Laboratorio di algoritmi e strutture dati

Modellazione con grafi*

Docente: Violetta Lonati

1 Sunnydale

1.1 Descrizione del problema

Contesto Sunnydale è una città che ospita un elevatissimo numero di vampiri. I vampiri non possono sopportare la luce solare; l'attraversamento delle gallerie sotterranee di Sunnydale è sempre stato il mezzo preferito dai vampiri per muoversi nella città. I continui crolli delle gallerie hanno creato dei fori nei soffitti, rendendone alcune troppo luminose per un attraversamento tranquillo e sereno.

Harmony, una ragazza-vampiro, si muove per le gallerie sotterranee secondo una regola tanto semplice quanto tassativa: *ad ogni svincolo sceglie sempre e comunque la galleria meno luminosa*. Harmony vuole andare a trovare la sua amica Sarah.

Avete a disposizione queste informazioni: N è il numero degli svincoli (numerati da 1 a N); M è il numero delle gallerie; H è l'indice dello svincolo dove abita Harmony; S è l'indice dello svincolo dove abita Sarah. Inoltre, per ogni galleria, sapete quali svincoli collega e conoscete la sua luminosità, indicata da un numero intero (maggiore è il numero, maggiore è la luminosità).

Problema Sapendo che non esistono due gallerie egualmente luminose, bisogna determinare se Harmony possa raggiungere la casa sotterranea di Sarah e, in caso affermativo, quante gallerie le sono necessarie per arrivare.

1.2 Modellazione

Modellate la situazione con un grafo:

1. Cosa rappresentano i nodi?
2. Cosa rappresentano gli archi?
3. Che caratteristiche ha il grafo (orientato? connesso? pesato?)
4. Formulate il problema in termini di grafo.

*Ultimo aggiornamento: 16 gennaio 2023 - 11:29:22

Progettate un algoritmo che risolve il problema.

1.3 Implementazione

Considerate nuovamente la situazione e il problema descritti in precedenza, e l'algoritmo che avete progettato per l'esercizio precedente.

Implementate l'algoritmo con un programma Go. Il formato di input deve essere il seguente:

- La prima riga dell'input è composta dai quattro numeri interi N, M, H e S definiti come sopra;
- Ognuna delle successive M righe descrive una galleria e contiene tre numeri interi A, B e L separati da uno spazio: i primi due rappresentano gli svincoli collegati dalla galleria mentre il terzo rappresenta il suo grado di luminosità.

L'output dovrà contenere un solo numero: il numero di gallerie che Harmony percorrerà per raggiungere la casa di Sarah, oppure -1.

Esempi di esecuzione

Esempio 1 Ricevendo questo input

```
5 6 1 5
1 2 5
2 3 1
3 4 3
4 5 2
5 1 6
1 4 4
```

il programma stampa

```
2
```

Esempio 1 Ricevendo questo input

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

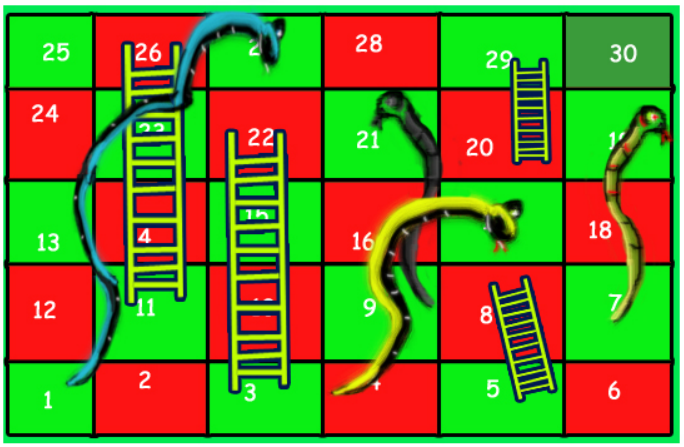
il programma stampa

```
-1
```

2 Scale e serpenti

2.1 Descrizione del problema

Regole del gioco Il gioco "Scale e serpenti" si gioca su una griglia rettangolare. che contiene numeri, scale e serpenti, come nella seguente figura:



Siano r e c il numero di righe e di colonne nella griglia, e $n = r \times c$ il numero totale di caselle. La griglia contiene i numeri da 1 a n, uno per casella. Si parte mettendo la pedina nella casella 1, si lancia un dado e si sposta avanti la pedina in base al numero ottenuto. Le scale e i serpenti determinano dei *salti*: quando si raggiunge una casella che contiene la base di una scala, si deve salire la scala fino a raggiungere la casella in cui si trova la cima della scala; quando si raggiunge una casella con la bocca di un serpente, si deve scendere fino alla casella in cui si trova la coda del serpente. L'obiettivo è raggiungere la casella numerata con n. Se con un lancio si supera il numero n, la partita è persa.

