Algoritmi e Strutture Dati (modulo II) - testo prova scritta 2/2/2026

docente: Luciano Gualà

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

Esercizio 1 [11 punti] Si consideri il problema del calcolo del massimo (*Max Flow problem)* in una rete di n nodi e m archi $G = (V, E, s, t, c)$ con sorgente s e pozzo t e capacità intera $c(e)$ per ogni arco $e \in E$.

1. Dire quale delle seguenti affermazioni è vera:

- Dato un flusso f di G, c'è sempre un s-t-taglio (A, B) la cui capacità è uguale a $v(f)$.
- L'algoritmo di Ford-Furkerson ha una complessità che in generale può essere esponenziale nella dimensione dell'istanza, ma è sempre polinomiale quando le capacità degli archi sono valori interi non più grandi di n^2.
- Se per ogni arco e $c(e) \geq \beta$, allora ogni augmanting step aumenta il flusso corrente di almeno β.
- Dato un flusso f, se nella rete residua G_f c'è un cammino da s a t, allora f non è massimo.
- Se G ha $\Theta(n\sqrt{n})$ archi, e le capacità degli archi sono tutte al più 2, allora l'algoritmo di Ford-Fulkerson ha complessità lineare, ovvero $O(n\sqrt{n})$.

2. Sia $G = (V, E, s, t, c)$ una rete di flusso di n nodi e m archi con capacità intere. Immaginate di aver già calcolato un flusso massimo f per G. Ora vi danno la possibilità di aumentare di una unità la capacità di un arco a vostra scelta. Mostrate che non è sempre possibile aumentare il valore del flusso massimo. E fornite un algoritmo di complessità $O(n + m)$ che decide se è possibile farlo o meno. (Max 5 righe.)

Esercizio 2 [11 punti] Si consideri il problema del calcolo del minimo albero ricoprente (*Minimum Spanning Tree problem)*.

1. Si definisca formalmente il problema. (Max 5 righe.)
2. Si enunci formalmente la proprietà del taglio (*cut property*). (Max 5 righe.)
3. Si fornisca una dimostrazione della proprietà del taglio. (Max 5 righe.)

Esercizio 3 [11 punti] *(La linea della metro D)*

Nella vostra città si scava per realizzare la linea D della metropolitana. A voi hanno affidato un particolare lavoro, quello di scavare in meno tempo possibile un tunnel che vi porti da una parte all'altra di due zone sotterranee che sono separate da una mole di terra che può essere rappresentata da una matrice di con n righe e m colonne. Voi vi trovate nella casella $(1, 1)$ e dovete arrivare in una qualsiasi casella della colonna m. Se siete in una casella (i, j), la tecnologia del vostro scavatore vi permette di scavare ogni volta di una casella o verso destra (che libererà la casella $(i, j + 1)$) o verso il basso (che libererà $(i + 1, j)$). Liberare una casella richiede esattamente un'ora .

Ora, si sa che il sottosuolo della vostra città è ricco di reperti del passato, che è una cosa bella, in generale. Ma non per voi. Infatti, ogni volta che scavando dissotterrate un reperto, questo va catalogato dagli archeologi e la cosa richiede tempo e voi dovete aspettare prima di riprendere a scavare. I reperti possono essere di 3 tipi: anfora (A), bifora (B), colonna dorica (C). Catalogare un'anfora richiede 3 ore mentre catalogare una bifora ne richiede 10. Se invece vi imbatteste in una colonna dorica... be', peggio per voi: perché in questo caso bisognerebbe interrompere i lavori per il rischio di rovinare una potenziale valle dei templi.

Per fortuna avete a disposizione un macchinario che prima di scavare è stato in grado di capire cosa c'è in ogni casella. Progettate un algoritmo di programmazione dinamica che, presa la matrice, calcoli il tunnel migliore da scavare e il tempo necessario per farlo.