1 Poeti Stocastici

Si consideri una struttura tipo dizionario le cui chiavi sono costituite da parole e ad ogni parola è associata una lista di frasi (le frasi non contengano spazi).
Si rappresenti tale struttura in maniera opportuna e si scriva un algoritmo che componga una poesia selezionando, stocasticamente, per ogni parola chiave scelta, una frase dalla relativa lista.
Per testare l'implementazione, viene dato in input un file contenente parole e corrispondente lista di frasi. Inoltre vengono anche indicate delle parole, scelte tra le parole chiave, da utilizzate per scegliere le frasi che comporranno la poesia.
Le frasi estratte vengono concatenate nell'ordine di estrazione e nell'output ogni frase è seguita da ". " (un punto e uno spazio).
Il seme (seed) da assegnare al generatore pseudocasuale che permette di selezionare le frasi, è fornito nella prima riga del file in input, nella seconda riga vi sono scritte tutte le chiavi del dizionario, mentre le parole chiave scelte per il test sono indicate nella terza riga.

N.B. Per la verifica dell'output estrarre le frasi nell'ordine in cui sono presentate le parole chiave scelte. In un'ultima riga del file di output riportare anche la serie dei valori casuali generati.

Un esempio di file di input è il seguente:

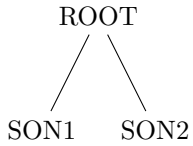
```
598 (seme)
Mare Strada Cielo
Mare Strada
lista delle frasi corrispondenti alle parole chiave:
Mare : [ IlBluDelMareNonHaLimiti , IlMareViveInOgnunoDiNoi ,
    ChiHaRobaInMareNonHaNulla ,
    UnMareCalmoNonHaMaiFattoUnBuonMarinaio ]
Cielo : [ NonMiStancoMaiDiUnCieloAzzurro ,
    SoloIlCieloTiParlaInPienoViso ,
    MeNeVadoInGiroConIlCieloInTascaAllInsaputaDeiPoeti ]
Strada : [ NonPuoiViaggiareSuUnaStradaSenzaEssereTuStessoLaStrada ,
    TuttiIPiuGrandiPensieriSonoConcepitiMentreSiCammina ,
    LaTartarugaBatteAchillePercheConosceLaStrada ,
    NonTiAffrettareIlBuonCamminatoreArriva ,
    SoloIDemoniPercorronoStradeDiritte ]
```

L'algoritmo produrrà per questo specifico input il seguente output:

```
ChiHaRobaInMareNonHaNulla. TuttiIPiuGrandiPensieriSonoConcepitiMentreSiCammina.
2, 1
```

2 Heap

Viene dato in input un albero binario non vuoto descritto in un'unica linea nel file della forma `( ROOT ( SON1 , SON2 ) )` dove ROOT è il nodo radice e `SON1` e `SON2` sono il figlio sinistro e il figlio destro rispettivamente. Se uno di questi nodi è nullo viene codificato con "`-`" ottenendo ad esempio `( ROOT ( SON , - ) )` quando manca il figlio destro e `( ROOT ( - , SON ) )` quando manca il figlio sinistro. Se uno dei nodi è una foglia lo si codifica con il suo valore numerico, se invece è un nodo che ha a sua volta altri figli viene codificato ricorsivamente nella maniera medesima.
N. B. tutti gli elementi sono volutamente separati da spazi.



Bisogna scrivere un algoritmo che controlli se l'albero dato in input è un heap. In caso negativo l'algoritmo dorà scrivere sul file di output la parola False, in caso positivo si scriva sul file di output l'implementazione dell'heap sotto forma di array.
Un esempio di file di input è il seguente:

```
( 1231 ( 1201 ( 684 , 732 ) , 1053 ( 150 , - ) ) )
```

Che rappresentando un heap bisogna produrre in output il file:

```
1231, 1201, 1053, 684, 732, 150
```

