Testi di esame precedenti a.a. e soluzioni

1 Appello del 25 gennaio 2010

Problema 1. Dimostrare che, per ogni costante intera positiva k, 2^{n^k} è una funzione limite.
Suggerimento: usare come subroutine la macchina di Turing che dimostra che n^k è una fuzione limite.

Problema 2. Una formula booleana è in *forma disgiuntiva normale* (DNF) se è nella forma

$$f(x_1,\ldots,x_n) = d_1 \vee d_2 \vee \ldots \vee d_m$$

dove, per $j = 1,\ldots,m$, $d_j = l_{j_1} \wedge l_{j_2} \wedge \ldots \wedge l_{j_h}$ e ciascun letterale l_{j_i} è una variabile in $\{x_1,\ldots,x_n\} \cup \{\neg x_1,\ldots,\neg x_n\}$.

Il problema decisionale DNF-SAT è definito nella maniera seguente: dati un insieme $X = \{x_1,\ldots,x_n\}$ ed una formula booleana f sull'insieme X in forma disgiuntiva normale, decidere se esiste una assegnazione di verità per l'insieme X che soddisfa f.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrare la sua appartenenza alla classe P presentando un algoritmo polinomiale che lo risolve.

Problema 3. Rispondere alla seguente domanda, dimostrando la propria affermazione: esiste un algoritmo polinomiale che trasforma una generica funzione booleana in forma disgiuntiva normale?

Problema 4. Si consideri il seguente problema decisionale: dato un grafo non orientato $G = (V,E)$ (in cui V è l'insieme dei nodi ed E l'insieme degli archi) ed un intero positivo k, decidere se l'insieme V può essere partizionato in al più k insiemi indipendenti.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne la NP-completezza.

1.1 Soluzione

Problema 1. Mostriamo di seguito lo pseudo-codice corrispondente ad una macchina di Turing che calcola2^{n^k} in cui:

- le variabili n1, n2, n3, n4 corrispondono ad altrettanti nastri semi-infiniti della macchina di Turing le cui celle sono numerate a partire dalla posizione 1;
- la variabile i2 corrisponde alla posizione della testina sul nastro n2;
- l'operatore + applicato a variabili di tipo nastro denota la concatenazione dei rispettivi contenuti;
- l'operatore + applicato a variabili di tipo testina denota lo spostamento a destra della stessa;
- l'istruzione calcolaPotenzaK-esima(n1, n2) denota l'utilizzo di un sottoprogramma che scrive sul nastro n2 il valore corrispondente al contenuto del nastro n1 elevato alla potenza k-esima.

```
n1 ← n;
calcolaPotenzaK-esima(n1, n2);
n3 ← 2;
i2 ← 2;
while (i2 ≤ n2) {
      n4 ← n3;
      n3 ← n3 + n4;
      i2 ← i2 + 1;
}
output: n3.
```

Per dimostrare che 2^{n^k} è una funzione limite occorre ancora mostrare che la macchina di Turing appena descritta opera in tempo $O(2^{n^k})$. A questo scopo, osserviamo che l'invocazione della routine calcolaPotenzaK-esima(n1, n2) richiede tempo $O(n^k)$ (in quanto n^k è una funzione limite. Rimane da analizzare il costo del ciclo **while**. All'inizio della i-esima iterazione ($i \geq 2$) il nastro n3 contiene il valore 2^{i-1} che viene immediatamente copiato sul nastro n4 e poi concatenato al contenuto del nastro n3 stesso (calcolando così il valore 2^i). Pertanto, l'iterazione i-esima richiede tempo $2^{i-1}+2^i$. Poiché il valore i è compreso fra 2 e n^k, il costo del ciclo **while** è

$$\sum_{i=2}^{n^k}\left[2^{i-1}+2^i\right] = 2^{n^k}-1-1+2^{n^k+1}-1-1-2 < 2^{n^k}+2^{n^k+1} = 2^{n^k}+2\cdot 2^{n^k} = 3\cdot 2^{n^k}.$$

Problema 2. Formalizziamo il problema in questione come segue:

- $\mathrm{I}_{DNF-SAT} = \{f(x_1,\ldots,x_n) = d_1 \vee d_2 \vee \ldots \vee d_m : d_j = l_{j_1} \wedge l_{j_2} \wedge \ldots \wedge l_{j_h} \forall j = 1,\ldots,m$ e ciascun letterale l_{j_i} è una variabile in $\{x_1,\ldots,x_n\} \cup \{\neg x_1,\ldots,\neg x_n\}\}$.
- $\mathrm{S}_{DNF-SAT}(f(x_1,\ldots,x_n)) = \{a : \{x_1,\ldots,x_n\} \to \{\textbf{vero}, \textbf{falso}\}^n$.
- $\pi_{DNF-SAT}(f(x_1,\ldots,x_n),a) = f(a(x_1,\ldots,x_n))$.

Osserviamo ora che una funzione booleana in forma disgiuntiva normale è soddisfacibile se e soltanto se *almeno una* delle d_j che la compongono è soddisfacibile. Dunque, per decidere se $f = d_1 \vee d_2 \vee \ldots \vee d_m$ è soddisfacibile è sufficiente verificare se esiste una d_j soddisfacibile:

A:DNFSAT
input:$f(x_1,\ldots,x_n) = d_1 \vee d_2 \vee \ldots \vee d_m$, ciascun $d_j = l_{j_1} \wedge l_{j_2} \wedge \ldots \wedge l_{j_h}$
e ciascun $l_{j_i} \in \{x_1,\ldots,x_n\} \cup \{\neg x_1,\ldots,\neg x_n\}$
output: accetta o rigetta;
    trovata $\leftarrow$ **falso**; // non e' stata trovata una d_j soddisfacibile
    j $\leftarrow 1$;
    while (j $\leq m \wedge$ **not** trovata) {
        // verifica se $d_j = l_{j_1} \wedge l_{j_2} \wedge \ldots \wedge l_{j_h}$ e' soddisfacibile
        // e se lo e' assegna trovata $\leftarrow$ **vero**
        j $\leftarrow$ j $+1$;
    }
    if (trovata) **output**: accetta;
    else output: rigetta.

Resta da chiarire come decidere se una d_j è soddisfacibile (le linee commentate del codice sopra). Poiché ciascuna d_j è una congiunzione di letterali, d_j è soddisfacibile se e soltanto se esiste una assegnazione di verità rispetto alla quale *tutti* i suoi letterali hanno valore **vero**: osserviamo ora che l'unico caso in cui è impossibile assegnare valore vero a *tutti* i letterali di un insieme è quando l'insieme contiene una variabile e la sua negazione. Pertanto, per decidere se d_j è soddisfacibile è sufficiente semplicemente verificare che essa non contenga una coppia di letterali che siano uno la negazione dell'altro (ad esempio, x_1 e $\neg x_1$). In conclusione, ecco di seguito l'algoritmo **A:DNFSAT** completo:

A:DNFSAT
input:$f(x_1,\ldots,x_n) = d_1 \vee d_2 \vee \ldots \vee d_m$, ciascun $d_j = l_{j_1} \wedge l_{j_2} \wedge \ldots \wedge l_{j_h}$
e ciascun $l_{j_i} \in \{x_1,\ldots,x_n\} \cup \{\neg x_1,\ldots,\neg x_n\}$
output: accetta o rigetta;
    trovata $\leftarrow$ **falso**; // non e' stata trovata una d_j soddisfacibile
    j $\leftarrow 1$;
    while (j $\leq m \wedge$ **not** trovata) {
        trovata $\leftarrow$ **vero**; // supponiamo d_j soddisfacibile ...
        for (p $\leftarrow 1$; p $< h$; p $\leftarrow$ p $+1$) // ... e lo verifichiamo
            for (q $\leftarrow$ p $+1$; q $\leq h$; q $\leftarrow$ q $+1$)
                if ($l_{j_p} = \neg l_{j_q}$) trovata $\leftarrow$ **falso** ; // contaddizione!
        j $\leftarrow$ j $+1$;
    }
    if (trovata) **output**: accetta;
    else output: rigetta.

Per quanto riguarda la complessità, iosserviamo che, poiché il massimo numero di letterali in una d_j è $2n$, l'algoritmo **A:DNFSAT** ha complessità O($n^2 \cdot m$).

Problema 3. Supponiamo per assurdo che esista un algoritmo polinomiale **A:TODNF** che, data una funzione booleana f, calcola una funzione booleana g in forma disgiuntiva normale tale che f è soddisfacibile se e soltanto se g è soddisfacibile. Allora, possiamo decidere se una funzione booleana (ad esempio, in forma congiuntiva normale) è soddisfacibile mediante il seguente algoritmo:

A:RGH
input: funzione booleana $f(x_1,\ldots,x_n)$;
output: accetta o rigetta;
    $g \leftarrow$ **A:TODNF**(f);
    if (**A:DNFSAT**(g) accetta) **output**: accetta;
    else output: rigetta.

Poiché per ipotesi l'algoritmo **A:TODNF** opera in tempo polinomiale, e poiché anche l'algorimo **A:DNFSAT** opera in tempo polinomiale (come affermato nel **Problema 2**), l'algoritmo **A:RGH** decide il problema SODDISFACIBILITÀ in tempo polinomiale. Ma, poiché SODDISFACIBILITÀ è un problema NP-completo, questo dimostrerebbe P= NP, da cui l'assurdo.

Problema 4. Il problema PARTIZIONE IN INSIEMI INDIPENDENTI (in breve, ISP) è formalizzato nella maniera seguente:

- $\mathrm{I}_{ISP} = \{\langle G = (V,E), k\rangle : G$ è un grafo non orientato e k è un intero positivo $\}$.
- $\mathrm{S}_{ISP}(\langle G = (V,E), k\rangle) = \{\langle V_1, \ldots, V_h\rangle : \forall 1 \leq i, j \leq h[V_i \cap V_j = \emptyset \wedge \cup_{i=1}^{h} V_i = V]\}$.
- $\pi_{ISP}(\langle G = (V,E), k\rangle, \langle V_1, \ldots, V_h\rangle) = \forall 1 \leq i \leq h[u, v \in V_i \Rightarrow (u,v) \notin E] \ \wedge \ h \leq k$.

Osserviamo ora che la definizione di tale problema coincide con quella del problema COLORABILITÀ: dunque, la NP-completezza di ISP segue immediatamente dalla NP-completezza di COLORABILITÀ, essendo i due problemi coincidenti (volendo, è possibile definire una riduzione polinomiale da COLORABILITÀ ad ISP la cui funzione di trasformazione da istanze di COLORABILITÀ ad istanze di ISP è la funzione identità).

2 Appello del 8 febbraio 2010

Problema 1. Utilizzando i risultati noti sulle funzioni limite dimostrare che, per ogni costante intera positiva k e per ogni $(k+1)$-pla $\langle a_0, a_1, \ldots, a_k \rangle$ di costanti intere positive, $f(n) = \sum_{i=0}^{k} a_i n^i$ è una funzione limite.

Problema 2. Si consideri il seguente problema decisionale: dato un grafo non orientato $G = (V, E)$ (in cui V è l'insieme dei nodi ed E l'insieme degli archi), decidere se V contiene un sottoinsieme V' che sia un ricoprimento tramite nodi per G di cardinalità 3.

Formalizzare la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, ed indicare la più piccola classe di complessità che lo contiene dimostrando la propria affermazione.

Problema 3. Si consideri il seguente problema decisionale: dato un grafo non orientato $G = (V, E)$ (in cui V è l'insieme dei nodi ed E l'insieme degli archi) ed un intero positivo k, decidere se l'insieme V può essere partizionato in al più k clique.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne la NP-completezza.

Problema 4. Si consideri la seguente istanza del problema `minimo Commesso Viaggiatore`: $n = 7$ (ossia, $V = \{1, 2, \ldots, 7\}$) e la funzione peso, simmetrica, definita come segue:

- $p(i, 2i) = 1$ e $p(i, 2i+1) = 1$ per $i = 1, 2, 3$;
- $p(4,6) = p(5,6) = p(5,7) = 2$;
- il peso di tutti gli archi rimanenti è 10.

Applicare a tale istanza l'algoritmo di Christofides e calcolare la misura della soluzione prodotta.

Una soluzione ottima per l'istanza in esame è il tour $\langle 1, 2, 4, 6, 5, 7, 3 \rangle$ e la sua misura è OTT$= 10$. Dopo aver calcolato l'errore relativo della soluzione prodotta dall'algoritmo di Christofides, esporre la ragione per la quale il fatto che esso risulti maggiore di 1 non contraddice la 1-approssimabilità dell'algoritmo di Christofides.

2.1 Soluzione

Problema 1. Si osservi che, essendo k e $a_0, \ldots, a_k$ valori costanti, non abbiamo bisogno di memorizzarli su nastro in quanto essi possono essere codificati direttamente negli stati interni della macchina di Turing che deve calcolare la funzione. In questo modo, ad esempio, con l'istruzione "n2 $\leftarrow a_0$" ci riferiamo in breve alla seguente sequenza di a_0 istruzioni della macchina di Turing:

1. se nello stato $q_{a_0,0}$ la testina sul nastro n2 legge un blank allora scrive un 1 e si sposta a destra e la macchina entra nello stato $q_{a_0,1}$;
2. se nello stato $q_{a_0,1}$ la testina sul nastro n2 legge un blank allora scrive un 1 e si sposta a destra e la macchina entra nello stato $q_{a_0,2}$;
3. ...
4. se nello stato q_{a_0,a_0-1} la testina sul nastro n2 legge un blank allora scrive un 1 e si sposta a destra e la macchina entra nello stato q_{a_0,a_0}.

Con questa convenzione, definiamo una macchina di Turing in cui:

- le variabili n1, n2, n3 corrispondono ad altrettanti nastri semi-infiniti della macchina di Turing le cui celle sono numerate a partire dalla posizione 1;
- la variabile i corrisponde all'insieme di stati necessari a calcolare il monomio $a_i n^i$; l'istruzione "i $\leftarrow$ i $+1$" indica il passaggio al calcolo del monomio $a_{i+1}n^{i+1}$ (o, a basso livello, il passaggio dallo stato q_{a_i,a_i} allo stato $q_{a_{i+1},0}$);
- la variabile j numera gli stati $q_{a_i,0}, \ldots q_{a_i,a_i}$; l'istruzione "j $\leftarrow$ j $+1$" indica il passaggio della macchina dallo stato $q_{a_i,j}$ allo stato $q_{a_i,j+1}$;
- l'operatore $+$ applicato a variabili di tipo nastro denota la concatenazione dei rispettivi contenuti;
- l'istruzione calcolaPotenza(n1, n3, i) denota l'utilizzo di un sottoprogramma che scrive sul nastro n3 il valore corrispondente al contenuto del nastro n1 elevato alla potenza i-esima.

Possiamo ora mostrare lo pseudo-codice corrispondente ad una macchina di Turing che calcola $f(n) = \sum_{i=0}^{k} a_i n^i$

```
n1 ← n;
n2 ← a_0;
i ← 1;
while (i ≤ k) {
    calcolaPotenza(n1, n3, i);
    j ← 1 ;
    while (j ≤ a_i) {
        n2 ← n2 + n3;
        j ← j +1;
    }
    i ← i +1;
}
output: n2.
```

Per dimostrare che $f(n) = \sum_{i=0}^{k} a_i n^i$ è una funzione limite, osserviamo che ciascuna invocazione "calcolaPotenza(n1, n3, i)" richiede tempo $p_i n^i$, per una opportunacostante p_i, e che il ciclo **while** interno (la cui unica operazione che

richiede tempo di calcolo è la concatenazione del contenuto del nastro n3 al contenuto del nastro n2) richiede tempo $a_i n^i$. Pertanto, il costo del ciclo **while** esterno richiede tempo

$$\sum_{i=1}^{k}(p_i + a_i)n^i \leq n^k \sum_{i=1}^{k}(p_i + a_i) = cn^k$$

dove il valore c è costante. Poiché le operazioni di inizializzazione che precedono il ciclo **while** esterno richiedono tempo costante o lineare in n, questo dimostra che $f(n)$ è una funzione limite.

Problema 2. Il problema 3-VERTEX COVER (in breve, 3VC) può essere formalizzato nella maniera seguente:

- $I_{3VC} = \{G = (V,E) : G \text{ è un grafo non orientato } \}$.
- $S_{3VC}(G) = \{V' \subseteq V : |V'| = 3\}$.
- $\pi_{3VC}(G,V') = \forall (u,v) \in E : u \in V' \vee v \in V'$.

Osserviamo innanzi tutto che il problema 3VC è un caso particolare del più generale VERTEX COVER (o RICOPRIMENTO TRAMITE NODI). Poiché VERTEX COVER è un problema in NP, il predicato $\pi_{3VC}(G,V')$ è decidibile in tempo polinomiale.

Ossreviamo ora che, poiché siamo alla ricerca di un sottoinsieme di V di 3 nodi, allora anche l'enumerazione delle soluzioni possibili (ossia, dell'insieme S_{3VC}) richiede tempo polinomiale: è infatti sufficiente considerare tutte le triple di tre elementi distinti appartenenti ad un insieme di n elementi, che richiede tempo $O(n^3)$.

Volendo esplicitare l'algoritmo appena descritto avremmo il seguente codice:

A:3VC
input: $G = (V,E)$, con $V = \{v_1, v_2, \ldots, v_n\}$
output: accetta o rigetta.

```
trovato ← falso;
for ( i ← 1; i ≤ n; i ← i +1)
    for ( j ← 1; j ≤ n; j ← j +1)
        for (h ← 1; h ≤ n; h ← h +1)
            trovato ← π_3VC(G, {v_i, v_j, v_h})
if (trovato) output: accetta;
else output: rigetta.
```

L'algoritmo **A:3VC** decide $G \in$ 3VC in tempo in $O(n^3 t_{\pi_{3VC}}(n))$, in cui $t_{\pi_{3VC}}(n)$ è il tempo necessario a verificare il predicato $\pi_{3VC}(G,V')$ che, come osservato in precedenza, è polinomiale in n. Dunque, il problema 3VC è contenuto nella classe P.

Volendo, nell'algoritmo **A:3VC** è possibile esplicitare il calcolo del predicato $\pi_{3VC}(G, \{v_1, v_2, v_3\})$:

```
π_3VC(G = (V,E), {v_1, v_2, v_3}) {
    ricoprimento ← vero;
    for ( (u,v) ∈ E )
        if ( u ∉ {v_1, v_2, v_3} ∧ v ∉ {v_1, v_2, v_3} )
            ricoprimento ← falso;
    return ricoprimento;
}
```

che richiede tempo in $O(|E|)$.

Problema 3. Il problema PARTIZIONE IN CLIQUE (in breve, PIC) è formalizzato nella maniera seguente:

- $I_{PIC} = \{\langle G=(V,E),k\rangle : G$ è un grafo non orientato e k è un intero positivo $\}$.
- $S_{PIC}(\langle G=(V,E),k\rangle) = \{\langle V_1,\ldots,V_h\rangle : \forall 1 \leq i,j \leq h[V_i \cap V_j = \emptyset \wedge \cup_{i=1}^{h} V_i = V] \wedge h \leq k\}$.
- $\pi_{ISP}(\langle G=(V,E),k\rangle,\langle V_1,\ldots,V_h\rangle) = \forall 1 \leq i \leq h[u,v \in V_i \Rightarrow (u,v) \in E] \ \wedge \ h \leq k$.

Il problema è in NP: infatti, una partizione dell'insieme dei nodi in k sottoinsiemi può essere generata non deterministicamente in tempo O(n). Indicato con $V = \{v_1, v_2, \ldots, v_n\}$ l'insieme dei nodi di G, si consideri allo scopo un albero delle computazioni di profondità n e grado di non determinismo k in cui il livello $j = 0,2,\ldots,n-1$ dell'albero corrisponde al nodo v_{j+1} di G e l'i-esimo arco uscente da un nodo del livello j corrisponde all'inserimento di v_{j+1} in V_i.

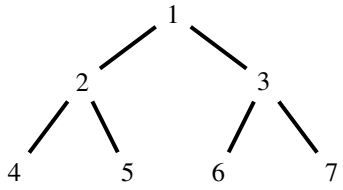
Per quanto riguarda la completezza, essa viene dimostrata mediante una semplice riduzione dal problema COLORABILITÀ. Si consideri un'istanza $\langle G=(V,E),k\rangle$ di COLORABILITÀ e si costruisca il grafo complemento di G $G^c = (V,E^c)$ tale che, per ogni coppia di nodi $u,v \in V$, $(u,v) \in E^c$ se e soltanto se $(u,v) \notin E$. Il grafo G^c è facilmente calcolabile in tempo in O($|G|$).

Consideriamo una istanza $\langle G=(V,E),k\rangle$ di COLORABILITÀ: allora, $\langle G=(V,E),k\rangle \in$ COLORABILITÀ **se e soltanto se** esiste una partizione di V in $h \leq k$ sottoinsiemi $V_1, V_2, \ldots, V_h$ tali che, per ogni $i = 1,2,\ldots,h$ e per ogni coppia di nodi $u,v \in V_i$, allora $(u,v) \notin E$. Ma questo significa che, per ogni $i = 1,2,\ldots,h$ e per ogni coppia di nodi $u,v \in V_i$, allora $(u,v) \in E^c$. E questo è vero **se e soltanto se** $\langle G^c = (V,E^c),k\rangle \in$ PIC.

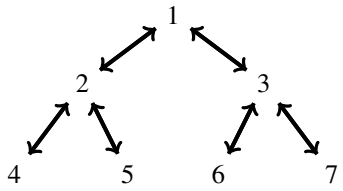
La dimostrazione di NP-completezza è così completata.

Problema 4. L'algoritmo di Christofides opera in quattro fasi:

1. viene calcolato l'albero ricoprente minimo del grafo G con radice nel nodo 1, mostrato nella seguente figura:



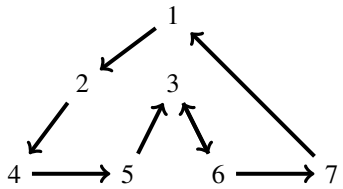
2. ciascun arco non orientato dell'albero ricoprente minimo calcolato al punto precedente viene sostituito dalla corrispondente coppia di archi orientati:



3. viene calcolato un ciclo euleriano sull'albero doppiamente orientato calcolato al punto precedente:

   $\langle 1,2,4,2,5,2,1,3,6,3,7,3,1\rangle$

4. dal ciclo euleriano calcolato al punto precedente viene estratto un ciclo hamiltoniano, sostituendo con archi singoli le porzioni del ciclo che ripassano su nodi già visitati :



Il costo della soluzione appena calcolata applicando l'algoritmo di Christofides è $m_C = 43$ e, quindi, l'errore relativo è

$$\frac{m_c - \mathrm{OTT}}{\mathrm{OTT}} = \frac{43-10}{10} = 3,3 > 1$$

e questo si giustifica osservando che l'algoritmo di Christofides garantisce errore relativo non maggiore di 1 quando viene applicato ad istanze del TSP che soddisfano la disuguaglianza triangolare, mentre il grafo considerato in questo problema non gode di tale proprietà: infatti, ad esempio,

$$p(5,3) = 10 > p(5,6) + p(6,3) = 3.$$

3 Appello del 14 giugno 2010

Problema 1. Sia $D = \{0,1,\ldots,9\}$ l'insieme delle cifre decimali, e sia $\mathscr{P}(D)$ l'insieme dei polinomi a coefficienti in D. Dimostrare che l'insieme $\mathscr{P}(D)$ è numerabile.

Problema 2. Sia $\overline{G}$ un grafo fissato.
Il problema 2-COLORABILEOPPUREISOMORFOA-$\overline{G}$ consiste nel chiedersi se un grafo G è 2-colorabile oppure è isomorfo a $\overline{G}$.

Formalizzare la definizione del suddetto problema mediante la tripla $\langle I,S,\pi\rangle$, ed indicare la più piccola classe di complessità che lo contiene dimostrando la propria affermazione.

Problema 3. Il problema 2-COLORABILEOPPURE3-COLORABILE consiste nel chiedersi se un grafo G è 2-colorabile oppure è 3-colorabile.

Formalizzare la definizione del suddetto problema mediante la tripla $\langle I,S,\pi\rangle$, ed indicare la più piccola classe di complessità che lo contiene dimostrando la propria affermazione.

Problema 4. Il problema $\{1,2\}$-MINIMO COMMESSO VIAGGIATORE è la restrizione del problema MINIMO COMMESSO VIAGGIATORE ad istanze $G = (V,E,w)$ in cui $G = (V,E)$ è un grafo completo non orientato di n nodi e w è una funzione peso tale che $w : E \to \{1,2\}$ (ossia, il peso di ogni arco è 1 oppure 2).

Progettare un algoritmo approssimante per il problema $\{1,2\}$-MINIMO COMMESSO VIAGGIATORE e calcolarne l'errore relativo.

3.1 Soluzione

Problema 1. Sia $p(x) = d_0 + d_1x + d_2x^2 + \ldots + d_nx^n$ un qualunque elemento in $\mathscr{P}(D)$ e consideriamo la seguente funzione che associa un numero naturale a ciascun polinomio in $\mathscr{P}(D)$:

$$f(p(x)) = f(d_0 + d_1x + d_2x^2 + \ldots + d_nx^n) = d_0 + 10d_1 + 10^2d_2 + \ldots + 10^nd_n.$$

Mostriamo ora che $f : \mathscr{P}(D) \to \mathbb{N}$ è una biezione.

- Per ogni numero naturale n esiste un elemento $p_n(x) \in \mathscr{P}(D)$ tale che $f(p_n(x)) = n$: infatti, è sempre possibile esprimere n nella forma $a_0 + 10a_1 + 10^2a_2 + \ldots + 10^ka_k$, per un opportuno $k \geq 0$, dove $a_0, a_1, \ldots, a_k$ sono cifre comprese fra 0 e 9; quindi, il polinomio $p_n(x) = a_0 + a_1x + a_2a^2 + \ldots + a_ka^k$ è tale che $f(p_n(x)) = n$. Questo prova che f è suriettiva.
- Siano $p(x) = a_0 + a_1x + a_2a^2 + \ldots + a_na^n$ e $q(x) = b_0 + b_1x + b_2a^2 + \ldots + b_ka^k$ due elementi di $\mathscr{P}(D)$ tali che $p(x) \neq q(x)$.
  Se $n < k$ allora $f(p(x)) < f(q(x))$, mentre se $n > k$ allora $f(p(x)) > f(q(x))$: in entrambi i casi $f(p(x)) \neq f(q(x))$.
  Resta da analizzare il caso $n = k$: in questo caso, poiché $p(x) \neq q(x)$, deve esistere almeno un indice $1 \leq i \leq n$ tale che $a_i \neq b_i$. Sia i_{max} il massimo indice tale che $a_{i_{max}} \neq b_{i_{max}}$ (ossia, per ogni $j > i_{max}$ accade che $a_j = b_j$). Di nuovo, se $a_{i_{max}} < b_{i_{max}}$ allora $f(p(x)) < f(q(x))$, altrimenti $f(p(x)) > f(q(x))$: in entrambi i casi $f(p(x)) \neq f(q(x))$. Questo prova che f è iniettiva.

Problema 2. Sia $\overline{G} = (\overline{V}, \overline{E})$. Osserviamo che $\overline{G}$ è un grafo costante e, quindi, non è parte dell'input.
Il problema 2-COLORABILEOPPUREISOMORFOA-$\overline{G}$ può essere formalizzato come segue:

- $I = \{G = (V,E) : G \text{ è un grafo }\}$;
- $S = \{< f : V \to \overline{V}, c : V \to \{1,2\}\}$, ossia, una soluzione possibile è un possibile isomorfismo fra G e $\overline{G}$ (f) ed una possibile 2-colorazione dei nodi di G (c);
- $\pi(G, < f, c >) = f$ è un effettivo isomorfismo da G a $\overline{G}$ (ossia, per ogni $u, v \in V$, $(u,v) \in E \Leftrightarrow (f(u), f(v)) \in \overline{E}$) oppure c è una effettiva 2-colorazione dei nodi di G (ossia, per ogni $u, v \in V$, $c(u) = c(v) \Rightarrow (u,v) \notin E$).

Osserviamo ora che, poiché $\overline{G}$ è un grafo costante, esistono solo un numero *finito* di grafi isomorfi a $\overline{G}$: in particolare, detto $\overline{n} = |\overline{V}$, esistono al più $\overline{n}!$ grafi isomorfi a $\overline{G}$ (ottenuti considerando tutte le permutazioni degli elementi di $\overline{V}$).

Allora, i problemi 2-COLORABILEOPPUREISOMORFOA-$\overline{G}$ e 2-COLORABILITÀ differiscono solo per un numero finito di istanze e, quindi, hanno le stessa complessità (chiusura rispetto a variazioni finite), ossia, sono entrambi in P.

Alternativamente, senza ricorrere al Teorema di chiusura rispetto alle variazioni finite, si può dimostrare l'appartenenza a P del problema 2-COLORABILEOPPUREISOMORFOA-$\overline{G}$ in maniera costruttiva, ossia, scrivendo direttamente l'algoritmo polinomiale che lo decide:

Input: grafo $G = (V,E)$ con $V = \{v_1, \ldots, v_n\}$.
if 2COL(G) **then** esito $\leftarrow$ **true**;
else
  if $n \neq |\overline{V}|$ **then** esito $\leftarrow$ **false**;
  else begin
    esito $\leftarrow$ **false**;
    for ogni permutazione π di $\{v_1, \ldots, v_n\}$ **do**
      if $\{(\pi(v_i), \pi(v_j)) : v_i, v_j \in V \wedge (v_i, v_j) \in E\} = \overline{E}$ **then**
        esito $\leftarrow$ **true**;
  end;

if esito = **true then accetta**;
else rigetta.

La parte **if** dell'istruzione **if-else** più esterna invoca la funzione booleana 2COL(G) che testa la 2-colorabilità di G, restituendo il valore **true** in caso affermativo, il valore **false** altrimenti. Come è noto, è possibile implementare tale funzione in modo che richieda tempo polinomiale nella dimensione di G.
La parte **else** della stessa istruzione esegue innanzi tutto un test per verificare se G ha tanti nodi quanti $\overline{G}$: in caso negativo, G non può essere isomorfo a $\overline{G}$ e, avendo già verificato che G non è 2-colorabile, possiamo concludere che G non appartiene al linguaggio. Se invece G e $\overline{G}$ hanno lo stesso numero di nodi (e, dunque, G ha dimensione costante) possiamo procedere alla verifica di isomorfismo, che viene eseguita dal loop **for**: tale loop, lavorando su un numero *costante* di elementi (il numero di nodi di $\overline{G}$) richiede tempo costante.
In conclusione, l'algoritmo proposto richiede tempo polinomiale.

Problema 3. Osserviamo subito che ogni 2-colorazione di un qualunque grafo è *anche* una 3-colorazione per esso che non usa uno dei tre colori consentiti. Pertanto,

- se un grafo G è 2-colorabile oppure 3-colorabile allora è anche 3-colorabile,
- se un grafo G è 3-colorabile allora è anche 2-colorabile oppure 3-colorabile,

ossia, il problema 2-COLORABILEOPPURE3-COLORABILE coincide con il problema 3-COLORABILITÀ ed è NP-completo. Dunque, esso può essere formalizzato come segue:

- $I = \{G = (V,E) : G \text{ è un grafo }\}$;
- $S = \{< c : V \to \{1,2,3\}\}$;
- $\pi(G,c) = c$ è una effettiva 3-colorazione dei nodi di G (ossia, per ogni $u,v \in V$, $c(u) = c(v) \Rightarrow (u,v) \notin E$).

Problema 4. Sono possibili due approcci differenti.

1. Ogni istanza del $G = (V,E,w)$ problema $\{1,2\}$-MINIMO COMMESSO VIAGGIATORE soddisfa la disuguaglianza triangolare. Infatti, siano u,v,z tre nodi di G: allora, $w(u,v) \leq 2$, $w(u,z) \geq 1$ e $w(z,v) \geq 1$, da cui

   $$w(u,v) \leq 2 = 1+1 \leq w(u,z) + w(z,v).$$

   Quindi, l'agoritmo di Christofides è approssimante con errore relativo 1 per il problema $\{1,2\}$-MINIMO COMMESSO VIAGGIATORE.

2. Sia $G = (V,E,w)$ una istanza del problema $\{1,2\}$-MINIMO COMMESSO VIAGGIATORE con $|V| = n$, e sia $T = \langle v_1, v_2, \ldots v_n \rangle$ un qualunque tour in G. Nel caso peggiore, ciascun arco di tale tour ha costo 2: per $i = 1 \ldots n-1$, $w(v_i, v_{i+1}) = 2$ e $w(v_n, v_1) = 2$. Dunque, il costo del tour è $c(T) \leq 2n$.
   D'altra parte, poiché ogni tour contiene n archi ed ogni arco ha peso almeno 1, il tour ottimo non può costare meno di n, ossia, $c^* \geq n$.
   In conclusione, sia $\mathscr{A}$ un algoritmo che calcola un qualunque tour T di G (ad esempio, il tour che visita i nodi in ordine di etichetta crescente): esso è un algoritmo approssimante il cui errore relativo è pari a

   $$\frac{c(T) - c^*}{c^*} \leq \frac{2-1}{1} = 1.$$

4 Appello del 15 settembre 2010

Problema 1. Sia L l'insieme delle stringhe $s = \langle x_1 x_2 \dots x_n \rangle$ di lunghezza pari e tali che:

- $x_i \in \{a,b\}$, per $i = 1,\dots,n/2$;
- $x_i \in \{c,d\}$, per $i = n/2+1,\dots,n$;
- $x_i = a \Leftrightarrow x_{n-i+1} = c$, per $i = 1,\dots,n/2$;
- $x_i = b \Leftrightarrow x_{n-i+1} = d$, per $i = 1,\dots,n/2$.

Esempio. Le stringhe *abacdc* e *aabbddcc* appartengono a L, mentre le stringhe *abadc* e *aabbadddcc* non appartengono a L.

Definire una macchina di Turing deterministica che riconosca L.

Problema 2. Il problema 2-COLORABILEE3-COLORABILE consiste nel chiedersi se un grafo G è 2-colorabile ed è anche 3-colorabile.

Formalizzare la definizione del suddetto problema mediante la tripla $\langle I,S,\pi \rangle$, e dimostrarne l'appartenenza a P oppure la NP-completezza.

Problema 3. Il problema 4-SODDISFACIBILITÀ consiste nel chiedersi se una funzione booleana in forma congiuntiva normale con clausole di esattamente 4 letterali ciascuna è soddisfacibile.

Formalizzare la definizione del suddetto problema mediante la tripla $\langle I,S,\pi \rangle$, e dimostrarne la NP-completezza.

Problema 4. Si consideri il seguente problema NO-CLIQUE: dato un grafo $G = (V,E)$ ed un intero $k > 0$, decidere se ogni sottoinsieme di V di almeno k nodi contiene almeno una coppia di nodi non adiacenti.

Formalizzare la definizione del suddetto problema mediante la tripla $\langle I,S,\pi \rangle$, ed indicare la più piccola classe di complessità che lo contiene dimostrando la propria affermazione.

4.1 Soluzione

Problema 1. Sia $s = \langle x_1 x_2 \dots x_n \rangle \in \{a,b,c,d\}^n$ e $\sigma(s) = \langle y_1 y_2 \dots y_n \rangle \in \{0,1\}^n$ la stringa binaria associata ad s secondo le regole seguenti:

- $y_i = 0 \Leftrightarrow x_i = a \vee x_i = c$, per $1 \leq i \leq n$;
- $y_i = 1 \Leftrightarrow x_i = b \vee x_i = d$, per $1 \leq i \leq n$.

