Aniello Murano

NP-Completezza (prima parte)

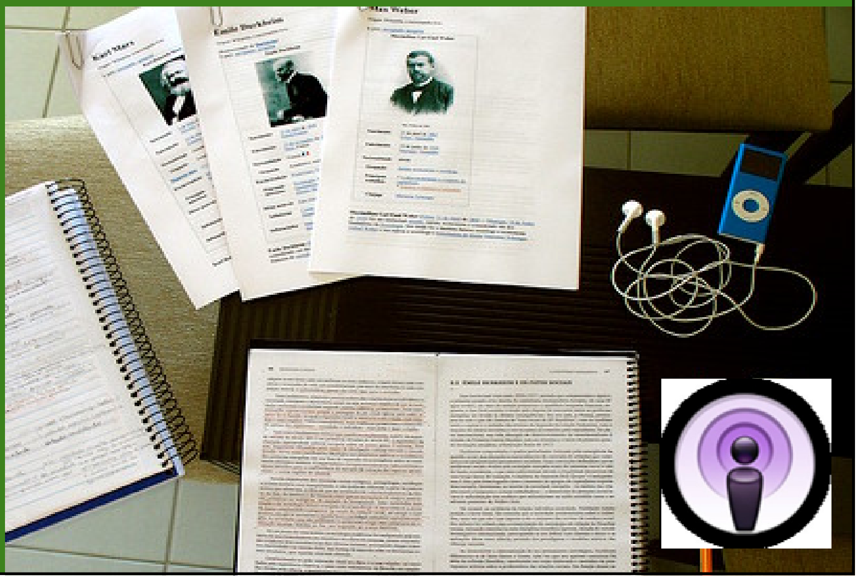
Lezione n.14
Parole chiave: Np-completezza

Corso di Laurea: Informatica

Codice:

Email Docente: murano@ na.infn.it

A.A. 2008-2009



Introduzione

- I problemi **NP-completi** formano un'importante sottoclasse di **NP**.
- Il concetto della **NP-completezza** è stato studiato nei primi anni del 1970 da Stephen Cook e Leonid Levin.
- Se esistesse un algoritmo in tempo polinomiale per tutti i problemi **NP-completi**, allora tutti i problemi **NP** sarebbero risolvibili in tempo polinomiale.
- Per dimostrare che **P** = **NP** è sufficiente prendere un particolare problema **NP-completo** A e dimostrare che **A**$\in$**P**.
- Per dimostrare che **P** $\neq$ **NP** è sufficiente prendere un particolare problema **NP-completo** A e dimostrare che **A**$\notin$**P**.
- Sul lato pratico, sapere che un dato problema A è **NP-completo** può evitare di perdere tempo alla ricerca di un (forse inesistente e comunque "difficile" da trovare) algoritmo polinomiale per A.

Un problema in NP: le formule Booleane

- **Variabili booleane**: x,y,... tale che possono assumere uno dei due seguenti valori 0 (false), 1(true).
- **Operatori booleani:** $\neg$ (NOT), $\wedge$ (AND), $\vee$ (OR).
- **Formule booleane:** sono costruiti con le variabili e le operazioni nel modo standard. Una volta che un assegnamento di verità per le variabili è dato, il valore di una formula composta è calcolato come segue:

$0 \wedge 0 = 0$	$0 \vee 0 = 0$
$0 \wedge 1 = 0$	$0 \vee 1 = 1$
$1 \wedge 0 = 0$	$1 \vee 0 = 1$
$1 \wedge 1 = 1$	$1 \vee 1 = 1$

Problema Sat

- Si dice che una formula booleana è soddisfacibile se e solo se vi è una assegnamento di 0 e 1 per le sue variabili che rende la formula composta vera (valutata 1).
  - Le seguenti formule sono soddisfacibili?

$$\underline{x}\wedge(x\vee y)$$
$$\underline{x}\wedge(x\wedge y)$$

- Il problema Sat consiste nel trovare un assegnamento tale che la formula ϕ risulti vera:
  - **SAT = {<ϕ> |** ϕ è una formula booleana soddisfacibile**}**

SAT $\in$ P?
SAT $\in$ NP?

Theorem 7.27 (Cook-Levin theorem) **SAT$\in$P** sse **P=NP**.

Riducibilità in tempo polinomiale

- **Definizione**: Una funzione calcolabile in tempo polinomiale è una funzione calcolabile in tempo polinomiale da una Macchina di Turing deterministica.