3 Semina

Sono dati N vasi in posizioni fissate lungo una dimensione. Bisogna piantare N semi inizialmente sparsi lungo la stessa dimensione in questi vasi sapendo che ogni vaso può contenere soltanto un seme. Un seme può essere sottoposto alle seguenti mosse elementari: essere lasciato nella posizione in cui si trova (z), essere spostato in avanti di una unità ($z \to z + 1$) o essere spostato indietro di una unità ($z \to z - 1$). I semi possono essere spostati contemporaneamente. Ogni passo[1] corrisponde ad un consumo di energia pari ad 1J (Joule).
Piantare tutti gli N semi negli N vasi tramite una certa composizione di mosse elementari in modo da minimizzare il consumo energetico.
Vengono dati in input due file: il primo contiene su una sola riga, separate da virgole, le posizioni iniziali dei semi, il secondo file contiene nello stesso formato le posizioni fisse dei vasi.
L'algoritmo deve produrre un file di output contenente il numero di Joule consumati per posizionare i semi.

Esempio:

File di input 1:

```
-8, -19, 20, 20, 37
```

File di input 2:

```
25, -11, 29, 32, 44
```

L'algoritmo produrra in output il seguente file:

```
33
```

[1] Un passo corrisponde alla mossa elementare su uno o più semi semi contemporaneamente. Ad esempio, spostare un solo seme di una unità corrisponde al consumo di 1J, lo spostamento di una unità di tre semi, se contemporaneo, corrisponde ugualmente al consumo di 1J.

4 Caccia al Tesoro

In una caccia al tesoro vengono collocati, in vari punti di un campo di gioco, degli indizi scritti su foglietti. Ogni foglietto contiene la descrizione di alcuni altri punti del campo di gioco dove sono collocati altri foglietti. Il gioco consiste nel partire da un punto del campo di gioco dove si trova un foglietto iniziale, per poi andare alla ricerca di altri foglietti, e si raggiunge la vittoria quando si trova un foglietto che contiene la descrizione del punto di partenza. Bisogna calcolare il minimo numero di foglietti che è necessario reperire e leggere per poter vincere il gioco.
In particolare bisogna scrivere un programma che riceve in input un file che contiene in una prima riga il numero di punti del campo di gioco in cui sono stati nascosti i foglietti: se tale numero è N, i punti sono identificati da 0, 1, 2, .. , N-1. Il punto di partenza è 0. Nel file, ci sono poi N righe, una per ogni punto del campo di gioco. Ogni riga contiene un numero M di posti descritti nel foglietto presente nel relativo punto (il primo numero della riga), seguito dall'indicazione di tali M punti.
Un esempio di file in input è il seguente:

```
3
2 1 2
2 1 0
2 1 2
```

Il programma deve generare in output un file che inizia con il numero K di foglietti minimo da leggere per vincere il gioco, seguito da K righe, indicanti i K posti in cui tali foglietti vengono trovati. Ad esempio, il precedente input avrà come output:

```
2
0
1
```

in quanto si può vincere il gioco leggendo due foglietti, uno al punto di partenza che descrive il punto 1, dove può essere trovato un foglietto che descrive il punto iniziale.

5 Pallina che Cade

Una pallina cade lungo un piano inclinato suddiviso in tante celle, organizzate in N righe ed M colonne. Ogni cella contiene un chiodo che impedisce alla pallina di scendere verticalmente; quindi quando la pallina raggiunge una cella, la successiva lungo il piano inclinato sarà una cella della riga sottostante, ma in una colonna a fianco (la prima a destra oppure la prima a sinistra). Ogni cella attraversata fornisce un punteggio. La pallina può iniziare a scendere a partire da una qualsiasi cella della prima riga, in alto nel piano inclinato. Bisogna calcolare il percorso che permette alla pallina di accumulare il massimo punteggio durante la discesa.
In particolare, bisogna scrivere un programma che riceve in input un file che contiene in una prima riga i due numeri interi N ed M (righe e colonne, rispettivamente), seguita da N righe, ognuna contenente i punteggi (in formato double) delle M celle della relativa riga.
Un esempio di input può essere:

```
3 3
0.1 2.6 3.7
2.1 4.5 6.5
13.7 5.5 9.8
```

In un file di output devono essere inseriti, per ogni riga, la colonna della cella che la pallina deve attraversare per accumulare il massimo punteggio.
L'output dell'esempio, dovrà quindi essere:

```
2
1
0
```