Algoritmi e Strutture Dati 1

Esonero del 31/03/2009

Compito n° 1

Esercizio 1

Dimostrare che non tutte le funzioni $f: \aleph \to \{0,1\}$ sono calcolabili.

Esercizio 2

Dire se, giustificando la risposta applicando la definizione delle notazioni asintotiche,

a) $n \log n = \theta(n^2)$
b) $n\sqrt{n} = \Omega(n)$

Esercizio 3

Scrivere una procedura **P1** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di A che sono presenti anche in B.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n^2)$, e non deve sfruttare nessuna delle procedure viste a lezione.

Esercizio 4

Scrivere una procedura **P2** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di A che sono presenti anche in B.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n \log n)$, e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 5

Sia $T(n) = \begin{cases} T(n/2) + \Theta(1) & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del teorema generale.

Esercizio facoltativo: si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del metodi di sostituzione e induzione..

Attenzione: non è ammesso per nessun motivo l'uso di telefoni cellulari, calcolatrici, etc…

Algoritmi e Strutture Dati 1

Esonero del 31/03/2009

Compito n° 2

Esercizio 1

Dimostrare che non tutte le funzioni $f: \aleph \to \{0,1\}$ sono calcolabili.

Esercizio 2

Dire se, giustificando la risposta applicando la definizione delle notazioni asintotiche,

a) $n\sqrt{n} = \theta(n^2)$
b) $n \log n = \Omega(n)$

Esercizio 3

Scrivere una procedura **P3** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di B che hanno il loro doppio presente in A.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n^2)$, e non deve sfruttare nessuna delle procedure viste a lezione.

Esercizio 4

Scrivere una procedura **P4** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di B che hanno il loro doppio presente in A.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n \log n)$, e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 5

Sia $T(n) = \begin{cases} 2T(n/2) + \Theta(1) & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del teorema generale.

Esercizio facoltativo: si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del metodi di sostituzione e induzione..

Attenzione: non è ammesso per nessun motivo l'uso di telefoni cellulari, calcolatrici, etc…

Algoritmi e Strutture Dati 1

Esonero del 31/03/2009

Compito n° 3

Esercizio 1

Dimostrare che non tutte le funzioni $f: \aleph \to \{0,1\}$ sono calcolabili.

Esercizio 2

Dire se, giustificando la risposta applicando la definizione delle notazioni asintotiche,

a) $5n + n^2 = \theta(n^2)$
b) $n \log n = O(n)$

Esercizio 3

Scrivere una procedura **P5** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di A che hanno il loro triplo presente in B.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n^2)$, e non deve sfruttare nessuna delle procedure viste a lezione.

Esercizio 4

Scrivere una procedura **P6** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di A che hanno il loro triplo presente in B.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n \log n)$, e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 5

Sia $T(n) = \begin{cases} 2T(n/2) + \Theta(n) & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del teorema generale.

Esercizio facoltativo: si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del metodi di sostituzione e induzione..

Attenzione: non è ammesso per nessun motivo l'uso di telefoni cellulari, calcolatrici, etc…

Algoritmi e Strutture Dati 1

Esonero del 31/03/2009

Compito n° 4

Esercizio 1

Dimostrare che non tutte le funzioni $f: \aleph \to \{0,1\}$ sono calcolabili.

Esercizio 2

Dire se, giustificando la risposta applicando la definizione delle notazioni asintotiche,

a) $n \log n + n^2 = \theta(n^2)$
b) $n + \sqrt{n} = O(n)$

Esercizio 3

Scrivere una procedura **P7** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di B il cui quadruplo è presente in A.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n^2)$, e non deve sfruttare nessuna delle procedure viste a lezione.

Esercizio 4

Scrivere una procedura **P8** in pseudocodice che presi in input due array A e B di interi (si assuma che sia A che B non contengano elementi ripetuti), restituisce in output il numero di elementi di B il cui quadruplo è presente in A.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n \log n)$, e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 5

Sia $T(n) = \begin{cases} T(n/2) + \Theta(1) & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del teorema generale.

Esercizio facoltativo: si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del metodi di sostituzione e induzione..

Attenzione: non è ammesso per nessun motivo l'uso di telefoni cellulari, calcolatrici, etc…

Algoritmi e Strutture Dati 1

Appello del 26/05/2009

Prima parte

Compito n° 1

Esercizio 1

Dire se, giustificando la risposta,

a) $n \log n = \theta(n \log n + n)$
b) $n\sqrt{n} = \Omega(n \log n)$

Esercizio 2

Si supponga di dover ordinare un array di *n* numeri interi contenente numeri da 0 a $n \log \log n$. Si dica qual è la complessità, nel caso medio e nel caso peggiore, degli algoritmi di ordinamento

- Merge Sort
- Quick Sort
- Selection Sort
- Counting sort

Dire quindi, giustificando la risposta, qual è l'algoritmo più conveniente nel caso medio e quale nel caso peggiore.

