Esercizi: la classe **P**

1 Problemi

Problema 8.1: 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 8.2: 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 8.3: 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 8.4: 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 8.5: 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 8.6: 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 8.7: 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 8.8: 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 8.9: 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 8.10: 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 $V-V'$ e 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 8.11: 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 8.12: Si consideri il seguente problema LARGE DOMINATING SET (in breve, LDS): decidere se, dato un grafo non orientato connesso $G = (V,E)$, esiste un 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 8.13: 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.

Problema 8.14: 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 8.15: 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 8.16: 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 8.17: 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 8.18: 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 8.19: 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 8.20: 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 8.21: 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 8.22: 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.

Problema 8.23: 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.

2 Soluzioni

Soluzione del problema 8.1

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 $\{x_1,\ldots,x_n\} \cup \{\neg x_1,\ldots,\neg x_n\}\}$.
- $\mathrm{S}_{DNF-SAT}(f(x_1,\ldots,x_n)) = \{a : \{x_1,\ldots,x_n\} \rightarrow \{$**vero**, **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 $O(n^2 \cdot m)$.

Soluzione del problema 8.2

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

- $I_{3VC} = \{G = (V,E) : G$ 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, ..., 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 $O(|E|)$.

Soluzione del problema 8.3

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:

- $I = \{G = (V,E) : G$ è un grafo $\}$;
- $S = \{< f : V \rightarrow \overline{V}, c : V \rightarrow \{1,2\}\}$, 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.

Soluzione del problema 8.4

Formalizzazione del problema:

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

Si osservi ora che una colorazione dei nodi di un grafo con 2 colori è anche una 3-colorazione dello stesso grafo. Infatti, sia $\chi_2 : V \rightarrow \{1,2\}$ tale che $\forall (u,v) \in E : \chi_2(u) \neq \chi_2(v)$ e definiamo la seguente funzione $\chi_3 : V \rightarrow \{1,2,3\}$: 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.

Soluzione del problema 8.5

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 $\}$.
- $S_{k-3SAT}(X,F) = \{a : X \rightarrow \{vero, falso\} : |\{x_i \in X : a(v) = vero\}| = k\}$.
- $\pi_{k-3SAT}(X,F,a) = F(a(X))$.

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.
    $S \leftarrow \{X' \subseteq X : |X'| = k\}$;
    $trovato \leftarrow$ **falso**;
    while ($S \neq \emptyset \wedge trovato = falso$) **do begin**
        estrai un elemento X' da S;
        for ($x \in X'$) **do** $a(x) \leftarrow vero$;
        for ($x \in X - X'$) **do** $a(x) \leftarrow falso$;
        $trovato \leftarrow \pi_{k-3SAT}(X,F,a)$;
    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**
        $\ldots$
            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 $h > 0$.

Soluzione del problema 8.6

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 $c_0 : V_1 \cup V_2 \rightarrow \{1,2,3\}$. 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**.

Soluzione del problema 8.7

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 $\}$.
- $S_{SUB-COL}(G,k) = \{V' \subseteq V\}$.
- $\pi_{SUB-COL}(G,k,V') = |V'| = k \wedge \exists c : V' \rightarrow \{1,2,\ldots k\} : [\forall u,v \in V' : (u,v) \in E \Rightarrow c(u) \neq c(v)]$.

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**.

Soluzione del problema 8.8

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 connesso 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**.

Soluzione del problema 8.9

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 $\}$.

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 8.1: Algoritmo che decide k-SAT.

- $S_{k-SAT}(X, f) = \{a : X \to \{vero, falso\}\}$.
- $\pi_{k-SAT}(X, f, a) = f(a(X))$.

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 8.1 dove, per semplicità di notazione, ciascuna clausola viene considerata come *insieme* di letterali.
In riferimento alla Tabella 8.1, 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**.

Soluzione del problema 8.10

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 $u \in V$

   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**.

Soluzione del problema 8.11

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 8.1) . Pertanto, la cardinalità minima $h_{min}(n)$ di un vertex cover per K_n

