Aniello Murano

Esercitazione su problemi NP-Completi

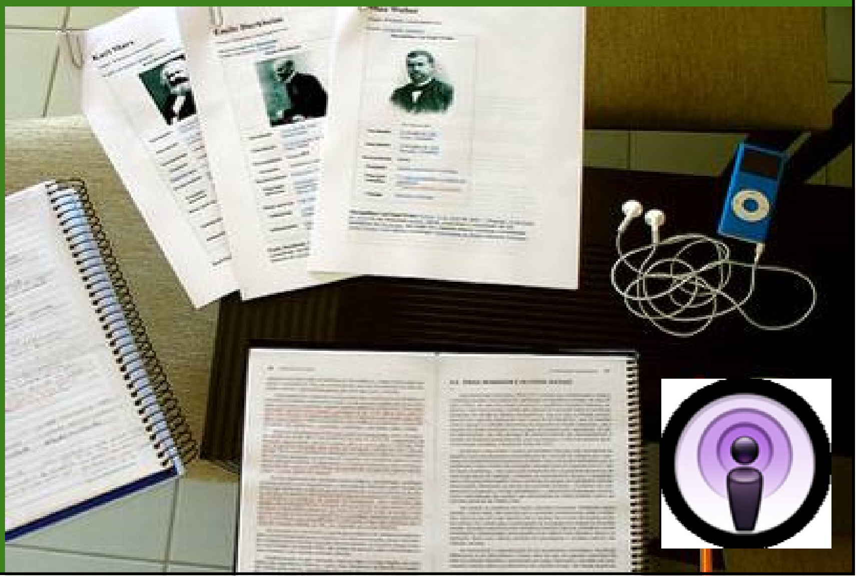
Lezione n.17
Parole chiave:
Np-complete: Esercitazione

Corso di Laurea:
Informatica

Codice:

Email Docente:
murano@ na.infn.it

A.A. 2008-2009



DOUBLE-SAT

- DOUBLE-SAT = { Φ | Φ è una formula Booleana con due assegnamenti di soddisfacibilità}.
- Si mostri che DOUBLE-SAT è NP-Complete. (Hint: Si usi una riduzione da SAT.)

1. DOUBLE-SAT ∊ NP. Basta indovinare non deterministicamente due differenti assegnamenti di verità per le variabili e verificare che in entrambi i casi la formula è vera
2. Per l'hardness usiamo una riduzione da SAT a DOUBLE-SAT:
   - Data una formula Φ, definiamo una nuova formula Φ′ ottenuta da Φ aggiungendo in congiunzione la clausola (x v $\underline{x}$), dove x è una variabile non appartenente a Φ.
   - Chiaramente la riduzione è polinomiale.
   - Resta da dimostrare che Φ ∊ SAT se e solo se Φ′ ∊ DOUBLE-SAT
   - Se Φ non è soddisfacibile, anche Φ′ è insoddisfacibile. Dunque Φ not in SAT → Φ′ not in DOUBLE-SAT
   - Se Φ è soddisfacibile, allora possiamo usare gli stessi assegnamenti di verità per dire che Φ′ è soddisfacibile. Inoltre, sia per x = 0 che per x = 1 la formula resta soddisfacibile.
   - Dunque Φ ∊ SAT → Φ′ ∊ DOUBLE-SAT.

SET-PARTITION (1)

- Il problema **SET-PARTITION** prende in input un insieme S di numeri e decide se l'insieme può essere partizionato in due insiemi A e il suo complemento

$$\sum_{x \in A} x = \sum_{x \in \overline{A}} x.$$

- Si mostri che SET-PARTITION è NP-Complete.
- Per l'appartenenza a NP mostriamo che esiste una macchina di Turing nondeterministica che in tempo polinomiale decide il problema:
  - Basta fare un "guess" su due partizioni e verificare che hanno la stessa somma
- Per l'hardness usiamo una riduzione polinomiale dal problema **SUBSET-SUM**, in modo tale che il problema **SET-PARTITION** ammette soluzione se e solo se **SUBSET-SUM** ammette soluzione

SET-PARTITION (2)

