili coinvolte.

Proprietà:

- La negazione è un'involuzione: $\neg(\neg f) \equiv f$
- La disgiunzione e la congiunzione sono commutative, associative e idempotenti:

$$f \vee g \equiv g \vee f, \qquad f \wedge g \equiv g \wedge f$$

$$f \vee (g \vee h) \equiv (f \vee g) \vee h, \qquad f \wedge (g \wedge h) \equiv (f \wedge g) \wedge h$$

$$f \vee f \equiv f, \qquad f \wedge f \equiv f$$

- La disgiunzione e la congiunzione insieme soddisfano le leggi distributive:

$$f \vee (g \wedge h) \equiv (f \vee g) \wedge (f \vee h), \qquad f \wedge (g \vee h) \equiv (f \wedge g) \vee (f \wedge h)$$

- e interagiscono con la negazione con le leggi di de Morgan:

$$\neg(f \vee g) \equiv (\neg f) \wedge (\neg g), \qquad \neg(f \wedge g) \equiv (\neg f) \vee (\neg g)$$

- In fine, ci sono le leggi tautologiche:

$$f \vee \neg f = 1, \qquad f \wedge \neg f = 0.$$

Le formule booleane possono essere trasformata in formule equivalenti applicando queste leggi.

Soddisfacibilità delle formule booleane

Una formula booleana è soddisfacente se esiste un'assegnazione di verità delle sue variabili per cui il valore della formula è 1.

In questo caso diciamo che la formula è soddisfatta dall'assegnazione.

Esempio: La formula $(\neg(x_1 \vee \overline{x_2}) \vee x_3) \wedge (x_2 \vee \overline{x_3})$ è soddisfacente dall'assegnazione di verità: x_1=0, $x_2 = 1$, $x_3 =$0

$$(\neg(0 \vee \overline{1}) \vee 0) \wedge (1 \vee \overline{0}) = (\neg(0 \vee 0) \vee 0) \wedge (1 \vee 1) = (\overline{0} \vee 0) \wedge 1 = (1 \vee 0) \wedge 1$$
$$= 1 \wedge 1 = 1$$

Osservazione: Non tutte le formule booleane sono soddisfacenti

Esempio: $x \wedge \overline{x}$

- $1 \wedge \overline{1} = 1 \wedge 0 = 0$
- $0 \wedge \overline{0} = 0 \wedge 1 = 0$

Questo pone il problema seguente:

Problema Soddisfacibilità booleana (Sat):
Dato: una formula booleana f,
Decidi: f è soddisfacente?

Osservazione: Sat appartiene a $\mathcal{NP}$

Dati opportuni valori delle variabili, si può verificare in tempo polinomiale che il valore della formula è effettivamente 1, essi costituiscono quindi un breve certificato

Cook e Levin hanno dimostrato, indipendentemente, che Sat è un esempio di un problema $\mathcal{NP} - completo:$

Teorema di Cook-Levin: Il problema Sat è $\mathcal{NP} - completo$

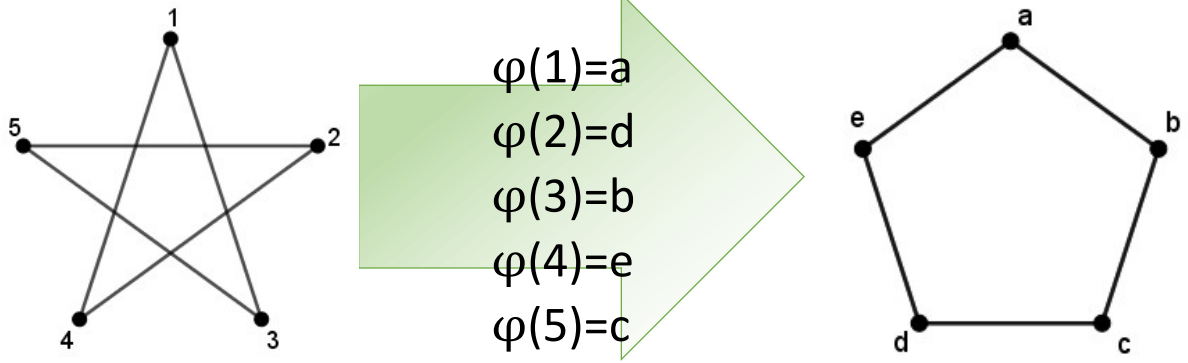
La dimostrazione di questo teorema utilizza la nozione di macchina di Turing e va oltre lo scopo di questo seminario.
Applicando questo teorema, Karp (1972) ha mostrato che molti problemi combinatori sono $\mathcal{NP} - completi.$

Notiamo che, nonostante quasi tutti i problemi decisionali che incontriamo appartengano alla classe $\mathcal{P}$ o $\mathcal{NPC}$, esiste una notevole eccezione ed è il problema dell'isomorfismo:

Problema isomorfismo del grafo:
Dati: due grafi G e H;
Decidi: G e H sono isomorfi?

I grafi isomorfi sono grafi il cui disegno può coincidere, a meno della numerazione dei vertici

Esempio:



G G'=φ(G)

La stato di complessità di questo problema rimane non conosciuto. Sebbene il problema appartenga chiaramente a $\mathcal{NP}$ (basta verificare l'isomorfismo dato), non è noto se appartenga anche a $\mathcal{P}$, a co- $\mathcal{NP}$ o a $\mathcal{NPC}$.
Sono stati trovati algoritmi polinomiali di test dell'isomorfismo per alcune classi di grafi, inclusi i grafi planari e i grafi di grado limitato, ma questi non sono validi per tutti i grafi.

Bibliografia:

- Algorithmic Graph Theory by Alan M. Gibbons
- Graph Theory by Adrian Bondy, U.S.R Murty
- Introduzione agli algoritmi e strutture dati di Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, McGraw-Hill
- M- Liverani, corso di IN440, Università Roma Tre, a.a. 2020/2021

(http://www.mat.uniroma3.it/users/liverani/doc/IN440_-_03_Ottimizzazione_Combinatoria_-_Grafi.pdf)

(http://www.mat.uniroma3.it/users/liverani/doc/IN440_-_01_Ottimizzazione_Combinatoria_-_Complessita.pdf)

(http://www.mat.uniroma3.it/users/liverani/doc/IN440_-_04_Ottimizzazione_Combinatoria_-_Algoritmi_elementari.pdf)