Esercizio 3

Scrivere una procedura **P1** in pseudocodice che preso in input un array A di numeri interi, restituisce in output il numero massimo di volte per cui un elemento è ripetuto in A.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n \log n)$, e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 4

Sia $T(n) = \begin{cases} T(n/2) + \Theta(1) & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso dei metodi di sostituzione e induzione.

Attenzione:

- Scrivere 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.

Algoritmi e Strutture Dati 1

Appello del 26/05/2009

Seconda parte

Compito n° 1

Esercizio 1

Si implementi in pseudocodice la strutture dati **O*rdered List,*** che consiste in una lista doppiamente collegata in cui gli elementi sono sempre ordinati in ordine crescente, e ogni intero non può essere presente più di una volta.

Si implementino per tale struttura dati le seguenti procedure:

- **Search** che prende in input una lista ordinata *OL* e un intero *x* e restituisce *true* se x è presente in OL, *false* altrimenti
- **Insert** che prende in input una lista ordinata *OL* e un intero *x*, e inserisce l'intero *x* in *OL* (se non è già presente)
- **Remove** che prende in input una lista ordinata *OL* e un intero *x*, e rimuove l'intero *x* da *OL* (se esso è presente)

Per ognuna delle procedure si stimi la complessità computazionale nel caso peggiore.

Esercizio 2

Si consideri una *tabella hash con lista di trabocco* contenente numeri interi.

- Si spieghi brevemente come una tale tabella possa essere implementata.

Supponendo che H(x) sia la funzione hash (già implementata) della tabella,

- scrivere una procedura **P3** che prende in input una tabella hash e restituisce il numero di elementi presenti nella più lunga lista di trabocco della tabella;
- scrivere una procedura **P4** che prende in input una tabella hash ed un intero *x* e, restituisce il più piccolo elemento maggiore di *x* presente in tabella (*NIL* se nessun elemento della tabella è maggiore di *x*).

Attenzione:

- Scrivere 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.

Algoritmi e Strutture Dati 1

Appello del 26/05/2009

Prima parte

Compito n° 2

Esercizio 1

Dire se, giustificando la risposta,

a) $n\sqrt{n} = \theta(n\sqrt{n} + n \ \ )$
b) $n \log n = \Omega(n)$

Esercizio 2

Si supponga di dover ordinare un array di *n* numeri interi contenente numeri da 0 a $n\sqrt{n}$. Si dica qual è la complessità, nel caso medio e nel caso peggiore, degli algoritmi di ordinamento

- Merge Sort
- Quick Sort
- Selection Sort
- Counting sort

Dire quindi, giustificando la risposta, qual è l'algoritmo più conveniente nel caso medio e quale nel caso peggiore.

Esercizio 3

Scrivere una procedura **P2** in pseudocodice che preso in input un array A di numeri interi, restituisce in output il numero minimo di volte per cui un elemento è ripetuto in A.

La procedura deve avere tempo di esecuzione nel caso peggiore $\theta(n \log n)$, e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 4

Sia $T(n) = \begin{cases} 2T(n/2) + \Theta(1) & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso dei metodi di sostituzione e induzione.

Attenzione:

- Scrivere 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.

Algoritmi e Strutture Dati 1

Appello del 26/05/2009

Seconda parte

Compito n° 2

Esercizio 1

Si implementi in pseudocodice la strutture dati ***Reverse List,*** che consiste in una lista doppiamente collegata in cui gli elementi sono sempre ordinati in ordine decrescente, e ogni intero non può essere presente più di una volta.

Si implementino per tale struttura dati le seguenti procedure:

- **Find** che prende in input una Reverse List *RL* e un intero *x* e restituisce *true* se x è presente in RL, *false* altrimenti
- **Add** che prende in input una Reverse List *RL* e un intero *x*, e inserisce l'intero *x* in *RL* (se non è già presente)
- **Remove** che prende in input una Reverse List *RL* e un intero *x*, e rimuove l'intero *x* da *RL* (se esso è presente)

Per ognuna delle procedure si stimi la complessità computazionale nel caso peggiore.

Esercizio 2

Si consideri una *tabella hash con lista di trabocco* contenente numeri interi.

- Si spieghi brevemente come una tale tabella possa essere implementata.

Supponendo che H(x) sia la funzione hash (già implementata) della tabella,

- scrivere una procedura **P5** che prende in input una tabella hash e restituisce il numero di elementi presenti nella più corta lista di trabocco della tabella (non si tengano in considerazione la posizioni della tabella a cui sono associate liste di trabocco vuote);
- scrivere una procedura **P6** che prende in input una tabella hash ed un intero *x* e, restituisce il più grande elemento minore di *x* presente in tabella (*NIL* se nessun elemento della tabella è minore di *x*).

