ALGORITMI - Complessità **Anno Accademico 2021-22**

Simulazione d'Esame 2022

Nome e Cognome: Matricola:

Modalità di svolgimento. Vanno svolti: l'esercizio 1; uno solo a scelta tra gli esericizi 2 e 3; uno solo a scelta tra gli esercizi 4 e 5.

Valutazione. Per la sufficienza è necessario rispondere correttamente ad almeno 4 domande nell'esercizio 1.

Importante: (i) Scrivere in modo **leggibile**; (ii) Ogni affermazione non ovvia va opportunamente giustificata; (iii) Le riduzioni vanno spiegate dettagliatamente.

Esercizio 1 - 20 punti

Si assuma che $NP \neq co{-}NP$.

Per ognuna delle seguenti affermazioni si dica se essa è Vera, Falsa o "Irrisolta". Per "Irrisolta" si intende che per quanto si sa, non risulta provata nè l'affermazione, nè la sua falsità, anche se si fa l'assunzione di cui sopra.

Si motivi brevemente ogni risposta. Risposte non motivate saranno valutate nulle.

1. $P \neq NP$.
2. $TIME(2^{O(n)}) = NTIME(2^{(O(n)})$
3. $P = APX$
4. VERTEX COVER $\notin$ co$-NP$
5. $NTIME(\log n) \subseteq P$
6. $APX \neq PTAS$
7. QSAT $\notin P$
8. SMALL FACTOR $\notin P$

Esercizio 2 - 15 punti

Si dimostri che il seguente problema è $\mathcal{NP}$-completo usando una riduzione da VERTEX COVER.

CLIQUE AND INDEPENDENCE (C&I)

Input: un grafo $G = (V, E)$ non orientato ed un valore k.

Output: *yes* se e solo se esistono due insiemi di vertici C e I entrambi di taglia k tali che C è una clique e I è un independent set.

Esercizio 3 - 20 punti

Data una stringa di parentesi tonde diciamo che le parentesi sono *propriamente accoppiate* se possono essere partizionate in coppie in modo tale che ogni coppia contenga una parentesi aperta che precede la sua corrispondente parentesi chiusa.

Esempio: Le parentesi in $\mathbf{s_1} = (((())())()())$ sono propriamente accoppiate, mentre le parentesi in $\mathbf{s_2} = ())()()(()$ non sono propriamente accoppiate.

Si dimostri che, sotto l'assunzione $P \neq NP$ il seguente problema non è NP-completo.

PARENTESIZZAZIONE
Input: Una stringa di parentesi tonde aperte e chiuse
Output: *yes* se e solo se le parentesi sono propriamente accoppiate

Domanda Bonus: Si dimostri che PARENTESIZZAZIONE appartiene alla classe $SPACE(\log n)$.

Esercizio 4 - 15 punti 2-APPROX

Il problema (di ottimizzazione) MAXCUT consiste nel partizionare i vertici di un grafo $G = (V, E)$ in due sottinsiemi A e B in modo da massimizzare il numero di archi $(a, b) \in E$ tali che $a \in A$ e $b \in B$.

Si consideri il seguente algoritmo greedy: siano $v_1, v_2, \ldots, v_n$ i vertici in V elencati in un ordine arbitrario. Si metta v_1 in A. Quindi, per ogni $i = 2, \ldots, n$ si contino gli archi da v_i verso i vertici già messi in A, sia a_i il loro numero; si contino gli archi da v_i verso i vertici già messi in B sia b_i il loro numero; se $a_i > b_i$ allora si aggiunga v_i a B altrimenti si aggiunga v_i ad A.

Si dimostri che questo algoritmo garantisce un'approssimazione 2 per MAX CUT.

Exercise 5 - 20 points

Definire esattamente il problema di ottimizzazione TSP ed il corrispondente problema di decisione, qui indicato con TSPDEC.

1. Supponiamo di avere un algoritmo polinomiale `TSPDec-Algo` che riesce a risolvere istanze del problema di decisione TSP-DEC in tempo $O(n^3)$ dove n rappresenta il numero di vertici del grafo.

   (a) Si progetti ed analizzi un algoritmo polinomiale che determina il peso di un tour ottimo.

   (b) Si progetti ed analizzi un algoritmo polinomiale che trova un tour ottimo.

2. Supponiamo di avere un algoritmo di 3-approssimazione per TSP. Cosa possiamo concludere circa la complessità di TSPDEC?
