

## Altri esempi esercizi d'esame (oltre a quelli su ADT)

**Attenzione: le soluzioni sono discusse nell'ultima pagina, non guardatele subito se volete provare a risolvere per conto vostro i problemi**

### 2-Sum problem

Dato un array di  $n$  interi con segno, riportare in output tutte le coppie di numeri la cui somma è 0.

E' possibile che poi venga chiesto di leggere/scrivere da standard input/output o file i dati in ingresso, oppure il risultato (con conseguenti controlli sui tipi di dato ed eventuali eccezioni).

### Esercizi sulla ricorsione

#### Conteggio di vocali

Data una stringa  $s$  contare il numero di vocali in essa presente implementando un metodo statico ricorsivo.

E' possibile che poi sia richiesto di leggere/scrivere da standard input/output o file i dati in ingresso, oppure il risultato (con conseguenti controlli sui tipi di dato ed eventuali eccezioni).

#### Bilanciamento di parentesi

Data una stringa che contiene solo parentesi tonde aperte o chiuse dire se le parentesi sono bilanciate. Proporre una soluzione che invochi un metodo ricorsivo.

E' possibile che poi sia richiesto di leggere/scrivere da standard input/output o file i dati in ingresso, oppure il risultato (con conseguenti controlli sui tipi di dato ed eventuali eccezioni).

### Esercizi complessità

Dato il seguente codice dire qual è la sua complessità e giustificare brevemente la risposta. Di seguito sono specificati diversi possibili metodi  $A_x$ ,  $x=1,2,3,4$  per "allenarsi". All'esame sarà assegnato un unico frammento di codice.

```
for (int i=0; i<n; i++){
    if(a<b){
        metodoAx(i,n);
    }
    else{
        b=a;
    }
}
```

----

```
public static void metodoA1(int x, int y){ // O(1)
    System.out.println(x);
}
```

```
public static void metodoA2(int x, int y){ //O(x)
    for(int i=0;i<x;i++){
        System.out.println(x);
    }
}
```

```
public static void metodoA3(int x, int y){
    for(int i=0;i<y^2;i++){
        System.out.println(x);
    }
}
```

```
public static void metodoA4(int x, int y){
    for(int i=0;i<y^2;i+=y){
        System.out.println(x);
    }
}
```

**Breve discussione sulle possibili soluzioni (il codice è presente nella cartella dove avete trovato questo documento).**

## **2-Sum problem**

Soluzione più semplice  $O(n^2)$  senza spazio aggiuntivo

Idea: per ogni elemento dell'array (ciclo for) cerco nell'array se c'è il suo opposto (ricerca sequenziale).

Soluzione  $O(n \log n)$  senza spazio aggiuntivo

Idea: Ordino l'array ( $n \log n$ )

Per ogni elemento dell'array (ciclo for), cerco se esiste il suo opposto con la ricerca binaria che posso applicare perché l'array è stato ordinato.

Soluzione  $O(n \log n)$  senza spazio aggiuntivo (proposta da uno studente)

Idea: Ordino l'array ( $n \log n$ )

Scansiono l'array utilizzando due indici: left parte da 0 e viene solo incrementato, right parte dall'ultimo elemento e viene solo decrementato. Ad ogni iterazione confronto gli elementi puntati dai due indici: se opposti, riporto la coppia e aggiorni entrambi gli indici; altrimenti aggiorni (incremento left o decremento right) l'indice corrispondente all'elemento considerato con valore assoluto maggiore. Il ciclo termina quando  $left \geq right$ , oppure se l'elemento di indice left è positivo oppure se l'elemento di indice right è negativo. Nota: non funziona in caso siano presenti valori duplicati che hanno un opposto. In questo caso serve un controllo aggiuntivo prima di incrementare gli indici.

Soluzione tempo atteso  $O(n)$  con spazio aggiuntivo  $O(n)$

Hashtable di dimensione n

Metto tutti gli elementi nell'hashtable

Per ogni elemento dell'array, cerco se esiste il suo opposto

## **Conteggio di vocali**

Caso base: la stringa è vuota, il numero di vocali è 0.