Attenzione:

- Scrivere 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.

Algoritmi e Strutture Dati 1

Appello del 23/06/2009

Esercizio 1

Scrivere una procedura **P1** in pseudocodice che preso in input un array **A** di numeri interi, restituisce in output la lunghezza della più lunga sequenza crescente presente nell'array. La procedura non può sfruttare nessuna procedura vista a lezione.

- Discutere la complessità computazionale in termini di tempo dell'algoritmo proposto, nel caso medio e nel caso peggiore.
- Qual è la miglior complessità computazionale nel caso peggiore ottenibile da un qualsiasi algoritmo che risolva il problema?
- L'algoritmo proposto ha complessità computazionale ottima nel caso peggiore?

Esercizio 2

Sia $T(n) = \begin{cases} 4T(n/2) + n^2 + \log n & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del **teorema generale**, giustificando la risposta.

Esercizio 3

Scrivere una procedura iterativa **P2** in pseudocodice che preso in input un albero binario di ricerca **T** e due interi **x** e **y**, restituisce il numero di elementi dell'albero aventi chiave compresa tra x e y, estremi inclusi.

Si assuma che **T** è semplicemente un puntatore al nodo radice.
Si analizzi la complessità computazionale nel caso peggiore della procedura, in funzione dell'altezza **h** dell'albero e del numero **m** di chiavi dell'albero comprese tra x e y.

La procedura può far uso UNICAMENTE della procedura **TreeSuccessor** vista a lezione, che preso in input il puntatore ad un nodo **v** dell'albero restituisce il puntatore al nodo avente come chiave il successore della chiave di **v** se un tale nodo esiste, NIL altrimenti.

Esercizio 4

a. Mostrare l'albero binario di ricerca che si ottiene a partire dall'albero vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **25, 2, 15, 6, 8, 90, 22, 24**
b. Scrivere una procedura ricorsiva **P3** che preso in input un albero binario di ricerca **T** e un intero **x** restituisce TRUE se x è presente in T, FALSE altrimenti.
   Si assuma che **T** è semplicemente un puntatore al nodo radice.

Attenzione:

- Scrivere 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.

Algoritmi e Strutture Dati 1

Appello del 21/07/2009

Esercizio 1

Scrivere una procedura **P1** in pseudocodice che preso in input un array **A** di numeri interi, lo modifica nel suo array palindromo. Ad esempio, l'array [7, 3 , 5, 2, 9] diviene [9, 2, 5, 3, 7].

La procedura non può far uso di nessun array ausiliario.

Discutere la complessità computazionale in termini di tempo dell'algoritmo proposto

Esercizio 2

Sia $T(n) = \begin{cases} 4T(n/2) + n^2 & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del **metodo di sostituzione e della dimostrazione per induzione**, giustificando la risposta.

Esercizio 3

Si implementi la procedura **EvenPriorityHashInsert(T,x)** di **inserimento** di una chiave **x** in una tabella hash **T** con liste di trabocco in cui le chiavi pari hanno priorità su quelle dispari.

In particolare, la tabella è tale che tutte le sue liste di trabocco hanno tutti gli elementi con chiave pari prima degli elementi con chiave dispari.

Esercizio 4

a. Mostrare l'heap di massimo che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato i seguenti elementi:
    - **25, 2, 15, 6, 8, 90, 22, 24**
b. Mostrare come è possibile implementare un heap tramite array.
c. Indicare, **giustificando adeguatamente le risposte**, la complessità delle seguenti operazioni per l'heap di massimo:
    - Individuazione del massimo
    - Estrazione del massimo
    - Inserimento

Attenzione:

- Scrivere 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.

Algoritmi e Strutture Dati 1

Appello del 10/09/2009

Esercizio 1 (16 punti)

Scrivere una procedura ricorsiva **P1** in pseudocodice che, facendo uso della tecnica *divide et impera*, preso in input un array **A** di **n** numeri interi, restituisce in output il valore del massimo intero presente in **A**.

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

Esercizio 2 (6 punti)

Scrivere una procedura iterativa **P1** in pseudocodice che, preso in input un array **A** di **n** numeri interi, restituisce in output il valore del massimo intero presente in **A**.

Si stimi la complessità dell'algoritmo proposto, e si dica, giustificando la risposta, se possono esistere algoritmi aventi tempo di esecuzione asintoticamente migliore.

