UNIVERSITÀ DELLA CALABRIA

Traccia prova d'esame

Un labirinto `M` è rappresentato da una matrice `n x m` dove ogni cella contiene il simbolo `.` oppure `#` rappresentanti, rispettivamente, un corridoio e un muro. È possibile muoversi soltanto attraverso i corridoi e solo nelle quattro direzioni principali (sopra, sotto, sinistra e destra) e non in diagonale. Una cella `<i,j>` è chiamata *punto di intersezione* se `<i,j>` è un corridoio e ha almeno 3 direzioni valide in cui è possibile muoversi, ovvero `<i,j>` è collegato direttamente con almeno 3 corridoi. Avendo in input `M`, si deve:

1. Trovare tutti i punti di intersezione in `M`,
2. Per ogni punto di intersezione `p` in `M`, si devono trovare tutti i punti di intersezione più vicini a `p`.

Si deve stampare in output il numero di punti di intersezione presenti in `M` e le coordinate di tutti i punti di intersezione, corredati delle coordinate dei punti di intersezione vicini. Un punto di intersezione è vicino di un altro punto di intersezione se è possibile raggiungere quest'ultimo a partire dal primo con un cammino tramite i corridoi. I punti di intersezione devono essere stampati in ordine a partire dalla cella in alto a sinistra della matrice, rappresentata dalle coordinate `<0,0>` (dati due punti `<a,b>` e `<c,d>`, `<a,b>` è più in alto a sinistra di `<c,d>` se `a<=c` e, solo nel caso `a=c`, deve essere `b<d`).

Struttura dell'input

L'input consiste in un certo numero di righe. La prima riga è nel formato `n m`, dove `n` e `m` sono interi rappresentanti rispettivamente il numero di righe e colonne del labirinto. Le successive `n` righe sono formate da `m` caratteri, dove ogni carattere può essere `.` oppure `#`.

Struttura dell'output

L'output della soluzione consiste in un certo numero di righe. La prima riga contiene un numero intero `x` che rappresenta il numero di punti di intersezione in `M`. Le successive `c` righe sono nel formato `(i, j) -> {(k1, k2), (k3, k4), …, (kz-1, kz)}`, dove `(i, j)` rappresenta la coordinata di un punto di intersezione in `M` e le coppie `(k1, k2), (k3, k4), …, (kz-1, kz)}`, rappresentano i punti di intersezione più vicini a `(i, j)`. Se per un punto di intersezione `(i, j)` non esistono punti di intersezione vicini raggiungibili, la riga sarà nel formato `(i, j) -> {}`.

Esempio input – output

```
4 10
##########
##.#...#.#
###...#...
##########
```

```
3
(1, 5) -> {(2, 4)}
(2, 4) -> {(1, 5)}
(2, 8) -> {}
```

Regole e istruzioni

- Si può scegliere se usare C++ o Java; in entrambi i casi, si presuppone che lo studente sappia compilare il codice sorgente e avviare l'eseguibile ottenuto tramite terminale/prompt dei comandi.
- Si può assumere che l'input sia sempre corretto.
- Il vostro programma deve **leggere da stdin** e **scrivere su stdout**. La lettura da file o l'hardcoding di un input nel vostro programma non rappresenta una soluzione corretta.
- Per effettuare le (vostre) varie prove, potete creare dei file testuali contenenti input di prova e

  - Scrivere o copiare riga per riga il vostro input al programma,
  - Reindirizzare il contenuto del file di input al vostro programma (consigliato).
- Il vostro programma verrà valutato su vari input utilizzando il secondo metodo.

Come posso reindirizzare su stdin?

- Supponendo di utilizzare C++ e di essere su Linux/OS X

  ```
  cat input.txt | ./programma
  ```

  dove input.txt è un file testuale contenente un input e programma è l'eseguibile ottenuto dalla compilazione del vostro codice sorgente (input.txt e programma devono essere nella stessa cartella)

- Supponendo di utilizzare C++ e di essere su Windows

  ```
  type input.txt | programma.exe
  ```

  dove input.txt è un file contenete un input e programma.exe è l'eseguibile ottenuto dalla compilazione del vostro codice sorgente (input.txt e programma.exe devono essere nella stessa cartella).