Laboratorio di Programmazione

Corso di laurea triennale in Informatica Musicale
Università degli Studi di Milano, A.A. 2016-2017

Nicola Basilico, Andrea Lanzi

Appello del 04 Luglio 2017

- L'esame ha una durata di 3 ore e 30 minuti.
- È possibile consultare il libro di testo, gli appunti e l'API Java.
- È possibile utilizzare qualsiasi ambiente di sviluppo installato sulla macchina.
- È proibito l'accesso ad Internet con qualsiasi mezzo.
- Ogni esercizio consegnato non deve generare errori di compilazione.
- Per ottenere la sufficienza è necessario consegnare almeno due esercizi corretti.

1 Permutazioni casuali di un array

nome del file sorgente: Permutazioni.java

Si scriva un programma che letto in input un array di 10 interi e un intero n stampi a video n permutazioni casuali degli elementi dell'array (si faccia uso della funzione *Math.random()*).

1.1 Esempio

Dati in input l'array:

```
1    2    3    4    5    6    7    8    9    10
```

e l'intero n: 4

Il programma stampa quattro permutazioni casuali come queste:

```
6    3    7    8    5    1    2    4    9    10
6    1    7    4    9    5    8    3    10    2
2    10   8    9    1    5    7    6    3    4
2    10   4    5    3    8    7    1    6    9
```

2 Matrice delle adiacenze di un grafo

nome del file sorgente: adiacenze.java

Un grafo è dato da un insieme di elementi detti *nodi*. Ogni nodo può essre collegato ad un altro nodo (incluso se stesso) tramite un *arco*. Gli archi hanno una direzione che va dal nodo di partenza a quello di arrivo.

Si scriva un programma che acquisisca una matrice *4 x 4* che rappresenti un grafo con 4 nodi. Ogni elemento di tale matrice sarà uno 0 oppure un 1. L'interpretazione è la seguente: se alla riga i e colonna j c'è un 1 allora c'è un arco che parte dal nodo i e arriva nel nodo j. Se, invece, c'è uno 0 nessun arco è presente dal nodo i al nodo j.

Una volta acquisita la matrice il programma deve: * dire se il gafo è completo (la matrice ha tutti 1); * stampare l'id dei nodi con massimo numero di archi uscenti; * dire se il grafo è simmetrico: questa condizione si verifica quando ogni volta che è presente l'arco da i a j allora è presente anche l'arco in direzione opposta (da j a i).

2.1 Esempio

Data in input questa matrice:

```
0     1     0     1
1     0     0     0
1     0     0     1
0     0     1     0
```

Il programma stampa:

```
Il grafo non è completo

Nodi con massimo numero di archi uscenti: 1, 3

Il grafo non è simmetrico
```

Data in input questa matrice:

```
0     1     1     1
1     0     0     0
1     0     0     0
1     0     0     0
```

Il programma stampa:

```
Il grafo non è completo

Nodi con massimo numero di archi uscenti: 1

Il grafo è simmetrico
```

3 Percorso dell'Alpinista

nome del file sorgente: Alpinista.java

Si scriva un programma che legga una sequenza di 10 numeri da interpretarsi come altitudini delle montagne in metri attraversate da un alpinista durante un percorso di trekking. Il programma accetti solo altitudini valide maggiori di 0 e stampi una serie di informazioni sul percorso di trekking totale:

- Si calcoli il percorso totale in Km dell'alpinista riguardante tutto il percorso; Per calcolare l'intero percorso si tenga conto che l'alpinista parte sempre dalla base della montagna che dista dal centro della base 3Km.
- Si dia un allarme (stampa a video) nel caso l'alpinista raggiunga due volte consecutive la death zone oltre gli 8000 m;
- Si calcoli la somma totale delle altitudine raggiunte dall'alpnista;
- Si annulli la gara (Stampando a video gara annullata) nel caso l'alpinista non raggiunga mai una vetta di 8000 metri.

3.1 Esempio

Data in input la sequenza:

```
8000, 4500, 6500, 8200, 8100
```

il programma stampa:

```
Distanza in Km: 76 Km (17 + 10,8 + 14,3 + 17,4 + 17,2)
Allarme superata la death zone 2 volte.
Somma totale: 35300 metri
Gara superata.
```

4 Intrusion Detection System

nome del file sorgente: intrusiondetection.java

Si scriva un programma che simuli un intrusion detection system. Tale sistema di sicurezza legge in input una stringa potenzialmente infinita che rappresenta il traffico di rete (nel nostro caso introdurre una stringa di almeno 40 caratteri) e una sequenza di tre stringhe che rappresentano i pattern di attacco da ricercare nel traffico di rete. Il programma deve dopo avere ricevuto il proprio input calcolare:

- Il numero di attacchi in termini di numero di signature e frequenza, apparse nel traffico di rete. La ricerca deve essere effettuata con la funzione java (charAt()), considerando il fatto che una signature puo' contenere un'altra signature al suo interno, in questo caso stampare solo la signature di attacco piu‘ lunga.
- L'attaccante puo‘ spezzare la stringa di attacco facendo interleaving di un byte, predisporre l'algoritmo per rilevare tale caso.
- L'algoritmo controlla inoltre se l'attacco rilevato e' un falso positivo. Cioe' nella stringa di rete gli attacchi che distano piu‘ di 10 byte dal suo successore sono falsi positivi.

4.1 Esempio

Data in input la sequenza:

traffico di rete: aaavvcffdfsfdsfsdfnfnsdnfndsfnsfsdfdsfddfffgggg

signature 1: aaa

signature 2: aaavv

signature 3: fsd

signature 4: fsdn

il programma stampa:

rilevato n: 1 attacco signature 2

rilevato n: 2 attacchi fsd di cui 1 e‘ falso positivo

rilevato n: 1 attacco fsdn