Esercizio 0

La seguente funzione `Part()` dovrebbe essere usata nel Quick-Sort per effettuare la partizione ma è errata (si ricorda che `Part()` dovrebbe ritornare un intero k con $a \leq k \leq b$ tale che ogni valore in $A[a...k]$ è minore od uguale a tutti i valori in $A[k+1...b]$). Esibire un esempio su cui questa versione di `Part()` sbaglia e poi proporre una piccola modifica allo pseudocodice che lo renda corretto.

```
Part(A,a, b)
input: A è un array di interi, a e b due indici che individuano inizio e fine della
porzione dell'array su cui Part lavora
output: intero k con a ≤ k ≤ b tale che ogni valore in A[a...k] è minore od uguale a
tutti i valori in A[k+1...b]).
k = a, j = b, v = A[a];
while (k < j) {
        if (A[k] > v) {
           SWAP(A[k], A[j])
           j--; }
        else k++; }
}
return k;
```

Esercizio 1.

Un algoritmo di ordinamento che procede per confronti e scambi è ideale se gode delle seguenti proprietà:
Operare in loco, richiedendo solo Θ(1) spazio di memoria extra.
Avere tempo di esecuzione Θ(n log(n)) nel caso peggiore.
Eseguire O(n) scambi.
Essere adattivo (o input sensitive): eseguire l'ordinamento più velocemente quando i dati sono in parte già nell'ordinamento richiesto o quando ci sono molti duplicati.
Nessun algoritmo ha tutte queste proprietà, quali tra queste sono invece presenti nel Merge Sort, nell'heapSort, nell'Insertion Sort e nel Selection Sort?

Esercizio 2.

Sia dato un Max-Heap e si disegni un algoritmo per trasformarlo in un Min-Heap. Si descriva innanzi tutto l'algoritmo proposto illustrandone l'idea in modo da convincere della sua correttezza, infine se ne dia lo pseudocodice ed il tempo di esecuzione asintotico.

Esercizio 3

Progettare un algoritmo che, dati in input due Maxheap H1 e H2, dia in output un unico Maxheap sugli elementi di H1 e di H2. Del- l'algoritmo presentato:
a.si dia la spiegazione a parole
b.si valuti la complessità computazionale c. si fornisca lo psudocodice.

Esercizi di analisi di algoritmi:

1.
```
analizzami(A,i,j)
Input: A è un array e i e j due indici che individuano la porzione di array in input
n=j-i+1
c=1
m = (i + j +1)/2
d=m
while d >1 do c++ ;
        d=d/2
return  c
```

2.
```
fun (array A, int i, int f)
Input: A è un array e i e f due indici che individuano la porzione di array in input
{n = f-i+1
   t=n;
   d = 1
   while t≥1 do {t = t-2; d++;}
 return d
}
```

3.
```
analizzami(int n) c= 1
m = n*n
while m>1 do
          for j=1 to m do c++
          m=m-2
```

3.
```
test (A,lo,hi)
Input: A è un array e lo e hi due indici che individuano la porzione di array in input
if hi ≤ lo then return 1
m = (lo+hi)/2
k=m
a=1
while k ≥ 1 do
          a = 2*a
          k = k/2
return
```

Esercizi analisi asintotica:

Per ogni affermazione seguente, si scriva se è vera o falsa. La verità di un'affermazione va provata, basandosi sulle definizioni di O, Θ e Ω. La falsità va dimostrata con un contro esempio, cioè fornendo le funzioni che falsificano l'affermazione.

1. $f(n)=\Theta(n)$ e $g(n)=\Omega(n) \Rightarrow f(n)g(n)=\Omega(n^2)$
2. 2. $f(n)=\Omega(n)$ e $g(n)=O(n^2) \Rightarrow f(n)/g(n)=O(n)$
3. 3. $f(n)=O(\log n) \Rightarrow 2^{f(n)} = O(n)$