28 Giugno 2018

In vista del prossimo appello di Algoritmi e principi dell'informatica si propongono i seguenti esercizi da completare in **1 ora e 15 minuti.**

Esercizio 1 (punti 7)
Si descrivano una MT a nastro singolo e una macchina RAM che riconoscono il linguaggio fatto da stringhe con lunghezza n divisibile per 3 su alfabeto {0, 1}, in cui l'ultimo carattere e quello in posizione 2n/3 sono uguali. Se ne valutino le complessità spaziali e temporali con tutti i criteri di costo applicabili.

Esercizio 2 (Punti 9)
Si consideri un insieme di N elementi *diversi tra loro* appartenenti a un insieme A.
E' definita una relazione d'ordine totale '<' su A. Dati due elementi qualunque e_1 e e_2, diciamo che e_1 è *migliore* di e_2, se $e_1 < e_2$.
Si vogliono determinare i migliori elementi dell'insieme.

1) Descrivere in modo succinto ma preciso un algoritmo che stampi i migliori 10 elementi dell'insieme (si assuma anche che N>10).
2) Valutare la complessità asintotica, in funzione di N, dell'algoritmo descritto al punto 1.

Si consideri ora, in aggiunta, un intero positivo k.

3) Descrivere in modo succinto ma preciso un algoritmo che stampi i migliori k elementi dell'insieme (si assuma anche che $N > k$).
4) Valutare la complessità asintotica, in funzione di N e k, dell'algoritmo descritto al punto 3.

Tracce di soluzioni

Esercizio 1
La MT a nastro singolo fa una serie di n scansioni dell'intero nastro marcando da sinistra 2 caselle e da destra 1; la lunghezza è divisibile per 3 se l'ultima casella da destra marcata è adiacente all'ultima casella da sinistra, che identifica la posizione 2n/3. A questo punto confronta il contenuto col carattere finale. $T(n) = \Theta(n^2)$, $S(n) = n$.

RAM, variante semplice. La RAM memorizza la stringa, calcolandone la lunghezza con un contatore; controlla che sia divisibile per 3; calcola la posizione 2n/3 e ne confronta il contenuto con quello dell'ultima.
Costo costante: $T(n) = \Theta(n)$, $S(n) = \Theta(n)$
Costo logaritmico: $T(n) = \Theta(n \log n)$, $S(n) = \Theta(n + \log n) = \Theta(n)$

Variante che ottimizza la complessità spaziale. Viene letta la stringa, tenendo: un contatore (M[1]) per la lunghezza, e una codifica binaria della stessa in M[2] (*2 per ogni 0, *2+1 per ogni 1 letti), inizializzato con 1 per evitare problemi con eventuali sequenze di 0 iniziali. Si memorizza poi l'ultimo simbolo ricevuto in un M[3] e si divide M[1] per 3. Si effettua dunque un ciclo, decrementando M[1] ad ogni passo, in cui M[2] viene diviso per 2: se il numero risultante è divisibile per 2, M[3] = 0, altrimenti deve essere M[3] = 1.
Costo costante: $T(n) = \Theta(n)$, $S(n) = \Theta(1)$
Costo logaritmico: $T(n) = \Theta(n^2)$, $S(n) = \Theta(n)$ infatti in M[2] c'è un numero di n cifre binarie.

Esercizio 2
Una prima semplice soluzione consiste nell'ordinare l'insieme degli N elementi secondo l'ordinamento indotto da $<$. Una volta ordinato l'insieme, basta quindi stampare i primi 10 elementi dell'insieme ordinato. L'ordinamento ha un costo $\Theta(N \log N)$ e le altre operazioni sono a costo costante. Complessivamente l'algoritmo ha un costo di $\Theta(N \log N)$.

Una soluzione più efficiente consiste nel mantenere un BST di al più 10 elementi.
Si leggono, in sequenza, i vari elementi dell'insieme dato. I primi 10 elementi vengono inseriti nel BST. Dall'11° in poi, un elemento viene inserito nel BST solo se è migliore del peggiore tra gli elementi del BST, e in tale caso, prima dell'inserimento, l'elemento peggiore viene cancellato.
Alla fine della lettura degli elementi dell'insieme, il BST conterrà i migliori 10 elementi.
La complessità di questa soluzione è determinata dalla scansione dell'insieme di N elementi e dal costo dell'inserimento e della cancellazione di un elemento nel BST. Come noto, tali operazioni sui BST hanno una complessità $O(\log n)$, dove n è la dimensione del BST, se si usano alberi rosso-neri.
Nel caso del punto 2, n=10 è costante, pertanto anche la complessità di queste operazioni è costante.
Complessivamente, la complessità è $\Theta(N)$.

Le stesse soluzioni possono essere adottate per i punti 3 e 4.
In particolare, nel primo caso la complessità risultante è $\Theta(k + N \log N)$ poiché, oltre all'ordinamento dell'insieme, occorre poi scandire i primi k elementi dell'insieme ordinato, che ha un costo $\Theta(k)$. Siccome però $k < N$, $\Theta(k + N \log N) = \Theta(N \log N)$.
Nel secondo caso, invece, il costo complessivo è $\Theta(N \log k)$, se si fa uso di un albero rosso-nero poiché le operazioni di inserimento e cancellazione hanno costo $\Theta(\log k)$, dal momento che l'albero ha k elementi. Se però il BST non è anche rosso-nero, allora nel caso pessimo il costo diventa $\Theta(N \cdot k)$.