Grafi

Esercizio 1

Si consideri una scacchiera $M[1..n, 1..n]$ di $n \times n$ elementi, con $n > 1$. Ogni cella della scacchiera può contenere il valore 0 oppure 1. Se $M[i, j] = 0$, allora la cella (i, j) è libera, altrimenti è occupata da un ostacolo. Un giocatore viene posto inizialmente nella casella $(1, 1)$, che si assume essere sempre libera. Ad ogni passo, il giocatore può muoversi in una delle caselle libere adiacenti. Specificamente, è possibile spostarsi dalla cella $M[i, j]$ ad una delle celle libere tra $M[i-1, j]$, $M[i+1, j]$, $M[i, j-1]$ e $M[i, j+1]$. Per semplicità, supponiamo che $M[i, j]$ sia definita per qualsiasi valore di i e j, e $M[i, j] = 1$ se almeno uno degli indici non è compreso tra 1 e n; quindi ad esempio, $M[0, 1] = M[1, 0] = 1$.

- Descrivere un algoritmo efficiente in grado di determinare il numero minimo di passi necessari per spostarsi dalla cella $(1, 1)$ alla cella (n, n), sempre che ciò sia possibile.
- Determinare il costo computazionale dell'algoritmo di cui al punto (1).

Esercizio 2

Si consideri un grafo non orientato $G = (V, E)$, non necessariamente connesso, in cui tutti gli archi hanno lo stesso peso positivo $\alpha > 0$.

- Scrivere un algoritmo efficiente che, dato in input il grafo $G = (V, E)$ di cui sopra e due nodi u e v, calcola la lunghezza del cammino minimo che collega u e v. Se i nodi non sono connessi, la distanza è infinita.
- Analizzare il costo computazionale dell'algoritmo proposto al punto (1).

Esercizio 3

Un **Euler tour** di un grafo diretto fortemente connesso $G = (V, E)$ è un ciclo che attraversa ogni arco di G esattamente una volta, anche se può visitare i vertici più di una volta.

- Mostrare che G ha un Euler tour se e solo se in-degree(v)=out-degree(v) $\forall v \in V$.
- Descrivere un algoritmo dal costo $O(E)$ che trova un Euler tour di G (se esiste).

Programmazione Dinamica

Esercizio 1

Supponiamo di avere n files aventi rispettivamente dimensione $F[1], F[2], \ldots, F[n]$; le dimensioni sono numeri interi strettamente positivi e sono espresse in MB. Disponiamo di un CD-ROM avente capacità 650 MB; sfortunatamente, il CD-ROM potrebbe non essere sufficientemente capiente per memorizzare tutti gli n files.

- Scrivere un algoritmo efficiente per determinare il numero massimo di files che è possibile memorizzare sul CD-ROM senza eccederne la capacità. Non è richiesto che l'algoritmo stampi anche quali file memorizzare.
- Analizzare il costo computazionale dell'algoritmo proposto.

Esercizio 2

Dovete affrontare l'esame di Algoritmi e Strutture Dati. L'esame si compone di n domande che valgono rispettivamente $p[1], \ldots, p[n]$ punti. In base alla vostra esperienza, stimate che le domande richiederanno rispettivamente $t[1], \ldots, t[n]$ minuti per essere svolte. Purtroppo il tempo a vostra disposizione è di T minuti, che potrebbe essere inferiore alla somma dei tempi necessari a rispondere a tutte le domande. I punteggi e i tempi sono interi strettamente positivi.

- Scrivere un algoritmo efficiente che, dati i vettori $p[1..n]$, $t[1..n]$ e il valore di T, restituisce il punteggio massimo che potete ottenere rispondendo correttamente ad un opportuno sottoinsieme delle n domande entro il tempo massimo di T minuti.
- Calcolare il costo computazionale dell'algoritmo proposto.