Segue dalle definizioni di L e di $\sigma(s)$ che $s \in L$ se e soltanto se $\sigma(s)$ è una stringa palindroma. Pertanto, la macchina di Turing richiesta è una banale modifica di quella vista a lezione per il linguaggio PALINDROMIA: T è definita sull'alfabeto $\{a,b,c,d,\Box\}$ ($\Box$ è il carattere blank) e sull'insieme degli stati $Q = \{q_0, q_a, q_b, q_c, q_d, q_{ind}, q_{acc}, q_{rig}\}$ in cui q_0 è lo stato iniziale, q_{acc} lo stato finale di accettazione e q_{rig} lo stato finale di rigetto. L'insieme delle quintuple di T è il seguente (per chiarezza di notazione, indichiamo con dx e sx, rispettivamente, lo spostamento a destra e a sinistra della testina)

1. $\langle q_0, a, \Box, q_a, dx \rangle$;
2. $\langle q_a, a, a, q_a, dx \rangle$, $\langle q_a, b, b, q_a, dx \rangle$, $\langle q_a, c, c, q_a, dx \rangle$, $\langle q_a, d, d, q_a, dx \rangle$;
3. $\langle q_a, \Box, \Box, q_c, sx \rangle$;
4. $\langle q_c, c, \Box, q_{ind}, sx \rangle$;
5. $\langle q_c, a, a, q_{rig}, f \rangle$, $\langle q_c, b, b, q_{rig}, f \rangle$, $\langle q_c, d, d, q_{rig}, f \rangle$, $\langle q_c, \Box, \Box, q_{rig}, f \rangle$;
6. $\langle q_0, b, \Box, q_d, dx \rangle$;
7. $\langle q_b, a, a, q_b, dx \rangle$, $\langle q_b, b, b, q_b, dx \rangle$, $\langle q_b, c, c, q_b, dx \rangle$, $\langle q_b, d, d, q_b, dx \rangle$;
8. $\langle q_b, \Box, \Box, q_d, sx \rangle$;
9. $\langle q_d, d, \Box, q_{ind}, sx \rangle$;
10. $\langle q_d, a, a, q_{rig}, f \rangle$, $\langle q_d, b, b, q_{rig}, f \rangle$, $\langle q_d, c, c, q_{rig}, f \rangle$, $\langle q_d, \Box, \Box, q_{rig}, f \rangle$;
11. $\langle q_{ind}, a, a, q_{ind}, sx \rangle$, $\langle q_{ind}, b, b, q_{ind}, sx \rangle$, $\langle q_{ind}, c, c, q_{ind}, sx \rangle$, $\langle q_{ind}, d, d, q_{ind}, sx \rangle$;
12. $\langle q_{ind}, \Box, \Box, q_0, dx \rangle$;
13. $\langle q_0, \Box, \Box, q_{acc}, f \rangle$.

Problema 2. Formalizzazione del problema:

- $I = \{G = (V,E) : G \text{ è un grafo }\}$;
- $S(G) = \{\langle c_2, c_3\rangle : c_2 : V \to \{1,2\} \wedge c_3 : V \to \{1,2,3\}\}$;
- $\pi(G, c_2, c_3) = \forall (u,v) \in E : c_2(u) \neq c_2(v) \wedge c_3(u) \neq c_3(v)$.

Si osservi ora che una colorazione dei nodi di un grafo con 2 colori è anche una 3-colorazione dello stesso grafo. Infatti, sia $\chi_2 : V \to \{1,2\}$ tale che $\forall (u,v) \in E : \chi_2(u) \neq \chi_2(v)$ e definiamo la seguente funzione $\chi_3 : V \to \{1,2,3\}$: scegliamo a caso un nodo $u_0 \in V$ e, per ogni nodo $u \in V$, assegniamo $\chi_3(u) = \chi_2(u)$ se $u \neq u_0$ e $\chi_3(u_0) = 3$. Dall'ipotesi che $\forall (u,v) \in E : \chi_2(u) \neq \chi_2(v)$ e dalla definizione di χ_3 segue immediatamente che $\forall (u,v) \in E : \chi_3(u) \neq \chi_3(v)$, ossia, che χ_3 è una 3-colorazione per G.

Pertanto, dato un grafo G, $G \in$2-COLORABILEE3-COLORABILE se e soltanto se $G \in$ 2-COLORABILITÀ. In conclusione, il problema 2-COLORABILEE3-COLORABILE è contenuto nella classe P.

Problema 3. Formalizzazione del problema:

- $I = \{\langle X = \{x-1,\ldots,x_n\}, f(X) = c_1 \wedge c_2 \wedge \ldots \wedge c_m\rangle : X \text{ è un insieme di variabili booleane e, per ogni } 1 \leq j \leq m, c_j = l_{j1} \vee l_{j2} \vee l_{j3} \vee l_{j4} \text{ con } l_{ji} \in X \text{ oppure } \neg l_{ij} \in X \text{ per } i = 1,2,3,4\}$;
- $S(X,f) = \{a : X \to \{$ **vero, falso** $\}\}$;
- $\pi(X,f,a) = f(a(X))$.

Una istanza $\langle X, f\rangle$ del problema 4-SODDISFACIBILITÀ è anche istanza del problema SODDISFACIBILITÀ: segue banalmente da questa osservazione che il problema 4-SODDISFACIBILITÀ è contenuto nella classe NP (formalmente, 4-SODDISFACIBILITÀ$\leq$SODDISFACIBILITÀ).

Per dimostrarne la completezza, mostriamo una riduzione polinomiale dal problema 3-SODDISFACIBILITÀ. Ricordiamo che il problema 3-SODDISFACIBILITÀ consiste nel chiedersi se una funzione booleana in forma congiuntiva normale con clausole di esattamente 3 letterali ciascuna è soddisfacibile.

Sia $X = \{x_1,\ldots,x_n\}$ un insieme di variabili booleane e $g(X) = d_1 \wedge \ldots \wedge D_m\rangle : X$ dove, per ogni $1 \leq j \leq m$, $d_j = l_{j1} \vee l_{j2} \vee l_{j3}$ con $l_{ji} \in X$ oppure $\neg l_{ij} \in X\}$ (ossia, $\langle X, g\rangle$ è un'istanza di 3-SODDISFACIBILITÀ). Costruiamo, a partire da $\langle X, g\rangle$, un'istanza $\langle X \cup Y, f(X \cup Y, g) = c_1 \wedge c'_1 \wedge c_2 \wedge c'_2 \wedge \ldots \wedge c_m \wedge c'_m\rangle$ di 4-SODDISFACIBILITÀ nella maniera seguente:

- definiamo l'insieme di variabili booleane $Y = \{y_1,\ldots,y_m\}$, contenente una variabile booleana per ciascuna clausola in g;
- per $j = 1,\ldots,m$, associamo alla clausola $d_j = l_{j1} \vee l_{j2} \vee l_{j3}$ (con $l_{ji} \in X$ oppure $\neg l_{ij} \in X$) di g la coppia di clausole $c_j = l_{j1} \vee l_{j2} \vee l_{j3} \vee y_j$ e $c'_j = l_{j1} \vee l_{j2} \vee l_{j3} \vee \neg y_j$.

Per costruzione, $\langle X \cup Y, f = c_1 \wedge c'_1 \wedge \ldots \wedge c_m \wedge c'_m\rangle$ è un'istanza di 4-SODDISFACIBILITÀ. Inoltre, essa può essere calcolata in tempo in $O(m)$ a partire da g. Resta da mostrare che che g è soddisfacibile se e soltanto se f è soddisfacibile:

1. se g è soddisfacibile allora esiste una assegnazione di verità a per le variabili in X che soddisfa ciascuna clausola $d_j$in g: poiché $c_j = d_j \vee y_j$ e $c'_j = d_j \vee \neg y_j$ allora a soddisfa sia c_j che c'_j qualunque sia l'assegnazione di verità alle variabili in Y;
2. se f è soddisfacibile allora esiste una assegnazione di verità b per le variabili in $X \cup Y$ che soddisfa ciascuna clausola c_j e ciascuna clausola c'_j in f. Mostriamo ora che, per ogni $j = 1,\ldots,m$, $b(X)$ soddisfa ciascuna clausola d_j: infatti, se $b(y_j) =$**vero** allora

$$\textbf{vero} = b(c'_j) = b(d_j \vee \neg y_j) = b(d_j) \vee b(\neg y_j) = b(d_j) \vee \textbf{falso} = b(d_j),$$

e se $b(y_j) =$**falso** allora

$$\textbf{vero} \ = b(c_j) = b(d_j \vee y_j) = b(d_j) \vee b(y_j) = b(d_j) \vee \ \textbf{falso} \ = b(d_j).$$

Allora, b è una assegnazione di verità che soddisfa g.

Si osservi che questa trasformazione (e la prova della sua correttezza) è la stessa che trasforma clausole di 2 letterali in clausole di 3 letterali nella riduzione da SODDISFACIBILITÀ a 3-SODDISFACIBILITÀ.

Problema 4. Formalizzazione del problema:

- $I = \{\langle G = (V,E), k\rangle : G$ è un grafo non orientato e k un intero positivo$\}$;
- osserviamo ora che per ciascuna istanza $\langle G,k\rangle$ di NO-CLIQUE esiste una *unica soluzione possibile*, ossia, *l'insieme di tutti i sottoinsiemi di V di cardinalità maggiore o uguale a k*: detto $\mathscr{V}_k = \{V' \subseteq V : |V'| \geq k\}$, allora $S(G,k) = \{\mathscr{V}_k\}$;
- $\pi(G,k,(V)_k) = \forall V' \in \mathscr{V}_k \exists u,v \in V' : (u,v) \notin E$.

Osserviamo subito che la dimensione della (unica) soluzione possibile di una istanza di NO-CLIQUE è $O(|V|^k)$: sembra, dunque, poco probabile riuscire a verificare il predicato π in tempo polinomiale in $|V|$ e k. Questa osservazione dovrebbe indurre a ritenere poco probabile che il problema in questione appartenga ala classe NP.

Passiamo ora all'analisi formale. Consideriamo ora una istanza $\langle G,k\rangle \in I$ che non appartenga al linguaggio NO-CLIQUE: allora, V contiene un sottoinsieme V' di almeno k nodi tale che ogni coppia di nodi di V' è collegata da un arco. Questo significa che V' è una clique in G di almeno k nodi. Questo dimostra che l'insieme delle istanze $\langle G,k\rangle \in I$ che *non* appartengono a NO-CLIQUE coicide con l'insieme delle istanze che appartengono a CLIQUE, ossia NO-CLIQUE=CLIQUE^c. Dunque, NO-CLIQUE$\in$Co-NP e, in particolare, poiché CLIQUE è completo rispetto ad NP, NO-CLIQUE è Co-NP-completo.

5 Appello del 26 gennaio 2011

Problema 1. Sia k un valore costante (ad esempio, $k = 5$). Scegliere opportunamente un modello di macchina di Turing e progettare una macchina rispondente alle caratteristiche di tale modello che, ricevendo in input k parole binarie $p_1 = x_{11}x12\ldots x_{1n}$, $p_2 = x_{21}x22\ldots x_{2n}$, $\ldots$, $p_k = x_{k1}xk2\ldots x_{kn}$, tutte aventi la stessa lunghezza n (non costante), esegue su tali parole il *controllo di parità orizzontale e verticale*, ossia, calcola le due parole o e v definite come segue:

- $o = o_1 o_2 \ldots o_k$, dove, per ogni $i = 1, \ldots, k$ $o_i = 1$ se p_i contiene un numero dispari di 1, $o_i = 0$ altrimenti (controllo orizzontale);
- $v = v_1 v_2 \ldots v_n$, dove, per ogni $i = 1, \ldots, n$ $v_i = 1$ se la parola $x_{1i}x2i\ldots x_{ki}$ contiene un numero dispari di 1, $v_i = 0$ altrimenti (controllo verticale).

Problema 2. Sia k un valore costante. Si consideri il seguente problema decisionale: dati un insieme di variabili booleane $X = \{x_1, x_2, \ldots, x_n\}$ e una funzione booleana F nelle variabili X in forma 3-congiuntiva normale, decidere se esiste una assegnazione di verità a per X che verifichi entrambe le seguenti proprietà

1. a soddisfa F
2. a assegna il valore vero ad esattamente k variabili in X.

Formalizzare la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, ed dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si ricordi la definizione del problema CIRCUITO HAMILTONIANO.

Il problema PERCORSO HAMILTONIANO consiste nel chiedersi, dati un grafo non orientato $G = (V, E)$ ed una coppia di nodi $u, v \in V$, se esiste in G un percorso da u a v che tocchi tutti i nodi in V una ed una sola volta.

Il problema LONGEST PATH consiste nel chiedersi, dati un grafo non orientato $G = (V, E)$, una coppia di nodi $u, v \in V$ ed un intero positivo k, se esiste in G un percorso da u a v di lunghezza almeno k.

Dopo aver formalizzato la definizione del problema LONGEST PATH mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **NP** e la completezza rispetto a tale classe. In particolare, al fine di dimostrarne la completezza si richiede di mostrare una riduzione da CIRCUITO HAMILTONIAMO a PERCORSO HAMILTONIANO e poi un riduzione da PERCORSO HAMILTONIANO a LONGEST PATH.

Problema 4. Si consideri il seguente problema decisionale: dato un grafo non orientato $G = (V, E)$ (in cui V è l'insieme dei nodi ed E l'insieme degli archi) ed un intero positivo k, decidere se ogni sottoinsieme di V di almeno k nodi contiene almeno una coppia di nodi adiacenti.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, collocarlo nella corretta classe di complessità.

5.1 Soluzione

Problema 1. Viene utilizzata una macchina di Turing a $k+2$ nastri a testine indipendenti: i primi k nastri contengono le k parole p_1, p_2, …, p_k, sul nastro $k+1$ viene scritta la parola o e sul nastro $k+2$ viene scritta la parola v.

La macchina opera in due fasi: durante la prima fase calcola la parola v e la scrive sul nastro $k+2$, durante la seconda fase calcola la parola o e la scrive sul nastro $k+1$.

Inizialmente, i primi k nastri contengono l'input, scritti a partire dalle cella di indica 0, la macchina si trova nello stato iniziale q_0 e le testine sono posizionate sulle celle di indice 0 dei rispettivi nastri.

Descriviamo la prima fase, che utilizza gli stati q_0, q_2^{v0}, q_2^{v1}, q_3^{v0}, q_3^{v1}, …, q_k^{v0}, q_k^{v1} e q_1^o. Nello stato q_0, se la testina del primo nastro legge 0 allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_2^{v0} e (soltanto) la testina del nastro 1 si muove a destra; se la testina del primo nastro legge 1 allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_2^{v1} e (soltanto) la testina del nastro 1 si muove a destra: $\forall x_2, \ldots, x_k$

$$\langle q_0, (0, x_2, \ldots x_k, \square, \square), (0, x_2, \ldots x_k, \square, \square), q_2^{v0}, (d, f, \ldots, f)\rangle$$

$$\langle q_0, (1, x_2, \ldots x_k, \square, \square), (1, x_2, \ldots x_k, \square, \square), q_2^{v1}, (d, f, \ldots, f)\rangle.$$

Poi, in generale, per $i = 2, \ldots, k-1$:

- nello stato q_i^{v0}, se la testina del nastro i legge 0 allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_{i+1}^{v0} e (soltanto) la testina del nastro i si muove a destra; se la testina del nastro i legge 1 allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_{i+1}^{v1} e (soltanto) la testina del nastro i si muove a destra;
- nello stato q_i^{v1}, se la testina del nastro i legge 0 allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_{i+1}^{v1} e (soltanto) la testina del nastro i muove a destra; se la testina del nastro i legge 1 allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_{i+1}^{v0} e (soltanto) la testina del nastro i si muove a destra.

Infine,

- nello stato q_k^{v0}, se la testina del nastro k legge 0, allora tutte le testine tranne la $k+2$ riscrivono quello che hanno letto, la testina $k+2$ scrive 0, la macchina entra nello stato q_0 e (soltanto) la testina del nastro $k+2$ si muove a destra; se la testina del nastro k legge 1 allora tutte le testine tranne la $k+2$ riscrivono quello che hanno letto, la testina $k+2$ scrive 1, la macchina entra nello stato q_0 e (soltanto) la testina del nastro $k+2$ si muove a destra;
- nello stato q_k^{v1}, se la testina del nastro k legge 0, allora tutte le testine tranne la $k+2$ riscrivono quello che hanno letto, la testina $k+2$ scrive 1, la macchina entra nello stato q_0 e (soltanto) la testina del nastro $k+2$ si muove a destra; se la testina del nastro k legge 1 allora tutte le testine tranne la $k+2$ riscrivono quello che hanno letto, la testina $k+2$ scrive 0, la macchina entra nello stato q_0 e (soltanto) la testina del nastro $k+2$ si muove a destra.

La prima fase termina quando, nello stato q_0 la testina del nastro 1 legge $\square$: in questo caso, tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_1 e le testine dei nastri 1, …, k si muovono a sinistra.

La seconda fase utilizza gli stati q_1^o, q_1^{o0}, q_1^{o1}, …, q_k^o, q_k^{o0}, q_k^{o1}, q_{k+1}^o. Negli stati q_i^o, q_i^{o0}, q_i^{o1} viene calcolato il bit o_i della parola o, per $i = 1, \ldots, k$:

- nello stato q_i^o, se la testina del nastro i legge 0, allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_i^{o0} e (soltanto) la testina del nastro i si muove a sinistra; se la testina del nastro i legge 1, allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_i^{o1} e (soltanto) la testina del nastro i si muove a sinistra;
- nello stato q_i^{o0}, se la testina del nastro i legge 0, allora tutte le testine riscrivono quello che hanno letto, la macchina rimane nello stato q_i^{o0} e (soltanto) la testina del nastro i si muove a sinistra; se la testina del nastro

i legge 1, allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_i^{o1} e (soltanto) la testina del nastro i si muove a sinistra; se la testina del nastro i legge $\square$ allora tutte le testine tranne la $k+1$ riscrivono quello che hanno letto, la testina $k+1$ scrive 0, la macchina entra nello stato q_{i+1}^{o} e (soltanto) la testina del nastro $k+1$ si muove a destra;

- nello stato q_i^{o1}, se la testina del nastro i legge 0, allora tutte le testine riscrivono quello che hanno letto, la macchina rimane nello stato q_i^{o1} e (soltanto) la testina del nastro i si muove a sinistra; se la testina del nastro i legge 1, allora tutte le testine riscrivono quello che hanno letto, la macchina entra nello stato q_i^{o0} e (soltanto) la testina del nastro i si muove a sinistra; se la testina del nastro i legge $\square$ allora tutte le testine tranne la $k+1$ riscrivono quello che hanno letto, la testina $k+1$ scrive 1, la macchina entra nello stato q_{i+1}^{o} e (soltanto) la testina del nastro $k+1$ si muove a destra.

La parola o è stata calcolata e lo stato q_{k+1}^{o} è lo stato finale.

Problema 2. Il problema in questione (che sarà denotato, in breve, k-3SAT) può essere formalizzato nella maniera seguente:

- I_{k-3SAT} = $\{\langle X = \{x_1,\ldots,x_n\}, F\langle :$ X è un insieme di variabili booleane ed F è una funzione nelle variabili in X in forma 3-CNF $\}$.
- $S_{k-3SAT}(X,F) = \{a : X \to \{vero, falso\} : |\{x_i \in X : a(v) = vero\}| = k\}$.
- $\pi_{k-3SAT}(X,F,a) = F(a(X))$.

Osserviamo innanzi tutto che il problema k-3SAT è un caso particolare del più generale 3SAT (o 3-SODDISFACIBILITÀ): in particolare, i predicati dei due problemi coincidono. Allora, poiché 3SAT è un problema in **NP**, il predicato $\pi_{3VC}(G,V')$ è decidibile in tempo polinomiale.

Osserviamo ora che una soluzione possibile è una assegnazione di verità per X che assegni il valore vero ad *esattamente* k variabili: pertanto, una soluzione possibile può essere vista anche come un sottoinsieme $X_V \subseteq X$ tale che $|X_V| = k$, dove X_V è il sottoinsieme di X delle variabili che ricevono il valore vero. Questo significa che $S = \{X' \subseteq X : |X'| = k\}$ e che, dunque, $|S_{k-3SAT}(X,F)| \leq |X|^k$, ossia, il numero di soluzioni possibili è polinomiale nelle dimensioni dell'istanza.

Le due osservazioni precedenti portano alla conclusione che il seguente algoritmo che decide se $\langle X,F\rangle \in k$-3SAT opera in tempo polinomiale in X e in F:

A:k-3SAT
input: $X = \{x_1,x_2,\ldots,x_n\}$ e F, funzione booleana in 3CNF nelle variabili in X
output: accetta o rigetta.

```
S ← {X' ⊆ X : |X'| = k};
trovato ← false;
while ( S ≠ ∅ ∧ trovato = falso) do begin
    estrai un elemento X' da S;
    for (x ∈ X') do a(x) ← vero;
    for (x ∈ X − X') do a(x) ← falso;
    trovato ← π_{k−3SAT}(X,F,a);
endif (trovato) output: accetta;
else output: rigetta.
```

Quindi, k-3SAT è in **P**.

È infine possibile, nell'algoritmo **A:k-3SAT**, esplicitare il calcolo dell'insieme S. Il seguente frammento di programma enumera in ordine lessicografico (rispetto all'indice delle variabili) tutti i sottoinsiemi di cardinalità k di $X = \{x_1,x_2,\ldots,x_n\}$ in tempo O(n^k):

$S \leftarrow \emptyset$;
for ($i_1 \leftarrow 1; i_1 \leq n; i_1 \leftarrow i_1 + 1$) **do**
    for ($i_2 \leftarrow i_1; i_2 \leq n; i_2 \leftarrow i_2 + 1$) **do**
        ...
            for ($i_k \leftarrow 1; i_k \leq n; i_k \leftarrow i_k + 1$) **do**
                $S \leftarrow S \cup \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$;

Alternativamente, è possibile utilizzare, allo scopo di enumerare tutti i sottoinsiemi di cardinalità k di X, il seguente algoritmo non deterministico:

$S \leftarrow \emptyset$;
for ($i \leftarrow 1; i \leq k; i \leftarrow i + 1$) **do begin**
    scegli $x \in X - S$;
    $S \leftarrow S \cup \{x\}$;
end

Tale algoritmo ha grado di non determinismo $n = |X|$ e richiede tempo in O(k) (ossia, costante). Quindi, esso può essere convertito in un algoritmo deterministico che richiede tempo in O(n^{hk}) per qualche valore costante $h > 0$.

Problema 3. Il problema LONGEST PATH (in breve, LP) è formalizzato nella maniera seguente:

- $I_{LP} = \{\langle G = (V,E), u, v, k\rangle : G$ è un grafo non orientato, $u \in V$, $v \in V$, e k è un intero positivo $\}$.
- $S_{LP}(\langle G = (V,E), u, v, k\rangle) = \{p = \langle v_1, v_1, \ldots, v_h\rangle : \forall 1 \leq i \leq h[v_i \in V]\}$.
- $\pi_{LP}(\langle G = (V,E), u, v, k\rangle, p = \langle v_1, \ldots, v_h\rangle) = \forall 0 \leq i < h[(v_i, v_{i+1} \in E)] \wedge u = v_1 \wedge v = v_h \wedge h \geq k+1$.

Il problema è in **NP**: infatti, un qualunque sottoinsieme dell'insieme dei nodi può essere generato non deterministicamente in tempo O(n). Un tale sottoinsieme, se di cardinalità h, corrisponde a $h!$ percorsi possibili in G, ove ciascun percorso corrisponde ad una permutazione degli elementi del sottoinsieme: ad esempio, se consideriamo il sottoinsieme $\{u,v,w,z\}$ di V, i percorsi possibili ad esso corrispondenti sono $\langle u,v,w,z\rangle$, $\langle u,v,z,w\rangle$, $\langle u,w,v,z\rangle$, $\langle u,w,z,v\rangle$,$\langle u,z,v,w\rangle$, $\langle u,z,w,v\rangle$, e così via. Pertanto, possiamo descrivere l'algoritmo non deterministico che decide LP nella maniera seguente:

$V' \leftarrow \emptyset$;
Fase 1: generazione di V'
    for ($x \in V$) **do**
        scegli se inserire o meno x in V';

Fase 2: generazione di una permutazione degli elementi di V'
    $i \leftarrow 1$;
    while ($V' \neq \emptyset$) **do begin**
        scegli $x \in V'$;
        $v_i \leftarrow x$;
        $V' \leftarrow V' - \{x\}$;
        $i \leftarrow i + 1$;
    end
    ora $\langle v_1, v_2, \ldots v_{|V'|}\rangle \in S_{LP}(G, u, v, k)$

Fase 3: verifica del predicato π_{LP}
    $\pi \leftarrow vero$;
    if ($v_1 \neq u \vee v_{|V'|} \neq v \vee |V'| < k+1$) **then** $\pi \leftarrow falso$;

$i \leftarrow 1$;
while $(i < |V'| \wedge \pi = vero)$ **do begin**
    if $((v_i, v_{i+1}) \notin E)$ **then** $\pi \leftarrow falso$;
    $i \leftarrow i+1$;
end

if $(\pi = vero)$ **then Output**: accetta;
else Output: rigetta;

Poiché la Fase 1 e la Fase 2 dell'algoritmo richiedono tempo in O(n) e la Fase 3 richiede tempo in O($n|E|$), tale algoritmo opera in tempo (non deterministico) polinomiale.

Dimostriamo, ora, la completezza di LP rispetto ad **NP** mediante una riduzione polinomiale dal problema PERCORSO HAMILTONIANO (in breve, HP).
Sia $\alpha = \langle G = (V,E), u, v\rangle$ un'istanza di HP: trasformiamo tale istanza nell'istanza $\beta = \langle G = (V,E), u, v, |V|-1\rangle$ di LP. Tale traformazione richiede tempo polinomiale in $|\alpha|$.
Supponiamo che $\alpha \in$HP: allora, G contiene un cammino da u a v che passa per ogni nodo in V una e una sola volta e, dunque, ha lunghezza $|V|-1$. Questo significa che $\beta \in$LP.
Viceversa, supponiamo che $\beta \in$HP: allora, G contiene un cammino da u a v che ha lunghezza $|V|-1$ e, dunque, passa necessariamente per ogni nodo in V una e una sola volta. Questo significa che $\alpha \in$LP.

Mostriamo, infine, una riduzione polinomiale dal problema CIRCUITO HAMILTONIANO (in breve, HC) al problema HP.
Sia $G = (V,E)$ un'istanza di HC, con $V = \{v_1, v_2, \ldots, v_n\}$; trasformiamo tale istanza nell'istanza $\alpha = \langle G_P = (V_P, E_P), a, b\rangle$ come di seguito specificato:

- a e b sono due nuovi nodi, non contenuti in V;
- $V_P = V \cup \{a,b\}$;
- $E_P = E \cup E_{ab}$, dove

$$E_{ab} = \{(a, v_1)\} \cup \{(b, v_i) : v_i \in V \wedge (v_1, v_i) \in E\}.$$

Supponiamo che $G \in$HC: allora, G contiene un ciclo $\langle v_1, v_{i_2}, \ldots, v_{i_n}\rangle$ che tocca tutti i nodi una ed una sola volta. Allora, $(v_1, v_{i_n}) \in E$ e, quindi, $(b, v_{i_n}) \in E_P$. Segue da ciò che $\langle a, v_1, v_{i_2}, \ldots, v_{i_n}, b\rangle$ è un percorso hamiltoniano in G_P da a a b e, quindi, $\alpha \in$HP.
Viceversa, supponiamo che $\alpha \in$HP: allora, G_P contiene un percorso hamiltoniano da a a b. Poiché l'unico nodo adiacente ad a in G_P è v_1, tale percorso sarà del tipo $\langle a, v_1, v_{i_2}, \ldots, v_{i_n}, b\rangle$; poiché $(v_{i_n}, b) \in E_P$, segue dalla definizione di E_{ab} che $(v_1, v_{i_n}) \in E$. Quindi, $\langle v_1, v_{i_2}, \ldots, v_{i_n}\rangle$ è un ciclo hamiltoniano in G.

Problema 4. Il problema in esame, che chiameremo T, è formalizzato nella maniera seguente:

- $I_T = \{\langle G = (V,E), k\rangle : G$ è un grafo non orientato e k un intero positivo$\}$;
- osserviamo ora che per ciascuna istanza $\langle G, k\rangle$ di T esiste una *unica soluzione possibile*, ossia, *l'insieme di tutti i sottoinsiemi di V di cardinalità maggiore o uguale a k*: detto $\mathscr{P}(V,k) = \{V' \subseteq V : |V'| \geq k\}$, allora $S_T(G,k) = \{\mathscr{P}(V,k)\}$;
- $\pi_T(G,k,(P)(V,k)) = \forall V' \in \mathscr{P}(V,k) \exists u, v \in V' : (u,v) \in E = \forall V' \subseteq V : |V'| \geq k [\exists u, v \in V' : (u,v) \in E]$.

Osserviamo subito che la dimensione della (unica) soluzione possibile di una istanza di T è $O(|V|^k)$: sembra, dunque, poco probabile riuscire a generare tale soluzione in tempo non deterministico polinomiale, in quanto per generarla occorre almeno enumerare tutti i suoi elementi. Sembra, quindi, poco probabile che il problema in questione appartenga alla classe **NP**.

Passiamo ora all'analisi formale. Consideriamo ora una istanza $\langle G,k\rangle \in I_T$. Se $\langle G,k\rangle \notin$T, allora esiste un sottoinsieme V' di V di almeno k che contiene una coppia di nodi collegati da un arco: quindi, G contiene un insieme indipendente di cardinalità almeno k, ossia, $\langle G,k\rangle \in$INSIEME INDIPENDENTE. Viceversa, se $\langle G,k\rangle \in$T, allora G non contiene alcun insieme indipendente di almeno k nodi, ossia, $\langle G,k\rangle \notin$INSIEME INDIPENDENTE. In conclusione, T = (INSIEME INDIPENDENTE)^c. Dunque, T$\in$Co-**NP**e, in particolare, poiché INSIEME INDIPENDENTE è completo rispetto ad **NP**, T è Co-**NP**-completo.

6 Appello dell'8 febbraio 2011

Problema 1. Sia $f(n)$ una funzione time-constructible. Dimostrare che, allora, anche $2^{f(n)}$ è una funzione time-constructible.

Problema 2. Un *grafo bipartito* non orientato è un grafo non orientato $G = (V,E)$ in cui l'insieme dei nodi V è partizionato in due sottoinsiemi V_1 e V_2 e non esistono archi fra coppie di nodi appartenenti allo stesso sottoinsieme. Formalmente, un grafo bipartito è un grafo $G = (V_1 \cup V_2, E)$ tale che $V_1 \cap V_2 = \emptyset$ e $(u,v) \in E \rightarrow [u \in V_1 \wedge v \in V_2] \vee [u \in V_2 \wedge v \in V_1]$.

Si consideri il seguente problema decisionale: dato un grafo bipartito non orientato $G = (V_1 \cup V_2, E)$, decidere se G è 3-colorabile.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si ricordi la definizione di colorabilità di un grafo.

Dati un grafo $G = (V,E)$ e $V' \subseteq V$, il *grafo indotto* in G da V' è il grafo $G' = (V', E')$ in cui, per ogni coppia di nodi $x, y \in V'$, $(x,y) \in E'$ se e soltanto se $(x,y) \in E$.

Si consideri il seguente problema decisionale: dato un grafo non orientato $G = (V,E)$ ed un intero positivo k, decidere se l'insieme V contiene un sottoinsieme V' di al più k nodi tale che il sottografo di G indotto da V' sia k-colorabile.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 4. Si consideri il problema di ottimizzazione `minimo Commesso Viaggiatore` ristretto ad istanze in cui i pesi degli archi sono compresi fra 1 e 4.

Progettare un algoritmo approssimante per tale problema e calcolarne l'errore relativo.

Precondizioni:	sul nastro N_1 è scritto il valore n codificato in unario
1	$n_2 \leftarrow f(n_1)$;
2	$n_4 \leftarrow 1$;
3	$i \leftarrow 1$;
4	**while** $(i \leq n_2)$ **do begin**
5	$\quad n_3 \leftarrow n_4$;
6	$\quad n_4 \leftarrow n_4 \oplus n_3$;
7	$\quad i \leftarrow i+1$;
8	**end**

Tabella 0.1: Algoritmo corrispondente alla macchina di Turing T che calcola $2^{f(n)}$.

6.1 Soluzione

Problema 1. Consideriamo una macchina di Turing T a 4 nastri: il nastro N_1 è utilizzato per registrare l'input n, il nastro N_2 per registrare il valore $f(n)$, il nastro N_3 è un nastro di lavoro ed il nastro N_4 è utilizzato per registrare l'output.

Ricordando quanto fatto nella dispensa 5, per il calcolo della funzione $2^{f(n)}$ descriviamo un algoritmo codificato in un qualche linguaggio ad alto livello: indichiamo con una variabile i la posizione della testina sul nastro N_2, con una variabile n_j il contenuto del nastro N_j ($j = 1,2,3,4$), con l'istruzione "$n_u \leftarrow n_v$;" la copia del contenuto del nastro N_v sul nastro N_u , e con "$\oplus$" l'operatore di concatenazione di due stringhe; l'algoritmo è descritto in Tabella 0.1.

Resta da calcolare il numero di passi eseguiti da T. L'istruzione 1 richiede $O(f(n))$ passi poiché, per definizione, $f(n)$ è una funzione limite. L'istruzione 2 scrive il valore 1 sul nastro N_4, in un numero costante di passi, e l'istruzione 3 posiziona la testina del nastro N_2 sul primo carattere che esso contiene. Quest'ultima operazione richiede $O(f(n))$ passi. Il ciclo **while** viene ripetuto $f(n)$ volte; in ciascuna iterazione, il contenuto del nastro N_4 viene copiato sul nastro N_3 (in n_4 passi) e poi il contenuto del nastro N_3 viene concatenato al contenuto del nastro N_4 (in n_3 passi), riposizionano, infine, la testina del nastro N_4 sul suo primo carattere (in $2n_3$ passi) e la testina del nastro N_2 avanza di una posizione. In definitiva, osservando che nel corso della j-esima iterazione del ciclo **while** viene calcolato il valore 2^j (o, equivalentemente, che all'ingresso del ciclo $n_4 = 2^{j-1}$), l'algoritmo proposto richiede un numero di passi pari a

$$
\begin{aligned}
O(f(n)) + \sum_{1\leq i\leq f(n)} \left[4\cdot 2^{i-1}+1\right] &= O(f(n)) + \sum_{1\leq i\leq f(n)} \left[2\cdot 2^{i}+1\right] \\
&= O(f(n)) + f(n) + \sum_{1\leq i\leq f(n)} 2\cdot 2^{i} \;=\; O(f(n)) + 2\sum_{1\leq i\leq f(n)} 2^{i} \\
&= O(f(n)) + 2\left[\frac{2^{f(n)+1}-1}{2-1} - 1\right] \;=\; O(f(n)) + 2\left[2^{f(n)+1}-2\right] \\
&= O(f(n)) + 4\cdot 2^{f(n)} - 4 \;=\; O(2^{f(n)})
\end{aligned}
$$

e questo dimostra che $2^{f(n)}$ è una funzione limite.

Problema 2. Il problema in questione (che sarà denotato, in breve, 3COL-BIP) può essere formalizzato nella maniera seguente:

- $I_{3COL-BIP} = \{G = (V_1 \cup V_2, E \subseteq V_1 \times V_2) :\ G$ è un grafo bipartito $\}$.
- $S_{3COL-BIP}(G) = \{c : V_1 \cup V_2 \rightarrow \{1,2,3\}$.
- $\pi_{3COL-BIP}(G,c) = \forall (u,v) \in E : c(u) \neq c(v)$.

Si osservi che la colorazione c_0 tale che $c_0(u) = 1$ per ogni $u \in V_1$ e $c_0(u) = 2$ per ogni $u \in V_2$ appartiene a $S_{3COL-BIP}(G)$ in quanto $c_0 : V_1 \cup V_2 \to \{1,2,3\}$. Inoltre, poiché

$$(u,v) \in E \Rightarrow [u \in V_1 \wedge v \in V_2] \vee [u \in V_2 \wedge v \in V_1],$$

allora,

$$\forall (u,v) \in E : [c_0(u) = 1 \wedge c_0(v) = 2] \vee [c_0(u) = 2 \wedge c_0(v) = 1],$$

ossia, $\forall (u,v) \in E : c_0(u) \neq c_0(v)$.