Passo ricorsivo: considero il primo carattere. Se è una vocale restituisco 1+ il valore restituito dal metodo ricorsivo invocato sulla stringa privata del primo carattere, altrimenti restituisco direttamente il valore restituito dalla chiamata ricorsiva sulla stringa privata del primo carattere.

## **Bilanciamento di parentesi**

Problema: devo in qualche modo tener conto del numero di parentesi che sono state aperte man mano che analizzo la stringa. Il metodo dovrà ricevere come parametro, oltre alla stringa, anche un intero che svolga questa funzione di contatore.

Caso base: se la stringa è vuota e il contatore è a 0 restituisco vero, se la stringa è vuota e il contatore è diverso da zero restituisco falso.

Passo ricorsivo: esamino il primo carattere: se è una parentesi aperta invoco il metodo in modo ricorsivo sulla stringa privata del primo carattere e incremento il secondo parametro di una unità. Se invece è una parentesi chiusa, invoco il

metodo in modo ricorsivo sulla stringa privata del primo carattere e decremento il secondo parametro.

Attenzione: se in qualsiasi istante la variabile che conta in numero di parentesi aperte diventa negativo significa che ho chiuso più parentesi di quante ne abbia aperte! Non ho più modo di rimediare al bilanciamento, pertanto se questa condizione si verifica posso interrompere l'elaborazione e restituire falso. Questo caso va gestito subito dopo i controlli del caso base, prima delle chiamate ricorsive.

E' stata illustrata anche una soluzione che non usa la ricorsione ma una pila come struttura dati d'appoggio. Sfruttando il meccanismo di funzionamento FIFO, scansiamo la stringa dal primo all'ultimo carattere: ogni volta che troviamo una parentesi aperta, la inseriamo nella pila; ogni volta che troviamo una parentesi chiusa, togliamo l'elemento in cima alla pila, in quanto le due sono parentesi accoppiate. Se pop lancia EmptyStackException, significa che abbiamo incontrato una parentesi chiusa di troppo: non c'è bilanciamento e posso restituire falso. Altrimenti, se arrivo alla fine della stringa, verifico lo stato della pila. Se è vuota vuol dire che tutte le parentesi aperte in essa inserite sono state rimosse in corrispondenza di una parentesi chiusa e posso restituire vero. Altrimenti, restituisco falso.

### **Complessità**

A1: metodo che esegue un'unica istruzione di stampa, quindi tempo costante. Il ciclo for esegue n iterazioni, in ciascuna iterazione ho un numero di operazioni  $O(1)$ , quindi in totale  $O(n)$ .

A2: metodo che esegue un numero di istruzioni di stampa pari al primo parametro passato. Guardando il ciclo for e la chiamata al metodo vedo che il parametro in questione è l'indice i. Quindi all'i-esima iterazione eseguo "i" iterazioni. Poiché i va da 0 a n-1, il numero di operazioni totali è dato dalla formula di Gauss, quindi quadratico.

A3: metodo che esegue un numero di istruzioni di stampa pari al quadrato del secondo parametro passato. Guardando il ciclo for e la chiamata al metodo vedo che il parametro in questione è n. Quindi all'i-esima iterazione eseguo  $n^2$  iterazioni. Poiché eseguo n iterazioni del ciclo for, il numero di operazioni totali è dato  $n \times n^2 = n^3$ .

A4:

metodo che esegue un numero di istruzioni di stampa pari secondo parametro passato. Infatti l'indice arriva fino al quadrato di y, ma y viene incrementato ogni volta di una quantità pari a se' stesso. Guardando il ciclo for e la chiamata al metodo vedo che il parametro in questione è n. Quindi all'i-esima iterazione eseguo n iterazioni. Poiché eseguo n iterazioni del ciclo for, il numero di operazioni totali è dato  $n \times n = n^2$ .