Problem Set 3

docente: Luciano Gualà

Esercizio 1 *(una domanda semplice ma non troppo)*
Dire se può esistere un algoritmo di ordinamento basato su confronti che ordina un insieme di 8 elementi facendo nel caso peggiore al più 15 confronti. Motivare la risposta.

Esercizio 2
Siano dati n punti disposti sul piano Euclideo, dove il punto p_i ha cordinate (x_i, y_i), $i = 1, \ldots, n$. Si progetti un algoritmo che, preso in input l'insieme degli n punti, un valore $k \in \{1, 2, \ldots, n\}$ e un ulteriore punto *target* t di coordinate (x_t, y_t), restituisca i k punti dell'insieme che sono più vicini a p_t (rispetto alla distanza euclidea). L'algoritmo deve avere complessità temporale (nel caso peggiore) $O(n + k \log n)$.

Esercizio 3
Sia A una matrice $n \times n$ di interi. Progettare un algoritmo che, data 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 quattro interi i, j, b, h, quale è la somma degli elementi della sottomatrice di A delimitata dai quattro elementi $A[i, j], A[i, j + b - 1], A[i + h - 1, j], A[i + h - 1, j + b - 1]$?

L'algoritmo di costruzione dell'oracolo deve avere complessità temporale $O(n^2)$, mentre l'algoritmo di interrogazione dell'oracolo, come già detto, deve avere complessità temporale $O(1)$.

Esercizio 4 *(tagliare un vettore in streaming)*
Vogliamo studiare il seguente problema: dato un array $A[1; n]$ di n bit, ovvero dove $A[i] \in \{0, 1\}$, per ogni i, vogliamo trovare un punto dove tagliare l'array in modo che la metà di sinistra contenga un numero di zeri uguale al numero di uni nella metà di destra. Più formalmente, vogliamo trovare un indice k (ammesso che esista), tale che la porzione $A[1; k]$ contenga un numero di zeri pari al numero di uni nella porzione $A[k + 1, n]$. Permettiamo anche i due tagli degeneri che corrispondono a $k = 0$ e a $k = n$, in tal caso assumiamo che $A[1; 0]$ e $A[n + 1; n]$ contengono 0 zeri e 0 uni.

Vogliamo progettare un algoritmo con complessità temporale $O(n)$ che risolve il problema in uno scenario in cui il vettore ha una dimensione tale da non poter essere mantenuto in memoria interamente. In particolare, assumiamo che possiamo usare solo una quantità di memoria costante (rispetto a n) e che l'unico modo per leggere il vettore è chiedere che ci vengano snocciolati uno alla volta, da sinistra a destra, i suoi elementi. Un algoritmo così fatto è spesso chiamato algoritmo in *streaming*, perché può leggere l'input una sola volta come se fosse un flusso di dati. Riassumendo, quindi, vogliamo progettare un algoritmo con complessità temporale $O(n)$, memoria ausiliaria $O(1)$, e che può leggere una sola volta il vettore A da sinistra a destra. Si argomenti in modo preciso la correttezza dell'algoritmo.