Questo significa che, qualunque sia il grafo bipartito G, la funzione c_0 è una soluzione effettiva per l'istanza G di 3COL-BIP. In altri termini, data una qualunque istanza di 3BIP-COL, essa è una istanza sì. Il problema è, quindi, deciso dall'algoritmo che, in tempo costante, accetta qualuque input e, dunque, appartiene alla classe **P**.

Problema 3. Il problema in questione (che sarà denotato, in breve, SUB-COL) può essere formalizzato nella maniera seguente:

- $I_{SUB-COL}$ = $\{\langle G = (V,E), k\rangle :$ G è un grafo non orientato e k è un intero positivo $\}$.
- $S_{SUB-COL}(G,k) = \{V' \subseteq V\}$.
- $\pi_{SUB-COL}(G,k,V') = |V'| = k \wedge \exists c : V' \to \{1,2,\ldots k\} : [\forall u,v \in V' : (u,v) \in E \Rightarrow c(u) \neq c(v)]$.

Si osservi che ogni grafo di n nodi può essere colorato con n colori rispettando il vincolo richiesto dal predicato del problema COLORABILITÀ, ossia, che nessuna coppia di nodi adiacenti sia colorata con lo stesso colore.

Consideriamo, ora, un qualsiasi grafo $G = (V,E)$ di $|V| = n$ nodi:

1. se $n < k$ allora V non contiene alcun sottoinsieme di k nodi e, quindi, G è una istanza no di SUB-COL;
2. se $n \geq k$ allora, per quanto appena osservato, dato qualunque sottoinsieme $V' = \{u_1, u_2, \ldots, u_k\}$ di V tale che $|V'| = k$, la funzione $c' : V' \to \{1,\ldots,k\}$ tale che $c'(u_i) = i$ soddisfa il vincolo $\forall u,v \in V' : (u,v) \in E \Rightarrow c(u) \neq c(v)$. In altri termini, ogni sottoinsieme V' di V con $|V'| = k$ è tale che $\pi_{SUB-COL}(G,k,V')$ assume il valore vero e, quindi, è una soluzione effettiva; di conseguenza G è una istanza sì di SUB-COL.

Riassumendo, l'algoritmo che, preso in input un grafo non orientato $G = (V,E)$ ed un intero positivo k, accetta se $|V| \geq k$ e rigetta altrimenti è un algoritmo che decide SUB-COL ed opera in tempo polinomiale. Dunque, SUB-COL appartiene a **P**.

Problema 4. Denotiamo $[1,4]$MIN-TSP il problema del minimo commesso viaggiatore vincolato a pesi degli archi in $[1,4]$. Si osservi che, in un'istanza $\langle G,p\rangle$ di $[1,4]$MIN-TSP, vale la seguente relazione:

$$\forall u,v,z \in V : p(u,v) \leq 4 \leq 2[1+1] \leq 2[p(u,z) + p(z,v)]. \tag{0.1}$$

Naturalmente, la precedente disuguaglianza può anche essere generalizzata:

$$\forall u,v,z_1,\ldots,z_k \in V : p(u,v) \leq 4 \leq 2[1+1] < 2[p(u,z_1) + p(z_1,z_2) + \ldots + p(z_{k-1},z_k) + p(z_k,v)]. \tag{0.2}$$

Si consideri, ora, l'algoritmo di Christofides e la sua analisi applicati ad istanze di $[1,4]$MIN-TSP, così come descritte nella dispensa 10 del corso. In riferimento alla dispensa, indicati con A, $c(A)$ e $c_{euler}(A)$, rispettivamente, il minimo albero ricoprente G, il suo costo ed il costo del ciclo euleriano costruito su A, valgono ancora le seguenti relazioni

$$c_{euler}(A) = 2c(A) \leq 2\omega_{[1,4]min-TSP}(G,p).$$

Ricordiamo che il passo 4) dell'algoritmo di Christofides sostituisce con "scorciatoie" tratti del ciclo euleriano costituiti da nodi attraverso i quali si è già passati. Ad esempio, il tratto di ciclo euleriano seguente

$$\langle \ldots a_s, a_{s+1}, \ldots, a_{s+k}, a_{s+k+1} = a_{s+k-1}, a_{s+k+2} = a_{s+k-2}, \ldots, a_{s+2k-1} = a_{s+1}, a_{s+2k} \neq a_s, \ldots \rangle$$

viene sostituito dal percorso $\langle \ldots, a_s, a_{s+1}, a_{s+2k}, \ldots \rangle$.
Osserviamo ora che, in virtù delle disuguaglianze (0.1) e (0.2) il costo della "scorciatoia" è non maggiore del doppio del costo del tratto di ciclo euleriano che essa ha sostituito, cioè,

$$\begin{aligned} c(\langle \ldots, a_s, a_{s+1}, \ldots, a_{s+k}, a_{s+k+1} = a_{s+k-1}, a_{s+k+2} = a_{s+k-2}, \ldots, a_{s+2k-1} = a_{s+1}, a_{s+2k} =\neq a_s, \ldots \rangle) \\ \leq 2c(\langle \ldots, a_s, a_{s+1}, a_{s+2k}, \ldots \rangle). \end{aligned}$$

In conclusione, se l'output dell'algoritmo è $\langle i_1, i_2, \ldots, i_n \rangle$, questo significa che

$$c(\langle i_1, i_2, \ldots, i_n \rangle) \leq 2c_{euler}(A) = 4c(A) \leq 4\omega_{[1,4]min-TSP}(G, p).$$

Pertanto, l'algoritmo di Christofides applicato ad istanze di $[1,4]$MIN-TSP fornisce soluzioni il cui errore relativo è

$$\frac{[c(\langle i_1, i_2, \ldots, i_n \rangle) - \omega_{[1,4]min-TSP}(G, p)]}{\omega_{[1,4]min-TSP}(G, p)} \leq \frac{[4\omega_{[1,4]min-TSP}(G, p) - \omega_{[1,4]min-TSP}(G, p)]}{\omega_{[1,4]min-TSP}(G, p)} \leq 3.$$

Input:	$a \in \mathbb{N}$ e $b \in \mathbb{N}$ con $a > 0$ e $b > 0$.
Output:	accetta o rigetta.
1	**if** $(a \geq b)$ **then begin**
2	$max \leftarrow a$;
3	$min \leftarrow b$;
4	**end**
5	**else begin**
6	$max \leftarrow b$;
7	$min \leftarrow a$;
8	**end**
9	**while** $(max \geq min)$ **do**
10	$max \leftarrow max - min$;
11	**if** $(max = 0)$ **then Output:** accetta;
12	**else Output:** rigetta.

Tabella 0.2: Algoritmo **A** : **Div** che decide se due interi positivi, ricevuti in input, sono divisibili.

7 Appello del 20 giugno 2011

Problema 1. Sia $\mathscr{M}$ l'insieme delle matrici quadrate ad elementi nell'insieme $\{1,2,3,4,5,6,7,8,9\}$. Dimostrare che $\mathscr{M}$ è numerabile.

Problema 2. Si consideri il seguente problema decisionale NUMERIDIVISIBILI: dati due numeri interi positivi a e b, decidere se essi sono divisibili (ossia se uno dei due è multiplo dell'altro). In Tabella 0.2 è mostrato l'algoritmo **A** : **Div** che decide se una data coppia $\langle a,b \rangle$ di interi positivi appartiene a NUMERIDIVISIBILI. Dimostrare se **A** : **Div** è un algoritmo polinomiale o meno.

Problema 3. Si ricordi la definizione di colorabilità di un grafo.

Dati un grafo $G = (V,E)$ e $V' \subseteq V$, il *grafo indotto* in G da V' è il grafo $G' = (V',E')$ in cui, per ogni coppia di nodi $x,y \in V'$, $(x,y) \in E'$ se e soltanto se $(x,y) \in E$.

Si consideri il seguente problema decisionale: dato un grafo non orientato $G = (V,E)$ ed un intero positivo k, decidere se l'insieme V contiene un sottoinsieme V' di almeno k nodi tale che il sottografo di G indotto da V' sia 1-colorabile.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I,S,\pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 4. Si consideri il problema decisionale seguente: dato un grafo (non orientato) $G = (V,E)$, decidere se ogni cammino in G costituito da nodi distinti ha lunghezza minore di $|V| - 1$.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I,S,\pi \rangle$, collocarlo nella corretta classe di complessità.

7.1 Soluzione

Problema 1. Sia $\oplus$ l'operatore di concatenazione fra stringhe. Definiamo $f : \mathscr{M} \to \mathbb{N}$ nella maniera seguente: per ogni $M \in \mathscr{M}$, con n righe ed n colonne, sia

$$f(M) = M_{11} \oplus M_{12} \oplus \ldots \oplus M_{1n} \oplus M_{21} \oplus M_{22} \oplus \ldots \oplus M_{2n} \oplus \ldots \oplus M_{n1} \oplus M_{n2} \oplus \ldots \oplus M_{nn},$$

dove M_{ij} denota l'elemento che si trova nella riga i e nella colonna j. Osserviamo che, pur avendo definito f mediante operazioni fra stringhe, $f(M)$ può essere facilmente interpretato come un valore intero in quanto ottenuto concatenando cifre in $\{1,\ldots,9\}$. In effetti, una definizione equivalente per f è la seguente:

$$f(M) = \sum_{i=1}^{n}\sum_{j=1}^{n} 10^{n^2-\left[\frac{(i+j-2)(i+j-1)}{2}+i\right]} \cdot M_{ij}.$$

Resta da mostrare che f è una biezione fra $\mathscr{M}$ ed un sottoinsieme di $\mathbb{N}$, ossia che, per ogni coppia $M, M' \in \mathscr{M}$ con $M \neq M'$, si ha $f(M) \neq f(M')$. Siano $M, M' \in \mathscr{M}$ con $M \neq M'$; allora, se n è il numero di righe e di colonne di M e n' è il numero di righe e di colonne di M', sono possibili i casi seguenti:

- $n < n'$: allora $f(M)$ ha meno cifre di $f(M')$ e, quindi, $f(M) < f(M')$;
- $n > n'$: allora $f(M)$ ha più cifre di $f(M')$ e, quindi, $f(M) > f(M')$;
- $n = n'$: allora, poiché $M \neq M'$, esistono i e j compresi fra 1 ed n tali che $M_{ij} \neq M'_{ij}$. Allora, la cifra in posizione $(n-i)+(n-j)$ di $f(M)$ è diversa dalla cifra in posizione $(n-i)+(n-j)$ di $f(M')$ e, quindi $f(M) \neq f(M')$.

In conclusione, abbiamo mostrato che f associa interi distinti a matrici distinte, ossia, che $\mathscr{M}$ è in corrispondenza biunivoca con un sottoinsieme di $\mathbb{N}$. Questo prova che $\mathscr{M}$ è numerabile.

Problema 2. Senza perdita di generalità, assumiamo che $a \geq b$. Allora, la complessità dell'algoritmo **A** : **Div** in Tabella 0.2 è $\mathbf{O}(\lceil \frac{a}{b} \rceil)$. In particolare, il caso peggiore per l'algoritmo **A** : **Div** si verifica quando $b = 1$. In questo caso, il tempo necessario all'algoritmo a calcolare il proprio output è $\mathbf{O}(a)$.
Osserviamo, ora, che la dimensione dell'input è $|a| + |b| = \log a + \log b$ che, nel caso $b = 1$, diventa $|a| = \log a$. Questo è sufficiente a concludere che l'algoritmo **A** : **Div** ha complessità esponenziale nella *dimensione* dell'istanza.

Problema 3. Segue dalla definizione di colorabilità di un grafo che

due nodi possono essere colorati con lo stesso colore solo se essi non sono adiacenti.

Allora, il problema in questione (che sarà denotato, in breve, SUB-1COL) può essere formalizzato nella maniera seguente:

- $I_{SUB-1COL}$ = $\{\langle G = (V,E), k\rangle :$ G è un grafo non orientato e k è un intero positivo $\}$.
- $S_{SUB-1COL}(G,k) = \{V' \subseteq V\}$.
- $\pi_{SUB-1COL}(G,k,V') = |V'| = k \wedge [\forall u, v \in V' : (u,v) \notin E]$.

Si osservi che la definizione di tale problema coincide con quella del problema INSIEME INDIPENDENTE. Dunque, il problema SUB-1COL=INDEPENDENT SET è **NP**-completo.

Problema 4. Il problema, che chiameremo SOLOCAMMINICORTI (in breve, SCC), può essere formalizzato nella maniera seguente:

- I_{SCC} = $\{\langle G=(V,E)\rangle : G$ è un grafo non orientato $\}$.
- $S_{SCC}(G) = \{\mathscr{P} = \{p : p$ è un cammino (ossia, una sequenza di archi consecutivi senza ripetizione di nodi) in $G\}\}$, ossia, per ogni istanza, esiste una sola soluzione possibile, costituita dall'insieme di *tutti* i cammini in G.
- $\pi_{SCC}(G,\mathscr{P}) = \forall p \in \mathscr{P} : |p| < |V|-1$, dove $|p|$ denota il numero di archi di cui p è costituito.

Consideriamo, ora, il suo problema complementare ESISTECAMMINOLUNGO (ECL):

- I_{ECL} = $\{\langle G=(V,E)\rangle : G$ è un grafo non orientato $\}$.
- $S_{ECL}(G) = \{p : p$ è un cammino (ossia, una sequenza di archi consecutivi senza ripetizione di nodi) in $G\}$.
- $\pi_{ECL}(G,u,v,p) = [|p| = |V|-1]$.

Il problema ECL è in **NP** ed inoltre esso è completo per **NP**. Per completezza, riportiamo nel seguito una semplice riduzione polinomiale dal ben noto problema **NP**-completo CAMMINO HAMILTONIANO (HP) che, sempre per completezza, è di seguito ricordato:

- I_{HP} = $\{\langle G=(V,E),u,v\rangle : G$ è un grafo non orientato $\wedge\ u \in V \wedge v \in V\}$.
- $S_{HP}(G,u,v) = \{p : p$ è un cammino (ossia, una sequenza di archi consecutivi senza ripetizione di nodi) da u a v in $G\}$.
- $\pi_{HP}(G,u,v,p) = [|p| = |V|-1]$.

Trasformiamo una istanza $\langle G=(V,E),u,v\rangle$ di HP in una istanza $\langle G'=(V',E')\rangle$ di ECL nel modo seguente: per ottenere V' aggiungiamo a V due nuovi nodi x e y, ossia, $V' = V \cup \{x,y\}$ e connettiamo x con u e y con v, ossia, $E' = E \cup \{(u,x),(v,y)\}$. È immediato verificare che esiste in G' un cammino di lunghezza $|V'|-1 = |V|+1$ se e solo se esso è un cammino hamiltoniano da x a y e, dunque, se e solo se esiste in G un cammino hamiltoniano da u a v.

Poiché il problema ECL è **NP**-completo, allora il problema SCC, ad esso complementare, è Co**NP**-completo.

8 Appello del 15 settembre 2011

Problema 1. Sia $\mathscr{P}(x)$ l'insieme dei polinomi nella variabile x a coefficienti interi. Ad esempio, il polinomio $-3x^4 + x^2 - 1$ è un elemento di $\mathscr{P}(x)$. Dimostrare che $\mathscr{P}(x)$ è numerabile e rispondere alla domanda seguente: perché, invece, l'insieme dei polinomi nella variabile x a coefficienti reali non è numerabile?

Suggerimento: come è possibile rappresentare un polinomo in forma di stringa?

Problema 2. Definiamo il seguente problema decisionale: dati un grafo non orientato $G = (V,E)$ (possibilmente non connesso e possibilmente contenente nodi isolati) ed un intero k, decidere se esiste una 2-colorazione per G con i colori giallo e verde tale che al più k nodi siano colorati con il colore giallo.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I,S,\pi\rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Sia +2SAT il problema decisionale seguente:

- $I_{+2SAT} = \{\langle f,k\rangle : f : X \to \{vero, falso\}$ è una funzione booleana in forma 2-congiuntiva normale non contenente alcun elemento di X negato e k è un intero positivo $\}$.
- $S_{+2SAT}(f,k) = \{a : X \to \{vero, falso\}\}$.
- $\pi_{+2SAT}(f,k,a) = f(a(X)) \wedge |\{x \in X : a(x) = vero\}| \leq k$.

Si dimostri la **NP**-completezza del suddetto problema.

Suggerimento: riduzione mediante VERTEX COVER.

Problema 4. Si consideri il problema di ottimizzazione seguente: dato un grafo non orientato $G = (V,E)$, calcolare il sottoinsieme $V' \subseteq V$ di cardinalità minima tale che

$$\forall u \in V\ [u \in V' \vee \forall (u,v) \in E : v \in V']$$

Dopo aver formalizzato la definizione del suddetto problema mediante la quintupla $\langle I,S,\pi,m,\tau\rangle$, individuare per esso un algoritmo approssimante calcolandone il corrispondente errore relativo.

8.1 Soluzione

Problema 1. Costruiamo una biezione f_S fra l'insieme $\mathscr{P}(x)$ e un sottoinsieme delle parole (ossia, stringhe di lunghezza finita) sull'alfabeto $\{0,1,\ldots,9,+,x\}$: poiché l'insieme delle parole su un alfabeto finito è numerabile, l'esistenza di tale biezione dimostra la numerabilità di $\mathscr{P}(x)$.
Allo scopo, sia $\oplus$ l'operatore di concatenazione fra stringhe, sia $s(n)$ la rappresentazione in forma di stringa del numero naturale n (ad esempio, $s(44) = 44$) e sia $\sigma(z)$ la rappresentazione in forma di stringa dell'intero $z \in (Z)$ comprendente il suo segno: ad esempio, $\sigma(-146) = -146$ e $\sigma(44) = +44$.
Consideriamo un elemento $p \in \mathscr{P}(x)$: $p = \Sigma_{i=0}^{n} a_i x^i$, con $n \in \mathbb{N}$ e $a_i \in \mathbb{Z}$ per $i = 0,\ldots,n$. Allora,

$$f_S(p) = \sigma(a_0) \oplus \sigma(a_1) \oplus x \oplus \sigma(a_2) \oplus x \oplus s(2) \oplus \ldots \oplus \sigma(a_n) \oplus x \oplus s(n).$$

Ad esempio, $f_S(1 - 14x^2 + 16x^{110}) = +1 - 14x2 + 16x110$.
Banalmente, ad ogni polinomio $p \in \mathscr{P}(x)$ è possibile associare una stringa $f_S(p)$. Inoltre, per ogni coppia di elementi distinti p_1 e p_2 di $\mathscr{P}(x)$ si ha che $f_S(p_1) \neq f_S(p_2)$. Infine, che, poiché il grado di ciascun polinomio in $\mathscr{P}(x)$ è finito e poiché ciascun coefficiente di ciascun poinomio è un numero con un numero finito di cifre (essendo un intero), per ogni $p \in \mathscr{P}(x)$ la sua rappresentazione $f_S(p)$ è una stringa di lunghezza finita. Dunque, f_S è una biezione fra $\mathscr{P}(x)$ ed un sottoinsieme di un insieme numerabile e questo prova che $\mathscr{P}(x)$ è numerabile.

Se, invece, q è un polinomio nella variabile x a coefficienti *reali*, qualcuno dei suoi coefficienti potrebbe essere irrazionale e non ammettere una rappresentazione finita. La stringa rappresentante q non sarebbe, in questo caso, finita.

Problema 2. Il problema, che denoteremo, in breve, 2COL-min, può essere formalizzato nella maniera seguente:

- $I_{2COL-min} = \{\langle G = (V,E), k\rangle : G$ è un grafo non orientato e k è un intero positivo $\}$.
- $S_{2COL-min}(G,k) = \{c : V \to \{giallo, verde\}\}$.
- $\pi_{2COL-min}(G,k,c) = [\forall (u,v) \in E : c(u) \neq c(v)] \wedge |\{v \in V : c(v) = giallo\}| \leq k$.

Si osservi che, poiché viene richiesta una 2-colorazione dei nodi di un grafo, in ogni componente connessa, una volta scelto il colore di un nodo, il colore assegnato a tutti gli altri nodi è una conseguenza necessaria di tale scelta. In altre parole, un grafo 2-colorabile ammette sempre una (unica) partizione dei nodi in due sottoinsiemi, ciascuno corrispondente ad uno dei due colori. Sia f_{2col} la funzione che calcola una 2-colorazione di un grafo F connesso: per quanto appena osservato, possiamo assumere che, se F è 2-colorabile allora $f_{2col}(F)$ calcola una partizione $\langle V_g, V_v\rangle$ dell'insieme dei nodi di F, altrimenti restituisce la coppia di insiemi vuoti. Consideriamo l'algoritmo seguente:

Input: grafo $G = (V,E)$ le cui componenti connesse sono $G_1 = (V_1,E_1), G_2 = (V_2,e_2), \ldots, G_h = (V_h,E_h)$;

1) $V_{giallo} \leftarrow \emptyset$;

2) **for** $i = 1,\ldots h$:

- **2.1)** $\langle V_g, V_v\rangle \leftarrow f_{2col}(G_i)$;
- **2.2)** **if** $V_g = \emptyset$ **then output** rigetta:
- **2.3)** **else if** $|V_g| < |V_v|$ **then** $V_{giallo} \leftarrow V_{giallo} \cup V_g$;
- **2.4)** **else** $V_{giallo} \leftarrow V_{giallo} \cup V_v$;

3) **if** $|V_{giallo}| \leq k$ **then output** accetta;

4) **else output** rigetta.

Per ogni componente connessa (2-colorabile), tale algoritmo assegna il colore giallo al più piccolo insieme dei nodi della componente che sono stati colorati con lo stesso colore dalla funzione f_{2col}. Pertanto, l'insieme V_{giallo} da esso

calcolato è l'insieme dei nodi di cardinalità minima che può essere colorato con lo stesso colore. È poi sufficiente confrontare $|V_{giallo}|$ con k per decidere correttamente circa l'accettazione.
Poiché il calcolo di f_{2col} richiede tempo polinomiale, l'algoritmo opera in tempo polinomiale.

In conclusione, il problema 2COL-min appartiene alla classe **P**.

Problema 3. Osserviamo, innanzi tutto, che il problema è in **NP**: infatti, possiamo generare in tempo non deterministico polinomiale tutte le assegnazioni di verità per l'insieme X (allo stesso modo in operiamo per il problema SAT) e poi, per ciascuna di esse, possiamo verificare il predicato π_{+2SAT} in tempo deterministico polinomiale.

Per quanto concerne la completezza, mostriamo ora che +2SAT $\leq$ VERTEX COVER (VC). Si ricordi che un'istanza di VC è una coppia $\langle G = (V,E), k\rangle$, in cui G è un grafo non orientato e k un intero positivo, che una soluzione possibile è un sottoinsieme $V' \subseteq V$ di al più k nodi di G, e che una soluzione possibile V' è una soluzione effettiva se, per ogni arco $(u,v) \in E$, $u \in V'$ o $v \in V'$.
Sia data, dunque, un'istanza $\langle G = (V,E), k\rangle$ di VC. Associamo a G la formula booleana f_G nel modo seguente:

- $X = \{x_u : u \in V\}$;
- per ogni arco $e = (u,v) \in E$, definiamo la clausola $c_e = x_u \vee x_v$;
- f_G è la congiunzione di tutte le clausole c_e: se $E = \{e_1, e_2, \ldots e_m\}$, allora

$$f_G(X) = \bigwedge_{e \in E} c_e = c_{e_1} \wedge c_{e_2} \wedge \ldots \wedge c_{e_m}.$$

Osserviamo che f_G rispetta i vincoli definiti per +2SAT. Dunque, $\langle f_G, k\rangle \in I_{+2SAT}$.
Supponiamo che $G = (V,E)$ ammetta un Vertex Cover di al più k nodi: allora, esiste $V' \subseteq V$ con $|V'| \leq k$ tale che, per ogni $(u,v) \in E$, $u \in V' \vee v \in V'$. Definiamo, allora, la seguente assegnazione di verità a per X: per ogni $u \in V'$ $a(x_u) = vero$, $u \notin V'$ $a(x_u) = falso$. Poiché $|V'| \leq k$, segue che $|\{x \in X : a(x) = vero\}| \leq k$; inoltre, poiché una clausola in f_G corrisponde ad un arco in G e ogni arco ha almeno un estremo in V', allora ogni clausola in f_G è soddisfatta da a, ossia, $f_G(a(X)) = vero$.
Supponiamo, ora, che esista una assegnazione di verità a per X che soddisfa f_G e tale che $|\{x \in X : a(x) = vero\}| \leq k$. Definiamo, allora, $V' = \{u \in V : a(x_u) = vero\}$. Per costruzione, $|V'| \leq k$; inoltre, poiché ad ogni arco in E corrisponde una clausola in f_G ed ogni clausola contiene almeno un letterale vero, allora ogni arco in E ha almeno un estremo in V', ossia, V' è un Vertex Cover per G.

Abbiamo quindi dimostrato che $\langle G = (V,E), k\rangle \in$ VC se e solo se $\langle f_G, k\rangle \in$ +2SAT. Poiché la costruzione di f_G richiede tempo polinomiale, il problema +2SAT è **NP**-completo.

Problema 4. Si consideri il problema di ottimizzazione seguente: dato un grafo non orientato $G = (V,E)$, calcolare il sottoinsieme $V' \subseteq V$ di cardinalità minima tale che

$$\forall u \in V \ [u \in V' \ \vee \ \forall (u,v) \in E : v \in V']$$

Si osservi che il problema in esame è il problema MIN-VERTEX COVER, formalizzato mediante la quintupla seguente:

- $I = \{G = (V,E) : G$ è un grafo non orientato $\}$;
- $S(G) = \{V' \subseteq V\}$;
- $\pi(G,V') = [u \in V' \ \vee \ \forall (u,v) \in E : v \in V'] = \forall (u,v) \in E : v \in V' \vee v \in V'$;
- $m(V') = |V'|$;
- $\tau = \min$.

È noto un algoritmo approssimante per il problema MIN-VERTEX COVER il cui errore relativo è pari a 1 (Esempio 8.10 del testo).

9 Appello del 1 febbraio 2012

Problema 1. Dimostrare che l'insieme dei polinomi a due variabili con coefficienti interi è numerabile.

Problema 2. Sia k un intero positivo fissato. Definiamo il seguente problema decisionale: data una funzione booleana f in forma congiuntiva normale costituita da k clausole, decidere se f è soddisfacibile.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Sia $G = (V, E)$ un grafo non orientato e sia $V' \subseteq V$; indichiamo con $G - V'$ il grafo non orientato che si ottiene rimuovendo da G i nodi in V' e gli archi incidenti qualche nodo in V'. Quindi, l'insieme dei nodi di $G - V'$ è $V - V'$ ed l'insieme dei suoi archi è il seguente sottoinsieme di E: $\{(u, v) \in E : u \notin V' \wedge v \notin V'\}$.

Si consideri il problema decisionale seguente: dati un grafo non orientato $G = (V, E)$ e un intero positivo k, decidere se, comunque si scelga un nodo $u \in V$, esistono k nodi $u_1, u_2, \ldots, u_k \in V$ tali che il nodo u è isolato nel grafo $G - \{u_1, \ldots, u_k\}$.

Collocare il suddetto problema nella corretta classe di complessità.

Problema 4. Si consideri il problema decisionale seguente: dati un grafo non orientato $G = (V, E)$ e un intero positivo k, decidere se, comunque si scelga un nodo $u \in V$, non esistono k nodi $u_1, u_2, \ldots, u_k \in V$ tali che $\langle u, u_1, \ldots, u_k \rangle$ è una clique in G.

Collocare il suddetto problema nella corretta classe di complessità.

9.1 Soluzione

Problema 1. Indichiamo con $\mathscr{P}(x,y)$ l'insieme dei polinomi a due variabili con coefficienti interi.
Sono possibili numerose soluzioni differenti a questo problema. In questa sede, ne presentiamo due.

1) Costruiamo una biezione f_S fra l'insieme $\mathscr{P}(x,y)$ e un sottoinsieme delle parole (ossia, stringhe di lunghezza finita) sull'alfabeto $\{0,1,\ldots,9,+,x\}$, analogamente a quanto illustrato nella soluzione del problema 1 proposto all'appello del 15/09/2011: poiché l'insieme delle parole su un alfabeto finito è numerabile, l'esistenza di tale biezione dimostra la numerabilità di $\mathscr{P}(x)$.
Allo scopo, sia $\oplus$ l'operatore di concatenazione fra stringhe, sia $s(n)$ la rappresentazione in forma di stringa del numero naturale n (ad esempio, $s(44) = 44$) e sia $\sigma(z)$ la rappresentazione in forma di stringa dell'intero $z \in (Z)$ comprendente il suo segno: ad esempio, $\sigma(-146) = -146$ e $\sigma(44) = +44$.
Consideriamo un elemento $p \in \mathscr{P}(x)$: $p = \Sigma_{i=0}^{n}\Sigma_{j=0}^{m} a_{ij}x^i y^j$, con $n,m \in \mathbb{N}$ e $a_{ij} \in \mathbb{Z}$ per $i = 0,\ldots,n$ e $j = 0,\ldots,m$.
Allora,

$$f_S(p) = \sigma(a_{00}) \oplus \sigma(a_{10}) \oplus x \oplus \ldots \oplus \sigma(a_{n0}) \oplus x \oplus s(n) \oplus \sigma(a_{01}) \oplus y \oplus \ldots \sigma(a_{0m}) \oplus y \oplus s(m) \oplus \sigma(a_{11}) \oplus x \oplus y \ldots \oplus \sigma(a_{nm})$$

La dimostrazione che $f_S(p)$ è una biezione è analoga a quella presentata per la soluzione del problema citato ed è, pertanto, omessa.

2) Poiché ogni polinomio in $\mathscr{P}(x,y)$ è il prodotto di un polinomio in $\mathscr{P}(x)$ e di un polinomio in $\mathscr{P}(y)$, allora $\mathscr{P}(x,y)$ è il prodotto cartesiano di $\mathscr{P}(x)$ e $\mathscr{P}(y)$, ossia, $\mathscr{P}(x,y) = \mathscr{P}(x) \times \mathscr{P}(x,y)$. Poiché $\mathscr{P}(x)$ è numerabile, come anche $\mathscr{P}(y)$, (problema 1 proposto all'appello del 15/09/2011) ed il prodotto cartesiano di due insiemi numerabili è numerabile, segue l'asserto.

Problema 2. Il problema, che denoteremo, in breve, k-SAT, può essere formalizzato nella maniera seguente:

- $I_{k-SAT} = \{\langle X, f\rangle : f$ è una funzione booleana in forma congintiva normale nelle variabili in X costituita da esattamente k clausole $\}$.
- $S_{k-SAT}(X,f) = \{a : X \to \{vero, falso\}\}$.
- $\pi_{k-SAT}(X,f,a) = f(a(X))$.

Si osservi che, se una clausola è la disgiunzione di $h \leq n$ letterali, ossia, $g_j = L_{j1} \vee l_{j2} \vee \ldots \vee l_{jh}$, allora esistono esattamente h *possibilità* di soddisfarla: assegnare $l_{j1} = vero$, oppure $l_{j2} = vero$, ..., oppure $l_{jh} = vero$. Per ciascuna di queste possibilità è necessario verificare se essa compatibile con almeno una possibilità di soddisfare ciascuna altra clausola. In altri termini, scegliamo un letterale da soddisfare in ciascuna clausola e verifichiamo che tale scelta non contenga una cotraddizione, ossia, una coppia di letterali l ed l' tali che $l = \neg l'$. Se questo non accade allora f è soddisfacibile, altrimenti viene scelta un'altra k-upla di letterali (uno per ciascuna clausola) e si ripete la verifica. L'algoritmo è mostrato in Tabella 0.3 dove, per semplicità di notazione, ciascuna clausola viene considerata come *insieme* di letterali.
In riferimento alla Tabella 0.3, poiché $|A| \leq n^k$ ed il controllo alla linea 8 viene ripetuto al più k^2 volte (ossia, un numero costante di volte), il costo dell'algoritmo è in $O(n^k)$, e questo prova che il problema è in **P**.

Problema 3. Scriviamo in maniera diversa la proprietà che deve essere soddisfatta da una istanza $\langle G = (V,E), k\rangle$ del problema affinché essa sia una istanza sì. Viene richiesto che: per ogni nodo $u \in V$ esistano k nodi rimossi i quali u non sia adiacente ad alcun altro nodo di G. Questo significa che ogni nodo del grafo può essere adiacente ad al più k nodi in G. Tale proprietà può essere controllata in tempo polinomiale in $|V| \cdot |E|$, come illustrato nel seguente algoritmo:

1) inizializza $esito \leftarrow vero$;
2) per ogni nodo $u \in V$

Input:	X, $f = c_1 \wedge c_2 \wedge c_k$ con $c_j = \{l_{j_1}, \ldots, l_{j_{h_j}}\}$, per ogni $j = 1, \ldots, k$.	
1	$A \leftarrow c_1 \times c_2 \times \ldots \times c_k$;	A è il prodotto cartesiano delle clausole: un suo elemento è una k-upla costituita da un letterale per ciascuna clausola
2	$sat \leftarrow falso$;	
3	**while** ($A \leq \emptyset \ \wedge \ sat = falso$) **do begin**	
4	estrai una k-upla $\langle l_1, l_2, \ldots, l_k \rangle$ da A;	
5	$sat \leftarrow vero$;	
6	**for** $i = 1$; $i < k$; $i \leftarrow i + 1$) **do**	
7	**for** $j = i + 1$; $j \leq k$; $j \leftarrow j + 1$) **do**	
8	**if** ($l_i = \neg l_j$) **then** $sat \leftarrow falso$;	
9	**end**	
10	**if** ($sat = vero$) **then Output**: accetta;	
11	**else Output**: rigetta.	

Tabella 0.3: Algoritmo che decide k-SAT.

2.1) inizializza $cont \leftarrow 0$;

2.2) per ogni arco $(u, v) \in E$ esegui $cont \leftarrow cont + 1$;

2.3) se $cont > k$ assegna $esito \leftarrow falso$;

3) se $esito = vero$ allora **Output**: accetta;

4) altrimenti **Output**: rigetta.

Questo prova che il problema appartiene alla classe **P**.

Problema 4. Si osservi che, dati un grafo non orientato $G = (V, E)$ e un intero positivo k, il problema consiste nel decidere se G non contiene alcuna clique di $k+1$ nodi. Il problema coincide, dunque, con il complemento del problema CLIQUE. Conseguentemente, esso è co**NP**-completo.

10 Appello del 22 febbraio 2012

Problema 1. Dimostrare se la funzione $f : \mathbb{N} \to \mathbb{N}$ di seguito descritta è time-constructible.

$$f(n) = \begin{cases} \frac{3n}{2} & \text{se } n \text{ è pari} \\ \frac{3(n-1)}{2} & \text{se } n \text{ è dispari} \end{cases}$$

Problema 2. Sia $\mathscr{G}_{clique} = \{G = (V, V \times V)$ la classe dei grafi completi e sia VERTEX COVER$(\mathscr{G}_{clique})$ il problema VERTEX COVER ristretto all'insieme delle istanze $\langle G = (V,E), k\rangle$ in cui $G \in \mathscr{G}_{clique}$.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi\rangle$, dimostrarne l'appartenenza alla classe **P**.

Rispondere, inoltre, alla seguente domanda: quale è la cardinalità minima di un ricoprimento tramite nodi per un grafo $G \in \mathscr{G}_{clique}$?

Problema 3. Sia $G = (V,E)$ un grafo non orientato e sia $D \subseteq V$; D è un insieme dominante per G se ogni nodo in $V - D$ ha un vicino in D. Il problema DOMINATING SET consiste nel decidere, dati un grafo $G = (V,E)$ e un intero k, se G contiene un insieme dominante di cardinalità $\leq k$.

1. Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi\rangle$, dimostrarne l'appartenenza alla classe **NP**.
2. La funzione f descritta di seguito è una riduzione polinomiale da VERTEX COVER a DOMINATING SET:

   sia $\langle G = (V,E), k\rangle$ una istanza di VERTEX COVER; allora $f(G,k) = \langle H = (W,F), k+1\rangle$ dove H è il grafo tale che

   - $W = V \cup X \cup \{a\}$, dove $X = \{x_e : e \in E\}$
     ed a è un nuovo nodo
   - $F = E \cup Y \cup Z$, con $Y = \{(u,x_e), (v,x_e) : e = (u,v) \in E\}$
     e $Z = \{(a,u) : u \in V$ è un nodo isolato$\}$.

   Dimostrare che essa può essere calcolata in tempo polinomiale in $|G|$ e k e che se G ammette un Vertex Cover di al più k nodi allora H ammette un Dominating Set di al più $k+1$ nodi.

Suggerimento: il nodo a serve a gestire eventuali nodi isolati presenti nel grafo G.

Problema 4. Si consideri il problema di ottimizzazione MINIMUM DOMINATING SET: dato un grafo non orientato $G = (V,E)$, calcolare l'insieme dominante di cardinalità minima per G.

Verificare se l'algoritmo approssimante per MINIMUM VERTEX COVER è approssimante anche per MINIMUM DOMINATING SET.

Suggerimento: considerare il caso in cui il grafo G è una clique di n nodi.

10.1 Soluzione

Problema 1. Definiamo una macchina di Turing T a 3 nastri (a testine indipendenti) che opera in quattro fasi distinte come di seguito descritto:

- il nastro N_1 contiene l'input, ivi memorizzato in unario all'inizio della computazione, preceduto e seguito da $\Box$;
- il nastro N_2 è il nastro di lavoro, inizialmente vuoto, sul quale, al termine della prima fase, si troverà il valore $n/2$, se n è pari, o $(n-1)/2$ se n è dispari;
- il nastro N_3 è il nastro di output, inizialmente vuoto, sul quale, al termine della fase k si trova scritto il valore $(k-1)n$, per $k = 2,3,4$.

Durante la prima fase, in cui viene scritto sul nastro N_2 il valore $n/2$ oppure il valore $(n-1)/2$, sono utilizzati gli stati q_0 (stato iniziale) e q_1^1 e q_0^1: il primo (oltre ad essere stato iniziale) indica che sul nastro N_1 è stato letto un numero pari di 1, il secondo che è stato letto un numero dispari di 1. Il valore $n/2$ o $(n-1)/2$ viene calcolato contestualmente al controllo di parità di n: ogni volta che viene letta una coppia di 1 sul nastro N_1 viene scritto un (singolo) 1 sul nastro N_2; così, al termine della scansione dell'input, se sono stati letti un numero dispari di 1 e viene letto un $\Box$ allora nulla viene scritto sul nastro N_2. Formalmente, le quintuple utilizzate nella prima fase sono:

$$\langle q_0,(1,\Box,\Box),(1,\Box,\Box),q_1^1,(d,f,f)\rangle \qquad \langle q_0,(\Box,\Box,\Box),(\Box,\Box,\Box),q^2,(f,f,f)\rangle$$

$$\langle q_1^1,(1,\Box,\Box),(1,1,\Box),q_0,(d,d,f)\rangle \qquad \langle q_1^1,(\Box,\Box,\Box),(\Box,\Box,\Box),q^2,(f,s,f)\rangle,$$

dove q^2 è lo stato iniziale della fase 2. Osserviamo che, al termine della prima fase, la testina del nastro N_2 è posizionata sul carattere 1 più a destra ivi contenuto.
La seconda fase scrive il valore $n/2$ oppure $(n-1)/2$ sul nastro N_3: questo corrisponde a copiare il contenuto del nastro N_2 sul nastro N_3 (con la testina del nastro N_2 che si muove da destra a sinistra e la testina del nastro N_3 che si muove da sinistra a destra). Allo scopo, è sufficiente utilizzare lo stato q^2:

$$\langle q^2,(\Box,1,\Box),(\Box,1,1),q^2,(f,s,d)\rangle \qquad \langle q^2,(\Box,\Box,\Box),(\Box,\Box,\Box),q^3,(f,d,f)\rangle,$$

dove q^3 è lo stato iniziale della fase 3. Osserviamo che, al termine della seconda fase, la testina del nastro N_2 è posizionata sul carattere 1 più a sinistra ivi contenuto.
La terza fase somma il valore $n/2$ oppure $(n-1)/2$ al contenuto del nastro N_3: questo corrisponde ad una concatenazione dei contenuti dei due nastri, ossia, a copiare il contenuto del nastro N_2 sul nastro N_3 a partire dal carattere 1 più a sinistra su quest'ultimo contenuto. In questa fase, le testine del nastro N_2 e del nastro N_3 si muovono entrambe da sinistra a destra. Allo scopo, è sufficiente utilizzare lo stato q^3:

$$\langle q^3,(\Box,1,\Box),(\Box,1,1),q^3,(f,d,d)\rangle \qquad \langle q^3,(\Box,\Box,\Box),(\Box,\Box,\Box),q^4,(f,s,f)\rangle,$$

dove q^4 è lo stato iniziale della fase 4. Osserviamo che, al termine della seconda fase, la testina del nastro N_2 è posizionata sul carattere 1 più a destra ivi contenuto.
La quarta fase concatena il contenuto del nastro N_2 al contenuto del nastro N_3 ed è del tutto simile alla fase 2:

$$\langle q^4,(\Box,1,\Box),(\Box,1,1),q^4,(f,s,d)\rangle \qquad \langle q^4,(\Box,\Box,\Box),(\Box,\Box,\Box),q_F,(f,f,f)\rangle,$$

dove q_F è lo stato finale.

Per dimostrare che $f(n)$ è una funzione time-constructible occorre ancora mostrare che la macchina di Turing appena descritta opera in tempo $O(f(n))$. A questo scopo, osserviamo che la prima fase richiede un numero di passi pari alla lunghezza del contenuto del nastro N_1 più 1, mentre ciascuna delle altre fasi richiede un numero di passi pari alla lunghezza del contenuto del nastro N_2 più 1. Quindi, il numero di passi totale è

$$t(n) = \begin{cases} n+1+3\left(\frac{n}{2}+1\right) & \text{se } n \text{ è pari} \\ \\ n+1+3\left(\frac{n-1}{2}+1\right) & \text{se } n \text{ è dispari,} \end{cases}$$

ossia, $t(n) \in O(f(n))$.

Problema 2. Osserviamo che, per ogni intero positivo n, esiste un soIo grafo di n nodi in $\mathscr{G}_{clique}$: indichiamo tale grafo con K_n e con $V = \{u_1, u_2, \ldots, u_n\}$ l'insieme dei suoi nodi. Il problema VERTEX COVER($\mathscr{G}_{clique}$) può essere formalizzato nella maniera seguente:

- $I_{VC(\mathscr{G}_{clique})} = \{\langle K_n, h\rangle : K_n$ è un grafo non orientato completo e h un intero positivo $\}$.
- $S_{VC(\mathscr{G}_{clique})}(K_n, h) = \{V' \subseteq \{u_1, \ldots, u_n\}\}$.
- $\pi_{VC(\mathscr{G}_{clique})}(K_n, h, V') = |V'| \leq h \wedge \forall (u,v) \in E[u \in V' \vee v \in V']$.

Per risolvere il problema, è sufficiente notare che K_n non ha ricoprimenti tramite nodi di cardinalità inferiore a $n-1$. Infatti, poiché K_n è un grafo simmetrico, è indifferente quale nodo scegliere inizialmente di inserire in V': qualunque nodo si scelga, rimarranno scoperti tutti gli archi ad esso non incidenti che costituiscono un grafo completo di $n-1$ nodi, ossia, il grafo K_{n-1} (si veda la Figura 0.1) . Pertanto, la cardinalità minima $h_{min}(n)$ di un vertex cover per K_n

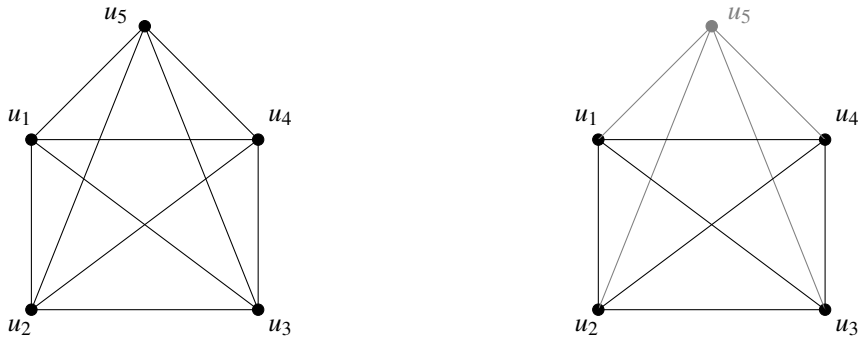


Figura 0.1: (a) Il grafo K_5; (b) gli archi non coperti dal nodo u_5 costituiscono il grafo K_4.

soddisfa la seguente relazione di ricorrenza:

$$h_{min}(n) = 1 + h_{min}(n-1)$$

che ha come soluzione $h_{min}(n) = n-1$, che risponde anche alla domanda posta a conclusione del problema 2.
In conclusione, $\langle K_n, h\rangle$ è una istanza sì di VERTEX COVER($\mathscr{G}_{clique}$) se e soltanto se $h \geq n-1$. Questo prova che il problema VERTEX COVER($\mathscr{G}_{clique}$) è in **P**.

Problema 3. Il problema DOMINATING SET può essere formalizzato nella maniera seguente:

- $I_{DS} = \{\langle G = (V,E), k\rangle : G$ è un grafo non orientato completo e k un intero positivo $\}$.
- $S_{DS}(G,k) = \{D \subseteq V\}$.
- $\pi_{DS}(G,k,D) = |D| \leq k \wedge \forall u \in V - D[\exists v \in D : (u,v) \in E]$.

Il problema è in **NP**: infatti, un sottoinsieme dell'insieme V dei nodi può essere generata non deterministicamente in tempo O(n). Allo scopo, indichiamo con $V = \{v_1, v_2, \ldots, v_n\}$ l'insieme dei nodi di G e consideriamo un albero delle computazioni di profondità n e grado di non determinismo 2 in cui un nodo a livello $j = 1, 2, \ldots, n-1$ corrisponde

ad un sottoinsieme X di $\{v_1, v_2, \dots v_j\}$: i due figli di tale nodo corrispondono ai due sottoinsiemi $X \cup \{v_{j+1}\}$ e X di $\{v_1, \dots v_{j+1}\}$. In tal modo, le foglie dell'albero corrispondono ai sottoinsiemi di V. Una volta generato un sottoinsieme V' di V, come appena descritto, è sufficiente verificare se $|V'| \leq k$ (in tempo deterministico in $O(\min(n,k))$) e se ogni nodo in $V - V'$ è adiacente a qualche nodo in V' (in tempo deterministico in $O(kn|E|)$).

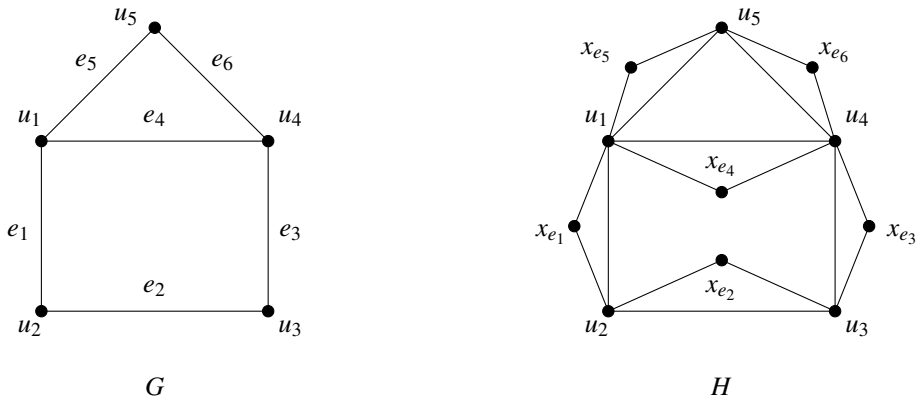


Figura 0.2: Il grafo G, istanza di VERTEX COVER ed il suo corrispondente H tramite la funzione f. In G sono stati anche evidenziati i nomi degli archi: all'arco e_i di G corrisponde il nodo x_{e_i} di H, per $i = 1, \dots, 6$.

Mostriamo ora che, dato un grafo non orientato G ed un intero positivo k, $f(G,k) = \langle H, k+1 \rangle$ è calcolabile in tempo polinomiale in $|G|$ e k. Infatti, detto $G = (V,E)$, l'insieme dei nodi del grafo H è costruito aggiungendo all'insieme V un nodo per ogni arco in E, mentre l'insieme degli archi di H è costruito aggiungendo all'insieme E una coppia di archi per ogni arco in E:

1) $X \leftarrow \emptyset$;
2) **for** $(e \in E)$ **do** $X \leftarrow X \cup \{x_e\}$;
3) $W \leftarrow V \cup X \cup \{a\}$;
4) $Y \leftarrow \emptyset$;
5) **for** $(e = (u,v) \in E)$ **do** $Y \leftarrow Y \cup \{(u,x_e),(v,x_e)\}$;
6) $Z \leftarrow \emptyset$;
7) **for** ($u \in V$ isolato in G) **do** $Z \leftarrow Z \cup \{(a,u)\}$;
8) $F \leftarrow E \cup Y \cup Z$;
9) **Output**: $H = (W,F)$.

Questo richiede tempo in $O(|E| + |V|)$.

Resta da far vedere che se $\langle G = (V,E), k \rangle$ è una istanza sì di VERTEX COVER allora $f(G,k) = \langle H = (V \cup X, E \cup Y), k+1 \rangle$ è una istanza sì di DOMINATING SET. A questo scopo, dato un insieme dominante C per un grafo, diciamo che un nodo b non contenuto in C è *dominato* da un nodo $c \in C$ se (b,c) è un arco del grafo.
Supponiamo, allora, che $G = (V,E)$ ammetta un Vertex Cover di k nodi: in tal caso, esiste $V' \subseteq V$ tale che $|V'| \leq k$ e, per ogni arco $(u,v) \in E$, $u \in V'$ oppure $v \in V'$. Proviamo, ora, che $D = V' \cup \{a\}$ è un Dominating Set per H. Infatti, per ogni $u \in W - D = W - (V' \cup \{a\})$, sono possibili i soli tre casi seguenti: $u \in V$ ed è adiacente a qualche nodo $v \in V$, oppure $u \in V$ ed è isolato in G, o, infine, $u \in X$. Di seguito, analizziamo separatamente i tre casi.

- $u \in V$ ed è adiacente a qualche nodo $v \in V$ (ossia, $(u,v) \in E$). In questo caso, poiché V' è un vertex cover per G e poiché $u \notin V'$, allora $v \in V' \subset D$ e, quindi, u è dominato da v.

- $u \in V$ ed è isolato in G. In questo caso, u è dominato da $a \in D$.
- $u \in X$. In questo caso, $u = x_e$, per qualche $e = (v_1, v_2) \in E$, e quindi, poiché $v_1 \in V' \vee v_2 \in V'$ e $(x_e, v_1) \in F \wedge (x_e, v_2) \in F$, $u = x_e$ è dominato da v_1 o da v_2.

Dunque, D è un insieme dominante per G.

Problema 4. L'algoritmo approssimante per MINIMUM VERTEX COVER, ricordiamo, sceglie, uno dopo l'altro, un insieme di archi che non hanno estremi in comune.
Sia K_n un grafo completo di n nodi e sia $V = \{v_1, \ldots v_n\}$ l'insieme dei suoi nodi. Con input K_n, l'algoritmo citato sceglie un arco, che, in virtù della simmetria di K_n, pssiamo assumere sia l'arco (v_1, v_2), ed inserisce v_1 e v_2 in V', poi sceglie un secondo arco disgiunto da (v_1, v_2), diciamo (v_3, v_4), ed inserisce v_3 e v_4 in V', e così via fino ad esaurire gli archi: osserviamo che, al termine dell'esecuzione dell'algoritmo, si avrà $V' = V$ se n è pari, oppure $V' = V - \{v_n\}$ se n è dispari.
D'altra parte, un insieme dominante di cardinalità minima per il grafo completo K_n è costituito da un unico nodo, in quanto tutti gli altri nodi sono ad esso adiacenti. Allora, indicati con $D^*(n)$ e con $V'(n)$, rispettivamente, un insieme dominante di cardinalità minima per K_n e l'output dell'algorimo approssimante per MINIMUM VERTEX COVER con input K_n, abbiamo:

$$\frac{|V'(n)| - |D^*(n)|}{|D^*(n)|} = \frac{|V'(n)| - 1}{1} \geq \frac{n-1-1}{1} = n-2$$

che non è limitato da alcun valore costante. Dunque, K_n ùn controesempio al fatto che l'algorimo approssimante per MINIMUM VERTEX COVER possa essere approssimante anche per MINIMUM DOMINATING SET.

11 Appello del 2 luglio 2012

Problema 1. Sia $k \in \mathbb{N}$ un valore costante. Dimostrare che le due funzioni $f_k : \mathbb{N} \to \mathbb{N}$ e $g_k : \mathbb{N} \to \mathbb{N}$ di seguito descritte sono time-constructible.

$$f_k(n) = \lceil \frac{n}{k} \rceil;$$

$$g_k(n) = \lfloor \frac{n}{k} \rfloor.$$

Problema 2. Sia L il problema decisionale che consiste nel decidere, dati un insieme di interi $A = \{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{N}$ e un intero $k \in \mathbb{N}$, se A contiene un sottoinsieme la somma dei cui elementi sia k. Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **NP**. Infine, progettare un algoritmo deterministico che decida tale problema calcolandone la complessità.
Suggerimento: un sottoinsieme di Z può essere rappresentato mediante una stringa binaria di n caratteri, che, a sua volta, può essere associata ad un numero intero compreso fra 0 e ....

Problema 3. Nel problema di ottimizzazione MAXIMUM SATISFIABILITY è data una funzione booleana f nelle n variabili $x_1, x_2, \ldots, x_n$, in forma congiuntiva normale, ed è richiesto di calcolare l'assegnazione di verità per le variabili $x_1, x_2, \ldots, x_n$ che soddisfi il massimo numero di clausole in f.

Sia $\mathscr{S}$ la classe delle funzioni booleane in forma congiuntiva normale che sono soddisfacibili, e sia MAXIMUM SATISFIABILITY($\mathscr{S}$) il problema MAXIMUM SATISFIABILITY ristretto all'insieme delle istanze $f \in \mathscr{S}$.

Verificare se MAXIMUM SATISFIABILITY($\mathscr{S}$) appartiene alla classe **PO**, dimostrando la propria affermazione.

Problema 4. Si ricordi che un insieme dominante per un grafo (non orientato) $G = (V, E)$ è un insieme $V' \subseteq V$ tale che ogni nodo in $V - V'$ ha un vicino in V'. Nel problema di ottimizzazione MINIMUM DOMINATING SET è richiesto di calcolare l'insieme dominante di cardinalità minima per un dato grafo non orientato $G = (V, E)$.

Sia $\mathscr{G}_{clique} = \{G = (V, V \times V)$ la classe dei grafi completi e sia MINIMUM DOMINATING SET($\mathscr{G}_{clique}$) il problema MINIMUM DOMINATING SET ristretto all'insieme delle istanze in cui $G = (V, E) \in \mathscr{G}_{clique}$.

Verificare se MINIMUM DOMINATING SET($\mathscr{G}_{clique}$) appartiene alla classe **PO**, dimostrando la propria affermazione.

11.1 Soluzione

Problema 1. Definiamo una macchina di Turing T a 3 nastri (a testine indipendenti) che calcola simultaneamente $\lceil \frac{n}{k} \rceil$ e $\lfloor \frac{n}{k} \rfloor$ come di seguito descritto:

- il nastro N_1 contiene l'input, ivi memorizzato in unario all'inizio della computazione, preceduto e seguito da $\Box$;
- il nastro N_2 è il nastro di lavoro e di output per la funzione $\lceil \frac{n}{k} \rceil$, inizialmente vuoto, sul quale, al termine della computazione, si troverà il valore $\lceil \frac{n}{k} \rceil$;
- il nastro N_3 è il nastro di lavoro e di output per la funzione $\lfloor \frac{n}{k} \rfloor$, inizialmente vuoto, sul quale, al termine della computazione, si troverà il valore $\lfloor \frac{n}{k} \rfloor$.

T utilizza gli stati q_0 (stato iniziale), q_1, q_2, ... $q_{k-1}k$ (si ricordi che k è una costante) e lo stato finale q_F: q_0 (oltre ad essere stato iniziale) indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k, q_1 indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k più un ulteriore 1, e, in generale, q_i ($i < k$) indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k più ulteriori i 1.
Il valore $\lceil \frac{n}{k} \rceil$ viene calcolato scrivendo un 1 sul nastro N_2 ogni volta che, nello stato q_0, viene letto un 1 sul nastro N_1: questo significa che n è almeno un multiplo di k (perché la macchina è nello stato q_0) più 1 (perché viene letto un 1). Così, se $n = hk + m$, con $m < k$, al termine della scansione dell'input, risultano scritti sul nastro N_2 h 1 più un eventuale ulteriore 1 se $m > 0$.
Analogamente, il valore $\lfloor \frac{n}{k} \rfloor$ viene calcolato scrivendo un 1 sul nastro N_2 ogni volta che, nello stato q_{k-1}, viene letto un 1 sul nastro N_1: questo significa che n è almeno un multiplo di k (perché la macchina è nello stato q_{k-1} e viene letto un 1). Così, se $n = hk + m$, con $m < k$, al termine della scansione dell'input, risultano scritti sul nastro N_3 h 1.

Formalmente, le quintuple utilizzate sono:

$$\begin{array}{ll}
\langle q_0, (1,\Box,\Box), (1,1,\Box), q_1, (d,d,f)\rangle & \\
\langle q_i, (1,\Box,\Box), (1,\Box,\Box), q_{i+1}i, (d,f,f)\rangle & \forall 1 \le i \le k-2 \\
\langle q_{k-1}, (1,\Box,\Box), (1,\Box,1), q_0, (d,d,f)\rangle & \\
\langle q_i, (\Box,\Box,\Box), (\Box,\Box,\Box), q_F, (f,f,f)\rangle, & \forall 1 \le i \le k-1.
\end{array}$$

Per dimostrare che $f_k(n)$ e $g_k(n)$ sono funzioni time-constructible occorre ancora mostrare che la macchina di Turing appena descritta opera in tempo O(n)=O($f_k(n)$)=O($g_k(n)$) (in quanto k è una costante). A questo scopo, è sufficiente osservare che la computazione descritta richiede un numero di passi pari alla lunghezza del contenuto del nastro N_1. Quindi, il numero di passi totale è $t(n) = n$.

Problema 2. Il problema L può essere formalizzato nella maniera seguente:

- $I_L = \{\langle A,k\rangle : A = \{a_1,\ldots,a_n\} \subset \mathbb{N}$ e k è un intero positivo $\}$.
- $S_L(A,k) = \{A' \subseteq A\}$.
- $\pi_L(A,k,A') = \sum_{a\in A'} a \le k$.

Per provare che il problema è in **NP** occorre progettare un algoritmo non deterministico che lo decide in tempo polinomiale. Tale algoritmo è rappresentabile mediante un albero delle computazioni binario e di altezza n: i nodi al livello i dell'albero (con $1 \le i \le n$) corrispondono a tutti i sottoinsiemi possibili dell'insieme $\{a_1,a_2,\ldots,a_i\}$. Esso è descritto dal seguente pseudo-codice:

Input: $A = \{a_1,\ldots,a_n\} \subset \mathbb{N}$, $k \in \mathbb{N}$.

$A' \leftarrow \emptyset$;

for ($i \leftarrow 1$; $i \le n$; $i \leftarrow i+1$) **do**

Input:	$A = \{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{N}, k \in \mathbb{N}$
1.	$sum \leftarrow 0$;
2.	$cont \leftarrow 0$;
3.	**while** ($cont \leq 2^n - 1 \wedge sum \neq k$) **do begin**
4.	$q \leftarrow cont$;
5.	$i \leftarrow 1$;
6.	**while** ($q > 0$) **do begin**
7.	$bin[i] \leftarrow q \ mod \ 2$;
8.	$q \leftarrow q/2$;
9.	$i \leftarrow i+1$;
10.	**end**
11.	$sum \leftarrow 0$;
12.	**for** ($i \leftarrow 1$; $i \leq n$; $i \leftarrow i+1$) **do**
13.	**if** ($bin[i] = 1$) **then** $sum \leftarrow sum + a_i$;
14.	$cont \leftarrow cont + 1$;
15.	**end**
16.	**if** ($sum = k$) **then Output**: esiste;
17.	**else Output**: non esiste.

Tabella 0.4: Algoritmo che verifica se A contiene un sottoinsieme di peso k.

scegli se inserire a_i in A'; *(passo non deterministico)*

if $\sum_{a \in A'} a \leq k$ **then Output:** accetta;

else Output: rigetta.

Affrontiamo ora la progettazione di un algoritmo deterministico che decida L. Come indicato nel suggerimento, fissato un insieme $A = \{a_1, a_2, \ldots a_n\}$, i suoi sottoinsiemi sono in corrispondenza biunivoca con l'insieme delle stringhe binarie di lunghezza n: la stringa $b = b_1 b_2 \ldots b_n$ corrisponde al sottoinsieme di A

$$A_b = \{a_i \in A : b_i = 1, i = 1, \ldots, n\}.$$

Ma una stringa binaria di lunghezza n rappresenta un intero compreso fra 0 e $2^n - 1$: la stringa $b = b_1 b_2 \ldots b_n$ corrisponde all'intero $n_b = \sum_{i=1}^{n} b_i \cdot 10^{i-1}$.
L'algoritmo deterministico che decide L basato sulle precedenti osservazioni è descritto in Tavola 0.4: esso è costituito da un ciclo **while** che conta gli interi da 0 a $2^n - 1$ e verifica se l'insieme corrispondente ha peso non superiore a k (linee 3-15). In particolare, durante un'iterazione del ciclo viene generata la stringa binaria corrispondente all'intero in forma di array (ciclo **while** alle linee 6-10) e viene calcolato il peso dell'insieme corrispondente (linee 12-13).
Il ciclo **while** esterno viene ripetuto 2^n volte, il costo di ciascuna iterazione è dominato dal ciclo **while** interno ($O(n)$) e dal ciclo **for** ($O(\log \sum_{i=1}^{n} a_i)$). Pertanto, il costo dell'algoritmo è $O(n 2^n \log \sum_{i=1}^{n} a_i)$.

Problema 3. Il problema MAXIMUM SATISFIABILITY(S) non è contenuto in **NPO** in quanto non è decidibile in tempo polinomiale se una funzione booleana f in forma congiuntiva normale è una istanza di MAXIMUM SATISFIABILITY(S): decidere sf è istanza di MAXIMUM SATISFIABILITY(S) è, infatti, equivalente a decidere se f è soddisfacibile, nota questione **NP**-completa. Dunque, MAXIMUM SATISFIABILITY(S) non è contenuto in **PO**.

Problema 4. Poiché decidere se un grafo è una clique richiede tempo polinomiale, il problema MINIMUM DOMINATING SET($\mathscr{G}_{clique}$) è in **NPO**. Inoltre, qualsiasi nodo di una clique è un insieme dominante per una clique. Quindi, l'algoritmo che prende in input una clique e restituisce in output uno qualsiasi dei suoi nodi calcola una soluzione ottima per MINIMUM DOMINATING SET($\mathscr{G}_{clique}$). In conclusione, il problema MINIMUM DOMINATING SET($\mathscr{G}_{clique}$) è in **PO**.

12 Appello del 13 settembre 2012

Problema 1. Progettare una macchina di Turing che calcoli le due funzioni $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ e $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ di seguito descritte:

$$f(n,k) = \lceil \frac{n}{k} \rceil; \qquad g(n,k) = \lfloor \frac{n}{k} \rfloor \qquad \text{con } k > 0.$$

Problema 2. Si consideri il seguente problema LARGE DOMINATING SET (in breve, LDS): decidere se, dato un grafo non orientato connesso $G = (V,E)$, esiste insieme $D \subseteq V$ di al più $\lfloor \frac{|V|}{2} \rfloor$ nodi tale che ogni nodo in $V - D$ ha almeno un vicino in D.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I_{LDS}, S_{LDS}, \pi_{LDS} \rangle$, si dimostri se il seguente algoritmo ne prova l'appartenenza alla classe **P**.

Input: $G = (V,E)$.

1) $T \leftarrow$ spanning tree di G;

2) $r \leftarrow$ radice di T;

3) $D_0 \leftarrow \{r\} \cup \{u \in V$ a distanza pari da r in $T\}$;

4) $D_1 \leftarrow \{u \in V$ a distanza dispari da r in $T\}$;

5) if $(|D_0| \leq \lfloor \frac{|V|}{2} \rfloor \vee |D_1| \leq \lfloor \frac{|V|}{2} \rfloor)$ **then Output: accetta**;

6) else Output: rigetta

In quale caso il precedente algoritmo rigetta?

Problema 3. Sia LARGE DOMINATING SET (in breve, LDS) il problema decisionale descritto al Problema 2. Dimostrare se la funzione $f : I_{2COL} \to I_{LDS}$ descritta di seguito è una riduzione polinomiale da 2-COLORABILITÀ a LARGE DOMINATING SET: dato un grafo non orientato $G = (V,E)$, istanza di 2-COLORABILITÀ, la corrispondente istanza di LARGE DOMINATING SET secondo f è il medesimo grafo $G = (V,E)$, ossia $f(G) = G$.

Problema 4. Si consideri il seguente problema decisionale: dato un grafo non orientato $G = (V,E)$, decidere se non esiste alcuna partizione dei nodi in due sottoinsiemi indipendenti V_1 e V_2. Collocare tale problema nella corretta classe di complessità dimostrando la propria affermazione.

12.1 Soluzione

Problema 1. Definiamo una macchina di Turing T a 4 nastri (a testine indipendenti) che calcola simultaneamente $\lceil \frac{n}{k} \rceil$ e $\lfloor \frac{n}{k} \rfloor$ come di seguito descritto:

- il nastro N_1 contiene l'input n ed il nastro N_2 contiene l'input k, ivi memorizzati in unario all'inizio della computazione, preceduti e seguiti da $\Box$;
- il nastro N_3 è il nastro di lavoro e di output per la funzione $\lceil \frac{n}{k} \rceil$, inizialmente vuoto, sul quale, al termine della computazione, si troverà il valore $\lceil \frac{n}{k} \rceil$;
- il nastro N_4 è il nastro di lavoro e di output per la funzione $\lfloor \frac{n}{k} \rfloor$, inizialmente vuoto, sul quale, al termine della computazione, si troverà il valore $\lfloor \frac{n}{k} \rfloor$.

T utilizza gli stati q_0 (stato iniziale), q_1, q_2 e lo stato finale q_F: q_0 (oltre ad essere stato iniziale) indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k, q_1 indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k più qualche ulteriore 1 (in numero inferiore a k), e q_2 indica che la scansione del nastro 2 è terminata avendo letto per ciascun 1 sul nastro N_2 un corrispondente 1 sul nastro N_1 (ossia, se la macchina entra per la h-esima volta nello stato q_2, allora n vale almeno hk) ed occorre riposizionare la testina di N_2 sul simbolo 1 più a sinistra.
Il valore $\lceil \frac{n}{k} \rceil$ viene calcolato scrivendo un 1 sul nastro N_3 ogni volta che, nello stato q_0, viene letto un 1 sul nastro N_1: questo significa che n è almeno un multiplo di k (perché la macchina è nello stato q_0) più 1 (perché viene letto un 1). Così, se $n = hk + m$, con $m < k$, al termine della scansione dell'input, risultano scritti sul nastro N_3 h 1 più un eventuale ulteriore 1 se $m > 0$.
Analogamente, il valore $\lfloor \frac{n}{k} \rfloor$ viene calcolato scrivendo un 1 sul nastro N_4 ogni volta che la macchina entra nello stato q_2: questo significa che n è almeno un multiplo di k. Così, se $n = hk + m$, con $m < k$, al termine della scansione dell'input, risultano scritti sul nastro N_4 h 1.

Formalmente, le quintuple utilizzate sono:

$$\begin{aligned}
&\langle q_0, (1,1,\Box,\Box), (1,1,1,\Box), q_1, (d,d,d,f)\rangle\\
&\langle q_0, (\Box,1,\Box,\Box), (\Box,1,\Box,\Box), q_F, (f,f,f,f)\rangle\\
&\langle q_1, (1,1,\Box,\Box), (1,1,\Box,\Box), q_1, (d,d,f,f)\rangle\\
&\langle q_1, (1,\Box,\Box,\Box), (1,\Box,\Box,1), q_2, (f,s,f,d)\rangle\\
&\langle q_1, (\Box,1,\Box,\Box), (\Box,\Box,\Box,\Box), q_F, (f,f,f,f)\rangle\\
&\langle q_1, (\Box,\Box,\Box,\Box), (\Box,\Box,\Box,1), q_F, (f,f,f,f)\rangle\\
&\langle q_2, (1,1,\Box,\Box), (1,1,\Box,\Box), q_2, (f,s,f,f)\rangle\\
&\langle q_2, (1,\Box,\Box,\Box), (1,\Box,\Box,\Box), q_0, (f,d,f,f)\rangle
\end{aligned}$$

Problema 2. Il problema *LDS* può essere formalizzato nella maniera seguente:

- $I_{LDS} = \{\langle G = (V,E)\rangle : G \text{ è un grafo connesso non orientato }\}$.
- $S_{LDS}(G) = \{D \subseteq V\}$.
- $\pi_{LDS}(G,D) = |D| \leq \lfloor \frac{|V|}{2} \rfloor$.

Per provare che l'algoritmo proposto decide LDS è sufficiente osservare che gli insiemi D_0 e D_1 sono entrambi insiemi dominanti per G: infatti, poiché T è un albero ricoprente per G, $V = D_0 \cup D_1$ e, quindi, ciascun nodo in $V - D_0 = D_1$ è adiacente a qualche nodo in D_0 (ossia, D_0 è un insieme dominante) e ciascun nodo in $V - D_1 = D_0$ è adiacente a qualche nodo in D_1 (ossia, D_1 è un insieme dominante). Poiché il calcolo di un albero ricoprente per un grafo richiede tempo polinomiale nella dimensione del grafo, e poiché i passi 3) e 4) richiedono una visita di T (eseguibile in tempo

polinomiale nella dimensione di T) e la condizione al punto 5) non è che un confronto fra valori interi, questo dimostra che LDS è un problema in **P**.

Si osservi, infine, che la condizione al punto 5) è sempre verificata: infatti, poiché $V = D_0 \cup D_1$ e $D_0 \cap D_1 = \emptyset$, allora $|V| = |D_0| + |D_1|$ e quindi $|D_0| \leq |V|/2$ oppure $|D_1| \leq |V|/2$.

Problema 3. Si ricordi che, affinché f sia una riduzione polinomiale da 2COL a LDS, è necessario che f sia caolcolabile in tempo polinomiale e che, per ogni grafo $G \in I_{2COL}$, si abbia che G è istanza sì per 2COL se e soltanto se $f(G) = G$ è istanza sì per LDS. La funzione f è ovviamente calcolabile in tempo polinomiale (essendo la funzione identità); tuttavia, essa non è una riduzione polinomiale da 2-COL a LDS. Per dimostrare tale affermazione è sufficiente mostrare un grafo che non sia 2 colorabile ma ammetta un insieme dominante contenente al più la metà dei suoi nodi oppure un grafo che sia 2 colorabile ma non ammetta un insieme dominante contenente al più la metà dei suoi nodi. In quanto segue descriviamo due grafi che rispettano, ciascuno, una delle due proprietà appena enunciate.

a) Sia K_3 il grafo completo di 3 nodi: K_3 non è 2 colorabile ma ammette un insieme dominante costituito da 1 nodo.

b) Sia I_n il grafo costituito da n nodi isolati: I_n è 2 colorabile (in effetti è sufficiente un solo colore per colorare i suoi nodi) ma l'unico insieme dominante per I_n contiene tutti i suoi nodi.