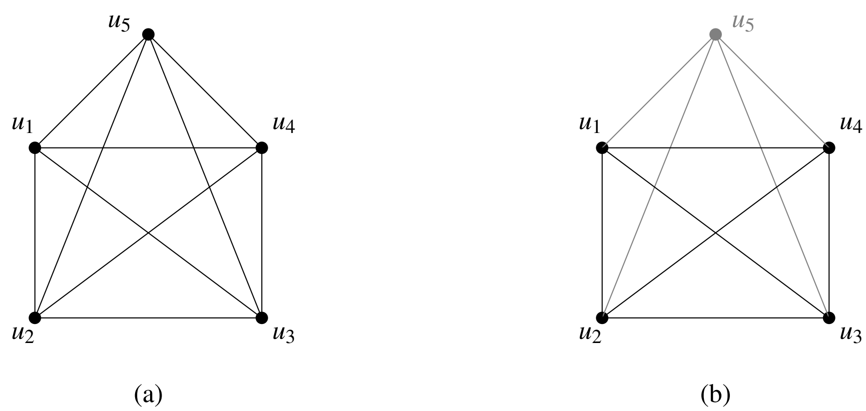

Figura 8.1: (a) Il grafo K_5; (b) gli archi non coperti dal nodo u_5 costituiscono il grafo K_4.

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**.

Soluzione del problema 8.12

Il problema *LDS* può essere formalizzato nella maniera seguente:

- $I_{LDS} = \{\langle G = (V,E)\rangle :$ G è un grafo connesso non orientato $\}$.
- $S_{LDS}(G) = \{D \subseteq V\}$.
- $\pi_{LDS}(G,D) = |D| \leq \lfloor \frac{|V|}{2} \rfloor \; wedge \; \forall \, u \in V - D \; [\; \exists \, v \in D : (u,v) \in E \;]$.

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 $|D_1| \leq |V|/2$.

Soluzione del problema 8.13

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**.

Soluzione del problema 8.14

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 8.2, 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**.

Soluzione del problema 8.15

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 $f(G) = 0$.

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 $j = 1, \ldots, m$.

$sat \leftarrow falso$;
$i_1 \leftarrow 1$;
for $(i \leftarrow 1;\ i \leq n;\ i \leftarrow i+1)$ **do** $a(x_i) \leftarrow falso$;
while $(i_1 \leq n-k+1\ \wedge\ sat = falso)$ **do begin**
$a(x_{i_1}) \leftarrow vero$;
$i_2 \leftarrow 1$;
while $(i_2 \leq n-k+2\ \wedge\ sat = falso)$ **do begin**
$a(x_{i_2}) \leftarrow vero$;
$i_3 \leftarrow 1$;
$\ldots$
$i_k \leftarrow 1$;
while $(i_k \leq n\ \wedge\ sat = falso)$ **do begin**
$a(x_{i_k}) \leftarrow vero$;
$sat \leftarrow f(a(X))$;
if $(sat = falso)$ **then** $a(x_{i_k}) \leftarrow falso$;
$i_k \leftarrow i_k + 1$;
end
if $(sat = falso)$ **then** $a(x_{i_{k-1}}) \leftarrow falso$;
$i_{k-1} \leftarrow i_{k-1} + 1$;
end
$\ldots$
end
if $(sat = falso)$ **then** $a(x_{i_1}) \leftarrow falso$;
$i_1 \leftarrow i_1 + 1$;
if $(sat = falso \wedge i_1 = n+1)$ **then**
for $(i \leftarrow 1;\ i \leq n;\ i \leftarrow i+1)$ **do** $a(x_i) \leftarrow falso$;
end
if $(sat = vero)$ **then Output:** accetta;
else Output: rigetta;

Tabella 8.2: Algoritmo che decide k-SAT.

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, $f \in \mathbf{FP}$.

Soluzione del problema 8.16

Formalmente, il problema 2IS descritto nella traccia è il seguente

- $I_{2IS} = \{G = (V,E) : G$ 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**.

Soluzione del problema 8.17

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

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

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**.

Soluzione del problema 8.18

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**.

Soluzione del problema 8.19

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 8.3 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 8.3, 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 $\in$ **P**.

Soluzione del problema 8.20

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

- $I_{LVC} = \{\langle G = (V,E), k\rangle : G$ è un grafo non orientato e $k \in \mathbb{N}\}$;
- $S_{LVC}(G,k) = \{V' \subseteq V\}$;