Esempio. Nella griglia di gioco rappresentata in figura, se si lancia il dato tre volte ottenendo i numeri 2, 6, 2, si procede come segue:

- dalla casella 1, avendo lanciato 2, si raggiunge la casella 3 che contiene la base di una scala, quindi si sale fino alla casella 22;
- dalla casella 22, avendo lanciato 6, si raggiunge la casella 28;
- infine, dalla casella 28, avendo lanciato 2, si raggiunge 30.

Si vince la partita anche con le sequenze di lanci (2, 2, 6), oppure (2, 5, 2, 6, 2), oppure (5, 6, 6, 6, 6).

Problema Vogliamo stabilire il numero minimo di mosse (lanci di dado) necessarie per vincere la partita.

2.2 Modellazione

1. Descrivete e formalizzate il gioco in termini di grafi, specificando in particolare cosa rappresentano i nodi e cosa rappresentano gli archi. Indicate inoltre le caratteristiche fondamentali del grafo (è orientato? è connesso? è pesato? ha delle regolarità particolari? ecc...)

2. Riformulate, in termini di grafi, il problema descritto alla fine della sezione 2.1.

Suggerimenti:

- Osservate innanzitutto che non è necessario memorizzare tutte le informazioni relative alla griglia di gioco. Ai fini del calcolo delle mosse, infatti, la griglia di gioco può essere descritta sinteticamente tramite il numero n e la sequenza di interi $a_1, a_2, \ldots a_n$ che indicano gli eventuali salti. Più precisamente:
  - se la casella numerata con i non contiene né la base di una scala né la bocca di un serpente, allora a_i è pari a 0;
  - altrimenti a_i è il numero della casella in cui saltare, una volta raggiunta la casella numerata con i.

  Ad esempio, la descrizione sintetica della griglia sopra disegnata è data dal numero 30 e dalla sequenza di salti
  $\{0,0,22,0,8,0,0,0,0,0,26,0,0,0,0,0,4,0,7,29,9,0,0,0,0,0,1,0,0,0\}$.

- E' utile modellare l'evoluzione del gioco con un grafo, dove ogni nodi rappresenta una configurazione del gioco (ovvero, la posizione della pedina) e tra due vertici $v1$ e $v2$ c'è un arco se dalla configurazione rappresentata da $v1$ è possibile raggiungere la configurazione rappresentata da $v2$ tramite il lancio di un dado.

2.3 Progettazione

1. Progettate un algoritmo che siano in grado di calcolare il numero minimo di lanci necessario per vincere la partita.
2. Modificate l'algoritmo in modo che stampi anche una sequenza di lanci di lunghezza minima che consente di vincere la partita.
3. Modificate gli algoritmi suddetti in modo che si possa specificare la casella di partenza.
4. Modificate ulteriormente gli algoritmi in modo da evitare le mosse che utilizzano scale e serpenti. Ad esempio, nel primo caso l'algoritmo deve calcolare il numero minimo di mosse necessarie per poter vincere la partita partendo da una data casella, ma senza usare né scale né serpenti.

Esempio Se la griglia di gioco è quella rappresentata nella figura, il numero minimo di lanci per raggiungere 30 partendo dalla casella 1 è 3 (ad esempio con i lanci 2, 6, 2). Senza usare scale né serpenti, il numero minimo di lanci invece è 5 (ad esempio con i lanci 5, 6, 6, 6, 6).

2.4 Implementazione e collaudo

Implementate l'algoritmo sviluppato al punto 2 della sezione precedente e collaudatelo usando griglie di gioco differenti. Scrivete almeno 5 griglie diverse: quali casi particolari è utile considerare?

Il programma deve leggere da standard input i dettagli relativi alla griglia e alla posizione di serpenti e scale, quindi deve stampare su standard input:

- il numero minimo di mosse necessario per vincere la partita;
- una sequenza di mosse (lanci di dado) di lunghezza minima che fanno vincere la partita.

Formato di input L'input è formato da:

- due numeri r e c;
- una serie di linee che rappresentano ciascuna la posizione di un serpente (per ogni serpente sono indicati i valori delle caselle che contengono rispettivamente la bocca e la coda del serpente) o di una scala (per ogni scala sono indicati il valore della casella che contiene la base della scala e il valore della casella che contiene la cima della scala).

Formato di input L'output deve essere formato da:

- il numero *min* di mosse minime necessarie per raggiungere la casella n partendo da 1
- una sequenza di mosse (lanci di dado) di lunghezza minima che consentono di raggiungere la casella n partendo da 1

Esempio di esecuzione Nel caso della griglia rappresentata in figura, il serpente con la coda nella casella 1 e la bocca nella casella 27 è indicato dalla coppia `27 1`; la scala che ha la base nella casella 11 e la cima nella casella 26 è indicata dalla coppia `11 26`. Nel complesso, l'input è dato da:

```
5 6
27 1
21 9
17 4
19 7
11 26
3 22
20 29
5 8
```

mentre l'output è

```
3
2 6 2
```

NB: la seconda riga potrebbe essere diversa, poiché possono esistere più sequenze di mosse di lunghezza 3.