Dunque, f non è una riduzione polinomiale da 2COL a LDS.

Problema 4. Indichiamo con NO INDEPENDENT SET PARTITION (in breve NISP) il problema decisionale in esame e consideriamone il complemento ISP: dato un grafo non orientato $G = (V, E)$, decidere se esiste una partizione dei nodi in due sottoinsiemi indipendenti V_1 e V_2. Si osservi che tale problema coincide con il problema 2COL: infatti, poiché è possibile assegnare lo stesso colore a due nodi distinti solo se essi non sono adiacenti, l'insieme dei nodi colorati con il colore 1 (e, equivalentemente, con il colore 2) è un insieme indipendente. Quindi, ISP (ossia, 2COL) è in **P**. Conseguentemente, il suo complemento NISP è anch'esso in **P**.

13 Appello del 31 gennaio 2013

Problema 1. Siano $\Sigma = \{a,b\}$ e $L = \{a^n b^{2n} : \ n \in \mathbb{N}\}$ ove, ricordiamo, a^n è la parola costituita dalla concatenazione di n caratteri a. Progettare una macchina di Turing che decide L.

Problema 2. Sia k un intero positivo fissato. Definiamo il seguente problema decisionale: data una funzione booleana f in forma congiuntiva normale, decidere se f è soddisfacibile da una assegnazione di verità che assegna il valore vero ad esattamente k variabili.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si consideri il seguente problema decisionale: dati un grafo (non orientato) $G = (V,E)$ ed un intero k, decidere se G soddisfa *almeno una* delle seguenti due proprietà:

a) G è un grafo di k nodi

b) esiste per G un Vertex Cover (ossia, un ricoprimento tramite nodi) di cardinalità al più k.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 4. Si consideri il problema di ottimizzazione seguente: data una funzione booleana f in forma congiuntiva normale e soddisfacibile, calcolare il minimo numero di variabili cui deve essere assegnato il valore vero in una assegnazione di verità che soddisfa f.
Dimostrare se il suddetto problema è contenuto nella classe **PO**.

13.1 Soluzione

Problema 1. La macchina T che decide L opera come di seguito descritto. Nello stato q_0, T legge il simbolo nella cella scandita dalla testina e, se tale simbolo è a, lo cancella, entra nello stato q_a, raggiunge la fine della parola e, se gli ultimi due caratteri sono una coppia di b, li cancella e torna all'inizio della parola. Se invece la parola termina con qualcosa di diverso, T entra nello stato di rigetto q_R. Analogamente, se nello stato q_0 T legge b allora entra nello stato q_R. Se, infine, nello stato q_0 T legge $\Box$ allora entra nello stato di accettazione q_A (in quanto la parola input è la parola vuota, che appartiene ad L, oppure tutti i suoi caratteri sono stati cancellati). Formalmente, T è definita dalle quintuple seguenti:

$$\langle q_0,a,\Box,q_a,destra\rangle \quad \langle q_0,b,b,q_R,fermo\rangle \quad \langle q_0,\Box,\Box,q_A,fermo\rangle \tag{0.3}$$
$$\langle q_a,a,a,q_a,destra\rangle \quad \langle q_a,b,b,q_{ab},destra\rangle \quad \langle q_a,\Box,\Box,q_R,fermo\rangle \tag{0.4}$$
$$\langle q_{ab},a,a,q_R,fermo\rangle \quad \langle q_{ab},b,b,q_{ab},destra\rangle \quad \langle q_{ab},\Box,\Box,q_b,sinistra\rangle \tag{0.5}$$
$$\langle q_b,b,\Box,q_{b1},sinistra\rangle \quad \text{gli altri casi non sono possibili} \tag{0.6}$$
$$\langle q_{b1},a,a,q_R,fermo\rangle \quad \langle q_{b1},b,\Box,q_1,sinistra\rangle \quad \langle q_{b1},\Box,\Box,q_R,fermo\rangle \tag{0.7}$$
$$\langle q_1,a,a,q_1,sinistra\rangle \quad \langle q_1,b,b,q_1,sinistra\rangle \quad \langle q_1,\Box,\Box,q_0,destra\rangle \tag{0.8}$$

Problema 2. Il problema, che denoteremo, in breve, k-SAT, può essere formalizzato nella maniera seguente:

- I_{k-SAT} = $\{\langle X,f\rangle :$ f è una funzione booleana in forma congiuntiva normale nelle variabili in X $\}$.
- $S_{k-SAT}(X,f) = \{a : X \to \{vero, falso\}\}$.
- $\pi_{k-SAT}(X,f,a) = f(a(X)) \wedge (\ |\{x \in X : a(x) = vero\}| = k\ )$.

Il vincolo sul numero di variabili cui si può assegnare il valore *vero* comporta che le assegnazioni di verità che potrebbero essere soluzioni effettive sono tutti e soli i sottoinsiemi di X di cardinalità k. Il numero di tali sottoinsiemi è al più $|X|^k$, cioè, poiché k è una costante, è polinomiale nella dimensione dell'istanza $\langle f,X\rangle$.
Questo fatto viene sfruttato nell'algoritmo in Tabella 0.5, che esegue una ricerca esaustiva nel sottoinsieme dell'insieme delle parti di X costituito da tutti (e soli) i sottoinsiemi di cardinalità k. Ogni ciclo **while** esegue $O(n)$ iterazioni e, quindi, poiché il numero di cicli nidificati è k, il numero totale di iterazioni è $O(n^k)$. Una volta scelto il sottoinsieme (ossia, nell'algoritmo, una volta scelte le variabili $x_{i_1}, a_{i_2}, \ldots, x_{i_k}$ cui assegnare il valore vero), sarà sufficiente verificare se l'assegnazione di verità corrispondente soddisfa f (linee $3k+3$ e $3k+4$): in caso affermativo, l'algoritmo termina accettando, in caso negativo viene eseguita (se possibile) l'iterazione successiva.
Poiché la verifica alle linee $3k+3$ e $3k+4$ può essere eseguita in tempo $O(nm)$, allora il tempo di calcolo complessivo dell'algoritmo è in $O(n^k+nm)$. Poiché k è una costante, questo prova che il problema è in **P**.

Problema 3. Il problema, che denoteremo, in breve, BVC (BOUNDED SIZE VERTEX COVER), può essere formalizzato nella maniera seguente:

- I_{BVC} = $\{\langle G=(V,E),k\rangle :$ G è un grafo non orientato $\}$.
- $S_{BVC}(G,k) = \{V' \subseteq V\}$.
- $\pi_{BVC}(G,k,V') = |V| = k\ \vee\ \{\ [\ \forall (u,v) \in E(\ u \in V' \vee v \in V'\ )]\ \wedge\ |V'| \leq k\ \}$.

Il problema BVC coincide con il problema VERTEX COVER. Infatti, consideriamo un grafo $G=(V,E)$ ed un intero k:

- se G è un grafo di k nodi, allora V è un Vertex Cover per G e, quindi, $\langle G=(V,E),k\rangle$ è una istanza sì sia per BVC che per VC;
- se $|V| > k$, allora $\langle G=(V,E),k\rangle$ è una istanza sì per BVC se e soltanto se $\langle G=(V,E),k\rangle$ è una istanza sì per VC.

Input:	$X = \{x_1, x_2, \ldots, x_n\}$, $f = c_1 \wedge c_2 \wedge c_m$ con $c_j = \{l_{j_1}, \ldots, l_{j_{h_j}}\}$, per ogni $j = 1, \ldots, m$.
1	$sat \leftarrow falso$;
2	$i_1 \leftarrow 1$;
3	**for** $(i \leftarrow 1; i \leq n; i \leftarrow i+1)$ **do** $a(x_i) \leftarrow falso$;
4	**while** $(i_1 \leq n-k+1 \ \wedge \ sat = falso)$ **do begin**
5	$a(x_{i_1}) \leftarrow vero$;
6	$i_2 \leftarrow 1$;
7	**while** $(i_2 \leq n-k+2 \ \wedge \ sat = falso)$ **do begin**
8	$a(x_{i_2}) \leftarrow vero$;
9	$i_3 \leftarrow 1$;
	$\ldots$
$3k$	$i_k \leftarrow 1$;
$3k+1$	**while** $(i_k \leq n \ \wedge \ sat = falso)$ **do begin**
$3k+2$	$a(x_{i_k}) \leftarrow vero$;
$3k+3$	$sat \leftarrow f(a(X))$;
$3k+4$	**if** $(sat = falso)$ **then** $a(x_{i_k}) \leftarrow falso$;
$3k+5$	$i_k \leftarrow i_k + 1$;
$3k+6$	**end**
$3k+7$	**if** $(sat = falso)$ **then** $a(x_{i_{k-1}}) \leftarrow falso$;
$3k+8$	$i_{k-1} \leftarrow i_{k-1} + 1$;
$3k+9$	**end**
	$\ldots$
$3k+3+3(k-1)$	**end**
$3k+3+3k-2$	**if** $(sat = falso)$ **then** $a(x_{i_1}) \leftarrow falso$;
$6k+2$	$i_1 \leftarrow i_1 + 1$;
$6k+3$	**if** $(sat = falso \wedge i_1 = n+1)$ **then**
$6k+4$	**for** $(i \leftarrow 1; i \leq n; i \leftarrow i+1)$ **do** $a(x_i) \leftarrow falso$;
$6k+5$	**end**
$6k+6$	**if** $(sat = vero)$ **then Output:** accetta;
$6k+7$	**else Output:** rigetta;

Tabella 0.5: Algoritmo che decide k-SAT.

Quindi, il problema è **NP**-completo.

Problema 4. Si ricordi che **PO** è una sottoclasse di **NPO** e che, affinché un problema sia incluso in **NPO** è necessario che l'insieme delle sue istanze sia decidibile in tempo deterministico polinomiale. Poiché l'insieme delle istanze del problema di ottimizzazione che stiamo considerando è il linguaggio **NP**-completo SAT, il problema di ottimizzazione non è in **PO** (modulo $\mathbf{P} \neq \mathbf{NP}$).

14 Appello del 25 febbraio 2013

Problema 1. Dimostrare che l'insieme dei sistemi lineari di due equazioni in due incognite con coefficienti interi è numerabile.

Problema 2 Si consideri il seguente problema decisionale: dati una funzione booleana f in forma congiuntiva normale ed un intero k, decidere se f è soddisfacibile da una assegnazione di verità che assegna il valore vero ad esattamente k variabili.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrare che l'esistenza di un algoritmo polinomiale che lo decide implica l'esistenza di un algoritmo polinomiale che decide SAT. Cosa si può dedurre da ciò?

Problema 3. Siano L_1 ed L_2 due linguaggi tali che

a) $|L_1 - L_2| = \{a_1, a_2, a_3\}$,

b) $|L_2 - L_1| = \{b_1, b_2\}$,

c) $L_2 \in$ **NPC**.

Dimostrare che, in queste ipotesi, $L_1 \in$ **NPC**.

Problema 4. Sia $\mathscr{G}_{123}$ la classe dei grafi completi pesati tali che, per ogni $G \in \mathscr{G}_{123}$, la funzione peso assegni ad ogni arco di G un peso in $\{1,2,3\}$.
Per quale ragione l'algoritmo di Christofides applicato a grafi $G \in \mathscr{G}_{123}$ non garantisce che il risultato sia una 1-approssimazione della soluzione ottima del problema MIN-TSP del minimo commesso viaggiatore (rispetto all'errore relativo)?
Valutare, dunque, l'errore relativo risultante dall'applicazione dell'algoritmo di Christofides ad un grafo $G \in \mathscr{G}_{123}$.

14.1 Soluzione

Problema 1. Indichiamo con $\mathscr{S}(x,y)$ l'insieme dei sistemi lineari a due variabili con coefficienti interi. Allora un elemento p di $\mathscr{S}(x,y)$ ha la forma

$$p = \begin{cases} a_1x+b_1y=c_1 \\ a_2x+b_2y=c_2 \end{cases} \tag{0.9}$$

Sono possibili numerose soluzioni differenti a questo problema. In questa sede, ne presentiamo una.

Costruiamo una biezione f_S fra l'insieme $\mathscr{S}(x,y)$ e un sottoinsieme delle parole (ossia, stringhe di lunghezza finita) sull'alfabeto $\{0,1,\ldots,9,+,,-,=,x,y,*\}$. analogamente a quanto illustrato nella soluzione del problema 1 proposto all'appello del 15/09/2011: poiché l'insieme delle parole su un alfabeto finito è numerabile, l'esistenza di tale biezione dimostra la numerabilità di $\mathscr{S}(x,y)$.
Allo scopo, sia $\oplus$ l'operatore di concatenazione fra stringhe, sia $\sigma(z)$ la rappresentazione in forma di stringa dell'intero $z \in (Z)$ comprendente il suo segno: ad esempio, $\sigma(-146) = -146$ e $\sigma(44) = +44$.
Allora, l'elemento $p \in \mathscr{S}(x,y)$ descritto in (0.9) corrisponde alla parola

$$\sigma(a_1)\oplus x\oplus\sigma(b_1)\oplus y\oplus = \oplus\sigma(b_1)\oplus *\oplus\sigma(a_2)\oplus x\oplus\sigma(b_2)\oplus y\oplus = \oplus\sigma(c_2).$$

La dimostrazione che la corrispondenza appena descritta fra sistemi e parole è una biezione è analoga a quella presentata per la soluzione del problema citato ed è, pertanto, omessa.

Problema 2. Il problema, che denoteremo, in breve, BSAT (*Bounded* SAT), può essere formalizzato nella maniera seguente:

- $I_{BSAT} = \{\langle X,f,k\rangle : \ f$ è una funzione booleana in forma congintiva normale nelle variabili in X costituita da m clausole $\wedge\ k \in \mathbb{N}\}$.
- $S_{BSAT}(X,f) = \{a : X \to \{vero, falso\}\}$.
- $\pi_{BSAT}(X,f,a) = f(a(X)) \wedge\ [\ |\{x \in X : a(x) = vero\}| = k]$.

Supponiamo che esista un algoritmo deterministico *bSAT* che, con input f, X, e k, decide in tempo polinomiale in $|f|$, $|X|$ e $|k|$ se f è soddisfacibile da una assegnazione di verità che assegna il valore vero ad esattamente k variabili. Allora, potremmo decidere SAT con l'algoritmo (deterministico) in Tabella 0.6.
Indichiamo con $c(m,n,k)$ il polinomio che, in accordo con la nostra ipotesi, limita il costo dell'algoritmo *bSAT* quando il suo input consiste di una funzione costituita da m clausole su un insieme di n variabili booleane e di un intero k; allora, il costo dell'algoritmo in Tabella 0.6 è

$$\sum_{i=0}^{n} c(m,n,i) \leq (n+1)\max\{c(m,n,i) : 0 \leq i \leq n\}.$$

Quindi, anche l'algoritmo in Tabella 0.6 che decide SAT avrebbe costo polinomiale nella dimensione dell'input. Ma, poiché SAT è un problema **NP**-completo, questo è possibile solo se **P=NP**.
Allora, se $\mathbf{P} \neq \mathbf{NP}$, BSAT$\notin$**P**.

Problema 3. Per dimostrare che L_1 è **NP**-completo è necessario dimostrare che $L_1 \in$ **NP** e che $L1$ è completo per **NP**. Per ciò che riguarda l'appartenenza ad **NP**, poiché non abbiamo informazioni sulla struttura di L_1 non abbiamo la possibilità di progettare per esso un algoritmo non deterministico polinomiale. Ricorriamo, allora, ad una riduzione, ossia, dimostriamo l'appartenenza di L_1 ad **NP** mediante una riduzione da L_1 ad L_2. Assumiamo che $L_1, L_2 \subseteq \Sigma^*$ e consideriamo la seguente funzione

$$\forall x \in \Sigma^*,\ f(x) = \begin{cases} x & \text{se } x \notin \{a_1,a_2,a_3,b_1,b_2\} \\ a_1 & \text{se } x \in \{b_1,b_2\} \\ b_1 & \text{se } x \in \{a_1,a_2,a_3\}. \end{cases}$$

Input:	X, $f = c_1 \wedge c_2 \wedge c_m$ con $c_j = \{l_{j_1}, \ldots, l_{j_{h_j}}\}$, per ogni $j = 1, \ldots, m$.
1	$sat \leftarrow falso$;
2	$k \leftarrow 0$;
3	**while** ($k \leq n \ \wedge \ sat = falso$) **do begin**
4	**if** ($bSAT(f, X, k)$ accetta) **then** $sat \leftarrow vero$;
5	**else** $k \leftarrow k + 1$;
7	**if** ($sat = vero$) **then Output**: accetta;
8	**else Output**: rigetta.

Tabella 0.6: Algoritmo che decide SAT.

Il calcolo di f richiede tempo costante, quindi $f \in \mathbf{FP}$. Inoltre, banalmente, $x \in L_1$ se e soltanto se $f(x) \in L_2$. Infatti:

- se $x \in L_1$, allora o $x \in \{a_1, a_2, a_3\}$ e $f(x) = b_1 \in L_2$, oppure $x \in L_1 - \{a_1, a_2, a_3\} \subset L_2$ e $f(x) = x \in L_2$;
- se $x \notin L_1$, allora o $x \in \{b_1, b_2\}$ e $f(x) = a_1 \notin L_2$, oppure $x \in \Sigma^* - (L_1 \cup \{b_1, b_2\}) \subset \Sigma^* - L_2$ e $f(x) = x \notin L_2$.

Questo prova che $L_1 \leq L_2$ e, quindi, $L_1 \in \mathbf{NP}$.

Per mostrare la completezza di L_1 rispetto ad **NP** dobbiamo dimostrare l'esistenza della riduzione inversa, ossia, dobbiamo mostrare che $L_2 \leq L_1$. In effetti, poiché L_1 e L_2 sono linguaggi sullo stesso alfabeto Σ, a questo scopo è sufficiente osservare che la stessa funzione f definita sopra è anche una riduzione da L_2 a L_1: infatti, in maniera analoga a quanto dimostrato sopra, possiamo mostrare che $x \in L_2$ se e soltanto se $f(x) \in L_1$. Questo prova che $L_2 \leq L_1$ e, quindi, che L_1 è completo per **NP**.

Problema 4. L'algoritmo di Christofides garantisce errore relativo 1 nel caso di grafi completi pesati in cui la funzione peso rispetti la disuguaglianza triangolare, ossia, per grafi completi pesati $G = (V, E, w)$ tali che

$$\forall u, v, z \in V : \ w(u, v) \ \leq \ w(v, z) \ + \ w(z, u).$$

Questo, in generale, non vale per i grafi in $\mathscr{G}_{123}$. Sia, infatti, G il grafo completo sui tre nodi a, b e c tale che $w(a, b) = w(b, c) = 1$ e $w(a, c) = 3$: in questo caso, $G \in \mathscr{G}_{123}$ ma in G non è verificata la disuguaglianza triangolare. Tuttavia, consideriamo un qualsiasi algoritmo A che sceglie un tour, ossia, un cammino hamiltoniano, in $G \in \mathscr{G}_{123}$ (ricordiamo che G è un grafo completo e contiene numerosi cicli hamiltoniani): la cosa peggiore che può accadere è che il tour ottimo in G sia costituito da soli archi di peso 1 mentre il tour scelto dall'algoritmo A sia costituito da soli archi di peso 3. In questo caso, il costo del tour ottimo sarebbe n, mentre il costo del tour calcolato dall'algoritmo sarebbe $3n$; allora, per qualunque grafo completo pesato G, indicati con $c^*(G)$ e con $c_A(G)$, rispettivamente, il costo del tour ottimo nel grafo G e il costo del tour in G calcolato dall'algoritmo A, l'errore relativo di A è

$$\frac{c_A(G) - c^*(G)}{c^*(G)} \leq \frac{3n - n}{n} = 2.$$

Poiché A è un qualunque algoritmo che sceglie un tour in G, questo vale anche per l'algoritmo di Christofides.

15 Appello del 1 luglio 2013

Problema 1. Sia Σ un alfabeto (finito) e siano $L_1 \subseteq \Sigma^*$ e $L_2 \subseteq \Sigma^*$ due linguaggi decidibili. Dimostrare che $L_1 \cup L_2$ è decidibile.

Problema 2 Siano $G = (V,E)$ un grafo e b_G la funzione booleana in forma 2-congiuntiva normale sull'insieme X_G di variabili definita nel seguito.

- Per ogni $u \in V$, X_G contiene la variabile booleana x_u; dunque: $X_G = \{x_u : u \in V\}$.
- Per ogni $(u,v) \in E$, b_G contiene la coppia di clausole

$$(x_u \vee x_v) \qquad \text{e} \qquad (\neg x_u \vee \neg x_v).$$

Si consideri, infine, la seguente funzione $f(G)$ che associa ad un grafo un valore in $\{0,1,2\}$

$$f(G) = \begin{cases} 0 & \text{se } b_G \text{ non è soddisfacibile,} \\ 1 & \text{se } b_G \text{ è soddisfacibile e } G \text{ non è 2-colorabile.} \\ 2 & \text{altrimenti.} \end{cases}$$

La funzione appena definita può effettivamente assumere tutti i valori del suo codominio?
Verificare se la funzione $f(G)$ appartiene a **FP**.

Problema 3. Siano L un linguaggio in **NPC** e $a \in L$.
Dimostrare che se **P**≠**NP** allora $L - \{a\} \notin \mathbf{P}$.

Problema 4. Si consideri il seguente problema di ottimizzazione: dati un insieme X di variabili booleane e una funzione booleana soddisfacibile f in forma 2-congiuntiva normale nelle variabili in X, calcolare l'assegnazione $a : X \to \{v,f\}$ che soddisfa f e che minimizza il numero di elementi di X cui è assegnato il valore $v = \textit{vero}$, ossia:

$$f(a(X)) = v \ \wedge \ |\{x \in X : a(x) = v\}| = \min\{|\{x \in X : b(x) = v\}| : b : X \to \{v,f\} \wedge f(b(X)) = v\}.$$

Verificare se detto problema è in **NPO**, dimostrando la propria affermazione.

15.1 Soluzione

Problema 1. Indichiamo con T_1 la macchina di Turing che decide L_1 e con T_2 la macchina di Turing che decide L_2. Utilizzando T_1 e T_2, definiamo ora una nuova macchina di Turing T che decide $L = L_1 \cup L_2$. Indichiamo con Q_1 e Q_2, rispettivamente, l'insieme degli stati di T_1 e l'insieme degli stati di T_2 ed assumiamo (senza perdita di generalità) La macchina Tè definita sull'alfabeto Σ e sull'insieme degli stati $Q = Q_1 \cup Q_2 \cup \{q_{A_T}, q_{R_T}\}$, dove $q_{A_T} \notin Q_1 \cup Q_2$ è lo stato di accettazione di T e $q_{R_T} \notin Q_1 \cup Q_2$ è lo stato di rigetto di T. T utilizza due nastri, su ciascuno dei quali, all'inizio della computazione, è scritto l'input $x \in \Sigma^*$ ed opera come di seguito descritto.

1) T inizia la sua computazione simulando sul nastro 1 la computazione $T_1(x)$: se tale computazione termina nello stato di accettazione di T_1 allora T entra nello stato di accettazione q_{A_T} e termina. Se, invece, $T_1(x)$ non termina nello stato di accettazione allora, poiché L_1 è un linguaggio decidibile ed è deciso da T_1, $T_1(x)$ termina nello stato di rigetto; in questo secondo caso, viene eseguito il passo 2).

2) T prosegue la sua computazione simulando sul nastro 2 la computazione $T_2(x)$: se tale computazione termina nello stato di accettazione di T_2 allora T entra nello stato di accettazione q_{A_T} e termina. Se, invece, $T_2(x)$ non termina nello stato di accettazione allora, poiché L_2 è un linguaggio decidibile ed è deciso da T_2, $T_2(x)$ termina nello stato di rigetto; in questo secondo caso, T entra nello stato di rigetto q_{R_T} e termina.

Quindi, la macchina T termina per ogni input x e, inoltre, termina nello stato di accettazione se e soltanto se $x \in L_1$ oppure $x \in L_2$, ossia, se e soltanto se $x \in L_1 \cup L_2$. Questo prova che L è decidibile.

Problema 2. Osserviamo che una assegnazione di verità a_G per le variabili in X_G corrisponde alla seguente assegnazione di colori c ai nodi di G: per ogni $u \in V$, se $a_G(x_u) = vero$ poniamo $c(u) = 1$, mentre se $a_G(u) = falso$ poniamo $c(u) = 2$. Inoltre, una assegnazione a_G per le variabili in X_G soddisfa la funzione b_G se e soltanto se, per ogni arco $(u,v) \in E$, la variabile corrispondente ad uno dei suoi estremi è vera mentre la variabile corrispondente all'altro suo estremo è falsa, ossia, uno dei nodi in $\{u,v\}$ è colorato 1 e l'altro è colorato 2. In altri termini, b_G è soddisfacibile se e soltanto se G è 2-colorabile. Pertanto, la funzione $f(G)$ non assume mai il valore 1.
Da quanto sopra osservato, deduciamo che $f(G)$ può essere calcolata mediante il seguente algoritmo:

1) calcola la funzione booleana b_G a partire da G;

2) verifica se b_G è soddisfacibile: in caso affermativo $f(G) = 2$, in caso negativo $f(G) = 0$.

La costruzione della funzione b_G al punto 1) richiede tempo polinomiale in $|G|$: infatti, la costruzione dell'insieme X_G richiede tempo $O(|V|)$, e la costruzione dell'insieme delle clausole richiede tempo $O(|E|)$. Inoltre, poiché b_G è in 2CNF, la decisione circa la sua soddisfacibilità richiede tempo polinomiale nella sua dimensione. Pertanto, $f \in \mathbf{FP}$.

Problema 3. Supponiamo, per assurdo, che $L' = L - \{a\} \in \mathbf{P}$: allora esistono una macchina di Turing deterministica T' ed un intero k tali che, per ogni $y \in \Sigma^*$, $T'(y)$ termina in tempo $O(|y|^k)$ ed inoltre $T'(y)$ accetta se e soltanto se $y \in L'$.
Possiamo, allora, utilizzare T' per definire una macchina di Turing deterministica che decide L, il cui comportamento è illustrato dal seguente algoritmo:

input: $x \in \Sigma_1^*$.

1) se $x = a$ allora T accetta;

2) altrimenti se $T'(x) = q_A$ allora T: accetta;

3) altrimenti T rigetta.

Poiché il passo 2) richiede tempo in $O(|x|^k)$, tale algoritmo richiede tempo polinomiale in $|x|$, e questo prova che il problema L appartiene alla classe **P**. MA L è **NP**-completo, dunque deve essere **P**=**NP**, contraddicendo l'ipotesi **P**$\neq$**NP**.

Problema 4. Si ricordi che un problema di ottimizzazione è una quintupla $\langle I, S, \pi, m, \tau \rangle$ e che un problema di ottimizzazione è in **NPO** se:

1) I è un linguaggio in **P**;
2) per ogni $x \in I$, $S(x)$ è un linguaggio in **P**;
3) esiste un polinomio p tale che, per ogni $x \in I$ e per ogni $y \in S(x)$, $|y| \leq p(x)$;
4) π è una funzione in **FP**;
5) m è una funzione in **FP**.

Nel caso del problema in esame, che chiameremo MIN-2SAT, abbiamo che:

1) l'insieme delle istanze è $I_{min-2Sat} = \{\langle X, f \rangle : \langle X, f \rangle \in \text{2SAT}\}$ e quindi decidere se $\langle X, f \rangle \in \text{SAT}$ richiede tempo polinomiale in $|X|$ e in $|f|$;
2) una soluzione possibile y è una assegnazione di verità per X, quindi $y \in S_{min-2Sat}(X, f)$ se e soltanto se $y = \langle y_1, y_2, \ldots, y_{|X|} \rangle$ con $y_i \in \{vero, falso\}$ per ogni $i = 1, \ldots, |X|$;
3) per ogni $\langle X, f \rangle \in I_{min-2Sat}$ e per ogni $y \in S_{min-2Sat}(X, f)$, $|y| = |X|$;
4) per ogni $\langle X, f \rangle \in I_{min-2Sat}$ e per ogni $y = \{y_1, \ldots, y_{|X|}\} \in S_{min-2Sat}(X, f)$, $\pi_{min-2Sat}(X, f, y) = vero$ se e soltanto se sostituendo in f ad ogni occorrenza di x_i il valore y_i e ad ogni occorrenza di $\neg x_i$ il valore negato di y_i, per $i = 1, \ldots, |X|$, f assume il valore *vero*; pertanto, il calcolo di $\pi_{min-2Sat}(X, f, y)$ richiede tempo proporzionale a $|X| \cdot |f|$;
5) per ogni $\langle X, f \rangle \in I_{min-2Sat}$ e per ogni $y = \{y_1, \ldots, y_{|X|}\} \in S_{min-2Sat}(X, f)$ tale che $\pi_{min-2Sat}(X, f, y) = vero$, $m(y) = |\{y_i = vero : \ 1 \leq i \leq |X|\}$; quindi, $m \in$ **FP**.

Questo prova che MIN-2SAT è in **NPO**.

16 Appello del 23 settembre 2013

Problema 1. Sia Σ un alfabeto (finito) e siano $L_1 \subseteq \Sigma^*$ e $L_2 \subseteq \Sigma^*$ due linguaggi accettabili. Dimostrare che $L_1 \cap L_2$ è accettabile.

Problema 2 Si ricordi il problema k-COLORABILITÀ, per ogni costante $k \in \mathbb{N}$, e se ne formalizzi la definizione mediante la tripla $\langle I, S, \pi \rangle$.
Si consideri la seguente funzione f che trasforma un grafo (non orientato) $G = (V, E)$ in un nuovo grafo (non orientato) $f(G) = G' = (V', E')$ tale che

- $V' = V \cup \{x\}$, ove $x \notin V$ (ossia, V' è ottenuto aggiungendo un nuovo elemento a V).
- $E' = E \cup \{(u, x) : u \in V\}$ (ossia, E' contiene tutti gli archi in E e tutti gli archi che connettono x ad un elemento di V).

Verificare se f è una riduzione polinomiale da 3-COLORABILITÀ a 4-COLORABILITÀ dimostrando la propria affermazione.

Problema 3. Dopo averne formalizzato la definizione mediante la tripla $\langle I, S, \pi \rangle$, si studi la complessità computazionale del problema seguente dimostrandone l'appartenenza alla classe **P** oppure la **NP**-completezza: dato un grafo non orientato $G = (V, E)$, decidere se V può essere partizionato in 2 insiemi indipendenti.

Problema 4. Si consideri il problema decisionale seguente: dato un grafo non orientato $G = (V, E)$, decidere se ogni ciclo in G ha lunghezza $< |V| - 1$. Studiare la complessità computazionale di tale problema.
Si ricordi che un ciclo di lunghezza k in un grafo non orientato $G = (V, E)$ è una sequenza $\langle v_1, v_2, \ldots v_k \rangle$ di elementi distinti di V tali che, per $i = 1, \ldots k-1$, $(v_i, v_{i+1}) \in E$ e, inoltre, $(v_k, v_1) \in E$.

16.1 Soluzione

Problema 1. Indichiamo con T_1 la macchina di Turing che accetta L_1 e con T_2 la macchina di Turing che accetta L_2. Utilizzando T_1 e T_2, definiamo ora una nuova macchina di Turing T che accetta $L = L_1 \cap L_2$. Indichiamo con Q_1 e Q_2, rispettivamente, l'insieme degli stati di T_1 e l'insieme degli stati di T_2 ed assumiamo (senza perdita di generalità) $Q_1 \cap Q_2 = \emptyset$. Analogamente, siano q_{01} e q_{02}, rispettivamente, gli stati iniziali di T_1 e T_2 e q_{A1} e q_{A2}, rispettivamente, gli stati finali di accettazione di T_1, e T_2 e q_{R1} e q_{R2}, rispettivamente, gli stati finali di rigetto di T_1 e T_2.
La macchina Tè definita sull'alfabeto Σ e sull'insieme degli stati $Q = Q_1 \cup Q_2 \cup \{q_{A_T}, q_{R_T}\}$, dove $q_{A_T} \notin Q_1 \cup Q_2$ è lo stato di accettazione di T e $q_{R_T} \notin Q_1 \cup Q_2$ è lo stato di rigetto di T; infine, q_{01} è lo stato iniziale di$T$. T utilizza due nastri, su ciascuno dei quali, all'inizio della computazione, è scritto l'input $x \in \Sigma^*$ ed opera come di seguito descritto.

1) T inizia la sua computazione simulando sul nastro 1 la computazione $T_1(x)$: se tale computazione termina nello stato q_{R1} allora T entra nello stato di rigetto q_{R_T} e termina. Se, invece, $T_1(x)$ termina nello stato q_{A1} allora T entra nello stato q_{02} ed esegue il passo 2).

2) T prosegue la sua computazione simulando sul nastro 2 la computazione $T_2(x)$: se tale computazione termina nello stato di accettazione di T_2 allora T entra nello stato di accettazione q_{A_T} e termina. Se, invece, $T_2(x)$ termina nello stato di rigetto, T entra nello stato di rigetto q_{R_T} e termina.

Quindi, la computazione $T(x)$ termina nello stato di accettazione se e soltanto se $x \in L_1 \cap L_2$, dunque T accetta $L_1 \cap L_2$. Osserviamo esplicitamente che nulla si può dire circa l'esito della computazione $T(x)$ per $x \notin L_1 \cap L_2$.

Problema 2. Ricordiamo che il problema k-COLORABILITÀ consiste nel decidere se i nodi di un dato grafo possono essere colorati con k colori in modo tale che gli estremi di nessun arco siano colorati con lo stesso colore. Formalmente,

- $I_{k-col} = \{G = (V,E) : G \text{ è un grafo non orientato}\}$;
- $S_{k-col}(G) = \{c : V \to \{1,2,\ldots,k\}\ \}$;
- $\pi_{k-col}(G,c) = \forall (u,v) \in E[c(u) \neq c(v)]$.

Consideriamo, ora, la funzione f della traccia. Dato un grafo non orientato $G = (V,E)$, calcolare $G' = f(G)$ richiede tempo lineare in $|V|$. Pertanto, $f \in \mathbf{FP}$.
Per dimostrare che f è una riduzione da 3-COLORABILITÀ a 4-COLORABILITÀ resta da far vedere che un grafo $G = (V,E)$ è 3-colorabile se e soltanto se il corrispondente $G' = f(G)$ è 4-colorabile.
Supponiamo, allora che G sia 3-colorabile, ossia, che esista una funzione $c : V \to \{1,2,3\}$ tale che $c(u) \neq c(v)$ per ogni $(u,v) \in E$. Definiamo una 4-colorazione c' per $G' = f(G)$ nella maniera seguente: per ogni $v \in V' = V \cup \{x\}$

$$c'(v) = \begin{cases} c(v) & \text{se } v \in V \\ 4 & \text{altrimenti.} \end{cases}$$

Sia ora $(y,z) \in E'$ un arco di G': se $y \in V$ e $z \in V$ allora $c'(y) = c(y) \neq c(z) = c'(z)$ in quanto c è una colorazione per G; se invece $y \in V$ e $z = x$ allora $c(y) \in \{1,2,3\}$ e $c(z) = c(x) = 4$ (e analogamente per $y = x$ e $z \in V$). Dunque, c' è una colorazione per G'.
Supponiamo, infine, che G' sia 4-colorabile, ossia, che esista una funzione $c' : V \to \{1,2,3,4\}$ tale che $c(y) \neq c(z)$ per ogni $(y,z) \in E'$. Allora, poiché $(x,u) \in E'$ per ogni $u \in V$, $c'(x) \neq c(u)$ per ogni $u \in V$. Questo significa che gli elementi di V sono colorati mediante c' con 3 colori in modo tale che nodi adiacenti hanno colore differente. Dunque, G è 3-colorabile.
Questo completa la prova che f è una riduzione polinomiale da 3-COLORABILITÀ a 4-COLORABILITÀ.