- **Definizione di Mapping riducibilità polinomiale:** Siano A e B due linguaggi con alfabeto Σ. Diremo che A è Mapping riducibile in tempo polinomiale a B, o semplicemente **riducibile in tempo polinomiale** a B, e scriveremo $A \leq_P B$, se esiste una funzione calcolabile in tempo polinomiale $f: \Sigma^* \to \Sigma^*$ t.c. per ogni stringa $w \in \Sigma^*$,

$$w \in A \quad \text{sse} \quad f(w) \in B.$$

Tale funzione f è chiamata **riduzione in tempo polinomiale** da A a B.

Una condizione sufficiente per P

- **Teorema:** Se $A \leq_P B$ e $B \in P$, allora $A \in P$.

- **Dimostrazione:**
  - Assumiamo che esiste una MdT M in grado di decidere B in tempo polinomiale. Sia f una
    riduzione in tempo polinomiale da A a B.
  - Il seguente è unalgoritmo polinomiale per decidere A:
  - N = "Con w in input:
    1. Calcola f(w).
    2. Simula M su ingresso f(w) e accetta se e solo se M accetta"
  - Chiaramente N lavora in tempo polinomiale perché è la composizione di due azioni, entrambe svolte in tempo polinomiale.

Problema 3SAT

- Un ***letterale*** è una variabile Booleana **x** o la sua negata $\underline{x}$.
- Una *clausola* è un insieme di letterali connesso con operatori Booleani. Per semplicità, ci limiteremo al simbolo OR ($\vee$). Per esempio $(x \vee \underline{y} \vee \underline{z} \vee t)$ **è una clausola**.
- Una formula Booleana è in forma normale congiuntiva (***conjunctive normal form)***, chiamata anche ***cnf-formula***, se comprende numerose clausole connesse dal simbolo AND ($\wedge$), per esempio:

  $$(x \vee \underline{y} \vee \underline{z} \vee t) \wedge (\underline{x} \vee z) \wedge (x \vee y \vee \underline{t})$$

  E' una formula in CNF.
- Una cnf-formula è detta ***3cnf-formula*** se tutte le sue clausole hanno 3 letterali, per esempio

  $$(x \vee \underline{y} \vee \underline{z}) \wedge (\underline{x} \vee z \vee t) \wedge (x \vee y \vee \underline{t}) \wedge (z \vee y \vee \underline{t})$$

  è' una 3cnf-formula.

- **Definizione problema 3Sat:**

  $$3SAT = \{\langle\phi\rangle \mid \phi \text{ è una 3cnf-formula soddisfacibile}\}$$

Riduzione di 3SAT a CLIQUE 1

- **Teorema:** 3SAT è riducibile in tempo polinomiale a CLIQUE.
- **Dimostrazione:** Sia ϕ una 3CNF-formula con k clausole, ovvero

  $$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \ldots \wedge (a_k \vee b_k \vee c_k)$$

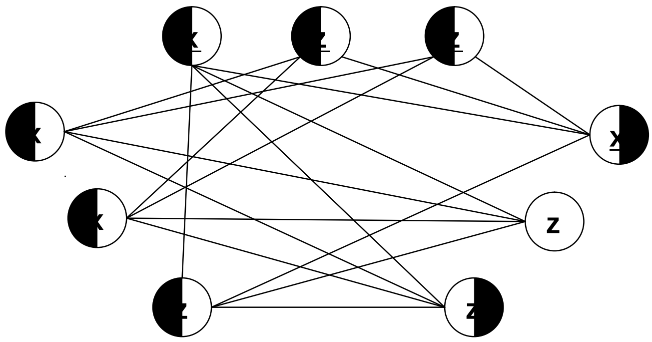
  - La nostra riduzione f consiste nel generare la stringa <G,k> dove G è un grafo non orientato definito come segue:
  - I nodi di G sono organizzati in k gruppi, $t_1, \ldots, t_k$, ognuno dei quali di tre nodi e chiamati triple.
  - Ciascuna tripla corrisponde ad una clausola di ϕ,
  - Ciascun nodo della tripla corrisponde ad un letterale della clausola ad esso associata.
  - Etichettiamo ciascun nodo di G con il letterale ad esso associato.

