

Algoritmi e Strutture Dati 1

Appello del 31/05/2010

Compito n° 1

Esercizio 1

I numeri di Fibonacci sono 0,1,1,2,3,5,8,13,21,34,... Più formalmente, $F(0)=0$, $F(1)=1$ e $F(n)=F(n-1)+F(n-2)$ per ogni n da 2 in poi.

Scrivere un metodo iterativo `public static boolean isFibonacci (int [] A)` in Java che preso in input un array **A** di n numeri interi, restituisce in output *true* se **A** contiene tutti e soli i primi n numeri di Fibonacci, *false* altrimenti. Ad esempio, se $A=[2,1,0,1,5,3]$, `isFibonacci` restituisce *true*; se $A=[1,0,1,5,3]$ `isFibonacci` restituisce *false*.

Il metodo può sfruttare tutti gli algoritmi visti a lezione (es: *InsertionSort*, *MergeSort*, *Quicksort*, *Binarysearch*, *LinearSearch*, etc...).

Si dia una stima del tempo di esecuzione dell'algoritmo proposto nel caso peggiore, **dimostrando formalmente il risultato ottenuto facendo uso della definizione di notazione asintotica**.

Esercizio 2

Scrivere un metodo ricorsivo `public static long prodotto (int []A, int i, int j)` in Java che, facendo uso della tecnica *divide et impera*, preso in input un array **A** di n numeri interi e due interi i e j , restituisce in output il valore del prodotto di tutti gli interi presenti in **A** tra la posizione i e la posizione j . Inoltre:

- Scrivere la ricorrenza **T(n)** del tempo di esecuzione dell'algoritmo.
- Risolvere la ricorrenza con il metodo di sostituzione e induzione.

INIZIO SECONDO PARZIALE

Esercizio 3

- a. Mostrare l'albero binario di ricerca che si ottiene a partire dall'albero vuoto inserendo nell'ordine indicato i seguenti elementi:
 - **18, 20, 60, 24, 35, 2, 25, 26, 23**
- b. Scrivere un metodo ricorsivo `public static int conto (Node x, int k)` che preso in input un nodo **x** ed un intero **k**, restituisce il numero di nodi nel sottoalbero radicato in **x** aventi chiave minore di **k**.

Esercizio 4

Si consideri una *tabella hash con lista di trabocco* contenente numeri interi, e si spieghi brevemente come una tale tabella possa essere implementata.

Data la funzione hash modulo, si mostri un esempio in cui l'inserimento di **n chiavi intere** $k_1, k_2, k_3, \dots, k_n$, in una tabella con **19** posizioni risulti in una tabella in cui la ricerca di una chiave **k** risulta avere il peggior tempo possibile di esecuzione. In particolare, si fornisca per ogni i da 1 a n il valore di k_i , (in funzione dell'indice i), ed il valore **k** della chiave da cercare.

Esercizio 5

Si implementi in java la classe **SimpleList**, che consiste in una lista singolarmente collegata di **SimpleElem**, in cui gli ogni SimpleElem contiene la chiave intera ed un puntatore al nodo successivo:

```
public class SimpleElem{
    int key;
    SimpleElem next;
    public SimpleElem (int k){
        key = k;
        next = null;
    }
}
```

Si implementino nella classe SimpleList i seguenti metodi:

- **public SimpleList ()** che costruisce una lista vuota.
- **public void Insert (SimpleElem e)** che prende in input un SimpleElem *e*, e lo inserisce in testa alla lista.
- **public boolean Search (int x)** che prende in input un intero *x* e restituisce *true* se *x* è presente nella lista, *false* altrimenti
- **public void Delete(SimpleElem e)** che prende in input un SimpleElem *e*, e lo rimuove dalla lista (se esso è presente)

I metodi non possono sfruttare nessun algoritmo visti a lezione.

Per ogni metodo implementato si stimi la complessità computazionale nel caso peggiore.

Attenzione:

- Scrivere **subito** nome, cognome, matricola e numero del compito su OGNI FOGLIO.
- non è ammesso **per nessun motivo** l'uso di telefoni cellulari, calcolatrici, etc...
- **non** è possibile consultare appunti, libri, dispense.