Problema 3. Formalmente, il problema 2IS descritto nella traccia è il seguente

- $I_{2IS} = \{G = (V,E) : G \text{ è un grafo non orientato}\}$;

- $S_{2IS}(G) = \{\langle V_1, V_2\rangle : V_1 \cup V_2 = V \ \wedge \ V_1 \cap V_2 = \emptyset\}$;
- $\pi_{2IS}(G,c) = \forall u,v \in V_1[(u,v) \notin E] \ \wedge \ \forall u,v \in V_1[(u,v) \notin E]$.

Osserviamo, ora, che G è una istanza sì di 2IS se e soltanto se G è 2-colorabile, ossia, i problemi 2IS e 2COLORABILITÀ coincidono. Dunque, 2IS è in **P**.

Problema 4. Si consideri il problema decisionale seguente: dato un grafo non orientato $G = (V,E)$, decidere se ogni ciclo in G ha lunghezza $< |V| - 1$. Studiare la complessità computazionale di tale problema.
Si ricordi che un ciclo di lunghezza k in un grafo non orientato $G = (V,E)$ è una sequenza $\langle v_1, v_2, \ldots v_k\rangle$ di elementi distinti di V tali che, per $i = 1, \ldots k-1$, $(v_i, v_{i+1}) \in E$ e, inoltre, $(v_k, v_1) \in E$.

Indichiamo con NOLONGCYCLE (in breve, *NLC*) il problema in questione e consideriamo il suo complemento NLC^c: decidere se un dato grafo $G = (V,E)$ contiene almeno un ciclo di lunghezza $|V| - 1|$. Il problema NLC^c può essere deciso mediante una serie di invocazioni dell'algoritmo non deterministico che decide il problema CIRCUITO HAMILTINIANO, come illustrato di seguito:

Input: $G = (V,E)$, con $V = \{v_1, v_2, \ldots, v_n\}$.

1) $i \leftarrow 1$;
2) $trovato \leftarrow falso$;
3) **while** ($i \leq n \ \wedge \ trovato = falso$) **do begin**

   (si sceglie quale nodo non considerare per costruire un ciclo di $n-1$ nodi)

   3.1) **if** (il grafo $G_i = (V - \{v_i\}, E_i)$ dove $E_i = E - \{(v, v_i) : v \in V\}$ contiene un ciclo hamiltoniano) **then**
      3.1.1) $esito \leftarrow vero$;

   end
4) **if** (G contiene un ciclo hamiltoniano) **then** $esito \leftarrow vero$;
5) **Output:** $esito$.

Il precedente algoritmo decide NLC^c ed opera in tempo non deterministico polinomiale. Dunque, NLC^c è in **NP** e *NLC* è in *Co*-**NP**.
Infine, il problema NLC^c è completo per **NP**. Per dimostrare questa affermazione consideriamo la seguente riduzione f da CIRCUITO HAMILTONIANO a NLC^c: dato un grafo non orientato $G = (V,E)$ (istanza di CIRCUITO HAMILTONIANO, $G' = f(G)$ è il grafo ottenuto aggiungendo a G un nodo isolato. Banalmente, f è in **FP** e, inoltre, G è una istanza sì di CIRCUITO HAMILTONIANO se e soltanto se G' è una istanza sì di NLC^c. Quindi, NLC^c è **NP**-completo e, in conclusione, *NLC* è *Co*-**NP**-completo.

17 Appello del 3 febbraio 2014

Problema 1. Siano $L_1 \subseteq \Sigma^*$ un linguaggio accettabile e $L_2 \subseteq \Sigma^*$ un linguaggio decidibile. Cosa si può dire circa la accettabilità/decidibilità del linguaggio $L = L_1 \cap L_2$? Motivare la risposta con una opportuna dimostrazione.

Problema 2. Un *grafo bipartito completo* è un grafo $G = (V,E)$ in cui $V = V_1 \cup V_2$, con $V_1 \cap V_2 = \emptyset$, e $E = V_1 \times V_2$. Definiamo il seguente problema decisionale: dato un grafo bipartito completo $G = (V_1 \cup V_2, V_1 \times V_2)$ ed un intero $k \in \mathbb{N}$, decidere se esiste per G un ricoprimento tramite nodi (Vertex Cover) di esattamente k nodi.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Sia $k \in \mathbb{N}$ una costante fissata. Si formalizzi il problema k-COLORABILITÀ mediante la tripla $\langle I, S, \pi \rangle$. Successivamente, si consideri la seguente funzione f: dato un grafo $G = (V,E)$, $f(G)$ è un nuovo grafo $G' = (V', E')$ tale che $V' = V \cup \{a,b,c\}$ (con $a,b,c \notin V$) e $E' = E \cup \{(a,b),(b,c),(c,a)\}$. Verificare se f è una riduzione da 4-COLORABILITÀ a 3-COLORABILITÀ, dimostrando la propria affermazione.

Problema 4. Si consideri il seguente problema decisionale: dato un grafo $G = (V,E)$ ed un intero $k \in \mathbb{N}$, decidere se, per ogni coppia di nodi $(u,v) \in V$, il percorso più lungo fra u e v ha lunghezza $\leq k$. Si studi la complessità computazionale di tale problema.

17.1 Soluzione

Problema 1. Poiché L_1 è accettabile, esiste una macchina di Turing T_1 tale che, per ogni $x \in \Sigma^*$, $T_1(x)$ accetta se e soltanto se $x \in L_1$; inoltre, poiché L_2 è decidibile, esiste una macchina di Turing T_2 tale che $L_2 = L(T_2)$.
Mostriamo ora che L è un linguaggio accettabile. La macchina T che accetta L opera come di seguito descritto. Con input $x \in \Sigma^*$, T esegue, inizialmente, la computazione $T_2(x)$ (che termina sempre): se tale computazione termina nello stato di rigetto, allora $x \notin L_2$ e, dunque, $x \notin L$. Se, invece, $T_2(x)$ termina nello stato di accettazione, allora $x \in L$ se e soltanto se $x \in L_1$: allora, la computazione $T(x)$ prosegue eseguendo la computazione $T_1(x)$ e, se essa termina nello stato di accettazione, allora anche $T(x)$ termina nello stato di accettazione. Quindi, T accetta L. Osserviamo esplicitamente che la computazione $T_1(x)$ potrebbe non terminare e, quindi, l'esistenza della macchina T non permette di affermare che L è decidibile.

In effetti, è anche possibile mostrare un esempio in cui il linguaggio intersezione fra un linguaggio accettabile ed uno decidibile è non decidibile. Infatti, siano L_H il linguaggio che definisce l'halting problem ed $L_2 = \Sigma^*$: ricordiamo che L_H è accettabile ma non decidibile e, inoltre, osserviamo che L_2 è banalmente decidibile (è sufficiente utilizzare una macchina di Turing che accetta ogni input). Allora, $L_1 \cap L_2 = L_H$.

Problema 2. Il problema, che denoteremo, in breve, BC-VC, può essere formalizzato nella maniera seguente:

- $I_{BC-VC} = \{\langle G = (V,E), k\rangle : V = V_1 \cup V_2 \wedge V_1 \cap V_2 = \emptyset \wedge E = V_1 \times V_2 \ \ wedge k \in \mathbb{N}\}$.
- $S_{BC-VC}(X, f) = \{V' \subseteq V\}$.
- $\pi_{k-SAT}(X, f, a) = |V'| = k \wedge \forall (u,v) \in E \ [\ u \in V' \vee v \in V']$.

Osserviamo preliminarmente che, se $k < |V|$, allora, banalmente, G non può contenere un Vertex Cover con un numero di elementi superiore al numero dei suoi nodi.

In secondo luogo, osserviamo che V_1 è in Vertex Cover per G, così come anche V_2 è un vertex cover per G. Inoltre, per ogni nodo $u \in V_1$ e per ogni Vertex Cover V' per G, se *u non* è in V' allora $V_2 \subseteq V'$: infatti, per ogni nodo $v \in V_2$, $(u,v) \in E$ e, quindi, per conprire tale arco, deve essere $v \in V'$. Un ragionamento analogo permette di affermare che, per ogni Vertex cover V' di G, se esiste $v \in V_2 - V'$ allora deve essere $V_1 \subseteq V'$. In conclusione, per ogni Vertex cover V' di G, deve essere $V_1 \subseteq V'$ oppure $V_2 \subseteq V'$ e, quindi, nessun Vertex Cover per G può avere cardinalità minore di $\min\{|V_1|, |V_2|\}$.
Dunque, se $k = \min\{|V_1|, |V_2|\}$ allora G contiene certamente il Vertex Cover richiesto, mentre se $k < \min\{|V_1|, |V_2|\}$ allora G non contiene un Vertex Cover di cardinalità k.
Supponiamo, infine, che sia $k \geq \min\{|V_1|, |V_2|\}$ (con $k \leq |V|$) e che $|V_1| \leq |V_2|$ (il caso $|V_2| < |V_1|$ è analogo): in questo caso un Vertex Cover V' per G di cardinalità *esattamente* k si ottiene inserendo in V' l'insieme V_1 e $k - |V_1|$ nodi di V_2.

Pertanto, una istanza $\langle G = (V_1 \cup V_2, V_1 \times V_2), k\rangle$ di BC-VC è una istanza sì se e soltanto se $\min\{|V_1|, |V_2|\} \leq k \leq |V|$. Questo prova che il problema è in **P**.

Problema 3. Il problema, che denoteremo, in breve, k-COL (mettendo in evidenza il ruolo di valore costante giocato da k), può essere formalizzato nella maniera seguente:

- $I_{k-COL} = \{G = (V,E) : G$ è un grafo non orientato $\}$.
- $S_{k-COL}(G) = \{V_1, \ldots, V_k : \forall i = 1, \ldots, k \ [\ V_i \subseteq V\ ]\ \}$.
- $\pi_{k-COL}(G, V_1, \ldots, V_k) = \forall i = 1, \ldots, k \ \forall u, v \in V_i \ [\ (u,v) \notin E\ ] \wedge \cup_{i=1}^{k} V_i = V$.

Sia G un grafo istanza di 4-COL; la funzione f applicata a G trasforma il grafo G in un nuovo grafo G' costituito da due componenti connesse, G_1 e G_2: una di tali componenti, diciamo G_1 è il grafo G stesso, l'altra componente, diciamo G_2 è il grafo completo sui tre nodi a, b, c.

È evidente che il calcolo di $f(G)$ richiede tempo costante. Per verificare se f è una riduzione da 4-COL a 3-COL, occorre verificare se è vero che:

G è 4-colorabile se e soltanto se $f(G)$ è 3-colorabile.

Supponiamo che G sia 4-colorabile: allora, la componente G_1 di $f(G)$ è anch'essa 4-colorabile. Questo, però, non significa che $G_1 = G$ sia 3-colorabile: a titolo di esempio, consideramo come grafo G il grafo completo di 4 nodi (che è 4-colorabile, ma non 3-colorabile). Quindi dalla 4-colorabilità di G non possiamo dedurre alcunché circa la 3-colorabilità di $f(G)$.

In conclusione, f non è una riduzione da 4-COL a 3-COL.

Problema 4. Sia $G = (V,E)$ un grafo non orientato e $k \in \mathbb{N}$. La coppia $\langle G,k \rangle$ è una istanza no del problema in esame se e soltanto se

esiste in G una coppia di nodi $u,v \in V$ tali che G contiene un percorso da u a v di lunghezza almeno k.

Ossia, $\langle G,k \rangle$ è una istanza no del problema in esame se e soltanto se $\langle G,k \rangle$ è una istanza sì del problema LONGEST PATH. Dunque, il problema in esame è il complemento di LONGEST PATH. Dalla **NP**-completezza del problema LONGEST PATH segue che il problema in esame è Co**NP**-completo.

18 Appello del 17 febbraio 2014

Problema 1. Siano $L_1 \subseteq \Sigma^*$ e $L_2 \subseteq \Sigma^*$ due linguaggi decidibili e sia

$$L = \{x \in L_1 : \exists k \in \mathbb{N} \, [\, |x| = 2k \,]\} \cup \{x \in L_2 : \exists k \in \mathbb{N} \, [\, |x| = 2k+1 \,]\}.$$

Dimostrare che il linguaggio L è decidibile.

Problema 2. Si ricordino le definizioni dei problemi SHORTEST PATH e LONGEST PATH.
Sia $G = (V,E)$ un grafo non orientato e $D \in \mathbb{N}$. Diciamo che G ha *diametro* D se

- esiste una coppia di nodi $u_0, v_0 \in V$ tali che D è la lunghezza del cammino più breve che collega in G u_0 e v_0, e inoltre
- non esiste alcuna coppia di nodi in G tali che la lunghezza del cammino più breve che li collega è maggiore di D.

Si consideri il seguente problema decisionale: dati un grafo $G = (V,E)$ ed un intero $D \in \mathbb{N}$, decidere se G ha diametro D.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si consideri il seguente problema decisionale: dati un grafo (non orientato) $G = (V,E)$ ed un intero k, decidere se i nodi di G possono essere colorati con almeno k colori in accordo alle seguenti regole:

a) per ogni $i, j = 1, \ldots, k-1$ tali che $i \neq j$, se due nodi hanno colore, rispettivamente, i e j allora essi non sono collegati da un arco;

b) nessun vincolo è posto per nodi colorati con il colore k.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 4. Si consideri il problema di ottimizzazione seguente: dato un grafo non orientato completo $G = (V,E)$, calcolare il Vertex Cover per G di cardinalitità minima.
Dimostrare se il suddetto problema è contenuto nella classe **PO**.

18.1 Soluzione

Problema 1. Poiché L_1 è decidibile, esiste una macchina di Turing T_1 tale che$L_1 = L(T_1)$; inoltre, poiché anche L_2 è decidibile, esiste una macchina di Turing T_2 tale che $L_2 = L(T_2)$. Assumiamo, senza perdita di generalità, che T_1 e T_2 siano macchine dotate di un solo nastro e che l'insieme degli stati di T_1 e di T_2 siano disgiunti. Indichiamo con q_{0_1} e q_{0_2}, rispettivamente, lo stato iniziale di T_1 e lo stato iniziale di T_2. Analogamente, indichiamo con $q_{A_1}, q_{A_2}, q_{R_1}, q_{R_2}$ gli stati finali (di accettazione e di rigetto) di T_1 e di T_2.
Mostriamo ora che L è un linguaggio decidibile. La macchina T che decide L utilizza un solo nastro e quattro nuovi stati: lo stato iniziale q_0, uno stato intermedio q_1, lo stato finale di accettazione q_A e lo stato finale di rigetto q_R. Con input $x \in \Sigma^*$, T esegue, inizialmente, una computazione che le permette di decidere se x ha lunghezza pari o dispari:

- se nello stato q_0 legge un simbolo diverso da $\Box$ riscrive tale simbolo, entra nello stato q_1 e muove la testina a destra di una posizione;
- se nello stato q_1 legge un simbolo diverso da $\Box$ riscrive tale simbolo, entra nello stato q_0 e muove la testina a destra di una posizione.

Quando la macchina incontra il simbolo $\Box$ ha terminato la scansione dell'input: se si trova nello stato q_0 allora l'input ha lunghezza pari e, per decidere l'appartenenza a L, è necessario decidere l'appartenenza ad L_1 mentre se si trova nello stato q_1 allora l'input ha lunghezza dispari e, per decidere l'appartenenza a L, è necessario decidere l'appartenenza ad L_2. Dunque, quando la macchina incontra il simbolo $\Box$

- se è nello stato q_0 muove la testina a sinistra fino a riportarla sul primo simbolo di x ed entra nello stato q_{0_1},
- se è nello stato q_1 muove la testina a sinistra fino a riportarla sul primo simbolo di x ed entra nello stato q_{0_2}.

Infine, se T entra nello stato q_{A_1} oppure nello stato q_{A_2} allora esegue un'ultima istruzione che la porta nello stato q_A, altrimenti se T entra nello stato q_{R_1} oppure nello stato q_{R_2} allora esegue un'ultima istruzione che la porta nello stato q_R.

Problema 2. Il problema, che denoteremo, in breve, DIAM, può essere formalizzato nella maniera seguente:

- $I_{DIAM} = \{\langle G = (V,E), D\rangle : \ G$ è un grafo non orientato e $k \in \mathbb{N}\ \}$.
- $S_{DIAM}(G,D) = \{P = \{p_{uv} : u,v \in V \wedge u \neq v \wedge p_{uv}$ è un percorso in G fra u e $v\}\}$ (ossia, una soluzione ammissibile P è un insieme di percorsi, ciascuno dei quali connette una diversa coppia di nodi, tale che, per ogni coppia di nodi distinti in V esiste in P un percorso che li connette).
- $\pi_{DIAM}(G,D,P) = \forall p \in P\ [\ |p| \leq D\ ] \wedge \exists \overline{p} \in P\ [\ |\overline{p}| = D\ ]$, ove $|p|$ indica il numero di archi costituenti p.

Osserviamo, preliminarmente, che ogni soluzione possibile $P \in S_{DIAM}(G,D)$ è costituita da un numero di percorsi quadratico nel numero di nodi di G, ossia, $|P| \in O(|V|^2)$.
Sia ora P_{sp} una soluzione possibile in $S_{DIAM}(G,D)$ costituita da soli shortest paths che connettono ogni coppia di nodi:

$$P_{sp} = \{p_{uv} : u,v \in V \wedge u \neq v \wedge p_{uv} \text{ è uno shortest path in } G \text{ fra } u \text{ e } v\}\}.$$

Osserviamo, ora, che $\langle G = (V,E), D\rangle$ è una istanza sì di DIAM se e soltanto se $\pi_{DIAM}(G,D,P_{sp})$ è vero, ossia, se e soltanto se P_{sp} è una soluzione effettiva.
Poiché è possibile calcolare uno shortest path fra una coppia di nodi in tempo $O(|V|^2)$ e, come osservato in precedenza, $|P_{sp}| \in O(|V|^2)$, è possibile calcolare P_{sp} in tempo $O(|V|^4)$. Inoltre, per verificare se $\pi_{DIAM}(G,D,P_{sp})$ ha valore vero è sufficiente confrontare la lunghezza di ogni elemento di P_{sp} con D, e questo richiede tempo polinomiale in $|P_{sp}| \cdot |D| \in O(|V|^2 \cdot |D|)$, ossia, polinomiale nella dimensione dell'istanza.
Questo prova che il problema è in **P**.

Problema 3. Il problema, che denoteremo, in breve, SC (SPECIAL COLORABILITY), può essere formalizzato nella maniera seguente:

- $I_{SC} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato } \}$.
- $S_{SC}(G,k) = \{c : V \to \{1,\ldots,k\}\ \}$.
- $\pi_{SC}(G,k,c) = \forall u,v \in V\ [\ c(u) < k \wedge c(v) < k \wedge c(u) \neq c(v) \Rightarrow\ (u,v) \notin E\ ]$.

Riduciamo problema INDEPENDENT SET (IS) al problema SC. Allo scopo, consideriamo una istanza $\langle G = (V,E), k\rangle$ di IS: l'istanza corrispondente di SC è $\langle G' = (V',E'), k+1\rangle$ in cui, semplicemente, viene incrementato il valore numerico e viene aggiunto un nodo all'insieme V collegandolo con tutti gli elementi di V: ossia, $V' = V \cup \{x\}$ (con $x \notin V$) e $E' = E \cup \{(x,u) : u \in V\}$. Mostriamo ora che $\langle G = (V,E), k\rangle$ è una istanza sì per IS se e soltanto se $\langle G' = (V',E), k+1\rangle \in$ è una istanza sì per SC.

- Se G contiene un insieme indipendente I di k nodi, allora coloriamo in G' ciascun elemento di I con uno dei primi k colori (nodi distinti avranno colori distinti) ed i nodi in $V' - I$ con il colore k. Poiché $x \notin V$, allora $x \in V' - I$, ossia, V' è non vuoto, e, dunque, tutti i $k+1$ colori sono effettivamente utilizzati. Tale colorazione rispetta i vincoli di SC e, quindi, $\langle G = (V,E), k\rangle$ è una istanza sì per SC.
- Se G non contiene un insieme indipendente di k nodi, allora non è possibile trovare in G k nodi distinti cui assegnare i primi k colori; inoltre, poiché x è collegato a tutti i nodi in V, x non può essere colorato con uno dei primi k colori. Quindi, $\langle G = (V,E), k\rangle$ è una istanza no per SC.

Quindi, il problema è **NP**-completo.

Problema 4. Chiamiamo MIN-CLIQUE VERTEX COVER il problema in esame.
Si ricordi che **PO** è una sottoclasse di **NPO**. Poiché MIN-CLIQUE VERTEX COVER è il problema MIN-VERTEX COVER ristretto ad una particolare classe di istanze, e poiché il problema MIN-VERTEX COVER è noto essere in **NP**, per provare l'apparteneza di MIN-CLIQUE VERTEX COVER ad **NPO** rimane da verificare l'insieme delle sue istanze sia decidibile in tempo deterministico polinomiale. Per decidere se un grafo $G = (V,E)$ è completo è sufficiente considerare tutte le coppie di suoi nodi e verificare se ciascuna di esse è collegata da un arco. Poiché il numero di coppie di nodi di G è $\frac{|V|(|V|-1)}{2}$ e poiché verificare se una coppia di nodi è collegata da un arco richiede tempo deterministico in $O(|E|)$, decidere se il grafo G è istanza di MIN-CLIQUE VERTEX COVER richiede tempo deterministico polinomiale. Dunque, MIN-CLIQUE VERTEX COVER è in **NPO**.
Osserviamo, ora, che ogni Vertex Cover di un grafo completo $G = (V,E)$ contiene almeno $|V!| - 1$ nodi e che, quindi, un Vertex Cover di G di cardinalità minima ha cardinalità $|V| - 1$. Infine, poiché ogni sottoinsieme V' di V di cardinalità $|V| - 1$ è un Vertex Cover per G, calcolare un Vertex Cover di cardinalità minima di un grafo completo richiede tempo (deterministico) polinomiale nella dimensione dell'istanza. Quindi, il problema MIN-CLIQUE VERTEX COVER è in **PO**.

19 Appello del 9 giugno 2014

Problema 1. Sia T_Σ una macchina di Turing definita sull'alfabeto $\Sigma = \{a,b,c,d\}$. Derivare da T_Σ una macchina T_B, equivalente a T_Σ, che opera sull'alfabeto $B = \{0,1\}$.

Nota: non viene richiesta la prova formale di equivalenza delle due macchine, è sufficiente limitarsi ad osservazioni intuitive.

Problema 2. Sia $k \in \mathbb{N}$ un valore fissato; si consideri il seguente problema decisionale: dati un insieme X di variabili booleane e una formula booleana f in forma 2-congiuntiva normale sull'insieme X, decidere se esiste un sottoinsieme X' di X di cardinalità k tale che, per ogni assegnazione di verità $b : X' \to \{vero, falso\}$ agli elementi di X', esiste una assegnazione di verità $c : X - X' \to \{vero, falso\}$ agli elementi di X tale che l'assegnazione di verità seguente

$$a(x) = \begin{cases} b(x) & \text{se } x \in X' \\ c(x) & \text{se } x \in X - X' \end{cases}$$

soddisfa f.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si consideri il seguente problema decisionale: dati un grafo (non orientato) $G = (V,E)$ ed un intero k, decidere se i nodi di G possono essere colorati con esattamente 2 colori in accordo alle seguenti regole:

a) ogni nodo colorato con il colore 1 è adiacente solo a nodi colorati con il colore 2;

b) il numero di nodi colorati con il colore 1 è almeno k.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 4. Si consideri la stessa regola a) di colorazione dei nodi di un grafo definita al Problema 3. In relazione ad essa, si consideri il seguente problema di ottimizzazione: dato un grafo (non orientato) $G = (V,E)$, calcolare una colorazione dei suoi nodi in accordo alla regola a) definita al Problema 3) che minimizzi la cardinalità dell'insieme dei nodi colorati colore 2.

Dimostrare se tale problema è in **PO** o in **APX** o se esso è non approssimabile.

19.1 Soluzione

Problema 1. Data l'equivalenza fra macchine di Turing ad un nastro e macchine di Turing che utilizzano più nastri, senza perdita di generalità possiamo assumere che T_Σ sia una macchina ad un nastro.
Innanzi tutto, fissiamo una codifica $\rho : \{a,b,c,d,\Box\} \to \{0,1,\Box\}^2$ dei simboli dell'alfabeto di lavoro $\Sigma \cup \Box$ di T_Σ nell'alfabeto di lavoro $B \cup \Box$ di T_B: scegliamo

$$\rho(a) = 00 \qquad \rho(b) = 01 \qquad \rho(c) = 10 \qquad \rho(d) = 11 \qquad \rho(\Box) = \Box\Box.$$

Codifichiamo, inoltre ogni simbolo $\Box$ eventualmente presente, o che scriveremo, all'interno della stringa input di T_Σ mediante una coppia di caratteri $\Box$: ossia, il blank viene rappresentato in T_B nella forma $\Box\Box$.

Ciò premesso, presentiamo due soluzioni differenti al problema in questione, la prima delle quali definisce una macchina T_B a due nastri a testine solidali, la seconda una macchina ad un solo nastro.

Soluzione 1. La macchina T_B proposta in questa soluzione è dotata di due nastri, N_1 ed N_2 sui quali vengono scritti, rispettivamente, il primo simbolo e il secondo simbolo della codifica binaria sopra descritta degli elementi di $\Sigma \cup \{\Box\}$ che costituiscono l'input: ad esempio, se l'i-esimo carattere dell'input di T_Σ è il carattere 'c' (ossia, la cella i-esima del nastro di T_Σ al tempo $t = 0$ contiene il carattere 'c'), allora nella cella $N_1[i]$ viene scritto il simbolo 1 e nella cella $N_2[i]$ viene scritto il simbolo 0. Assumiamo, quindi, che, all'istante iniziale, la codifica dell'input di T_Σ sia scritta sui due nastri di T_B in accordo alla regola appena descritta.

Mostriamo, ora, come derivare le quintuple di T_B da quelle di T_Σ; cominciamo con l'osservare che, per costruzione, se all'istante iniziale la testina di T_Σ legge un carattere $\sigma \in \Sigma$ allora, all'istante iniziale, le due testine di T_B leggono i due bit $b_1(\sigma)$ e $b_2(\sigma)$ corrispondenti alla codifica di σ. Allora, per ogni quintupla di T_Σ del tipo $\langle q_0, \sigma, \tau, q_2, m\rangle$, la macchina T_B contiene la quintupla

$$\langle q_0, (b_1(\sigma), b_2(\sigma)), (b_1(\tau), b_2(\tau)), q_2, m\rangle.$$

Intuitivamente, l'osservazione appena fatta può essere generalizzata (la dimostrazione formale è non richiesta): se all'istante t la testina di T_Σ legge un carattere $\sigma \in \Sigma$ allora, all'istante t, le due testine di T_B leggono i due bit $b_1(\sigma)$ e $b_2(\sigma)$ corrispondenti alla codifica di σ. Allora, per ogni quintupla di T_Σ del tipo $\langle q_1, \sigma, \tau, q_2, m\rangle$, la macchina T_B contiene la quintupla

$$\langle q_1, (b_1(\sigma), b_2(\sigma)), (b_1(\tau), b_2(\tau)), q_2, m\rangle.$$

Soluzione 2. La macchina T_B proposta in questa soluzione è dotata di un solo nastro sul quale viene scritta inizialmente la codifica binaria dell'input di T_Σ.
Consideriamo, ora, una quintupla $\langle q_1, s_1, s_2, q_2, m\rangle$ di T_Σ, con $s_1 \in \Sigma$ e $s_2 \in \Sigma \cup \{\Box\}$, e trasformiamola in una serie di quintuple di T_B. Mostriamo questa trasformazione nel caso particolare in cui $s_1 = a$, $s_2 = b$ e $m = S$ (gli altri casi sono analoghi, ricordando che il simbolo $\Box$ di T_Σ è codificato in T_B mediante $\Box\Box$):

$$\langle q_1, 0, 0, q_1^0, D\rangle, \qquad \langle q_1^0, 0, 1, q_2^0, S\rangle, \qquad \langle q_2^0, 0, 0, q_2^{sin,1}, S\rangle, \qquad \forall x \in \{0,1\}\ \langle q_2^{sin,1}, x, x, q_2, S\rangle$$

La prima quintupla sopra, a partire dallo stato q_1 verifica che il carattere letto possa essere il primo caratere della codifica di a: se questo è vero, entra nello stato q_1^0 (che tiene traccia dello stato di partenza q_1 e del carattere letto 0) e si muove a destra per controllare se il carattere successivo è proprio il secondo carattere della codifica di a; in tal caso, la seconda quintupla sopra inizia l'esecuzione della quintupla $\langle q_1, s_1, s_2, q_2, m\rangle$ di T_Σ scrivendo il secondo carattere di b (1), entrando nello stato q_2^0 (che tiene traccia del primo carattere di b che deve essere scritto nella cella immediatamente a sinistra e dello stato in cui dovrà entrare al termine della esecuzione della quintupla) e muovendosi a sinistra. La terza quintupla sopra, scrive il carattere 0 (il primo carattere della codifica di b, specificato all'interno dello stato attuale q_2^0) e si prepara a spostare la testina a sinistra di due posizioni (perché il movimento della quintupla

di T_Σ è S) entrando nello stato $q_2^{sin,1}$ e spostandosi a sinistra. Infine, la quarta quintupla, indipendentemente dal carattere letto, sposta la testina a sinistra ed entra nello stato q_2, predisponendosi a simulare un'altra quintupla di T_Σ.

Consideriamo, infine, una quintupla $\langle q_3, \Box, s, q_4, m\rangle$ di T_Σ, con $s \in \Sigma \cup \{\Box\}$, e trasformiamola in una serie di quintuple di T_B. Mostriamo questa trasformazione nel caso particolare in cui $s_2 = c$ e $m = D$ (gli altri casi sono analoghi):

$$\langle q_3, \Box, \Box, q_3^{\Box}, D\rangle, \qquad \langle q_3^{\Box}, \Box, 0, q_4^1, S\rangle, \qquad \langle q_4^1, \Box, 1, q_4^{des,1}, D\rangle, \qquad \forall x \in \{0,1\}\ \langle q_4^{des,1}, x, x, q_4, D\rangle$$

La spiegazione di questo ultimo gruppo di quintuple è analoga a quella del gruppo precedente.

Per affermare che T_B simula T_Σ, consideriamo la computazione $T_\Sigma(x)$, per una qualsiasi parola $x \in \Sigma^*$, e la computazione $T_B(\rho(x)$, dove $\rho(x)$ è la codifica di x mediante ρ. Osserviamo, allora, che, per come abbiamo costruito le quintuple di T_B, se ad un dato istante $t \geq 1$ della computazione $T_\Sigma(x)$[1] viene eseguita la quintupla $\langle q_1, s_1, s_2, q_2, m\rangle$ e che all'istante $t+1$ T_Σ si trova nello stato q_2 e la sua testina è posizionata su una cella che contiene il carattere $s \in \Sigma \cup \{\Box\}$, allora all'istante $4t$ della computazione $T_B(x)$ inizia l'esecuzione delle quattro quintuple corrispondenti descritte sopra e all'istante $4t+4$ T_B si trova nello stato q_2 e la sua testina è posizionata su una cella che contiene il primo carattere della codifica di $\rho(s)$ di s.

Una dimostrazione formale di quanto affermato sopra richiede un semplice ragionamento induttivo che, comunque, non era necessario ai fini dell'esame.

Problema 2. Chiamiamo k-SUBSET 2SAT (in breve, k-S2SAT) il problema in questione, che può essere formalizzato come di seguito descritto:

- $I_{k-S2SAT} = \{\langle X, f\rangle : X$ è un insieme di variabili booleane e f è una formula in forma 2-CNF nelle variabili in $X\}$;
- $S_{k-S2SAT}(X, f) = \{X' \subseteq X : |X'| = k\}$;
- $\pi_{k-S2SAT}(X, f, X') = \forall b : X' \to \{vero, falso\} \exists c : X - X' \to \{vero, falso\}\ [\ f(b(X'), c(X - X'))\ ]$.

L'algoritmo in Tabella 0.7 costruisce l'insieme $\mathscr{P}$ delle soluzioni possibili (linea 1) e, per ciascuna di esse, verifica se il predicato $\pi_{k-S2SAT}$ è soddisfatto (ciclo **while** alle linee 3-14). In particolare, il ciclo **while** interno (linee 8-13) verifica se, per ogni assegnazione di verità b alle variabili in X', esiste una assegnazione di verità c per le variabili in $X - X'$ che soddisfa la formula ottenuta sostituendo in f alle variabili in X' il valore di verità assegnato ad esse da b (tale formula è quella calcolata alla linea 11): la verifica dell'esistenza della assegnazione c avviene mediante invocazione dell'algoritmo che decide 2SAT (linea 12). Se l'assegnazione c non esiste, allora il sottoinsieme X' considerato nella attuale iterazione del ciclo **while** esterno non è la soluzione cercata, ed una nuova iterazione ha inizio.

Per quanto riguarda la complessità dell'algoritmo in Tabella 0.7, osserviamo che

1) il ciclo **while** esterno viene ripetuto al più $|\mathscr{P}| \in O(|X|^k)$ volte,
2) il ciclo **while** interno viene ripetuto al più 2^k volte (il numero di assegnazioni di verità ad un insieme di k variabili booleane),
3) il costo di ogni iterazione del ciclo **while** interno è dominato dal test alla linea 12, che richiede tempo polinomiale nella dimensione dell'istanza.

In definitiva, poiché k è una costante, l'algoritmo richiede tempo polinomiale e, dunque, k-S2SAT $\in \mathbf{P}$.

Problema 3. Mostriamo di seguito la formalizzazione del problema in questione, che indicheremo, in breve, $\{1,2\}-$1-COL:

[1] Assumiamo che la computazione inizi al tempo $t = 1$.

Input: insieme X di variabili booleane, funzione booleana f, in 2CNF, sulle variabili in X

```
1   𝒫 ← { Y ⊆ X : |Y| = k };
2   trovato ← falso;
        // la variabile trovato verifica se l'insieme X' è stato trovato
3   while (𝒫 ≠ ∅ ∧ trovato = falso) do begin
4       scegli X' ∈ 𝒫;
5       𝒫 ← 𝒫 − {X'};
6       trovato ← vero;
            // con l'assegnazione sopra, supponiamo che l'insieme X'
              che abbiamo scelto al punto 4 sia parte della soluzione effettiva
7       ℬ ← {b : X' → {vero, falso} };
8       while (ℬ ≠ ∅ ∧ trovato = vero) do begin
9           scegli b ∈ ℬ;
10          ℬ ← ℬ − {b};
11          f' ← f(b(X'));
                // ossia, f' viene ottenuta sostituendo in f le variabili in X'
                  con i valori che esse ottengono mediante l'assegnazione b
12          if (f' ∉ 2SAT) then trovato ← falso;
13      end
14  end
15  Output: trovato
```

Tabella 0.7: Algoritmo che decide se $\langle X, f \rangle \in k$-S2SAT.

- $I_{\{1,2\}-1-COL} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{\{1,2\}-1-COL}(G,k) = \{V' \subseteq V\}$;
- $\pi_{\{1,2\}-1-COL}(G,k,V') = |V'| \geq k \ \wedge \ \forall (u,v) \in E \ [\ u \notin V' \vee v \notin V'\ ]$,

in cui V' è l'insieme dei nodi colorati con il colore 1.
A questo punto, è sufficiente osservare che il problema $\{1,2\}-1$-COL coincide con il problema INDEPENDENT SET per provarne la **NP**-completezza.

Problema 4. Si osservi che, dato un grafo $G = (V,E)$, l'insieme dei nodi di G colorati colorati con il colore 2 (in accordo alle regole definite al precedente problema) sono un Vertex Cover per G; di conseguenza, minimizzare il numero di tali nodi equivale a ricercare un MIN-VERTEX COVER per G. Il problema è, dunque, in **NPO**-**PO** (modulo **P**$\neq$ **NP**) ed in **APX**.

