Esercizio 1 del 31 Maggio 2019

Dimostrare che un problema X è NP-hard richiede diversi passaggi:

- Scegli un problema Y che sai essere NP-hard (perché l'hai visto a lezione).
- Descrivi un algoritmo per risolvere Y usando un algoritmo per X come subroutine. Tipicamente questo algoritmo ha la seguente forma: data un'istanza di Y, trasformala in un'istanza di X, quindi chiama l'algoritmo magico black-box per X.
- Dimostra che l'algoritmo è corretto. Ciò richiede sempre due passaggi separati, che di solito hanno la seguente forma:
  - Dimostra che il tuo algoritmo trasforma istanze "buone" di Y in istanze "buone" di X.
  - Dimostra che il tuo algoritmo trasforma istanze "cattive" di Y in istanze "cattive" di X. Equivalentemente: Dimostra che se la tua trasformazione produce un'istanza "buona" di X, allora era partita da un'istanza "buona" di Y.
- Mostra che il tuo algoritmo per Y funziona in tempo polinomiale, a meno della chiamata (o delle chiamate) all'algoritmo magico black-box per X. (Questo di solito è banale.)

Un circuito Hamiltoniano in un grafo G è un ciclo che attraversa ogni vertice di G esattamente una volta. Stabilire se un grafo contiene un circuito Hamiltoniano è un problema NP-completo.

Un *circuito quasi Hamiltoniano* in un grafo G è un ciclo che attraversa esattamente una volta tutti i vertici del grafo *tranne uno*. Il *problema del circuito quasi Hamiltoniano* è il problema di stabilire se un grafo contiene un circuito quasi Hamiltoniano.

1. Dimostrare che il problema del circuito quasi Hamiltoniano è in NP fornendo un certificato per il *Sì* che si può verificare in tempo polinomiale.
2. Mostrare come si può risolvere il problema del circuito Hamiltoniano usando il problema del circuito quasi Hamiltoniano come sottoprocedura.
3. Il problema del circuito quasi Hamiltoniano è un problema NP-completo?
   - ☐ No, è un problema in P
   - ☐ No, è un problema NP ma non NP-completo
   - ☐ Si

Esercizio 2 del 31 Maggio 2019

Dimostrare che un problema X è NP-hard richiede diversi passaggi:

- Scegli un problema Y che sai essere NP-hard (perché l'hai visto a lezione).
- Descrivi un algoritmo per risolvere Y usando un algoritmo per X come subroutine. Tipicamente questo algoritmo ha la seguente forma: data un'istanza di Y, trasformala in un'istanza di X, quindi chiama l'algoritmo magico black-box per X.
- Dimostra che l'algoritmo è corretto. Ciò richiede sempre due passaggi separati, che di solito hanno la seguente forma:
  - Dimostra che il tuo algoritmo trasforma istanze "buone" di Y in istanze "buone" di X.
  - Dimostra che il tuo algoritmo trasforma istanze "cattive" di Y in istanze "cattive" di X. Equivalentemente: Dimostra che se la tua trasformazione produce un'istanza "buona" di X, allora era partita da un'istanza "buona" di Y.
- Mostra che il tuo algoritmo per Y funziona in tempo polinomiale, a meno della chiamata (o delle chiamate) all'algoritmo magico black-box per X. (Questo di solito è banale.)

"Colorare" i vertici di un grafo significa assegnare etichette, tradizionalmente chiamate "colori", ai vertici del grafo in modo tale che nessuna coppia di vertici adiacenti condivida lo stesso colore. Il problema kCOLOR è il problema di trovare una colorazione di un grafo non orientato usando k colori diversi. Sappiamo che il problema 3COLOR è NP-completo.

1. Dimostrare che il problema 4COLOR (colorare un grafo con 4 colori) è in NP fornendo un certificato per il *Sì* che si può verificare in tempo polinomiale.
2. Mostrare come si può risolvere il problema 3COLOR (colorare un grafo con 3 colori) usando 4COLOR come sottoprocedura.
3. Per quali valori di k il problema kCOLOR è NP-completo?
   - ☐ Per nessun valore: kCOLOR è un problema in P
   - ☐ Per tutti i $k \geq 3$
   - ☐ Per tutti i valori di k

Esercizio 3 del 31 Maggio 2019

Dimostrare che un problema X è NP-hard richiede diversi passaggi:

- Scegli un problema Y che sai essere NP-hard (perché l'hai visto a lezione).
- Descrivi un algoritmo per risolvere Y usando un algoritmo per X come subroutine. Tipicamente questo algoritmo ha la seguente forma: data un'istanza di Y, trasformala in un'istanza di X, quindi chiama l'algoritmo magico black-box per X.
- Dimostra che l'algoritmo è corretto. Ciò richiede sempre due passaggi separati, che di solito hanno la seguente forma:
  - Dimostra che il tuo algoritmo trasforma istanze "buone" di Y in istanze "buone" di X.
  - Dimostra che il tuo algoritmo trasforma istanze "cattive" di Y in istanze "cattive" di X. Equivalentemente: Dimostra che se la tua trasformazione produce un'istanza "buona" di X, allora era partita da un'istanza "buona" di Y.
- Mostra che il tuo algoritmo per Y funziona in tempo polinomiale, a meno della chiamata (o delle chiamate) all'algoritmo magico black-box per X. (Questo di solito è banale.)

Il problema SetPartitioning chiede di stabilire se un insieme di numeri interi S può essere suddiviso in due sottoinsiemi disgiunti S_1 e S_2 tali che la somma dei numeri in S_1 è uguale alla somma dei numeri in S_2. Sappiamo che questo problema è NP-completo.

Considerate il seguente problema, che chiameremo SubsetSum: dato un insieme di numeri interi S ed un valore obiettivo t, stabilire se esiste un sottoinsieme $S' \subseteq S$ tale che la somma dei numeri in S' è uguale a t.

1. Dimostrare che il problema SubsetSum è in NP fornendo un certificato per il *Sì* che si può verificare in tempo polinomiale.
2. Mostrare come si può risolvere il problema SetPartitioning usando il problema SubsetSum come sottoprocedura.
3. Il problema SubsetSum è un problema NP-completo?
   - ☐ No, è un problema in P
   - ☐ No, è un problema NP ma non NP-completo
   - ☐ Si