Algoritmi e Strutture Dati (modulo II) - testo prova scritta 9/09/2025

docente: Luciano Gualà

Cognome:..... Nome:..... Matr.:.....

Esercizio 1 [11 punti] Si consideri il problema del calcolo del minimo albero ricoprente (*Minimum Spanning Tree problem)* che prende in input un grafo non orientato e pesato $G = (V, E, w)$ di n nodi.

1. Dire quale delle seguenti affermazioni è vera:

- Se tutti gli archi hanno peso 1, allora l'algoritmo di Prim applicato su un nodo iniziale s calcola un MST che è necessariamente anche un albero dei cammini minimi con sorgente s.
- Si assuma che per ogni arco e vale $w(e) \in \{1, 2\}$, e sia T un MST di G. Allora ogni arco del grafo che non appartiene a T deve avere peso 2.
- Sia T un MST di G e sia f un arco che non appartiene a T, allora l'aggiunta di f a T forma un ciclo e tutti gli archi del ciclo hanno un peso che è minore o uguale a quello di f.
- Sia T un MST di G e sia e un arco di T, allora l'arco e è l'arco più leggero di almeno un ciclo in G.
- Se G ha $\Theta(n\sqrt{n})$ archi, allora l'algoritmo di Kruskal che implementa la Union-Find con la *QuickFind* con euristica *union by size* ha complessità lineare, ovvero $\Theta(n\sqrt{n})$.

2. Si consideri la seguente affermazione e si dica se è (sempre) vera o (se può essere) falsa, argomentando la risposta. (Max 5 righe.)

Claim: Sia $G = (V, E, w)$ un grafo non orientato e pesato. Sia T un MST di G e sia f un arco non in T. Si consideri il grafo $G' = (V, E, w')$ ottenuto da G alzando il peso dell'arco f a un valore $w'(f) > w(f)$. Allora T è un MST anche di G'.

Esercizio 2 [11 punti] Si consideri il problema dell'*Interval Partitioning*.

1. Si definisca formalmente il problema. (Max 5 righe.)
2. Si motivi perché un algoritmo greedy che ordina gli intervalli per finish time non trova la soluzione ottima. (Max 5 righe.)
3. Si descriva invece l'algoritmo greedy ottimo per il problema discutendone la complessità computazionale (non si discuta invece la correttezza). (Max 5 righe.)

Esercizio 3 [11 punti] Devi assegnare n job, numerati da 1 a n, a due macchine con l'obiettivo di minimizzare il costo totale. Ogni job può essere eseguito da entrambe le macchine, ma le politiche di costo delle due macchine sono diverse. In particolare, la macchina A non ha costi fissi e può eseguire il generico job i ad un costo di a_i. La macchina B, invece, ha dei costi per job più bassi ma ha dei costi fissi che dipendono dal numero totale di job che decidi di assegnare alla macchina. Più precisamente, il costo della macchina B per eseguire il job i è b_i (e vale sempre $b_i \leq a_i$). Però devi pagare un costo pari a c_k se decidi di assegnare k job in totale alla macchina B. Ovviamente vale $c_1 \leq c_2 \leq \cdots \leq c_n$.

Progettate un algoritmo di programmazione dinamica che calcoli il costo minimo a cui è possibile eseguire tutti i job. Si discuta la complessità temporale dell'algoritmo proposto.