Problem Set 4

docente: Luciano Gualà

Esercizio 1 *(Aggiungendo un'operazione a una Pila)*
Progettare una struttura dati che implementa un tipo di dato *Pila* che mantiene una sequenza di elementi con chiave e che, oltre le classiche operazioni di `Top`, `Pop` e `Push`, consente un'operazione aggiuntiva chiamata `Min`. Tale operazione restituisce il puntatore all'elemento di chiave minima contenuto nella pila. Tutte le operazioni devono avere complessità temporale $O(1)$ nel caso peggiore.

Esercizio 2 *(Un dizionario un po' più ricco)*
Progettare una struttura dati che implementa un tipo di dato *Dizionario* che, oltre le classiche operazioni di `Search`, `Insert` e `Delete`, fornisce anche le seguenti due operazioni aggiuntive:

- `ElementOfRank`(i): dato un intero i, ritorna il puntatore all'elemento di *rango* i, ovvero l'i-esimo minimo contenuto nel dizionario.
- `RankOf`(x): dato il puntatore x a un elemento, ritorna il rango dell'elemento puntato da x, ovvero la posizione dell'elemento nella sequenza ordinata (in ordine crescente di chiave) degli elementi nel dizionario.

Tutte le operazioni devono avere complessità temporale $O(\log n)$ dove n è il numero di elementi presenti nel dizionario.

Suggerimento: si modifichi un albero AVL aggiungendo a ogni nodo v un campo $size(v)$ che contiene il numero di nodi presenti nel sottoalbero radicato in v. Come è possibile usare tale informazione aggiuntiva per implementare le due nuove operazioni? E' possibile mantenere efficientemente questa informazione aggiuntiva?

Esercizio 3 *(Ancora sugli oracoli: il problema del* Minimum Range Query*)*
Sia A un vettore di n valori reali. Progettare un algoritmo che, dato A, costruisca un *oracolo* (ovvero una struttura dati) che sia in grado di rispondere in tempo $O(1)$ a *query* (ovvero domande) del seguente tipo: dati due interi i, j, calcolare l'indice dell'elemento di valore minimo nella porzione $A[i; j]$ del vettore.

Si noti che una soluzione semplice al problema è quella di precalcolare tutte le risposte alle $\Theta(n^2)$ query e memorizzarle in una matrice. In questa soluzione, però, l'oracolo (ovvero la matrice delle risposte) ha dimensione $\Theta(n^2)$. Vogliamo, invece, fare meglio in termini di memoria occupata dall'oracolo la cui dimensione richiediamo essere $O(n \log n)$. Non imponiamo invece nessun vincolo sulla complessità temporale necessaria per costruire l'oracolo.

Suggerimento: un'idea potrebbe essere quella di memorizzare solo le risposte a un sottoinsieme delle $\Theta(n^2)$ query. Tale sottoinsieme deve avere dimensione $O(n \log n)$ e deve comunque consentire di rispondere a una generica query in tempo costante.