Esercizio 3 (11 punti)

a. Mostrare *l'albero binario di ricerca* che si ottiene a partire dall'albero vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **25, 90, 18, 7, 15, 6, 8, 22**
b. Scrivere una procedura iterativa **P2** che preso in input un albero binario di ricerca **T** e un intero **x** restituisce TRUE se x è presente in T, FALSE altrimenti.

Attenzione:

- Scrivere 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.

Algoritmi e Strutture Dati 1

Esonero del 23/03/2010

Compito n° 1

Esercizio 1

Dire se, giustificando la risposta applicando la definizione delle notazioni asintotiche,

a) $n \log n - 5n = \theta(n \log n)$
b) $n\sqrt{n} + n = \mathrm{O}(n)$

Esercizio 2

Un array di numeri interi si dice *chiuso rispetto al complemento a* ***k*** se per ogni suo elemento x, A contiene anche un elemento y tale che x+y=k.

Scrivere una funzione `public static boolean` **`closed`** `(int [] A, int k)` in Java che preso in input un array **A** di interi ed un intero **k**, restituisce `true` se A è chiuso rispetto al complemento a k, e restituisce `false` altrimenti.

La procedura deve avere tempo di esecuzione nel caso peggiore $O(n \log n)$, e può sfruttare tutte le procedure viste a lezione.

Esercizio 3

Scrivere una funzione `public static int` **`maxSubArray`** `(int [] A)` in Java che preso in input un array **A** di interi, restituisce il valore della somma degli elementi del sottoarray di A (contenente elementi consecutivi) avente somma massima. Per esempio, se A=[-10,20,-15,30,-1], **`maxSubArray`** restituisce **35**.

La procedura non può sfruttare nessuna delle procedure viste a lezione. Si stimi il suo tempo di esecuzione nel caso peggiore (se esso è $O(n^2)$, l'esercizio dà luogo ad una valutazione migliore).

Esercizio 4

Sia $T(n) = \begin{cases} 9T(n/3) + 3\mathrm{n}^2 - 2\mathrm{n} & se\ n > 1 \\ c_1 & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del **teorema generale**.

Attenzione: non è ammesso per nessun motivo l'uso di telefoni cellulari, calcolatrici, etc…

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: *InserctionSort, 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, …, 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 booelean 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.

Algoritmi e Strutture Dati 1

Appello del 6/07/2010

Esercizio 1

Si consideri la seguente classe Java:

```
public class Elem{
      public int blocco;
      public int valore;
}
```

Scrivere un metodo **`public static void sort (Elem [] A)`** in Java che preso in input un array **A** di ***n*** oggetti di tipo Elem, ordina l'array in modo che gli elementi con lo stesso numero di `blocco` siano tutti consecutivi nell'array, e ordinati in modo non decrescente rispetto a `valore`. Inoltre i blocchi devono essere ordinati in modo crescente rispetto al numero di blocco.

Suggerimento: scegliere un qualsiasi algoritmo di ordinamento visto a lezione (per esempio il Selection Sort riportato qui sotto) e modificarlo in modo che lavori correttamente con array di Elem. In particolare, sarà necessario definire un metodo statico che confronta due Elem identificando chi va prima nell'ordinamento (`public static boolean isLessOrEqual (Elem x, Elem y)`).

```
public static void SelectionSort (int [] A) {
      int n=A.length;
      for (int i=0; i<n; i++) {
            int posmin=i;
            for (int j=i+1; j<n; j++) {
                  if (A[j]<A[posmin]) {posmin=j;}
            }
            int aux=A[i];
            A[i]=A[posmin];
            A[posmin]=aux;
      }
}
```

Si dia una stima del tempo di esecuzione dell'algoritmo proposto nel caso peggiore.

Esercizio 2

Sia $T(n) = \begin{cases} 25T\left(n/5\right) + \mathrm{n}^2 & se\ n > 1 \\ c_1 & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso

- del **metodo di sostituzione ed induzione**;
- del **teorema generale** *(mostrare formalmente perché è valido il caso che si applica).*

INIZIO SECONDO PARZIALE

Esercizio 3

a. Mostrare l'heap che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **18, 20, 60, 24, 35, 2, 25, 26, 23**
b. Aggiungere alla classe Heap un metodo **`public static int getsecondMax ()`** che, senza modificare l'heap su cui lavora, restituisce il secondo masssimo presente nell'heap, facendo uso di tutti gli altri metodi già presenti nella classe Heap ed implementati a lezione. Si stimi la complessità computazionale di getSecondMax in funzione di **n**, dove ***n*** è il numero di elementi nell'heap.