Riduzione di 3SAT a CLIQUE 2

- Per esempio consideriamo la seguente formula

$$\phi = (x \vee x \vee z) \wedge (\underline{x} \vee \underline{z} \vee \underline{z}) \wedge (\underline{x} \vee z \vee z)$$

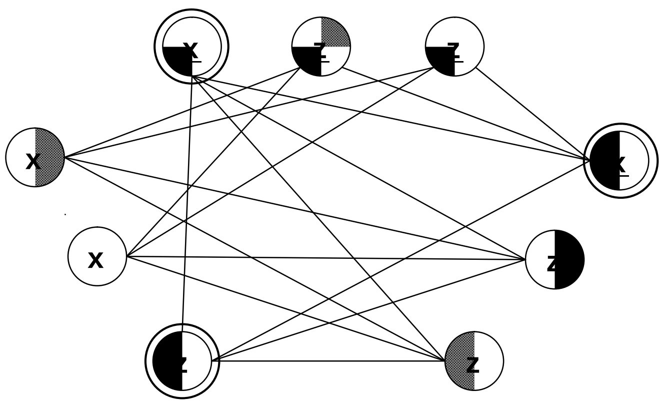
  G ha i nodi come indicato in figura
- In G tutti i nodi sono connessi tra loro, tranne due tipologie di coppie:
  - (1) non è presente alcun arco tra due nodi nella stessa tripla,
  - (2) non è presente alcun arco tra nodi con etichette contraddittorie, come per **x** e **$\underline{x}$**.



Si può osservare che trasformare ϕ in un grafo G richiede tempo polinomiale. [dettagli lasciati per esercizio agli studenti]

Riduzione di 3SAT a CLIQUE 3

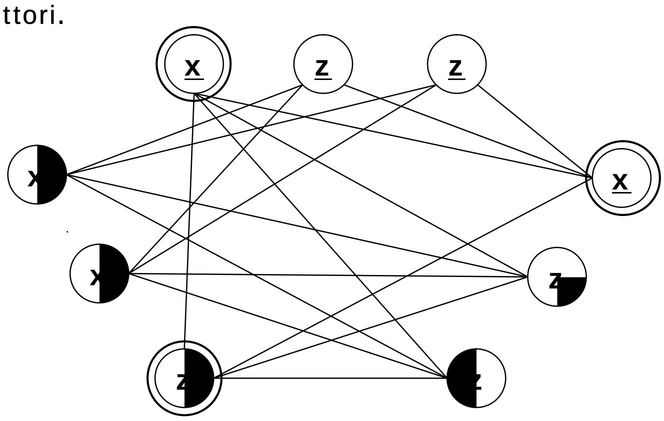
Proviamo adesso che se ϕ ∈ **3SAT**, allora **<G,k>** ∈ **CLIQUE**, e viceversa, se **<G,k>** ∈ **CLIQUE**, allora ϕ ∈ **3SAT**. Iniziamo dalla seconda parte



$$\phi = (x \vee x \vee z) \wedge (\underline{x} \vee \underline{z} \vee \underline{z}) \wedge (\underline{x} \vee z \vee z)$$

Riduzione di 3SAT a CLIQUE 4

- Supponiamo che ϕ abbia un assegnamento soddisfacibile. Allora almeno un letterale deve essere vero in ogni clausola.
  Selezionare uno di questi letterale in ogni clausola, e selezionare i nodi corrispondenti nel grafo.
  Quei nodi formano un k-clique! Infatti, essi sono esattamente k; ogni coppia è connessa da un arco visto che sono in differenti triple e i rispettivi letterali non contraddittori.



Definizione di NP-COMPLETEZZA

- **Definizione Np-complete:** Un linguaggio **B** è ***NP-complete*** se soddisfa le seguenti due condizioni:
  1. **B** è in **NP**, e
  2. Ogni linguaggio in **NP** è riducibile in tempo polinomiale a **B**.

- **Teorema:** Se un linguaggio **B** è in NP-complete e **B**$\in$**P**, allora **P=NP**.

Definizione di NP-COMPLETEZZA

- **Teorema:** Se **$C \in NP$**, **B** è NP-complete e **$B \leq_P C$**, allora **C** è NP-complete.
- **Dimostrazione:**
  - Sappiamo che $C \in NP$, quindi abbiamo solo bisogno di dimostrare che **$A \leq_P C$** per ogni $A \in NP$.
    Siccome B è NP-completo, per definizione abbiamo che **$A \leq_P B$**,
  - Supponiamo che f è una funzione di riduzione da A a B.
  - Supponiamo che $B \leq_P C$ e che g è la funzione di riduzione polinomiale da B a C.
  - Costruiamo ora h come la composizione di f e g, t.c., $h(w) = g(f(w))$.
  - h è la composizione di due funzioni polinomiali, dunque è anch'essa polinomiale, ed è una riduzione da A a C. Quindi **$A \leq_P C$**.