Problem Set 1

docente: Luciano Gualà

Esercizio 1 (*notazione asintotica*)
Siano $f(n), g(n), h(n)$ tre funzioni asintoticamente positive. Inoltre, sia $c > 1$ una costante reale positiva. Si dimostrino o confutino le seguenti affermazioni:

1. $2^{f(n)+2^c} = \Theta(2^{f(n)})$.
2. $g(n) = \Theta(1)$ implica $2^{f(n)+g(n)} = O(2^{f(n)})$.
3. $g(n) = o(f(n))$ implica $2^{f(n)+g(n)} = O(2^{f(n)})$.
4. $f(n) + g(n) + h(n) = \Theta(\max\{f(n), g(n), h(n)\})$.
5. $f(n) = \Theta(\log n)$ implica $\log n^{f(n)} = O(\log^c n^{g(n)})$.
6. $f(n) = \Theta(f(c \cdot n))$.
7. $f(n) = \Theta(f(c + n))$.

Esercizio 2 (*trovare l'intero mancante*)
Sia $A[1 : n]$ un vettore ordinato di n interi distinti compresi fra 1 e $n + 1$. Chiaramente A contiene tutti gli elementi dell'insieme $\{1, 2, \ldots, n+1\}$ tranne uno. Progettare un algoritmo con complessità temporale $o(n)$ che trova l'elemento mancante.

Esercizio 3 *(Uno strano algoritmo di ordinamento)*
L'algoritmo ricorsivo `gualasort` è un algoritmo di ordinamento ricorsivo, la cui chiamata iniziale `gualasort`$(V, 1, n)$ ordina il vettore $V[1; n]$ di n numeri su cui è chiamato. O almeno si spera. Dimostrate la correttezza dell'algoritmo e discutetene la sua complessità computazionale.

Algorithm 1: `gualasort`(V, i, j)

```
N ← j − i + 1 ;
if N = 2 then
    if V[i] > V[i + 1] then
        scambia V[i] e V[i + 1]
if N ≥ 2 then
    m1 = i + ⌊N/3⌋ ;
    m2 = i − 1 + ⌈2/3 N⌉ ;
    gualasort(V, i, m2);
    gualasort(V, m1, j);
    gualasort(V, i, m2);
```

Esercizio 4 *(Un gioco di rimozione di pedine)*
Il gioco si gioca con un vettore di n elementi. Inizialmente, ogni posizione del vettore può essere vuota o contenere una pedina bianca o contenere una pedina nera. Ad ogni turno potete eseguire una mossa del seguente tipo: potete prendere due pedine di diverso colore che si trovano in posizioni adiacenti e rimuoverle dal vettore (e quindi le posizioni prima occupate da queste due pedine dopo la mossa diventano vuote). Il gioco termina quando non potete più fare nessuna mossa. Il vostro obiettivo è rimuovere il maggior numero possibile di pedine dal vettore.

Progettate un algoritmo veloce che, presa un'istanza del gioco, calcola una strategia ottima, ovvero una sequenza di mosse che rimuove il massimo numero di pedine. Dimostrate la correttezza dell'algoritmo e fornitene l'analisi della complessità computazionale.