ALGORITMI - Complessità	**Anno Accademico 2014-15**

Appello 2 - Luglio 2015

Nome e Cognome: Matricola:

Importante: Usare solo i fogli forniti dal docente, riportando Nome Cognome e Matricola su ogni foglio. Scrivere in modo **leggibile** e **giustificare** ogni affermazione. Risposte non giustificate saranno valutate **0**. Per ogni algoritmo proposto va provata la correttezza e se necessario al problema, la complessità.
Si chiede di risolvere 3 esercizi:
(i) uno a scelta tra 1 e 2, (ii) uno a scelta tra 3 e 4, (iii) uno a scelta tra 5 e 6.

Esercizio 1 - 15 punti

Definiamo il seguente problema

HalfCycle (HalfC)
Input: Un grafo diretto $G = (V, E)$
Output: *yes* se e solo il ciclo piú lungo in G è di lunghezza esattamente $\lfloor |V|/2 \rfloor$.

Si dimostri che
(i) HalfC è $\mathcal{NP}$-hard
(ii) HalfC è co-$\mathcal{NP}$-hard
(iii) se $\mathcal{PH} \neq$ co$-\mathcal{NP}$ allora HalfC non è $\mathcal{NP}$-completo.

Esercizio 2 - 15 punti

Dati due problemi $\mathbb{A}$ e $\mathbb{B}$ definiti sullo stesso insieme di istanze, denotiamo con $\mathbb{A} \cap \mathbb{B}$ il problema le cui istanze *yes* sono tutte e solo le istanze che risultano essere contemporaneamente *yes* per $\mathbb{A}$ e $\mathbb{B}$.

Denotiamo con $\mathbb{A} \setminus \mathbb{B}$ il problema le cui istanze *yes* sono tutte e solo le istanze che risultano essere contemporaneamente *yes* per $\mathbb{A}$ e *no* per $\mathbb{B}$.

Definiamo le classi di problemi

$$\mathcal{DP}_0 = \{\mathbb{C} = \mathbb{A} \cap \mathbb{B} \mid \mathbb{A} \in \mathcal{NP}, \mathbb{B} \in \text{co}-\mathcal{NP}\}$$

$$\mathcal{DP}_1 = \{\mathbb{C} = \mathbb{A} \setminus \mathbb{B} \mid \mathbb{A}, \mathbb{B} \in \mathcal{NP}\}$$

Con riferimento alla definizione del problema HalfC data nell'esercizio precedente, dimostrare che
(i) $\mathcal{DP}_0 = \mathcal{DP}_1$
(ii) HalfC $\in \mathcal{DP}_1$
(iii) $\mathcal{DP}_1 = \mathcal{NP} \cup \text{co}-\mathcal{NP}$ se e solo se $\mathcal{NP} = \text{co}-\mathcal{NP}$.

Nota: Per la (iii) si possono usare come assunzioni (i) e (ii) dell'esercizio precedente.

Esercizio 3 - 15 punti

Definiamo il seguente problema

FACILITY LOCATION PROBLEM (FLP)
Input: Un grafo $G = (V, E)$ e per ogni vertice v un valore b_v, ed un valore di soglia B. Due giocatori $P1$ e $P2$ si alternano nello scegliere un nuovo nodo di V rispettando la condizione che i nodi scelti formino sempre un insieme indipendente. $P2$ vince se alla fine i nodi da lui selezionati hanno valore totale almeno pari a B
Output: *yes* se e solo $P2$ puó vincere.

Dimostrare che

(i) FLP $\in \mathcal{PSPACE}$.

(ii) se per ogni problema $\mathbb{A} \in \mathcal{PSPACE}$ risulta che $\mathbb{A} \leq_L$ allora FLP $\notin \mathcal{NL}$. Dove per $\leq_L$ intendiamo una riduzione log-space.

Bonus 10 punti. FLP è $\mathcal{PSPACE}$-completo

Esercizio 4 - 15 punti

Si fornisca la definizione esatta di riducibilità log-space tra problemi e se ne dimostri dettagliatamente la proprietà di transitività .

Esercizio 5 - 20 punti

Fornire la seguente L-riduzione

MAX-3-SAT $\leq_L$ MAX-2-SAT

Considerato quanto visto a lezione per MAX-3-SAT, e la L-riduzione fornita, cosa possiamo concludere riguardo all'inapprossimabilità di MAX-2-SAT? (Fornire una costante α tale che MAX-3-SAT non è α-approssimabile se $\mathcal{P} \neq \mathcal{NP}$)

Nota: Si può assumere che nella formula MAX-3-SAT ogni clausola abbia esattamente 3 letterali e che per MAX-2-SAT ogni clausola possa avere uno o due letterali.

Esercizio 6 - 20 punti

Si consideri la seguente variante del problema dello SCHEDULING a minimo makespan visto a lezione

SCHEDULING WITH DIFFERENT MACHINES (SC2M)
Input: Un insieme di n job, m processori lenti e k processori veloci. ogni job j, impiega tempo t_j se eseguito su un processore lento e tempo $\frac{1}{2}t_j$ su un processore veloce.
Output: Assegnamento dei job ai processori in modo da minimizzare il makespan, cioè il massimo tempo di esecuzione tra tutti i processori.

Si dimostri che SC2M $\in \mathcal{APX}$

Suggerimento: Si può considerare l'assegnamento *greedy* usato dall'algoritmo visto a lezione e analizzarlo in maniera analoga per valutare l'approssimazione garantita dall'algoritmo che si ottiene.
