

Matricola_____ Cognome_____ Nome_____

Fondamenti di Informatica

Prova d'esame del 18/7/2014

Regole d'esame:

1. È vietato parlare con altri studenti
2. È vietato consultare appunti, dispense, libri, in qualunque formato
3. È vietato tenere accesi i telefoni cellulari

Esercizio 1

Si consideri il seguente metodo che riceve in input un array di interi **a**. Indicare, spiegandola brevemente, la complessità asintotica di caso peggiore in funzione della lunghezza **n** dell'array **a**.

```
public static int esercizio(int[] a){
    int n=a.length;
    int s=0;
    int i=0;
    while(i<n){
        int h=i;
        while(h<n && h<i+2){
            s+=a[h];
            h++;
        }
        i++;
    }
    return s;
}
```

Il corpo del ciclo `while` più esterno viene ripetuto n volte (per i che assume tutti i valori interi da 0 a $n-1$). Ad ogni iterazione di tale ciclo si esegue il ciclo `while` più interno il cui corpo si ripete 2 volte, cioè per h che assume i valori interi i e $i+1$ (quando i è pari a $n-1$ allora h assume un unico valore). Pertanto le istruzioni `s+=a[h]` e `h++` (che sono le istruzioni dominanti del metodo) vengono eseguite $\sum_{i=0}^{n-1} 2 = 2n = O(n)$ volte. La complessità del metodo è quindi $O(n)$.

Esercizio 2

Scrivere un metodo di classe ricorsivo che riceve in input una stringa **s** ed altri parametri se necessario, e restituisce la stringa che si ottiene rimuovendo da **s** tutti gli spazi. Se, ad esempio, la stringa **s** fosse "Nel mezzo del cammin di nostra vita" il metodo dovrebbe restituire "Nelmezzodelcammindinostravita".

Note.

1. Una soluzione non ricorsiva non sarà considerata corretta
2. Se ritenuto utile o necessario è possibile scrivere un metodo per l'avvio della ricorsione
3. Il metodo non deve utilizzare variabili che non siano locali

```

public static String rimuoviSpazi(String s, int i){
    String r;
    if(i==s.length())
        r="";
    else{
        if(s.charAt(i)==' ')
            r=rimuoviSpazi(a,i+1);
        else
            r=s.charAt(i)+rimuoviSpazi(a,i+1);
    }
    return r;
}

public static String rimuoviSpazi (String s){
    return rimuoviSpazi(s,0);
}

```

Esercizio 3

Nell'ambito di un'applicazione per gestire dati cinematografici, si vuole realizzare un metodo per stampare un elenco di **personaggi**. Per ogni **personaggio** interessano il nome e cognome e il sesso. Inoltre a seconda della tipologia di attività possono interessare altre informazioni. Ad esempio, per gli **attori** interessa il numero di premi Oscar vinti, mentre per i **registi** interessa il genere cinematografico preferito. Le tipologie di attività a cui si è interessati non sono fissate una volta per tutte ma se ne possono aggiungere di nuove in futuro. Il metodo di stampa deve poter continuare a funzionare senza modifiche anche a seguito dell'aggiunta di nuove tipologie di attività.

Si descriva una possibile soluzione del problema. In particolare si illustri come modellare i concetti descritti in maniera da consentire la scrittura del metodo per la stampa secondo le indicazioni date.

Si definisce una classe **Personaggio** i cui campi saranno nome, cognome e sesso. Tale classe sarà dotata, tra gli altri, del metodo `toString()` che restituirà una descrizione testuale del personaggio in cui verranno riportate le informazioni di interesse (nome, cognome e sesso). Vengono poi definite le classi **Attore** e **Regista** come classi derivate da **Personaggio**. Ognuna di queste classi avrà degli attributi specifici in aggiunta a quelli ereditati da **Personaggio**: per **Attore** avremo l'attributo `numOscarVinti`, per **Regista** avremo `generePreferito`. Ogni classe inoltre sarà dotata di un metodo `toString()` che restituirà una descrizione testuale in cui verranno riportate tutte le informazioni di interesse (nome, cognome, sesso e `numOscarVinti` per **Attore** e nome, cognome, sesso e `generePreferito` per **Regista**).

Il metodo di stampa potrà essere realizzato come segue

```

public static void stampa(Personaggio[] p){
    for(int i=0;i<p.length;i++)
        System.out.println(p[i].toString());
}

```

Si osservi che gli elementi dell'array `p` possono essere istanze di **Personaggio**, di **Attore** o di **Regista**. Per ogni elemento, l'istruzione `toString()` (grazie al binding dinamico) restituirà la descrizione opportuna. Si osservi

inoltre che qualora volessimo aggiungere nuove tipologie di personaggi (ad esempio Sceneggiatore) sarebbe sufficiente definire una nuova classe con gli attributi di interesse e dotata dell'opportuno metodo toString; il metodo stampa continuerebbe a funzionare senza bisogno di modifiche.

Esercizio 4

Si illustri il funzionamento dell'algoritmo di ricerca binaria visto a lezione con riferimento al seguente array di input ed assumendo che la chiave di ricerca sia pari a 27:

3	4	17	19	23	30	37	57	65	67	75	76	81	84	91	94
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Per rispondere alla domanda si deve mostrare, per ogni passo, lo spazio di ricerca, l'elemento confrontato con la chiave e l'esito del confronto.

3 4 17 19 23 30 37 | 57 | 65 67 75 76 81 84 91 94 sx: 0 dx: 15 ct: 7 - 57>27: continuo a sinistra

3 4 17 | 19 | 23 30 37 * * * * * sx: 0 dx: 6 ct: 3 - 19>27: continuo a destra

* * * * * 23 | 30 | 37 * * * * * sx: 4 dx: 6 ct: 5 - 30>27: continuo a sinistra

* * * * * | 23 | * * * * * sx: 4 dx: 4 ct: 4 - 23>27: continuo a destra

27 non è contenuto nell'array

Esercizio 5

Sia **n** il numero rappresentato in esadecimale come **BD4A**. Si converta **n** in ottale e in binario mostrando anche il procedimento di conversione.

$BD4A_{16} = (1011)(1101)(0100)(1010) = 1011110101001010_2 = (001)(011)(110)(101)(001)(010) = 136512_8$