- Si ricordi che SUBSET-SUM è definito nel modo seguente:
  - Dato un insieme S di interi e un numero target t, occorre verificare che esiste un sottoinsieme di numeri di S la cui somma è uguale a t
- Sia s la somma degli elementi di S.
- Consideriamo due casi distinti.
- Se s = 2t, usiamo come input per il SET-PARTITION l'insieme S stesso. Ovviamente, S appartiene a SET-PARTITION se e solo se (S, t) è in SUBSET-SUM.
- Se s è diverso da 2t, usiamo come input per SET-PARTITION l'insieme S' = S U {2s, s+2t}. Si noti che la somma di tutti gli oggetti in S' è 4s + 2t.
- Si supponga adesso che (S, t) sia in SUBSET-SUM. Allora, esiste un sottoinsieme A di S' la cui somma è t. Sia A' = A U {2s}. Con semplici calcoli si vede che la somma degli elementi in A' e il suo complemento rispetto S' coincide. Dunque, S' è in SET-PARTITION.
- Si supponga adesso che S' sia partizionato in due insiemi: A' e il suo complemento. Per ognuno di questi insiemi, la somma dei suoi elementi è 2s+t. Si noti adesso che gli elementi 2s e s+2t non possono stare nello stesso insieme perchè la loro somma è maggiore di 2s+t.
- Senza perdita di generalità, si assuma che 2s è in A'.
- Sia adesso A uguale ad A' \ {2s}. Allora la somma di tutti gli elementi di A è 2s+t-2s=t e tutti gli elementi di A sono elementi di S. Dunque, (S,t) è in SUBSET-SUM

Il problema del Triangolo in un Grafo (3-CLIQUE)

- Un triangolo in un grafo indiretto è una 3-clique. Si mostri che il problema di verificare che un grafo abbia un triangolo è in P. Formalmente, il problema è così definito:

TRIANGLE = { <G> | G contiene una 3-clique}

- Per mostrare che TRIANGLE è in P è sufficiente mostrare un algoritmo che risolve il problema in tempo polinomiale nella taglia dell'input.
- Dato un grafo di n vertici, iteriamo sulle n^3 combinazioni di 3 vertici in G.
- Per ogni combinazione, controlliamo se tutti gli archi sono presenti.
- Se e solo se esiste una combinazione per cui il controllo sugli archi risponde affermativamente, allora rispondiamo "SI" al problema.
- Questo algoritmo risolve correttamente il problema TRIANGLE con un algoritmo di forza bruta che prende tempo $O(n^3)$. Si noti che verificare l'esistenza dei 3 archi richiede tempo O(1) time.
- Quindi, TRIANGLE è in P.
- **Esercizio per gli studenti:** PENT = { <G> | G contiene una 5-clique} è in P?

Half-Clique è NP-Completo

- Sia HALF-CLIQUE = { <G> | G è un grafo inderetto di n nodi e (G, n/2) è una CLIQUE}. Si mostri che HALF-CLIQUE è NP-completo.
- Per l'appartenenza in NP, usiamo l'algoritmo di verifica per k-Clique, già usato in precedenza, per k qualsiasi.
- Per la NP-hardness, usiamo una riduzione da k-CLIQUE a HALF-CLIQUE in tempo polinomiale.
- Dividiamo la riduzione in due parti:
- Se $k \geq n/2$, allora si procede come segue:
  1. Si costruisce da G un nuovo grafo G′ aggiungendo $2k - n$ vertici, tutti senza archi uscenti (cioè tutti i nodi aggiunti sono disconnessi in G′).
  2. G′ ha $n+(2k-n) = 2k$ vertici ed è in HALF-CLIQUE se e solo se ha una k-Clique.
- Se $k < n/2$, allora si procede come segue:
  1. Si costruisce da G un nuovo grafo G′ aggiungendo ai suoi vertici V, un insieme di n-2k vertici V′. Così facendo, G′ ha $n+n-2k = 2n-2k = 2(n-k)$ vertici.
  2. Si collegano tutte le coppie di vertici in V′ in modo che formino una (V′, n-2k)-Clique.
  3. Si collegano poi tutti i vertici in V con quelli in V′.
  4. Se G ha una k-clique, allora G′ deve avere una Clique di taglia $k+(n-2k)=n-k$, perché tutti i k vertici in G formano una clique e sono connessi a $n-2k$ veritici in V′ che formano una clique e tutti questi sono, a loro volta, collegati tra loro.
  5. Se G′ ha una Clique ti taglia $n - k$, allora G′ appartiene ad HALF-CLIQUE.

  Si noti che se G non ha una k-clique, allora è impossibile avere una (n-k)-Clique in G′.