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, …, k_n,** in una tabella con **19** posizioni risulti in una tabella in cui la ricerca di una chiave **k** risulta avere il **miglior** tempo possibile di esecuzione a prescindere dalla politica implementata per l'inserimento in lista di nuovi elementi (in testa, in coda,…). 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

a. Mostrare l'albero binario di ricerca che si ottiene a partire dall'albero vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **13, 2, 60, 22, 67, 5, 25, 12, 3, 24, 28**
b. Mostrare l'ordine di visita dei nodi dell'albero costruito al punto (a) rispetto ai seguenti algoritmi di visita:
   - Visita in ordine intermedio
   - Visita in preordine
   - Visita in postordine

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.

Algoritmi e Strutture Dati 1

Appello dell' 8/09/2010

Esercizio 1

- Si presenti un algoritmo di ordinamento a scelta visto a lezione per ordinare in modo non decrescente un array di numeri interi, e si dia una stima della sua complessità computazionale.
- Si consideri ora la seguente classe Java:

```
public class Elem{
public int blocco;
public int valore;
}
```

Modificando l'algoritmo presentato, scrivere un metodo **`public static void sort (Elem [] A)`** in Java che preso in input un array **A** di ***n*** oggetti di tipo Elem, ordina l'array in modo che gli elementi con lo stesso numero di `blocco` siano tutti consecutivi nell'array, e ordinati in modo **non crescente rispetto a `valore`**. Inoltre i blocchi devono essere ordinati in modo **non decrescente rispetto a `blocco`**.

Suggerimento: definire un metodo statico che confronta due Elem identificando chi va prima nell'ordinamento (`public static boolean isLessOrEqual (Elem x, Elem y)`).

Esercizio 2

Dire se, giustificando la risposta applicando la definizione delle notazioni asintotiche,

a) $n \log n - 5n = \theta(n^2)$
b) $n\sqrt{n} + n = O(n^2)$

INIZIO SECONDO PARZIALE

Esercizio 3

a. Mostrare l'heap di massio che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **13, 2, 60, 22, 67, 5, 25, 12, 3, 24, 28**
b. Scrivere FUORI dalla classe Heap un metodo **`public static int getfourthMax (Heap h)`** che, senza modificare l'heap su cui lavora e senza poter accedere direttamente all'albero dell'heap, ma solo potendo utilizzare tutti i metodi già presenti nella classe Heap ed implementati a lezione, restituisce il quarto masssimo presente nell'heap, facendo uso di tutti gli altri. Si stimi la complessità computazionale di **`getfourthMax`** in funzione di **n**, dove ***n*** è il numero di elementi nell'heap.

Esercizio 4

Si descriva brevemente come può essere risolto il problema delle collisioni nelle tabelle Hash (sia con lista di trabocco che con indirizzamento aperto).

Esercizio 5

Si consideri un albero binario di ricerca. Scrivere un metodo ricorsivo **`public static int height (Node x)`** che preso in input un nodo **x**, restituisce l'altezza del sottoalbero radicato al nodo **x**.

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.

Algoritmi e Strutture Dati 1

Appello del 22/12/2010

Esercizio 1

Scrivere un metodo **`public static booelan LEQ (int [] A, int[] B)`** in Java che presi in input due array **A** e **B** di ***n*** ed ***m*** oggetti rispettivamente, restituisce ***true*** se e solo se ogni elemento di A è minore o uguale di tutti gli elementi di B.

La complessità computazionale della procedura deve essere $\boldsymbol{\theta(n+m)}$. Dimostrare formalmente il risultato di complessità del metodo proposto.

Esercizio 2

$$\text{Sia } T(n) = \begin{cases} T(n/3) + c_1 & se\ n > 1 \\ c_2 & se\ n = 1 \end{cases}$$

dove c_1 e c_2 sono due costanti.

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso

- del **metodo di sostituzione ed induzione**;
- del **teorema generale** *(mostrare formalmente perché è valido il caso che si applica).*

Esercizio 3

a. **Mostrare** l'albero binario di ricerca che si ottiene a partire dall'albero vuoto inserendo nell'ordine indicato i seguenti elementi:

   13, 2, 1, 18, 60, 24, 59, 100, 3, 34

   Che altezza ha l'albero ottenuto?

b. Dopo aver scritto la classe **Node** vista a lezione, scrivere un metodo ricorsivo **public static int SUM (Node x)** che preso in input un nodo **x**, restituisce la somma delle chiavi contenute nell'albero radicato al nodo **x**. Si discuta la complessità computazionale della procedura proposta.

Esercizio 4

Si consideri una *tabella hash con lista di trabocco* contenente numeri interi. Data la funzione hash **modulo**, si mostri un esempio in cui l'inserimento di **n chiavi intere k_1, k_2, k_3, …, k_n,** in una tabella con **31** 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.

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.

