Problem Set 2

docente: Luciano Gualà

Esercizio 1 (*equazioni di ricorrenza*)
Si risolvano le seguenti equazioni di ricorrenza. Si assuma sempre $T(1) = 1$.

(*a*) $T(n) = T(n-10) + 10$.

(*b*) $T(n) = T(n/2) + 2^n$.

(*c*) $T(n) = T(n/3) + T(n/6) + n^{\sqrt{\log n}}$.

(*d*) $T(n) = T(\sqrt{n}) + \Theta(\log\log n)$.

(*e*) $T(n) = T(n/2 + \sqrt{n}) + \Theta(1)$.

(*f*) $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

Esercizio 2 (*ricerca di un picco in una dimensione*)
Sia $V[1 : n]$ un vettore di n numeri non negativi. Un *picco* in V è una posizione $p \in \{1, \ldots, n\}$ il cui elemento ha a sinistra e a destra elementi non più grandi di lui, ovvero $V[p] \geq \max\{V[p-1], V[p+1]\}$. Quando p è sul bordo, la condizione è da considerarsi relativa solo all'unico elemento esistente adiacente a p. Pertanto, per semplicità e in modo equivalente considereremo $V[i] = -\infty$, quando $i < 1$ o $i > n$. Progettare un algoritmo che, dato V, trovi un picco in tempo $o(n)$.

Esercizio 3 (*ricerca di un picco in due dimensioni*)
Sia $M[1 : n, 1 : m]$ una matrice con n righe, m colonne, tale che $M[i, j]$ è un numero non negativo. Un *picco* in M è una posizione (p, q), $p \in \{1, \ldots, n\}$ e $q \in \{1, \ldots, m\}$ il cui elemento ha a sinistra, a destra, sopra e sotto elementi non più grandi di lui, ovvero $M[p, q] \geq \max\{M[p-1, q], M[p+1, q], M[p, q-1], M[p, q+1]\}$. Quando (p, q) è sul bordo, la condizione è da considerarsi relativa solo agli elementi esistenti adiacenti a (p, q). Pertanto, per semplicità e in modo equivalente considereremo $M[i, j] = -\infty$, quando uno dei due indici i o j è minore di 1 o maggiore di n.

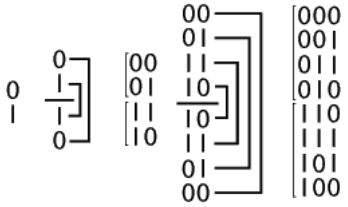
- Si consideri il seguente algoritmo che essenzialmente esegue una ricerca di un picco spostandosi sempre su elementi più grandi. Più in dettaglio, l'algoritmo parte da una data posizione (i, j). Se tale elemento non è un picco si controlla gli (al più 4) elementi adiacenti a (i, j), si sposta sull'elemento di valore massimo, e procede ricorsivamente. Si dimostri se tale algoritmo è corretto e se ne studi la complessità asintotica nel caso peggiore.

- Si progetti un algoritmo che trovi un picco in M con complessità temporale $o(nm)$ e che usi la tecnica *divide et impera*.

Esercizio 4 (*sul codice Gray*)
Un *codice Gray* di lunghezza n è costituito da tutte le 2^n sequenze di n bit e consente di rappresentare tutti gli interi da 0 a $2^n - 1$. La caratteristica di un tale codice è che le codifiche di due valori consecutivi differiscono di esattamente un bit. Un esempio è mostrato in figura. E' possibile costruire ricorsivamente un codice gray a n bits partendo da uno ad $n-1$ bits nel seguente modo (si veda la figura):

i	Gray(*i*)
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

(a) esempio condice Gray con n=4 bit



(b) Generazione codice per riflessione

Figura 1:

- Le prime 2^{n-1} parole del codice ad n bits sono uguali a quelle del codice ad $n-1$ bits estese con uno 0 come bit più significativo;
- Le seconde 2^{n-1} parole del codice ad n bits sono uguali a quelle del codice ad $n-1$ bits ma elencate in ordine inverso (riflesso) ed estese con un 1 come bit più significativo.

Si progetti un algoritmo ricorsivo che, dato n e un indice $i \in \{0, 1, \ldots, 2^n - 1\}$, calcoli la codifica di i in tempo $O(n)$.