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

1 $\mathscr{P} \leftarrow \{ \, Y \subseteq X : |Y| = k \, \}$;
2 $trovato \leftarrow falso$;
    // la variabile $trovato$ verifica se l'insieme X' è stato trovato
3 **while** ($\mathscr{P} \neq \emptyset \, \wedge \, trovato = falso$) **do begin**
4   scegli $X' \in \mathscr{P}$;
5     $\mathscr{P} \leftarrow \mathscr{P} - \{X'\}$;
6     $trovato \leftarrow vero$;
        // con l'assegnazione sopra, supponiamo che l'insieme X'
        che abbiamo scelto al punto 4 sia parte della soluzione effettiva
7     $\mathscr{B} \leftarrow \{ b : X' \rightarrow \{vero, falso\} \, \}$;
8   **while** ($\mathscr{B} \neq \emptyset \, \wedge \, trovato = vero$) **do begin**
9   scegli $b \in \mathscr{B}$;
10         $\mathscr{B} \leftarrow \mathscr{B} - \{b\}$;
11         $f' \leftarrow f(b(X'))$;
            // ossia, f' viene ottenuta sostituiendo in f le variabili in X'
            con i valori che esse ottengono mediante l'assegnazione b
12   **if** ($f' \not\in$ 2SAT) **then** $trovato \leftarrow falso$;
13   **end**
14 **end**
15 **Output:** $trovato$

Tabella 8.3: Algoritmo che decide se $\langle X, f \rangle \in k$-S2SAT.

- $\pi_{LVC}(G, k, V') = |V'| \geq k \, \wedge \, [\, \forall (u, v) \in E : u \in V' \, \vee \, v \in V' \,]$.

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**.

Soluzione del problema 8.21

Chiamiamo k-PATH (in breve, kP) il problema in questione, che può essere formalizzato come di seguito descritto:

- $I_{kP} = \{ G = (V, E) : G$ è un grafo non orientato $\}$;
- $S_{kP}(G) = \{ V' = \{v_1, v_2, \ldots, v_{k-1} : \forall i, j = 1, \ldots, k-1 : i \neq j \, [v_i \neq v_j \,] \wedge \forall i = 1, \ldots, k-1 \, [\, u \neq v_i \wedge v \neq v_i \,] \, \} \subseteq V \}$;
- $\pi_{kP}(G, V' = \{v_1, v_2, \ldots, v_{k-1}\}) = (u, v_1) \in E \, \wedge \, (v_{k-1}, v) \in E \, \wedge \, \forall i = 1, \ldots, k-2 \, [\, (v_i, v_{i+1}) \in E \,]$.

Si osservi che, essendo k una costante, essa non compare nella descrizione dell'insieme delle istanze del problema. In effetti, il fatto che k sia costante implica che è possibile scrivere un algoritmo che utilizza k variabili, $i_1, i_2, \ldots, i_k$, ognuna delle quali governa un ciclo che sceglie uno dei k nodi che fa parte di una soluzione possibile del problema. Pertanto, è possibile considerare il seguente algoritmo che, con input $G = (V, E)$, ove $V = \{v_1, v_2, \ldots, v_n\}$, costruisce l'insieme $S_{kP}(G)$ e, successivamente, verifica se esiste $V' \in S_{kP}(G)$ che soddisfa $\pi_{kP}(G, V')$:

1) costruisci $S_{kP}(G)$:
   $S_{kP}(G) \leftarrow \emptyset$;
   for ($i_1 \leftarrow 1; i_1 \leq |V| - k + 2; i_1 \leftarrow i_1 + 1$) **do**
     for ($i_2 \leftarrow i_1 + 1; i_2 \leq |V| - k + 3; i_2 \leftarrow i_2 + 1$) **do**
       $\ldots$
         for ($i_{k-1} \leftarrow i_{k-2} + 1; i_{k-1} \leq |V|; i_{k-1} \leftarrow i_{k-1} + 1$) **do begin**
           $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{k-1}}\}$;

$S_{kP}(G) \leftarrow S_{kP}(G) \cup \{V'\}$;
end;

2) trovata $\leftarrow$ `falso`;

3) **while** ($S_{kP}(G) \neq \emptyset \wedge$ trovata $=$ `falso`) **do begin**

   3.1) estrai $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{k-1}}\}$ da $S_{kP}(G)$;

   3.2) **if** ($\pi_{kP}(G, V')$) **then** trovata $\leftarrow$ `vero`;

4) **end**

5) **if** (trovata $=$ `vero`) **then Output:** accetta;

6) **else Output:** rigetta;

È ora sufficiente osservare che costruire l'insieme $S_{kP}(G)$ al passo 1) dell'algoritmo richiede tempo $\mathbf{O}(|V|^{k-1})$, che il ciclo **while** al passo 3) esegue al più $|S_{kP}(G)| \leq \mathbf{O}(|V|^{k-1})$ iterazioni, e che, per ogni $V' \in S_{kP}(G)$, è possibile verificare $\pi_{kP}(G, V')$ in tempo $\mathbf{O}(|V'| \cdot |E|) = \mathbf{O}(|E|)$, per concludere che il precedente algoritmo, che decide se un grafo $G = (V, E)$ è istanza sì del problema k-PATH, ha costo polinomiale in $|G|$. In conclusione, il problema è in **P**.