Input:	$G = (V,E)$.
1	$E' \leftarrow \emptyset$;
2	**while** $(E \neq \emptyset)$ **do begin**
3	scegli $(u,v) \in E$;
4	$E' \leftarrow E' \cup \{(u,v)\}$;
5	$E \leftarrow E - \{(u,v)\} - \{(u,z) : (u,z) \in E\} - \{(v,z) : (v,z) \in E\}$;
	elimina da E l'arco (u,v) *e tutti gli archi coperti da* (u,v)
6	**end**
7	**Output:** E';

Tabella 0.8: Algoritmo che calcola un EDGE COVER di un grafo

20 Appello del 22 settembre 2014

Problema 1. Siano $L_1 \subseteq \Sigma^*$ un linguaggio decidibile e $L_2 \subseteq \Sigma^*$ un linguaggio accettabile. Cosa si può dedurre circa le proprietà di accettabilità/decidibilità di $L = L_1 - L_2^c$?
(Si ricordi che L_2^c è il linguaggio complemento di L_2).

Problema 2. Si ricordi la definizione di Vertex Cover di un grafo e si consideri il seguente problema decisionale: dati un grafo $G = (V,E)$ ed un intero k, decidere se esiste un Vertex Cover V' per G tale che $k \leq |V'| \leq |V|$. Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si ricordi il problema decisionale DOMINATING SET: dati un grafo (non orientato) $G = (V,E)$ ed un intero k, decidere se esiste $D \subseteq V$ tale che $|D| \leq k$ e, per ogni $u \in V - D$, esiste $v \in D$ tale che $(u,v) \in E$.

Si consideri il problema decisionale EDGE COVER seguente: dati un grafo (non orientato) $G = (V,E)$ ed un intero k, decidere se esiste $E' \subseteq E$ tale che $|E'| \leq k$ e, per ogni $(u,v) \in E$, esiste $z \in V$ tale che $(u,z) \in E'$ oppure $(v,z) \in E'$. Dopo aver formalizzato la definizione dei suddetti problemi mediante la tripla $\langle I, S, \pi \rangle$, si consideri la trasformazione di istanze di EDGE COVER in istanze di DOMINATING SET mediante la funzione $f(G,k) = \langle \overline{G} = (E, \overline{E}), k \rangle$ in cui $(e_1, e_2) \in \overline{E}$ se e solo se e_1 ed e_2 hanno in G un estremo in comune. Si verifichi se detta trasformazione è una riduzione polinomiale da EDGE COVER a DOMINATING SET e si verifichi se essa è sufficiente a provare che EDGE COVER è **NP**-completo.

Problema 4. Si consideri il problema di ottimizzazione MIN-EDGE COVER corrispondente al problema decisionale EDGE COVER descritto al Problema 2. Si chiede di analizzare le prestazioni dell'algoritmo in Tabella 0.8 per verificare se esso calcola un soluzione ottima per MIN-EDGE COVER, oppure una soluzione approssimata (e, in questo caso, si chiede la valutazione dell'errore), oppure se la soluzione da esso calcolata non garantisce alcun limite sull'errore.
Suggerimento: si ricordi l'analisi dell'algoritmo approssimante per il problema MIN-VERTEX COVER (del tutto simile a quello in Tabella 0.8).

20.1 Soluzione

Problema 1. Si osservi che $L = L_1 - L_2^c = L_1 \cap L_2$. Definiamo la macchina di Turing T che, con input $x \in \Sigma^*$, opera come segue:

1) verifica se $x \in L_1$: se $x \notin L_1$ allora rigetta, altrimenti, se $x \in L_1$, esegue il passo 2);

2) verifica se $x \in L_2$: se $x \in L_2$ allora accetta, se $x \notin L_2$ allora rigetta.

Poiché L_1 è un linguaggio decidibile, il passo 1) termina per ogni $x \in \Sigma^*$. Poiché L_2 è un linguaggio accettabile, il passo 2) temina per ogni $x \in L_2$ (e nulla si può dire se $x \notin L_2$). Quindi, se $x \in L_1 \cap L_2 = L$ la computazione $T(x)$ termina nello stato di accettazione (e nulla si può dire se $x \notin L$), ossia, L è un linguaggio accettabile.

Problema 2. Chiamiamo LARGE VERTEX SET (in breve, LVC) il problema in questione, che può essere formalizzato come di seguito descritto:

- $I_{LVC} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{LVC}(G,k) = \{V' \subseteq V\}$;
- $\pi_{LVC}(G,k,V') = |V'| \geq k \ \wedge\ [\ \forall (u,v) \in E : u \in V' \ \vee\ v \in V'\ ]$.

È ora sufficiente osservare che, per ogni grafo $G = (V,E)$, l'intero insieme V dei nodi è un Vertex Cover per G; quindi, $\langle G = (V,E), k\rangle$ è una istanza sì per LVC se e soltanto se $|V'| \geq k$. In conclusione, il problema è, banalmente, in **P**.

Problema 3. Mostriamo di seguito la formalizzazione dei due problemi in questione, che indicheremo, in breve, rispettivamente, DS e EC:

- $I_{DS} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{DS}(G,k) = \{D \subseteq V\}$;
- $\pi_{DS}(G,k,D) = |D| \geq k \ \wedge\ \forall u \in V - D\ [\ \exists v \in D : (u,v) \in E\ ]$
- $I_{EC} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{EC}(G,k) = \{E' \subseteq E\}$;
- $\pi_{EC}(G,k,E') = |E'| \geq k \ \wedge\ \forall (u,v) \in E - E'\ [\ \exists z \in V : (u,z) \in E' \ \vee\ (v,z) \in E'\ ]$

Sia ora $\langle G = (V,E), k\rangle \in I_{EC}$ e sia $f(G,k) = \langle \overline{G} = (\overline{V}, \overline{E}), k\rangle$.
Se $\langle G = (V,E), k\rangle \in I_{EC}$ è una istanza sì per EC allora esiste $E' \subseteq E$, con $|E'| \leq k$, tale che, per ogni $(u,v) \in E - E'$, esiste $z \in V$ tale che $(u,z) \in E'$ o $(v,z) \in E'$. Definiamo, allora, $D = \{e \in \overline{V} = E : e \in E'\}$ e dimostriamo che esso è un Dominating set per $\overline{G}$. Sia $e \in \overline{V} - D$ con $e = (u,v)$; allora, per definizione di D, $e \in E - E'$ e quindi, poiché E' è un Edge Cover per G, esiste un arco $e' \in E'$ che è incidente su u oppure su v. Allora, $e' \in D$ e $(e,e') \in \overline{E}$. Questo prova che D è un Dominating Set per $\overline{G}$ e, poiché $|D| = |E'| \leq k$, questo prova che $f(G,k)$ è una istanza sì si DS.
Viceversa, se $f(G,k) = \langle \overline{G} = (\overline{V}, \overline{E}), k\rangle$ è una istanza sì di DS, allora esiste $D \subseteq \overline{V}$, con $|D| \leq k$, tale che, per ogni $e \in \overline{V} - D$ esiste $d \in D$ tale che $(e,d) \in \overline{E}$. Definiamo, allora, $E' = \{e \in E = \overline{V} : e \in D\}$ e dimostriamo che esso è unEdge Cover per G. Sia $e \in E - E'$; allora $e \in \overline{V} - D$ e quindi, poiché D è un Dominating Set per $\overline{V}$, esiste $d \in D$ tale che $(e,d) \in \overline{E}$. Ma $(e,d) \in \overline{E}$ significa che gli archi $e \in E$ e $d \in E$ hanno un estremo in comune, cioè, che l'arco e è coperto dall'arco d in G. Questo prova che E' è un Edge Cover per G e, poiché $|E'| = |D \leq k$, questo prova che $\langle G,k\rangle$

è una istanza sì si EC.
Poiché f è calcolabile in tempo polinomiale in $|G|$, questo prova che f è una riduzione polinomiale da EC a DS.

Per provare la **NP**-completezza di EC è necessario ridurre ad esso un problema noto **NP**-completo, e non ridurre esso ad un problema **NP**-completo (come fa la funzione f). Quindi, l'esistenza della funzione f non dimostra che EC è **NP**-completo.

Per completezza, osserviamo che, se interpretiamo f come una trasformazione di istanze di DS ad istanze di EC, essa non una riduzione da DS ad EC. Per provare questa affermazione, consideriamo un grafo $G = (V,E)$ in cui $V = \{u,v,z,x,y\}$ e $E = \{(u,v),(u,z),(u,x),(u,y)\}$ (stella centrata in u): è facile verificare che $D = \{u\}$ è un insieme dominante per G, ma che non esiste alcun Edge Cover di cardinalità 1 per $\overline{G}$. Infatti, $\overline{G}$ è una clique di 4 nodi (e, quindi, 6 archi) e per coprire tutti i suoi archi è necessario utilizzare un insieme di almeno 2 archi.

Problema 4. Si osservi, innanzi tutto, che l'algoritmo opera nello stesso tempo in cui opera l'algoritmo approssimante per MIN-VERTEX COVER ed è, pertanto un algoritmo polinomiale.
Poi, osserviamo che l'algoritmo calcola effettivamente un Edge Cover per il suo input $G = (V,E)$. Infatti, ad ogni iterazione rimuove da E (evitando, così di considerare il loro futuro inserimento in E') tutti e soli gli archi coperti dal nodo che in quella iterazione è stato inserito in E'.
Per quanto riguarda l'analisi della qualità della soluzione, anche essa è analoga a quella relativa all'algoritmo approssimante per MIN-VERTEX COVER. Infatti, in questo caso, l'algoritmo restituisce un insieme E' di archi che non hanno estremi in comune. Consideriamo una coppia qualsiasi di archi in E' che, per quanto appena osservato, saranno della forma (a,b) e (u,v) con $a \notin \{u,v\}$ e $b \notin \{u,v\}$. Tali archi devono necessariamente essere coperti e potrebbero essere coperti da meno di due archi (come avviene in E', dove sono coperti da sé stessi) solo se esistesse in G uno degli archi (a,u) o (a,v) o $(b.u)$ o (b,v): in tal caso, sostituendo in E' i due archi (a,b) e (u,v) con uno di tali archi si potrebbe forse ottenere un Edge Cover di cardinalità minore (osserviamo che questo accadrebbe solo se il nuovo arco coprisse anche tutti gli archi coperti da (a,b) e (u,v)). Questo significa che potrebbe esistere in G un Edge Cover E'' di cardinalità $|E'|/2$ in cui coppie di archi di E' sono coperte (e sostituite) da singoli archi: si pensi, ad esempio, al grafo costituito dall'insieme di nodi $V = \{x,y,z,w\}$ e dall'insieme di archi $E = \{(x,y),(y,z),(z,w)\}$, all'insieme $E' = \{(x,y),(z,w)\}$ e all'insieme $E'' = \{(y,z)\}$. D'altra parte, proprio perché gli archi in E' non hanno estremi in comune, un singolo arco non può coprire più di una coppia di archi in E' e, poiché tutti gli archi in E' devo essere coperti, un insieme $\hat{E} \subset E$ di cardinalità minore di $|E'|/2$ non potrebbe essere un Edge Cover per G (in quanto lascerebbe scoperto almeno un elemento E'). Questo prova che l'Edge Cover E' calcolato dall'algorimo ha cardinalità al più doppia dell'Edge Cover ottimo E^* e, quindi,

$$\frac{|E'| - |E^*|}{|E^*|} \leq \frac{2|E^*| - |E^*|}{|E^*|} \leq 1.$$

21 Appello straordinario del 26 gennaio 2015

Problema 1. Progettare una macchina di Turing di tipo trasduttore che legga in input un intero n, rappresentato in unario, e, in tempo in $O(n)$, calcoli la funzione

$$f(n) = \left\lfloor \frac{n}{3} \right\rfloor + \left\lceil n - 3 \left\lfloor \frac{n}{3} \right\rfloor \right\rceil$$

scrivendo il risultato (in unario) sul nastro di output (si osservi che il secondo addendo della funzione è il resto della divisione intera di n per 3).
Si può affermare che $f(n)$ è time-constructible?

Problema 2. Sia k una costante positiva. Si consideri il seguente problema: dati un grafo (non orientato) $G = (V,E)$ ed una coppia di nodi $u, v \in V$, decidere se esiste in G un percorso da u a v di lunghezza (esattamente) k.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si consideri il seguente problema: dati un grafo (non orientato) $G = (V,E)$ ed una coppia di nodi $u, v \in V$, decidere se esiste in G un percorso semplice (ossia, che non passi più volte per lo stesso nodo) da u a v di lunghezza (esattamente) $\left\lceil \frac{|V|}{2} \right\rceil - 1$.
Si consideri, ora, la seguente funzione f che trasforma istanze del problema CAMMINO HAMILTONIANO (HP) in istanze del problema in esame: data una istanza $G = (V,E)$ ed una coppia di nodi $u, v \in V$, $f(G,u,v) = \langle G' = (V',E), u, v \rangle$ in cui $V' = V \cup \overline{V}$ e $\overline{V}$ è un insieme di $|V|$ nodi, ossia, G' consiste del grafo $G = (V,E)$ e di ulteriori $|V|$ nodi isolati e u e v rimangono invariati.
Dimostrare se f è una riduzione polinomiale da HP al problema in esame, e se questo dimostra che il problema in esame è **NP**-completo.

Problema 4. Sia k una costante positiva. Si consideri il seguente problema: dato un grafo (non orientato) $G = (V,E)$, decidere se in G non esiste alcun ciclo di (esattamente) k nodi.
Studiare la complessità computazionale del suddetto problema, collocandolo nella corretta classe di complessità.

21.1 Soluzione

Problema 1. Si osservi che $L = L_1 - L_2^c = L_1 \cap L_2$. Definiamo la macchina di Turing T che, con input $x \in \Sigma^*$, opera come segue:

1) verifica se $x \in L_1$: se $x \notin L_1$ allora rigetta, altrimenti, se $x \in L_1$, esegue il passo 2);

2) verifica se $x \in L_2$: se $x \in L_2$ allora accetta, se $x \notin L_2$ allora rigetta.

Poiché L_1 è un linguaggio decidibile, il passo 1) termina per ogni $x \in \Sigma^*$. Poiché L_2 è un linguaggio accettabile, il passo 2) temina per ogni $x \in L_2$ (e nulla si può dire se $x \notin L_2$). Quindi, se $x \in L_1 \cap L_2 = L$ la computazione $T(x)$ termina nello stato di accettazione (e nulla si può dire se $x \notin L$), ossia, L è un linguaggio accettabile.

Problema 2. Chiamiamo LARGE VERTEX SET (in breve, LVC) il problema in questione, che può essere formalizzato come di seguito descritto:

- $I_{LVC} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{LVC}(G,k) = \{V' \subseteq V\}$;
- $\pi_{LVC}(G,k,V') = |V'| \geq k \ \wedge \ [\ \forall (u,v) \in E : u \in V' \ \vee \ v \in V'\ ]$.

È ora sufficiente osservare che, per ogni grafo $G = (V,E)$, l'intero insieme V dei nodi è un Vertex Cover per G; quindi, $\langle G = (V,E), k\rangle$ è una istanza sì per LVC se e soltanto se $|V'| \geq k$. In conclusione, il problema è, banalmente, in **P**.

Problema 3. Mostriamo di seguito la formalizzazione dei due problemi in questione, che indicheremo, in breve, rispettivamente, DS e EC:

- $I_{DS} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{DS}(G,k) = \{D \subseteq V\}$;
- $\pi_{DS}(G,k,D) = |D| \geq k \ \wedge \ \forall u \in V - D \ [\ \exists v \in D : (u,v) \in E\ ]$
- $I_{EC} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{EC}(G,k) = \{E' \subseteq E\}$;
- $\pi_{EC}(G,k,E') = |E'| \geq k \ \wedge \ \forall (u,v) \in E - E' \ [\ \exists z \in V : (u,z) \in E' \ \vee \ (v,z) \in E'\ ]$

Sia ora $\langle G = (V,E), k\rangle \in I_{EC}$ e sia $f(G,k) = \langle \overline{G} = (\overline{V}, \overline{E}), k\rangle$.
Se $\langle G = (V,E), k\rangle \in I_{EC}$ è una istanza sì per EC allora esiste $E' \subseteq E$, con $|E'| \leq k$, tale che, per ogni $(u,v) \in E - E'$, esiste $z \in V$ tale che $(u,z) \in E'$ o $(v,z) \in E'$. Definiamo, allora, $D = \{e \in \overline{V} = E : e \in E'\}$ e dimostriamo che esso è un Dominating set per $\overline{G}$. Sia $e \in \overline{V} - D$ con $e = (u,v)$; allora, per definizione di D, $e \in E - E'$ e quindi, poiché E' è un Edge Cover per G, esiste un arco $e' \in E'$ che è incidente su u oppure su v. Allora, $e' \in D$ e $(e,e') \in \overline{E}$. Questo prova che D è un Dominating Set per $\overline{G}$ e, poiché $|D| = |E'| \leq k$, questo prova che $f(G,k)$ è una istanza sì si DS.
Viceversa, se $f(G,k) = \langle \overline{G} = (\overline{V}, \overline{E}), k\rangle$ è una istanza sì di DS, allora esiste $D \subseteq \overline{V}$, con $|D| \leq k$, tale che, per ogni $e \in \overline{V} - D$ esiste $d \in D$ tale che $(e,d) \in \overline{E}$. Definiamo, allora, $E' = \{e \in E = \overline{V} : e \in D\}$ e dimostriamo che esso è unEdge Cover per G. Sia $e \in E - E'$; allora $e \in \overline{V} - D$ e quindi, poiché D è un Dominating Set per $\overline{V}$, esiste $d \in D$ tale che $(e,d) \in \overline{E}$. Ma $(e,d) \in \overline{E}$ significa che gli archi $e \in E$ e $d \in E$ hanno un estremo in comune, cioè, che l'arco e è coperto dall'arco d in G. Questo prova che E' è un Edge Cover per G e, poiché $|E'| = |D \leq k$, questo prova che $\langle G,k\rangle$

è una istanza sì si EC.
Poiché f è calcolabile in tempo polinomiale in $|G|$, questo prova che f è una riduzione polinomiale da EC a DS.

Per provare la **NP**-completezza di EC è necessario ridurre ad esso un problema noto **NP**-completo, e non ridurre esso ad un problema **NP**-completo (come fa la funzione f). Quindi, l'esistenza della funzione f non dimostra che EC è **NP**-completo.

Per completezza, osserviamo che, se interpretiamo f come una trasformazione di istanze di DS ad istanze di EC, essa non una riduzione da DS ad EC. Per provare questa affermazione, consideriamo un grafo $G = (V,E)$ in cui $V = \{u,v,z,x,y\}$ e $E = \{(u,v),(u,z),(u,x),(u,y)\}$ (stella centrata in u): è facile verificare che $D = \{u\}$ è un insieme dominante per G, ma che non esiste alcun Edge Cover di cardinalità 1 per $\overline{G}$. Infatti, $\overline{G}$ è una clique di 4 nodi (e, quindi, 6 archi) e per coprire tutti i suoi archi è necessario utilizzare un insieme di almeno 2 archi.

Problema 4. Si osservi, innanzi tutto, che l'algoritmo opera nello stesso tempo in cui opera l'algoritmo approssimante per MIN-VERTEX COVER ed è, pertanto un algoritmo polinomiale.
Poi, osserviamo che l'algoritmo calcola effettivamente un Edge Cover per il suo input $G = (V,E)$. Infatti, ad ogni iterazione rimuove da E (evitando, così di considerare il loro futuro inserimento in E') tutti e soli gli archi coperti dal nodo che in quella iterazione è stato inserito in E'.
Per quanto riguarda l'analisi della qualità della soluzione, anche essa è analoga a quella relativa all'algoritmo approssimante per MIN-VERTEX COVER. Infatti, in questo caso, l'algoritmo restituisce un insieme E' di archi che non hanno estremi in comune. Consideriamo una coppia qualsiasi di archi in E' che, per quanto appena osservato, saranno della forma (a,b) e (u,v) con $a \notin \{u,v\}$ e $b \notin \{u,v\}$. Tali archi devono necessariamente essere coperti e potrebbero essere coperti da meno di due archi (come avviene in E', dove sono coperti da sé stessi) solo se esistesse in G uno degli archi (a,u) o (a,v) o $(b.u)$ o (b,v): in tal caso, sostituendo in E' i due archi (a,b) e (u,v) con uno di tali archi si potrebbe forse ottenere un Edge Cover di cardinalità minore (osserviamo che questo accadrebbe solo se il nuovo arco coprisse anche tutti gli archi coperti da (a,b) e (u,v)). Questo significa che potrebbe esistere in G un Edge Cover E'' di cardinalità $|E'|/2$ in cui coppie di archi di E' sono coperte (e sostituite) da singoli archi: si pensi, ad esempio, al grafo costituito dall'insieme di nodi $V = \{x,y,z,w\}$ e dall'insieme di archi $E = \{(x,y),(y,z),(z,w)\}$, all'insieme $E' = \{(x,y),(z,w)\}$ e all'insieme $E'' = \{(y,z)\}$. D'altra parte, proprio perché gli archi in E' non hanno estremi in comune, un singolo arco non può coprire più di una coppia di archi in E' e, poiché tutti gli archi in E' devo essere coperti, un insieme $\hat{E} \subset E$ di cardinalità minore di $|E'|/2$ non potrebbe essere un Edge Cover per G (in quanto lascerebbe scoperto almeno un elemento E'). Questo prova che l'Edge Cover E' calcolato dall'algorimo ha cardinalità al più doppia dell'Edge Cover ottimo E^* e, quindi,

$$\frac{|E'| - |E^*|}{|E^*|} \leq \frac{2|E^*| - |E^*|}{|E^*|} \leq 1.$$

22 Appello del 15 giugno 2015

Problema 1. Si ricordi che, per ogni $k \in \mathbb{N}$ costante, la funzione n^k è time-constructible, e sia, per ogni $k \in \mathbb{N}$ costante, T_k il trasduttore deterministico che certifica la time-constructibility di n^k. Utilizzando le macchine T_3 e T_2, dimostrare che esiste una macchina di Turing T di tipo trasduttore che calcola la funzione

$$f(n) = \left\lfloor \frac{n^3+2}{n^2+1} \right\rfloor$$

(parte intera della divisione fra n^3+2 e n^2+1) e tale che $dtime(T,n) \in \mathbf{O}(n^3)$.

Problema 2. Si consideri il problema seguente: dato un grafo (non orientato) $G = (V,E)$, decidere se G è 3-colorabile oppure è un grafo completo.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si ricordi il problema DOMINATING SET (DS): dati un grafo non orientato $G = (V,E)$ ed un intero $k \in \mathbb{N}$, decidere se esiste $D \subseteq V$ tale che $|D| \leq k$ e, per ogni $u \in V - D$, esiste $v \in D$ tale che $(u,v) \in E$.

Si consideri, ora, il seguente problema 2-HOPS-DOMINATING SET (2HDS): dati un grafo non orientato $G = (V,E)$ ed un intero $k \in \mathbb{N}$, decidere se esiste $H \subseteq V$ tale che $|H| \leq k$ e, per ogni $u \in V - H$, esistono $v \in H$ e $z \in V$ tali che $(u,z) \in E$ e $(z,v) \in E$.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, si dimostri se la funzione $f : I_{DS} \to I_{2HDS}$ di seguito descritta è una riduzione polinomiale da DS a 2HDS.
Sia $\langle G = (V,E), k \rangle$ una istanza di DS; allora $f(G,k) = \langle G' = (V',E'), k \rangle$, con

- $V' = V \cup E$;
- $E' = \{(x,y) : x \in V \ \wedge \ y \in E \wedge$ in G l'arco y è incidente sul nodo $x\ \}$, o, equivalentemente,
  $E' = \{(x,y) : x \in V \ \wedge \ y \in E \ \wedge \ \exists z \in V \ [\ y = (x,z)\ ]\ \}$.

22.1 Soluzione

Problema 1. La macchina T che testimonia la time-constructibility della funzione $f(n)$ utilizza 5 nastri:

- sul nastro N_1 è scritto l'input n, in unario;
- sul nastro N_2 verrà scritto il valore n^2 e, immediatamente dopo, il valore n^2+1;
- sul nastro N_3 verrà scritto il valore n^3 e, immediatamente dopo, il valore n^3+2;;
- sul nastro N_4 verrà scritto il valore di output $f(n)$;
- il nastro N_5 è il nastro di lavoro.

La computazione $T(n)$ è suddivisa in 5 fasi, descritte di seguito.

Fase 1) T simula il comportamento della macchina T_2 e, utilizzando il nastro N_5 come nastro di lavoro, scrive il valore n^2 (in unario) sul nastro N_2.

Fase 2) T scrive un carattere '1' a destra dell'ultimo carattere '1' scritto, nella fase precedente, sul nastro N_2: al termine di questa fase, sul nastro N_2 è scritto il valore n^2+1 (in unario).

Fase 3) T simula il comportamento della macchina T_3 e, utilizzando il nastro N_5 come nastro di lavoro, scrive il valore n^3 (in unario) sul nastro N_3.

Fase 4) T scrive due caratteri '1' a destra dell'ultimo carattere '1' scritto, nella fase precedente, sul nastro N_3: al termine di questa fase, sul nastro N_3 è scritto il valore n^3+2 (in unario).

Fase 5) Utilizzando i contenuti dei nastri N_2 e N_3, T calcola il valore $f(n)$ verificando quante volte il valore scritto sul nastro N_2 è contenuto nel valore scritto sul nastro N_3. All'inizio della Fase 5, le testine sui nastri N_2 e N_3 sono posizionate sui caratteri '1' più a destra su ciascun nastro; dunque la computazione prosegue alternando fra le due sottofasi di seguito descritte:

5.1) se le testine sui nastri N_2 e N_3 leggono entrambe '1', esse vengono spostate di una posizione a sinistra; se la testina sul nastro N_2 legge '□' e la testina sul nastro N_3 legge '1', allora la macchina scrive '1' sul nastro N_4, sposta a destra la testina sul nastro N_2 (lasciando ferma la testina sul nastro N_3) e passa ad eseguire la sottofase 5.2); se le testine sui nastri N_2 e N_3 leggono entrambe '□', allora la macchina scrive '1' sul nastro N_4 e termina; se la testina sul nastro N_2 legge '1' e la testina sul nastro N_3 legge '□', allora la macchina termina (senza scrivere '1' sul nastro N_4);

5.1) se le testine sui nastri N_2 e N_3 leggono entrambe '1', la testina sul nastro N_2 viene spostata di una posizione a destra e la testina sul nastro N_3 viene spostata di una posizione a sinistra; se la testina sul nastro N_2 legge '□' e la testina sul nastro N_3 legge '1', allora la macchina scrive '1' sul nastro N_4, sposta a sinistra la testina sul nastro N_2 (lasciando ferma la testina sul nastro N_3) e passa ad eseguire la sottofase 5.1); se le testine sui nastri N_2 e N_3 leggono entrambe '□', allora la macchina scrive '1' sul nastro N_4 e termina; se la testina sul nastro N_2 legge '1' e la testina sul nastro N_3 legge '□', allora la macchina termina (senza scrivere '1' sul nastro N_4).

Valutiamo, ora, $dtime(T,n)$. Dalla definizione delle macchine T_2 e T_3, segue che le fasi 1 e 3 terminano, rispettivamente, in $\mathbf{O}(n^2)$ e $\mathbf{O}(n^3)$ passi; pertanto, poiché le fasi 2 e 4 richiedono tempo costante, la fase 5 ha inizio dopo $\mathbf{O}(n^3)$ passi. Per quanto riguarda la fase 5, osserviamo che essa termina non appena viene letto '□' sul nastro N_3; poiché durante la fase 5 la testina sul nastro N_3 non inverte mai la direzione del proprio movimento (si vuove sempre verso destra), e poiché essa rimane ferma per una istruzione ogni volta che la testina sul nastro N_2 legge '□', ossia, al più $f(n)$ istruzioni, il simbolo '□' sul nastro N_3 viene incontrato dopo aver eseguito al più $\mathbf{O}(n^3)+f(n)$ istruzioni. Quindi, osservando che $f(n) \in \mathbf{O}(n^3)$, possiamo concludere che $dtime(T,n) \in \mathbf{O}(n^3)$.

Problema 2. Chiamiamo 3 COLORABILITY OR COMPLETE GRAPH (in breve, 3COLC) il problema in questione, che può essere formalizzato come di seguito descritto:

- $I_{3COLC} = \{G = (V,E) : G \text{ è un grafo non orientato } \}$;
- $S_{3COLC}(G) = \{c : V \to \{1,2,3\} \}$;
- $\pi_{3COLC}(G, S_{3COLC}(G)) = \left\{ \exists c \in S_{3COLC}(G) : \ \forall (u,v) \in E \ [\ c(u) \neq c(v)\ ] \right\} \ \vee \ G$ è un grafo completo.

Il problema è in **NP**: infatti, un certificato per una istanza $G = (V,E)$ è una coppia $\langle c, G \rangle$, in cui $c : V \to \{1,2,3\}$ è una colorazione dei nodi dei G mediante 3 colori, che ha lunghezza $\mathbf{O}(|G|)$, e verificare se un certificato è una soluzione effettiva significa verificare il predicato

$$\eta_{3COLC}(G,c) = \forall (u,v) \in E \ [\ c(u) \neq c(v)\ ] \ \vee \ \forall u,v \in V \ [\ (u,v) \in E\ ],$$

verifica, quest'ultima, realizzabile in tempo $\mathbf{O}(|E||V|^2)$.

Inoltre, il problema è completo per **NP**. Per dimostrarlo, mostriamo una riduzione polinomiale f dal problema 3COL. Prima di procedere, osserviamo che ogni istanza di 3COL costituita da un grafo contenente uno o due nodi è, banalmente, un'istanza no. In altre parole, il problema 3COL ristretto a grafi di uno o due nodi è un problema in **P**. Di conseguenza, poiché 3COL è un problema **NP**-completo in generale, esso rimane **NP**-completo per grafi che contengono almeno 3 nodi. In conclusione, nella riduzione che stiamo per presentare assumeremo, senza perdita di generalità, che il grafo istanza di 3COL contenga almeno 3 nodi.
Presentiamo, ora, la riduzione: data una istanza di 3COL $G = (V,E)$ (con $|V| \geq 3$), la corrispondente istanza di 3COLC è $f(G) = G' = (V',E')$, dove

- $V' = V \cup \{a\}$, in cui a è un nuovo nodo (ossia, $a \notin V$),
- $E' = E$.

Informalmente, il grafo G' è ottenuto semplicemente aggiungendo un nodo isolato a G.
Banalmente, calcolare G' da G richiede tempo lineare in $|G|$.

Mostriamo ora che f è effettivamente una riduzione da 3COL a 3COLC. Allo scopo, osserviamo che ogni 3-colorazione di G è banalmente trasformabile in una 3-colorazione di G' (assegnando ad a uno qualsiasi dei 3 colori): dunque, se G è una istanza sì di 3COL, allora G' è una istanza sì di 3COLC. Viceversa, anche ogni 3-colorazione di G' è banalmente trasformabile in una 3-colorazione di G e, quindi, se G' è una istanza sì di 3COLC, allora G è una istanza sì di 3COL.

Questo completa la prova di **NP**-completezza del problema.

Problema 3. Il problema 2HDS può essere formalizzato come di seguito descritto:

- $I_{2HDS} = \{\langle G = (V,E), k \rangle : G \text{ è un grafo non orientato } \wedge k \in \mathbb{N}\}$;
- $S_{2HDS}(G,k) = \{H \subseteq V \}$;
- $\pi_{2HDS}(G,k,S_{2HDS}(G)) = \exists H \in S_{2HDS}(G,k) : \ |H| \leq k \wedge \forall\, u \in V - H \left[\ \exists v \in H,\ \exists z \in V : \ (u,z) \in E \wedge (z,v) \in E\ \right]$.

Cominciamo con l'osservare che tempo $\mathbf{O}(|E||V|$ è sufficiente per calcolare $f(G,k)$.

Sia, ora, $G = (V,E)$ il grafo costituito due nodi collegati da un arco: in questo caso, $V = \{u,v\}$ e $E = \{e = (u,v)\}$. Consideriamo l'istanza $\langle G, 1 \rangle$ di DS; essa è una istanza sì, e gli unici insiemi dominanti costituiti da un singolo nodo sono gli insiemi $D_1 = \{u\}$ e $D_2 = \{v\}$.

L'istanza $f(G,1) = \langle G' = (V',E'), 1 \rangle$ di 2HDS corrispondente a $\langle G, 1 \rangle$ è, in questo caso, individuata dal grafo $G' = (V',E')$ tale che $V' = \{u,v,e\}$ e $E' = \{(u,e),(e,v)\}$ (si veda la Figura 0.3). In questo caso, mostriamo che nessun sottoinsieme H di V' costituito da un solo elemento soddisfa la condizione che deve essere soddisfatta da una soluzione effettiva di 2HDS.

Figura 0.3: Il grafo G, istanza di DOMINATING SET ed il suo corrispondente G', istanza di 2HDS, tramite la funzione f.

- Se $H = \{u\}$, il nodo e non soddisfa la condizione: infatti, l'unica coppia di nodi in $V' - \{e\}$ è u, v con $u \in H$ e $(e,v) \in E'$ ma $(v,u) \not\in E'$.
- Se $H = \{v\}$, di nuovo il nodo e non soddisfa la condizione: infatti, l'unica coppia di nodi in $V' - \{e\}$ è u, v con $v \in H$ e $(e,u) \in E'$ ma $(u,v) \not\in E'$.
- Se $H = \{e\}$, il nodo u e il nodo v non soddisfano entrambi la condizione: la verifica di questa affermazione è simile alle verifiche dei due casi precedenti ed è lasciata per esercizio.

23 Appello del 15 luglio 2015

Problema 1. Siano $L_1 \subseteq \mathbb{N} \times \mathbb{N}$ un liguaggio decidibile e $L_H \subseteq \mathbb{N} \times \mathbb{N}$ il linguaggio che definisce l'Halting Problem. Dimostrare se il linguaggio

$$L_2 = L_1 - L_H^c$$

è accettabile (ove, ricordiamo, L_H^c è il linguaggio complemento di L_H).

Problema 2. Si consideri il problema seguente: dati un grafo (non orientato) $G = (V,E)$ ed un intero $k \in \mathbb{N}$, decidere se G è 3-colorabile oppure contiene un sottografo completo di almeno $|V| - k$ nodi.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

Problema 3. Si consideri il problema seguente: dati un insieme X di variabili booleane ed una funzione booleana in forma 3-congiuntiva normale f definita sull'insieme X, decidere se esiste una assegnazione di verità agli elementi di X che soddisfa f e che assegna il valore `vero` ad esattamente due elementi di X.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, se ne dimostri l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

23.1 Soluzione

Problema 1. Si osservi, innanzi tutto, che $L_2 = L_1 \cap L_H$. Ricordiamo, inoltre, che L_H è un linguaggio accettabile. Siano T_1 e T_H, rispettivamente, la macchina di Turing che accetta L_1 e la macchina di Turing che accetta L_H. Descriviamo, ora, una macchina T_2 che utilizza 3 nastri e simula le due macchine T_1 e T_H operando in due fasi, come di seguito descritto. Con input x scritto sul primo nastro,

fase 1) simula $T_1(x)$ sul secondo nastro: se la computazione $T_1(x)$ rigetta allora anche la macchina T_2 rigetta, mentre se la computazione $T_1(x)$ accetta allora T_2 esegue la fase 2;

fase 2) simula $T_H(x)$ sul terzo nastro: se la computazione $T_H(x)$ rigetta allora anche la macchina T_2 rigetta, mentre se la computazione $T_H(x)$ accetta allora anche T_H accetta.

Dimostriamo che la macchina T_2 appena descritta accetta L_2. Sia $x \in L_2$: allora, per quanto osservato, $x \in L_1$ e $x \in L_H$. Allora, poiché T_1 accetta L_1 e $x \in L_1$, la prima fase termina quando $T_1(x)$ raggiunge lo stato di accettazione e, quindi, per definizione della macchina T_2, ha inizio la fase 2. Poiché T_H accetta L_H e $x \in L_H$, la seconda fase termina quando $T_H(x)$ raggiunge lo stato di accettazione e, quindi, per definizione della macchina T_2, anche $T_2(x)$ accetta.