Algoritmi e Strutture Dati 1

Appello del 25/01/2011

Esercizio 1

Il **rango** di un array A è definito come il minimo numero di sottoarray (di elementi consecutivi) in cui l'array A deve essere suddiviso in modo tale che ognuno di tali sottoarray sia ordinato in ordine non decrescente.

Scrivere un metodo **`public static booelan rango (int [] A, int k)`** in Java che presi in input un array **A** di ***n*** interi ed un intero k, restituisce ***true*** se e solo se l'array A ha rango minore o uguale a k.

Si stimi **formalmente** la complessità dell'algoritmo proposto, e si dica, giustificando la risposta, se possono esistere algoritmi aventi tempo di esecuzione asintoticamente migliore.

Nota*: algoritmi più efficienti danno luogo ad una migliore valutazione rispetto ad algoritmi meno efficienti.*

Esercizio 2

I numeri di **Catalan** sono definiti ricorsivamente nel seguente modo:

$$Catalan(n) = \begin{cases} 1 & se\ n = 0 \\ \sum_{i=0}^{n-1} Catalan(i) \cdot Catalan\ (n-1-i) & se\ n > 0 \end{cases}$$

I primi numeri di Catalan sono: 1, 1, 2, 5, 14, 42

Scrivere un metodo **ricorsivo** *public static int Catalan (int n)* che calcola l'ennesimo numero di Catalan.

Esercizio 3

a. Mostrare *l'heap di massimo* che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **3, 25, 2, 18, 20, 8, 39, 36, 24**
b. Scrivere FUORI dalla classe Heap un metodo **`public static int getMaxProduct (Heap h)`** che, assumendo che tutte le chiavi dell'heap siano non negative, senza poter accedere direttamente all'albero dell'heap, ma solo potendo utilizzare tutti i metodi già presenti nella classe Heap ed implementati a lezione, restituisce il massimo numero intero ottenibile come prodotto di due chiavi presenti nell'heap, e rimuove dall'Heap un unico elemento, che deve essere l'elemento massimo. Si stimi la complessità computazionale di **`getMaxProduct`** in funzione di **n**, dove ***n*** è il numero di elementi nell'heap.

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.

Algoritmi e Strutture Dati 1

Appello del 16/02/2011

Esercizio 1 (8 punti)

Lo **spettro** di un array A di numeri interi è definito come la differenza tra il massimo e il minimo intero presente in A. Ad esempio, lo spettro dell'array `[1,7,-4,8,10,7]` è **14**.

Scrivere un metodo **`public static int spettro (int [] A)`** in Java che preso in input un array **A** di ***n*** interi, restituisce lo spettro di A**.**

L'algoritmo deve avere complessità temporale $\boldsymbol{\theta(n)}$ nel caso peggiore.

Esercizio 2 (8 punti)

Data la seguente definizione ricorsiva:

$$F(n) = \begin{cases} 10 & se\ n \leq 3 \\ 2F(n-1) + F(n-2) & se\ n \geq 4 \end{cases}$$

scrivere un metodo **<u>ricorsivo</u>** *public static int F (int n)* che calcola F(n).

Esercizio 3 (4 punti + 8 punti)

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 **`public static int conto(Node x)`** che preso in input un nodo **x**, restituisce il numero di nodi nel sottoalbero radicato in **x** aventi chiave che è un multiplo della profondità del nodo stesso. Si assuma che il nodo x ha profondità 0. *Si consiglia di utilizzare un metodo ricorsivo ausiliario.*

Esercizio 4 (8 punti)

Scrivere un metodo <u>ricorsivo</u> **`public static long minimo (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 minimo 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.

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.

Algoritmi e Strutture Dati 1

Appello del 25/05/2011

Esercizio 1 (9 punti)

Scrivere un metodo **`public static int sommaInsieme (int [] A)`** in Java che preso in input un array A di numeri interi restituisce in output la somma degli elementi presenti in A, **contando una sola volta gli elementi ripetuti**. La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,3,2,3,5,1,3], il metodo deve restituire 1+3+2+5=11, senza contare più di una volta nella somma gli interi 1 e 3 che sono ripetuti.

La procedura deve avere tempo di esecuzione nel caso peggiore $O(n^2)$, e NON può sfruttare nessuna delle procedure viste a lezione.

Esercizio 2 (9 punti)

Scrivere un metodo **`public static int sommaInsiemeBis (int [] A)`** in Java che preso in input un array A di numeri interi restituisce in output la somma degli elementi presenti in A, **contando una sola volta gli elementi ripetuti**. La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,3,2,3,5,1,3], il metodo deve restituire 1+3+2+5=11, senza contare più di una volta nella somma gli interi 1 e 3 che sono ripetuti.

