**INTRODUZIONE AGLI ALGORITMI**
**10 Giugno 2019**
**Prof.ssa Calamoneri - Prof.ssa Fachini - Prof.ssa Petreschi**
**Prova completa (2 ore e mezza)**

**Esercizio 1.**
Un algoritmo di ordinamento che procede per confronti e scambi è ideale se gode delle seguenti proprietà:

- Operare in loco, richiedendo solo $\Theta(1)$ spazio di memoria extra.
- Avere tempo di esecuzione $\Theta(n \log(n))$ 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'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.**
Siano dati 2 alberi bilanciati (AVL o rosso/neri), $T_1$ e $T_2$. Si costruisca un unico albero bilanciato (AVL o rosso/nero) $T$ , dato dall'unione di $T_1$ e $T_2$. Gli alberi sono memorizzati con strutture a puntatori, ogni nodo ha quattro campi:

- *left*, per il puntatore al figlio sinistro,
- *right*, per il puntatore al figlio destro,
- *key*, per la chiave e
- *FB*, per il fattore di bilanciamento oppure *col* per il colore.

Si illustri l'idea algoritmica in modo da convincere della sua correttezza ne se ne analizzi il tempo di esecuzione asintotico nei casi migliore e peggiore.

**Per chi deve fare solo la prima parte** (2 ore)**:**

**Esercizio 1.1.**

Dimostrare la verità o falsità delle seguenti affermazioni:

*f(n) = n(2n²+3n +4)* è *O(n⁴)*

f(n) = *n(n+1)/2* è $\Omega(n)$

f(n) = $n^5 - n^2$ è $\Theta(n^5)$

**Esercizio 2.1.**

Un algoritmo di ordinamento si dice stabile quando non modifica le posizioni relative di elementi duplicati. Il selection sort non è stabile, si mostri con un esempio come mai. Si modifichi l'algoritmo in modo che sia stabile, mantenendo come criterio di ordinamento la scelta del minimo tra i rimanenti da ordinare.

**Esercizio 3.1.**

Siano dati *A(1,…n)*, vettore ordinato di interi, e *x*, valore intero. Trovare un algoritmo per vedere se ci sono due numeri la cui somma sia uguale ad *x*.

**Per chi deve fare solo la seconda parte** (2 ore)**:**

**Esercizio 1.2**

Si confrontino gli ABR e gli AVL e si spieghi cosa cambia per:

1. il tempo di esecuzione delle operazioni principali (ricerca, inserimento e cancellazione)
2. lo spazio occupato in memoria
3. la costruzione della struttura a partire da un array qualunque.

**Esercizio 2.2**

Si scriva la relazione di ricorrenza che esprime il tempo di esecuzione della seguente funzione e la si risolva usando il metodo della sostituzione.

```
fun(A,i,j) // A è un array di interi
n =  j - i +1
if n ≤ 4 then return (A[n]);
x = 0;
for k = 1 to 4 do
        for h = 1 to n − k do A[h] = A[h] − A[n − h];
       m= i + n/4
      x = x+fun(A,i,m);
return x
```

**Esercizio 3.2**

Il costo di un cammino radice-foglia è la somma delle chiavi dei nodi del cammino. Si scriva una algoritmo che dà in output il costo del cammino più costoso. Si descriva 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.