Osserviamo, infine, che, poiché L_1 è decidibile, la fase 1 di qualunque computazione della macchina T_2 termina per ogni $x \in \mathbb{N} \times \mathbb{N}$. D'altra parte, poiché L_H è un linguaggio non decidibile, esiste $y \in \mathbb{N} \times \mathbb{N}$ tale che $T_H(y)$ non termina: pertanto, se esiste $z \in L_1$ tale che $T_H(z)$ non termina, allora $T_2(z)$ non termina. Pertanto, nulla è possibile concludere circa la decidibilità di L_2.

Problema 2. Chiamiamo 3-COLORABILITY OR CLIQUE (in breve, 3COL$\vee$C) il problema in questione, che può essere formalizzato come di seguito descritto:

- $I_{3COL\vee C} = \{\langle G = (V,E), k\rangle : G \text{ è un grafo non orientato e } k \in \mathbb{N}\}$;
- $S_{3COL\vee C}(G,k) = \{\langle c, V'\rangle : \ c : V \to \{1,2,3\} \ \wedge \ V' \subseteq V \ \}$;
- $\pi_{3COL\vee C}(G,k,S_{3COL\vee C}(G,k)) = \exists \langle c, V'\rangle \in S_{3COL\vee C}(G,k) : \ \{ \ \forall (u,v) \in E \ [\ c(u) \neq c(v) \ ] \ \} \ \vee \ \{ \ |V'| \geq |V| - k \ \wedge \ \forall \ u,v \in V' \ [\ (u,v) \in E \ ] \ \}$.

Il problema è in **NP**: infatti, un certificato per una istanza $G = (V,E)$ è una coppia $\langle c, V'\rangle$, in cui $c : V \to \{1,2,3\}$ è una colorazione dei nodi dei G mediante 3 colori, che ha lunghezza $\mathbf{O}(|G|)$, e V' è un sottoinsieme di V, che ha lunghezza $\mathbf{O}(|V|)$. Inoltre, verificare se un certificato è una soluzione effettiva significa verificare il predicato

$$\eta_{3COL\vee C}(G,k,c,V') = \{ \ \forall (u,v) \in E \ [\ c(u) \neq c(v) \ ] \ \} \ \vee \ \{ \ |V'| \geq |V| - k \ \wedge \ \forall u,v \in V \ [\ (u,v) \in E \ ] \ \},$$

verifica, quest'ultima, realizzabile in tempo $\mathbf{O}(|E||V|^2)$.

Inoltre, il problema è completo per **NP**. Per dimostrarlo, mostriamo una riduzione polinomiale f dal problema 3-COLORABILITÀ (3COL): data una istanza $G = (V,E)$ di 3COL, la corrispondente istanza di 3COL$\vee$C è $f(G) = \langle \overline{G} = (\overline{V}, \overline{E}), k\rangle$, dove

- $\overline{V} = V \cup \{a\}$, ove $a \notin V$,
- $\overline{E} = E$,
- $k = 0$.

Informalmente, il grafo $\overline{G}$ è ottenuto semplicemente aggiungendo un nuovo nodo isolato a G: avendo scelto $k = 0$, garantiamo in questo modo che, qualunque sia G, il grafo $\overline{G}$ non è un grafo completo, o, in altri termini, $\overline{G}$ non contiene

un sottografo completo di $|\overline{V}| - k = |\overline{V}|$ nodi.
Banalmente, calcolare $\overline{G}$ da G richiede tempo lineare in $|G|$.

Mostriamo ora che f è effettivamente una riduzione da 3COL a 3COL$\vee$C.
Sia $G = (V,E)$ una istanza sì di 3COL; allora, esiste $c : V \to \{1,2,3\}$ tale che, per ogni $(u,v) \in E$, $c(u) \neq c(v)$. Consideriamo, allora, la seguente funzione $\overline{c} : \overline{V} \to \{1,2,3\}$: per ogni $\overline{u} \in \overline{V}$,

$$\overline{c}(\overline{u}) = \begin{cases} 1 & \text{se } \overline{u} = a, \\ c(\overline{u}) & \text{se } \overline{u} \neq a. \end{cases}$$

Banalmente, poiché c è una 3-colorazione valida per G, anche $\overline{c}$ è una 3-colorazione valida per $\overline{G}$. Questo prova che $\langle \overline{G}, k \langle$ è una istanza sì di 3COL$\vee$C.
Sia $\langle G = (V,E) \rangle$ una istanza no di 3COL e supponiamo che $\langle \overline{G}, k = 0 \rangle$, l'istanza di 3COL$\vee$C corrispondente a G, sia invece una istanza sì per 3COL$\vee$C. Poiché, come abbiamo già osservato, $\overline{G}$ non contiene un sottografo completo di $|V| - k$ nodi, affinché $\langle \overline{G}, k = 0 \rangle$ sia una istanza sì per 3COL$\vee$C deve accadere che $\overline{G}$ sia 3-colorabile: dunque, deve esistere una funzione $\overline{c} : \overline{V} \to \{1,2,3\}$ tale che, per ogni $(\overline{u},\overline{v}) \in \overline{E}$, sia $\overline{c}(\overline{u} \neq \overline{c}(\overline{v})$. Definiamo, allora, la seguente funzione $c : V \to \{1,2,3\}$:

$$\text{per ogni } u \in V,\ c(u) = c(u).$$

Poiché $\overline{E} = E$, e poiché $\overline{c}$ è una 3-colorazione per $\overline{G}$, per ogni $(u,v) \in E$, si ha che $c(u) \neq c(v)$. Ma questo significa che c è una 3-colorazione valida per G, contraddicendo, in tal modo, il fatto che G non è 3-colorabile. Quindi, $\overline{G}$ è una istanza no di 3COL$\vee$C.

Questo completa la prova di **NP**-completezza del problema.

Problema 3. Il problema in esame, che indicheremo, in breve, con l'acronimo 2-3SAT, può essere formalizzato come di seguito descritto:

- $I_{2-3SAT} = \{\langle X, f \rangle : f \text{ è una funzione booleana in 3CNF nelle variabili in } X\}$;
- $S_{2-3SAT}(X,f) = \{a : X \to \{\texttt{vero}, \texttt{falso}\}\ \}$;
- $\pi_{2-3SAT}(X,f,S_{2-3SAT}(X,f)) = \exists a \in S_{2-3SAT}(X,f) :\ f(a(X)) \ \wedge\ |\ \{\ x \in X : a(x) = \texttt{vero}\ \}\ | = 2.$

Osserviamo che una soluzione ammissibile $a \in S_{2-3SAT}(X,f)$ è una soluzione effettiva soltanto se essa soddisfa il predicato $\alpha(a,X) = |\ \{\ x \in X : a(x) = \texttt{vero}\ \}\ | = 2$. Inoltre, il numero di assegnazioni di verità a per X che soddisfano il predicato $\alpha(a,X)$ è al più $|X|^2$; pertanto, possiamo generare le assegnazioni di vertà candidate ad essere soluzioni effettive in tempo deterministico $\mathbf{O}(|X|^2)$. Per ciascuna di esse, è poi possibile verificare in tempo (deterministico) polinomiale la soddisfacibilità di f: dunque, il problema 2-3SAT è in **P**.

In Tabella 0.9 è riportata una possibile implementazione nel linguaggio **PascalMinimo** dell'idea di algoritmo sopra indicata. I due cicli **while** alle linee 4 e 6 scelgono le due variabili (x_i e x_h) cui assegnare il valore `vero`. Il ciclo **while** alla linea 9 verifica se tutte le clausole sono soddisfatte dall'assegnazione corrente ($x_i = \texttt{vero}$, $x_h = \texttt{vero}$, $x_k = \texttt{falso}$ per $k \neq i$ e $k \neq h$): se viene trovata una clausola non soddisfatta esso si interrome con $j \leq m$ e alla linea 8 alla variabile sat viene assegnato nuovamente il valore `falso`.
È immediato verificare che il costo dell'algoritmo nell'implementazione in Tabella 0.9 è in $\mathbf{O}(n^2 m)$.

Input:	$f = \{c_1, c_2, \ldots, c_m\}$, con $c_j = (\ell_{j_1} \vee \ell_{j_2} \vee \ell_{j_3})$ e $\ell_{j_i} \in \{x_1, \ldots, x_n\}$ per ogni $i = 1,2,3$ e $j = 1, \ldots, m$.
Output:	accetta o rigetta.

```
1   X_A ← ∅;
2   sat ← falso;
3   i ← 1;
4   while (i < n ∧ sat = falso) do begin
5       h ← i + 1;
6       while (h ≤ n ∧ sat = falso) do begin
7           j ← 1;
8           sat ← vero;
9           while (j ≤ m ∧ ( uno dei letterali di c_j è x_i ∨ uno dei letterali di c_j è x_h ∨
                             uno dei letterali di c_j è ¬x_k con k ≠ i e k ≠ h) ) do j ← j + 1;
10          if (j < m + 1) then sat ← falso;
11      end
12      i ← i + 1;
13  end
14  if (sat = vero) then Output: accetta;
15  else Output: rigetta.
```

Tabella 0.9: Algoritmo `A:2-3SAT`.

24 Appello del 16 settembre 2015

Problema 1. Sia $k \in \mathbb{N}$ una costante, e sia NT_k una macchina di Turing non deterministica con grado di non determinismo pari a k. Definire una macchina di Turing non deterministica NT_2 con grado di non determinismo pari a 2 che sia equivalente a NT_k.

Problema 2. Due grafi $G = (V,E)$ e $G' = (V',E')$ sono *isomorfi* se essi sono uguali a meno di una ridenominazione dei nodi di uno dei due. Si vedano, ad esempio, i tre grafi riportati in Figura 0.4.
Più formalmente, G e G' sono isomorfi se esiste una biezione $b : V \to V'$ tale che, per ogni coppia di nodi $u, v \in V$ di G, si ha che $(u,v) \in E$ se e soltanto se $(b(u), b(v)) \in E'$. In riferimento alla figura, l'isomorfismo fra i grafi (a) e (b) è individuato dalla seguente biezione: $b(u_1) = v_3$, $b(u_2) = v_4$, $b(u_3) = v_1$, $b(u_4) = v_2$.

Il problema GRAFI NON ISOMORFI GNI) è definito nel modo seguente: dati due grafi (non orientati) $G = (V,E)$ e $G' = (V',E')$, decidere se G e G' non sono isomorfi.
Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, dimostrarne l'appartenenza alla classe **P** o alla classe **NP** o alla classe co**NP**.

Problema 3. Si consideri il problema seguente: dati un insieme X di variabili booleane, una funzione booleana in forma 3-congiuntiva normale f definita sull'insieme X, ed un intero $k \in \mathbb{N}$, decidere se esiste una assegnazione di verità agli elementi di X che soddisfa f e che assegna il valore `vero` ad al più k elementi di X.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I, S, \pi \rangle$, se ne dimostri l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

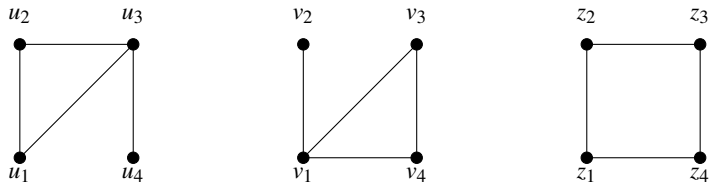


Figura 0.4: I grafi (a) e (b) sono isomorfi, ma non il grafo (c).

24.1 Soluzione

Problema 1. Indichiamo con Σ, con Q_k, e con P_k, rispettivamente, l'alfabeto, l'insieme degli stati, e l'insieme delle quintuple che definiscono NT_k. Poiché NT_k è una macchina non deterministica, è possibile che, per qualche $x \in \Sigma_k$ e per qualche $q \in Q_k$, P_k contenga più di una quintupla i cui primi due elementi siano, rispettivamente, q e x. Indichiamo, dunque, per ogni $x \in \Sigma_k$ e per ogni $q \in Q_k$, con $P_k(q,x)$ l'insieme delle quintuple in P_k i cui primi due elementi sono q e x (si osservi che tale insieme può essere vuoto). D'altra parte, poiché il grado di non deteminismo di NT_k è k, per ogni $x \in \Sigma_k$ e per ogni $q \in Q_k$, $|P_k| \leq k$.
Siano, dunque, $x \in \Sigma$ e $q \in Q_k$ tali che

$$P_k(q,x) = \langle q,x,x_1,q_1,m_1\rangle,\ \langle q,x,x_2,q_2,m_2\rangle,\ \ldots\ \langle q,x,x_h,q_h,m_h\rangle$$

con $h > 2$ (e, ovviamente, $h \leq k$). Ricordiamo che il significato dell'insieme $P_k(q,x)$ di quintuple è il seguente: *se* la macchina si trova nello stato q *e* legge sul nastro il simbolo x *allora* deve eseguire o la quintupla $\langle q,x,x_1,q_1,m_1\rangle$ o la quintupla $\langle q,x,x_2,q_2,m_2\rangle$,... o la quintupla $\langle q,x,x_h,q_h,m_h\rangle$. Per ottenere lo stesso comportamento con una macchina che abbia grado di non determinismo 2, ragioniamo nel modo seguente: *se* la nuova macchina si trova nello stato q *e* legge sul nastro il simbolo x *allora* deve eseguire la quintupla $\langle q,x,x_1,q_1,m_1\rangle$ *oppure* non deve eseguirla - ossia, deve eseguire un'altra quintupla $\langle q,x,x,q'_1(x),\texttt{ferma}\rangle$; a questo punto, nello stato $q'_1(x)$ e leggendo il simbolo x, deve eseguire la quintupla $\langle q'_1(x),x,x_2,q_2,m_2\rangle$ *oppure* non deve eseguirla - ossia, deve eseguire un'altra quintupla $\langle q,x,x,q'_2(x),\texttt{ferma}\rangle$, e così via.

Quanto appena descritto è illustrato in Figura 0.5: la parte (a) mostra il comportamento della macchina NT_k quando si trova nello stato interno q e legge sul nastro il simbolo x (e, dunque, esegue una quintupla scelta in $P_k(q,x)$), mentre la parte (b) mostra la sequenza di passi che devono essere eseguiti dalla macchina NT_2 per ottenere un un comportamento equivalente.

Lo schema illustrato in Figura 0.5 può essere implementato in linguaggio **PascalMinimo**. A questo scopo, assumiamo. come di consueto, che lo stato interno della macchina sia memorizzato nella variabile q, il contenuto del nastro nell'array$N$, la posizione della testina della variabile intera t e che il movimento della testina sia indicato da un intero in $\{-1,0,1\}$. Allora, il passo non deterministico di grado k è implementato nel seguente frammento di codice

```
1   scegli una quintupla ⟨q,N[t],x_i,q_i,m_i⟩ nell'insieme P_k(q,n[t]:
2   N[t] → x_i;
3   q → q_i;
4   t → t + m_i;
```

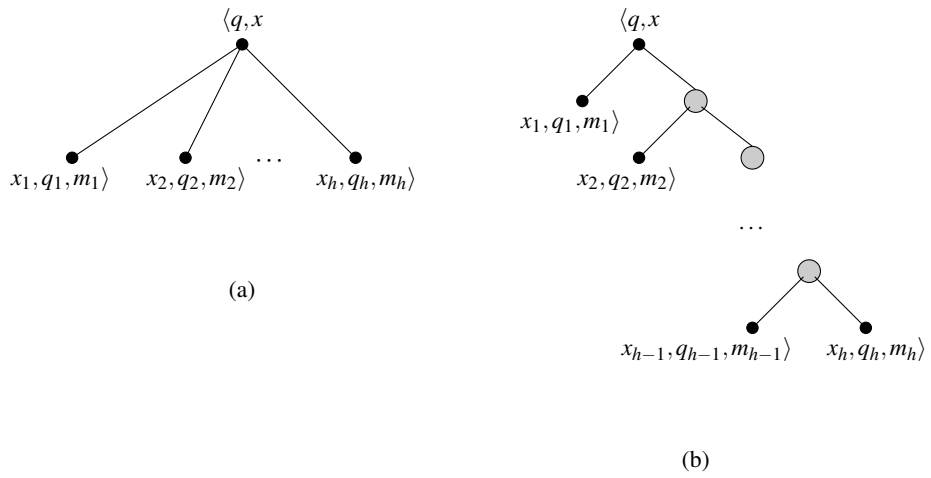


Figura 0.5: Un passo non deterministico di grado $h > 2$ in (a) e gli $h-1$ passi non deterministici di grado 2 ad esso equivalenti in (b).

mentre la sequenza di passi non deterministici di grado 2 è implementato nel seguente frammento di codice

```
1     i → 1; scelta → falso;
2     while (i ≤ |P_k(q,N[t] − 1) ∧ scelta → falso) do begin
3         scegli se eseguire la quintupla ⟨q,N[t],x_i,q_i,m_i⟩ oppure no:
4         if (hai scelto di eseguire la quintupla ⟨q,N[t],x_i,q_i,m_i⟩) then begin
5             N[t] → x_i;
6             q → q_i;
7             t → t + m_i;
8             scelta → vero;
8         end;
9         else i → i + 1;
9     end;
10    if (scelta → falso) then begin
11        N[t] → x_h;
12        q → q_h;
13        t → t + m_h;
14    end.
```

Definiamo, ora, formalmente, la macchina NT_2, il cui alfabeto di lavoro è ancora Σ ed il cui insieme degli stati Q_2 contiene propriamente Q_k. Indichiamo con P_2 l'insieme delle quintuple di NT_2 e, per ogni $x \in \Sigma_k$ e per ogni $q \in Q_k$, con $P_2(q,x)$ l'insieme delle quintuple in P_2 i cui primi due elementi sono q e x.
Se

$$P_k(q,x) = \langle q,x,x_1,q_1,m_1\rangle,\ \langle q,x,x_2,q_2,m_2\rangle,\ \ldots\ \langle q,x,x_h,q_h,m_h\rangle$$

e $2 < h \leq k$, allora introduciamo l'insieme dei nuovi stati interni $Q_2(q,x) = \{q'_1(q,x),\ldots,q'_{h-2}(q,x)\}$ e definiamo

l'insieme $P_2(q,x)$ come costituito dalle quintuple seguenti:

$$\begin{array}{rl}
\langle q,x,x_1,q_1,m_1\rangle & \langle q,x,x,q'_1(q,x),\texttt{ferma}\rangle \\
\langle q'_1(q,x),x,x_2,q_2,m_2\rangle & \langle q'_1(q,x),x,x,q'_2(q,x),\texttt{ferma}\rangle \\
\langle q'_2(q,x),x,x_3,q_3,m_3\rangle & \langle q'_2(q,x),x,x,q'_3(q,x),\texttt{ferma}\rangle \\
\multicolumn{2}{c}{\ldots} \\
\langle q'_{h-2}(q,x),x,x_{h-1},q_{h-1},m_{h-1}\rangle & \langle q'_{h-2}(q,x),x,x_h,q_h,m_h\rangle.
\end{array}$$

Se, invece $|P_k(q,x)| \leq 2$, allora definiamo $P_2(q,x) = P_k(q,x)$ e $Q_2(q,x) = \emptyset$. Infine, poniamo $P_2 = \cup_{q\in Q_k \,\wedge\, x\in\Sigma} P_2(q,x)$. Allora, per costruzione, per ogni $q \in Q_k$ e per ogni $x \in \Sigma$, il numero delle quintuple in $P_2(q,x)$ che iniziano con la stessa coppia stato-simbolo è al più 2. Infine, sempre per costruzione, $Q_2(q,x) \cap Q_2(q',x') = \emptyset$ se $q \neq q'$ o $x \neq x'$, e questo completa la prova che il grado di non determinismo di NT_2 è 2.

Problema 2. Il problema GNI può essere formalizzato come di seguito descritto:

- $I_{GNI} = \{\langle G = (V,E), G' = (V',E')\rangle : G \text{ e } G' \text{ sono grafi non orientati }\}$;
- $S_{GNI}(G,G') = \{b : V \to V' :\ b \text{ è una biezione fra } V \text{ e } V'\}$;
- $\pi_{GNI}(G,G',S_{GNI}(G,G')) = \forall b \in S_{GNI}(G,G')\ \big[\ \exists u,v \in V :\ \big(\ (u,v) \in E \wedge (b(u),b(v)) \notin E\big)\ \vee\ \big(\ (u,v) \notin E \wedge (b(u),b(v)) \in E\big)\ \big]$.

Osserviamo che il predicato π_{GNI} del problema richiede di verificare che *ogni* soluzione possibile soddisfi la proprietà:

$$\exists u,v \in v :\ \big(\ (u,v) \in E \wedge (b(u),b(v)) \notin E\big)\ \vee\ \big(\ (u,v) \notin E \wedge (b(u),b(v)) \in E\big).$$

La necessità di eseguire questa verifica per ogni elemento di $S_{GNI}(G,G')$ suggerisce che il problema sia in co**NP**. Per dimostrarlo, consideriamo il problema complemento, GRAFI ISOMORFI (GI): dati due grafi non orientati $G = (V,E)$ e $G' = (V',E')$, decidere se G e G' sono isomorfi. Di seguito, la sua formalizzazione:

- $I_{GI} = \{\langle G = (V,E), G' = (V',E')\rangle : G \text{ e } G' \text{ sono grafi non orientati }\}$;
- $S_{GI}(G,G') = \{b : V \to V' :\ b \text{ è una biezione fra } V \text{ e } V'\}$;
- $\pi_{GI}(G,G',S_{GNI}(G,G')) = \exists b \in S_{GNI}(G,G') :\ \forall u,v \in V\ \big[\ (u,v) \in E \Leftrightarrow (b(u),b(v)) \in E\ \big]$.

Il problema GI è in **NP**: infatti, un certificato per una istanza $\langle G = (V,E), G' = (V',E')\rangle$ è una funzione $b : V \to V'$, che ha lunghezza $\mathbf{O}(|V|)$. Inoltre, verificare se un certificato è una soluzione effettiva significa verificare il predicato

$$\eta_{GI}(G,G',b) = \forall u,v \in V\ \big[\ (u,v) \in E \Leftrightarrow (b(u),b(v)) \in E\ \big],$$

verifica, quest'ultima, realizzabile in tempo $\mathbf{O}(|V|^2|E'|)$. Questo completa la dimostrazione che GI è in **NP** e, conseguentemente, che il suo complemento GNI è in co**NP**.

Problema 3. Il problema in esame, che indicheremo, in breve, con l'acronimo MaxTrue3SAT, può essere formalizzato come di seguito descritto:

- $I_{\texttt{MaxTrue3SAT}} = \{\langle X,f,k\rangle : f \text{ è una funzione booleana in 3CNF nelle variabili in } X \text{ e } k \in \mathbb{N}\}$;
- $S_{\texttt{MaxTrue3SAT}}(X,f,k) = \{a : X \to \{\texttt{vero},\texttt{falso}\}\}$;
- $\pi_{\texttt{MaxTrue3SAT}}(X,f,k,S_{\texttt{MaxTrue3SAT}}(X,f,k)) = \exists a \in S_{\texttt{MaxTrue3SAT}}(X,f,k) :\ f(a(X))\ \wedge\ |\ \{\ x \in X : a(x) = \texttt{vero}\ \}\ | \leq k$.

Un certificato per una istanza $\langle X, f, k\rangle$ di MaxTrue3SAT è un elemento $a \in S_{\texttt{MaxTrue3SAT}}(X, f, k)$ e, dunque ha lunghezza $\mathbf{O}(|X|)$. Per verificare un certificato è necessario verificare che esso soddisfa f (e sappiamo dal problema 3SAT che tale verifica richiede tempo polinomiale in $|f|$ e $|X|$) e che $|\{ x \in X : a(x) = \texttt{vero} \}| \leq k$: banalmente, quest'ultimo test richiede tempo lineare in $|X|$ e $|k|$. Questo prova che il problema è in **NP**.

Dimostriamo, ora, la completezza del problema per la classe **NP** mediante una riduzione polinomiale dal problema 3SAT. Sia $\langle X, f\rangle$ una istanza di 3SAT: ad essa facciamo corrispondere l'istanza $\langle X, f, |X|\rangle$ di MaxTrue3SAT, che, banalmente, è calcolabile in tempo polinomiale da $\langle X, f\rangle$.
Se $\langle X, f\rangle$ è una istanza sì di 3SAT, allora esiste una assegnazione di verità per X tale che $f(a(X)) = \texttt{vero}$. Ovviamente, $|\{ x \in X : a(x) = \texttt{vero} \}| \leq |X| = k$, e, quindi, $\langle X, f, |X|\rangle$ è una istanza sì di MaxTrue3SAT.
Se, invece, $\langle X, f, |X|\rangle$ è una istanza sì di MaxTrue3SAT, allora esiste una assegnazione di verità per X tale che $f(a(X)) = \texttt{vero}$ e $|\{ x \in X : a(x) = \texttt{vero} \}| \leq |X| = k$: tale assegnazione, dunque, soddisfa f e, quindi, $\langle X, f\rangle$ è una istanza sì di 3SAT.
Questo completa la prova di **NP**-completezza di MaxTrue3SAT.

25 Appello del 17 febbraio 2016

Problema 1. Sia $\Sigma = \{0,1\}$ e sia $L \subseteq \Sigma^*$ l'insieme delle parole della forma 1^n01^k tali che $n \in \mathbb{N}$ è un multiplo di $k \in \mathbb{N}$. Dimostrare che L è decidibile.

Problema 2. Dato un grafo $G = (V,E)$, sia $\chi(G) = \langle M,P \rangle$ una sua codifica in cui M è la matrice di adiacenza di G e P è l'insieme di tutti i sottoinsiemi di V.

Si consideri il seguente problema decisionale: dati un grafo (non orientato) $G = (V,E)$ ed un intero k, esiste un sottoinsieme V' di V di cardinalità al più k e tale che, per ogni $u \in V - V'$, tutti i nodi adiacenti a u sono in V'?

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I,S,\pi \rangle$, descrivere un algoritmo che, prendendo in input $\chi(G)$ e k, decide se $\langle G,k \rangle$ è una istanza sì del problema in tempo polinomiale in $|\chi(G)|$ e $|k|$.

Rispondere, infine, alla seguente domanda: l'esistenza di tale algoritmo è sufficiente a dimostrare l'appartenenza alla classe **P** del problema?

Problema 3. Si consideri il problema seguente: dati un insieme X di variabili booleane, una funzione booleana in forma 3-congiuntiva normale f definita sull'insieme X, ed un intero positivo $k \in \mathbb{N}^+$, decidere se esiste una assegnazione di verità agli elementi di X che soddisfa f e che assegna il valore `vero` ad almeno k elementi di X.

Dopo aver formalizzato la definizione del suddetto problema mediante la tripla $\langle I,S,\pi \rangle$, se ne dimostri l'appartenenza alla classe **P** o, in alternativa, la **NP**-completezza.

25.1 Soluzione

Problema 1. Dimostriamo la decidibilità di L descrivendo una macchina di Turing T, a due nastri e a testine indipendenti, che lo decide.
L'alfabeto di lavoro di T è lo stesso $\Sigma = \{0,1\}$ ed il suo insieme degli stati è $Q = \{q_0, q_1, q_2, q_A, q_R\}$, ove q_0 è lo stato iniziale, q_A lo stato di accettazione e q_R lo stato di rigetto.
All'inizio della computazione, l'input è scritto sul nastro N_1 mentre il nastro N_2 è vuoto.
Descriviamo, ora, le quintuple di T illustrando, contestualmente, una computazione della macchina.

1) la macchina copia sul nastro N_2 il contenuto del nastro N_1, cancellandolo da N_1, fino a quando non incontra il carattere '0' (ossia, se l'input appartiene ad L, copia 1^n sul nastro N_2):
   $\langle q_0, (1,\Box), (\Box,1), q_0, (d,d)\rangle$, $\langle q_0, (0,\Box), (\Box,\Box), q_1, (d,s)\rangle$.
   A questo punto, la testina del nastro N_1 è posizionata sul carattere a destra del primo carattere '0' che ha incontrato, e la testina del nastro N_2 è posizionata sul carattere '1' più a destra che vi ha scritto. Inoltre, se l'input appartiene ad L, sul nastro N_1 è scritto 1^k, ossia, una parola non contenente il carattere '0' (poiché l'input è una parola di Σ^* e quindi non può contenere il carattere '$\Box$').

2) La macchina verifica che la parola sul nastro N_1 non contenga '0' e che n sia un multiplo di k mediante il seguente procedimento:
   fino a quando non legge '$\Box$' sul nastro N_1
   prova a cancellare dal nastro N_2 tanti '1' quanti ne incontra sul nastro N_1:
   se non vi riesce (perché non sono rimasti abbastanza '1' sul nastro N_2), allora rigetta.
   Tale procedimento è implementato dalle quintuple seguenti:
   $\langle q_1, (1,1), (1,\Box), q_1, (d,s)\rangle$ (cancella un '1' da N_2 se legge '1' sia su N_1 che su N_2)
   $\langle q_1, (\Box,1), (\Box,1), q_2, (s,f)\rangle$ (si predispone a riposizionare la testina sul nastro N_1)
   $\langle q_2, (1,1), (1,1), q_2, (s,f)\rangle$, $\langle q_2, (\Box,1), (\Box,1), q_1, (d,f)\rangle$ (ha riposizionato la testina sul nastro N_1 sul carattere '1' più a sinistra e si predispone ad eseguire una nuova sequenza di cancellazioni sul nastro N_2)
   $\langle q_1, (0,x), (0,x), q_R, (f,f)\rangle$ (la parola su N_1 contiene uno '0' e, quindi, l'input non appartiene al linguaggio)
   $\langle q_1, (1,\Box), (1,\Box), q_R, (f,f)\rangle$ (n non è multiplo di k)
   $\langle q_1, (\Box,\Box), (\Box,\Box), q_A, (f,f)\rangle$ (n è multiplo di k).

Problema 2. Il problema decisionale considerato, che chiameremo A, può essere formalizzato come di seguito descritto:

- $I_A = \{\langle G = (V,E), k\rangle : G$ è un grafo non orientato e $k \in \mathbb{N}\}$;
- $S_A(G,k) = \{V' :\subseteq V\ \}$;
- $\pi_A(G,k,S_A(G,k)) = \exists V' \in S_A(G,k) : \forall u \in V - V'\ \forall (u,v) \in E\ [\ v \in V'\ ]$.

Osserviamo che il predicato π_A del problema può essere espresso nella maniera seguente:

$$\exists V' \in S_A(G,k) : \forall (u,v) \in E\ [\ u \in V'\ \vee\ v \in V'\ ],$$

e, quindi, il problema A coincide con il problema VERTEX COVER.

L'algoritmo richiesto nel testo, che riceve in input un intero k, la matrice di adiacenza M di un grafo $G = (V,E)$ e l'insieme P di tutti i sottoinsiemi di V, è descritto nel seguente frammento di codice:

```
1    trovato ← falso;
2    while (P ≠ ∅ ∧ trovato= falso) do begin
3        estrai un elemento V' da P:
```

```
4         if (|V'| ≤ k) then begin
5             trovato ← vero;
6             for (u ∈ V − V') do
7                 for (v ∈ V) do
8                     if (M[u,v] = 1 wedge v ∉ V') then trovato ← falso;
9         end;
10    end.
```

Analizziamo, ora, la complessità del frammento di codice appena descritto.
Poiché accedere ad un elemento della matrice M ha costo costante, e verificare se un nodo è in V' ha costo proporzionale a $|V'|$, l'istruzione **if** alla linea 8 ha costo $\mathbf{O}(|V|)$; pertanto, il doppio ciclo **for** alle linee 6 e 7 ha costo $\mathbf{O}(|V|^3)$.
Testare la condizione dell'istruzione **if** alla linea 4 ha costo $\mathbf{O}(k|V|) \subseteq \mathbf{O}(|V|^2)$ (si osservi, infatti, che $k \leq |V|$).
Il numero di iterazioni del ciclo **while** alla linea 2 è $|P|$, e, quindi, il costo del frammento di codice è $\mathbf{O}(|P| \cdot |V|^2)$, ossia, è polinomiale *nella dimensione dell'input* o, in altri termini, è polinomiale in $|\chi(G)|$.

Osserviamo, infine, che la codifica $\chi(G)$ non è una codifica ragionevole di G: infatti, $|P| = 2^{|V|}$ e, quindi, la codifica di G mediante la sola matrice di adiacenza (che codifica tutte le informazioni necessarie ad individuare un grafo) è esponenzialmente più corta di $\chi(G)$. Poiché un problema è in **P** se esiste un algoritmo deterministico che richiede tempo polinomiale nella dimensione di una *codifica ragionevole* delle sue istanze, il frammento di codice non permette di affermare che il problema A è in **P** (e, in effetti, esso coincide con VERTEX COVER ed è, quindi, **NP**-completo).

Problema 3. Il problema in esame, che indicheremo, in breve, con l'acronimo MinTrue3SAT, può essere formalizzato come di seguito descritto:

- $I_{\texttt{MinTrue3SAT}} = \{\langle X, f, k\rangle : f \text{ è una funzione booleana in 3CNF nelle variabili in } X \text{ e } k \in \mathbb{N}^+\}$;
- $S_{\texttt{MinTrue3SAT}}(X, f, k) = \{a : X \to \{\texttt{vero}, \texttt{falso}\}\ \}$;
- $\pi_{\texttt{MinTrue3SAT}}(X, f, k, S_{\texttt{MinTrue3SAT}}(X, f, k)) = \exists a \in S_{\texttt{MinTrue3SAT}}(X, f, k) :\ f(a(X)) \ \wedge\ |\ \{\ x \in X : a(x) = \texttt{vero}\ \}\ | \geq k$.

Un certificato per una istanza $\langle X, f, k\rangle$ di MinTrue3SAT è un elemento $a \in S_{\texttt{MinTrue3SAT}}(X, f, k)$ e, dunque ha lunghezza $\mathbf{O}(|X|)$. Per verificare un certificato è necessario verificare che esso soddisfa f (e sappiamo dal problema 3SAT che tale verifica richiede tempo polinomiale in $|f|$ e $|X|$) e che $|\ \{\ x \in X : a(x) = \texttt{vero}\ \}\ | \geq k$: banalmente, quest'ultimo test richiede tempo lineare in $|X|$ e $|k|$. Questo prova che il problema è in **NP**.

Dimostriamo, ora, la completezza del problema per la classe **NP** mediante una riduzione polinomiale dal problema 3SAT. Prima di procedere, osserviamo che è richiesto che il valore k nell'istanza di MinTrue3SAT sia strettamente positivo.
Sia $\langle X, f\rangle$ una istanza di 3SAT e siano $y_1, y_2, y_3 \notin X$: ad essa facciamo corrispondere l'istanza $\langle X', f', 1\rangle$ di MinTrue3SAT tale che: $X' = X \cup \{y_1, y_2, y_3\}$ e $f' = f\ \wedge\ (y_1 \vee y_2 \vee y_3)$. Banalmente, $\langle X', f', 1\rangle$ è calcolabile in tempo polinomiale da $\langle X, f\rangle$.
Se $\langle X, f\rangle$ è una istanza sì di 3SAT, allora esiste una assegnazione di verità per X tale che $f(a(X)) = \texttt{vero}$. Consideriamo l'assegnazione $a' : X \cup \{y_1, y_2, y_3\}$ definita come segue:

$$a'(x) = \begin{cases} a(x) & \text{se } x \in X, \\ \textbf{vero} & \text{se } x = y_1, \\ \textbf{falso} & \text{se } x = y_2 \ \vee\ x = y_3. \end{cases}$$

Per costruzione, a' soddisfa f' e, inoltre, $|\ \{\ x \in X' : a'(x) = \texttt{vero}\ \}\ | \geq 1 = k$, e, quindi, $\langle X', f', 1\rangle$ è una istanza sì di MinTrue3SAT.
Se, invece, $\langle X', f', 1\rangle$ è una istanza sì di MinTrue3SAT, allora esiste una assegnazione di verità a' per X' tale che $f'(a'(X')) = \texttt{vero}$: ancora per costruzione, la restrizione a di tale assegnazione alle variabili in X soddisfa f e, quindi, $\langle X, f\rangle$ è una istanza sì di 3SAT.
Questo completa la prova di **NP**-completezza di MinTrue3SAT.