La procedura deve avere tempo di esecuzione nel caso peggiore $O(n \log n)$, e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 3 (9 punti)

Si consideri la classe SImpleList

```
public class SimpleList{
      int key;
      SimpleList next;
      ...
}
```

che rappresenta un elemento di una lista e possiede una chiave intera ed il puntatore all'elemento successivo. Si noti come una lista è identificata in modo naturale dal suo elemento di testa.

Scrivere un metodo **ricorsivo** **`public static booelan ricerca (SimpleList L, int k)`** in Java che data una SimpleList L ed una chiave k restituisce ***true*** se e solo se k è presente nella lista L.

Esercizio 4 (3 punti + 6 punti)

a. Data le seguente sequenza di chiavi intere:
   - **24, 8, 74, 31, 12, 9, 22, 10, 18, 99**

   mostrare, a partire da strutture dati vuote, ed inserendo nell'ordine indicato le chiavi su elencate
   1. l'**albero binario di ricerca** che si ottiene
   2. l'**heap di minimo** che si ottiene
   3. la **tabella Hash** di **dimensione 11** con **liste di trabocco** e funzione hash *H(x) = x mod 11* che si ottiene

b. Mostrare l'ordine di visita dei nodi dell'albero costruito al punto (a.1) rispetto ai seguenti algoritmi di visita:
   - Visita in ordine intermedio
   - Visita in preordine
   - Visita in postordine

Attenzione:

- Per svolgere il compito si hanno a disposizione **90** minuti.
- Scrivere **subito** nome, cognome, matricola e numero del compito su OGNI FOGLIO.
- Durante la prova scritta non è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- **Non** è possibile consultare appunti, libri, dispense o qualsiasi altro materiale.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**

Algoritmi e Strutture Dati 1

Appello del 22/06/2011

Esercizio 1 (9 punti)

Scrivere un metodo **`public static int piuFrequente (int [] A)`** in Java che preso in input un array A di numeri interi restituisce in output uno degli interi presenti in A il maggior numero di volte. La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,5,2,5,5,1,3], il metodo deve restituire 5.

La procedura deve avere tempo di esecuzione nel caso peggiore $\boldsymbol{O(n^2)}$, e NON può sfruttare nessuna delle procedure viste a lezione.

Esercizio 2 (9 punti)

Scrivere un metodo **`public static int piuFrequenteBis (int [] A)`** in Java che preso in input un array A di numeri interi restituisce in output uno degli interi presenti in A il maggior numero di volte. La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,5,2,5,5,1,3], il metodo deve restituire 5.

La procedura deve avere tempo di esecuzione nel caso peggiore $\boldsymbol{O(n \log n)}$**,** e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 3 (9 punti)

Scrivere un algoritmo **ricorsivo** di ricerca "ternaria" **`public static int ricercaTernaria (int [] A, int i, int j, int x)`**, che prende in input un array A ordinato e un intero, e cerca l'intero x in A (tra le posizioni i e j, estremi inclusi) restituendo *true* se e solo se x è presente in A (tra le posizioni i e j, estremi inclusi). [si assuma che la prima volta il metodi è richiamato con i parametri *i* e *j* uguali a *0* e *A.length-1*, rispettivamente]

Ad ogni passo **`ricercaTernaria`**, quando invocato su un array di lunghezza L, confronta l'elemento x con quello in posizione L/3 ed eventualmente con quello in posizione 2L/3, ed eventualmente chiama ricorsivamente se stesso su un opportuno sottoarray lungo approssimativamente L/3.

Si dia la stima ricorsiva **T(n)** del tempo di esecuzione e, dopo averla risolta con un metodo a piacere, si confronti la ricerca ternaria con la ricerca binaria vista a lezione.

Esercizio 3 (9 punti)

a. Mostrare l'heap di **minimo** che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato i seguenti elementi:
    - **24, 2, 6, 15, 8, 90, 5, 22, 3**
b. Mostrare come è possibile implementare un heap tramite array (in particolare illustrare la corrispondenza tra nodi dell'albero dell'heap e posizioni dell'array).
c. Indicare, **giustificando adeguatamente le risposte**, la complessità delle seguenti operazioni per l'heap di massimo **in funzione del numero n di chiavi presenti nell'heap**:
    - Individuazione del massimo
    - Estrazione del massimo
    - Inserimento

Algoritmi e Strutture Dati 1

Appello del 6/07/2011

Esercizio 1 (9 punti)

Scrivere un metodo **`public static int secondMax (int [] A)`** in Java che preso in input un array A di numeri interi restituisce in output il secondo elemento più grande presente nell'array A.

La procedura **NON può modificare** l'array A. Si assuma che l'array A contiene almeno due elementi.

Ad esempio, se A=[1,5,2,5,-1,3], il metodo deve restituire 5. Se invece A=[1,5,2,-1,3] il metodo deve restituire 3.