Soluzione del problema 8.22

Il problema, che chiameremo k-CICLO (in breve, kC) è molto simile al problema k-PATH. In effetti, l'insieme delle istanze e delle soluzioni possibili possono essere formalizzati come di seguito descritto:

- $I_{kC} = \{G = (V, E) : G$ è un grafo non orientato $\}$;
- $S_{kC}(G) = \{V' = \{v_1, v_2, \ldots, v_k : \forall i, j = 1, \ldots, k : i \neq j \; [v_i \neq v_j] \} \subseteq V\}$.

Si osservi nuovamente che, essendo k una costante, essa non compare nella descrizione dell'insieme delle istanze del problema e che il fatto che k sia costante implica che è possibile scrivere un algoritmo che utilizza k variabili, $i_1, i_2, \ldots, i_k$, ognuna delle quali governa un ciclo che sceglie uno dei k nodi che fa parte di una soluzione possibile del problema. Pertanto, è possibile considerare il seguente algoritmo che, con input $G = (V, E)$, ove $V = \{v_1, v_2, \ldots, v_n\}$, costruisce l'insieme $S_{kP}(G)$ e, successivamente, verifica se esiste $V' \in S_{kC}(G)$ che soddisfa $\pi_{kC}(G, V')$:

1) costruisci $S_{kC}(G)$:
   $S_{kC}(G) \leftarrow \emptyset$;
   for ($i_1 \leftarrow 1; i_1 \leq |V| - k + 1; i_1 \leftarrow i_1 + 1$) **do**
      for ($i_2 \leftarrow i_1 + 1; i_2 \leq |V| - k + 2; i_2 \leftarrow i_2 + 1$) **do**
         ...
            for ($i_k \leftarrow i_{k-1} + 1; i_k \leq |V|; i_k \leftarrow i_k + 1$) **do begin**
               $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$;
               $S_{kC}(G) \leftarrow S_{kC}(G) \cup \{V'\}$;
            end;

2) trovata $\leftarrow$ `falso`;

3) **while** ($S_{kC}(G) \neq \emptyset \wedge$ trovata $=$ `falso`) **do begin**

   3.1) estrai $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ da $S_{kC}(G)$;

   3.2) **if** ($\pi_{kC}(G, V')$) **then** trovata $\leftarrow$ `vero`;

4) **end**

5) **if** (trovata $=$ `vero`) **then Output:** rigetta;

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 $X_A \leftarrow \emptyset$;
2 sat $\leftarrow$ `falso`;
3 $i \leftarrow 1$;
4 **while** ($i < n \wedge$ sat $=$ `falso`) **do begin**
5     $h \leftarrow i+1$;
6   **while** ($h \leq n \wedge$ sat $=$ `falso`) **do begin**
7         $j \leftarrow 1$;
8   sat $\leftarrow$ `vero`;
9   **while** ($j \leq m\ \wedge$ (uno dei letterali di c_j è $x_i \vee$ uno dei letterali di c_j è $x_h \vee$ uno dei letterali di c_j è $\neg x_k$ con $k \neq i$ e $k \neq h$)) **do** $j \leftarrow j+1$;
10   **if** ($j < m+1$) **then** sat $\leftarrow$ `falso`;
11   **end**
12     $i \leftarrow i+1$;
13 **end**
14 **if** (sat $=$ `vero`) **then Output**: accetta;
15 **else Output**: rigetta.

Tabella 8.4: Algoritmo `A:2-3SAT`.

6) **else Output:** accetta;

È ora sufficiente osservare che costruire l'insieme $S_{kC}(G)$ al passo 1) dell'algoritmo richiede tempo $\mathbf{O}(|V|^k)$, che il ciclo **while** al passo 3) esegue al più $|S_{kC}(G)| \leq \mathbf{O}(|V|^k)$ iterazioni, e che, per ogni $V' \in S_{kC}(G)$, è possibile verificare $\pi_{kC}(G,V')$ in tempo $\mathbf{O}(|V'| \cdot |E|) = \mathbf{O}(|E|)$, per concludere che il precedente algoritmo, che decide se un grafo $G = (V,E)$ è istanza sì del problema k-CICLO, ha costo polinomiale in $|G|$. In conclusione, il problema è in **P**.

Soluzione del problema 8.23

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

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

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 8.4 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 ($x_i =$ `vero`, $x_h =$ `vero`, $x_k =$ `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 8.4 è in $\mathbf{O}(n^2 m)$.