Il metodo deve avere tempo di esecuzione nel caso peggiore $\boldsymbol{O(n)}$, e NON può sfruttare nessuno dei metodi visti a lezione.

Si stimi la complessità dell'algoritmo proposto, e si dica, giustificando la risposta, se possono esistere algoritmi aventi tempo di esecuzione asintoticamente migliore.

Esercizio 2 (9 punti)

$$\text{Sia } T(n) = \begin{cases} T\left(n/5\right) + c_1 & se\ n > 1 \\ c_2 & se\ n = 1 \end{cases}$$

dove c_1 e c_2 sono due costanti.

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso del **metodo di sostituzione ed induzione**.

Esercizio 3 (9 punti)

Si consideri la classe SimpleList

```
public class SimpleList{
      int key;
      SimpleList next;
      ...
}
```

che rappresenta un elemento di una lista e possiede una chiave intera ed il puntatore all'elemento successivo. Si noti come una lista è identificata in modo naturale dal suo elemento di testa.

Scrivere un metodo **iterativo** **`public static SimpleList ricerca (SimpleList L, int k)`** in Java che data una SimpleList L ed una chiave k restituisce ***il puntatore ad un elemento che contiene la chiave k*** se tale chiave k è presente nella lista L, **null** altrimenti.

Esercizio 4 (9 punti)

a. Mostrare l'albero binario di ricerca che si ottiene a partire dall'albero vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **24, 2, 6, 15, 8, 90, 5, 22**
b. Si consideri un albero binario. Scrivere un metodo **ricorsivo** **`public static int height (Node x)`** che preso in input un nodo **x**, restituisce l'altezza del sottoalbero radicato al nodo **x** [se x è un foglia l'albero radicato in x ha altezza 0].

Algoritmi e Strutture Dati 1

Appello del 7/09/2011

Esercizio 1 (9 punti)

Scrivere un metodo **`public static int Selection (int [] A, int k)`** in Java che preso in input un array A di n numeri interi ed intero k compreso tra 1 ed n, restituisce in output il k-esimo elemento più grande presente nell'array A.

La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,5,2,5,-1,3] e k=2, il metodo deve restituire 5. Se invece A=[1,5,2,-1,3] e k=3, il metodo deve restituire 2.

La procedura deve avere tempo di esecuzione nel caso peggiore $\boldsymbol{O(n \log n)}$**,** e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 2 (9 punti)

Scrivere un algoritmo **ricorsivo** di ricerca "ternaria" **`public static boolean ricercaTernaria (int [] A, int i, int j, int x)`**, che prende in input un array A ordinato e un intero, e cerca l'intero x in A (tra le posizioni i e j, estremi inclusi) restituendo *true* se e solo se x è presente in A (tra le posizioni i e j, estremi inclusi). [si assuma che la prima volta il metodo è richiamato con i parametri *i* e *j* uguali a *0* e *A.length-1*, rispettivamente]

Ad ogni passo **`ricercaTernaria`**, quando invocato su un array di lunghezza L, confronta l'elemento x con quello in posizione L/3 ed eventualmente con quello in posizione 2L/3, ed eventualmente chiama ricorsivamente se stesso su un opportuno sottoarray lungo approssimativamente L/3.

Si dia la stima ricorsiva **T(n)** del tempo di esecuzione e, dopo averla risolta con un metodo a piacere, si confronti la ricerca ternaria con la ricerca binaria vista a lezione.

Esercizio 3 (9 punti)

Si consideri la classe SimpleList

```
public class SimpleList{
      int key;
      SimpleList next;
      ...
}
```

che rappresenta un elemento di una lista e possiede una chiave intera ed il puntatore all'elemento successivo. Si noti come una lista è identificata in modo naturale dal suo elemento di testa.

Scrivere un metodo **iterativo** **`public static booelan gemelli (SimpleList L)`** in Java che data una SimpleList L restituisce ***true*** se e solo se in L sono presenti due elementi consecutivi aventi la stessa chiave.

Esercizio 4 (9 punti)

a. Mostrare la tabella Hash di dimensione 17 con liste di trabocco che si ottiene a partire dalla tabella vuota inserendo nell'ordine indicato i seguenti elementi:
    - **13, 2, 60, 22, 67, 5, 25, 12, 3, 24, 28**
b. Si descriva brevemente come può essere risolto il problema delle collisioni nelle tabelle Hash (sia con lista di trabocco che con indirizzamento aperto)

Attenzione:

- Per svolgere il compito si hanno a disposizione **90** minuti.
- Scrivere **subito** nome, cognome, matricola e numero del compito su OGNI FOGLIO.
- Durante la prova scritta non è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- **Non** è possibile consultare appunti, libri, dispense o qualsiasi altro materiale.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**