S2-Grafos

Camino entre dos vértices usando DFS - Básico

Camino entre dos vértices usando BFS - Básico

Camino entre dos vértices usando DFS - Básico

Ficheros requeridos: Caminos.py (Descargar)
Tipo de trabajo: Individual

Camino entre dos vértices usando DFS

En el videojuego Age of Empires 1: Rise of Rome, el objetivo del juego es conquistar otras civilizaciones. Para lograrlo, es necesario enviar ejercito entre diferentes puntos del mapa. El mapa se representa internamente como un grafo.

Un problema que se presenta es la existencia de islas porque esto significa que el grafo no es conexo y, por consiguiente, no se puede llegar desde cualquier punto del grafo a cualquier otro punto.

Usando el recorrido primero en profundidad (siglas en inglés de DFS), implemente un programa que retorne si hay camino o no entre un vértice a y un vértice b en un grafo que no es necesariamente conexo.

Usando el recorrido primero en profundidad (siglas en inglés de DFS), implemente un programa que retorne si hay camino o no entre un vértice a y un vértice b en un grafo que no es necesariamente conexo.

Entrada: vértice **a** (vértice inicial del camino), vértice **b** (vértice final del camino), matriz de adyacencias m que representa el grafo, es decir el mapa.

La matriz de adyacencias binaria, contiene 1 si hay conexión entre dos vértices y 0 si no hay conexión entre esos vértices.

Salida: (booleano) True si existe camino, **False** si no existe camino.

Ficheros requeridos

Caminos.py

```
def camino(a,b,mat):
    # codigo aqui
```

S2 - Una ciudad unida - Sencillo

S2 - Una ciudad unida - Sencillo

Ficheros requeridos: Ciudad.py (Descargar)
Tipo de trabajo: Individual

Una Ciudad Unida

Unos ambiciosos ingenieros tienen pensado construir una gran ciudad a partir de unos planos que elaboraron. El plano de la ciudad consta de *N* esquinas y *M* calles bidireccionales. Cada calle une a lo mas a dos equinas. Se dice que existe un camino que une a dos esquinas v y w, si v = w, si hay una calle conectando directamente a las dos esquinas, o si existe una secuencia de esquinas a1, a2, a3, ..., ak, tal que a1 = v, ak = w, y toda ai (menos ak) está unida directamente con ai+1 por una calle.

Los ingenieros se preguntan si habrán planificado mal algo, como por ejemplo, que si una persona vive en una esquina *A* y quiere ir a visitar a un amigo que vive en una esquina *B*, no exista un camino que le permita llegar a la esquina *B*.

Problema: Debes hacer un programa que dado un plano de la ciudad, determine cuantos pares de esquinas hay tales que no exista un camino que las una.

Entrada:

Descripción: Se pasa a la función dos números enteros *N* (número de esquinas) y *M* (calles), además se pasa la matriz bidimensional *mat* con *M* filas, en cada fila está almacenada información sobre la calle correspondiente, en cada fila hay 2 números enteros representando los números de las esquinas que une la calle.

Salida:

Descripción: Un solo número entero, el cuál contiene el número de pares de esquinas para las cuales no existe camino que las una.

Ejemplo:

Entrada:

N=5

M=3

mat =[[1,2],

[3,4],

[3,5]]

Salida: 6

Consideraciones:

0<2000<N

0<100000<M

Ficheros requeridos

Ciudad.py

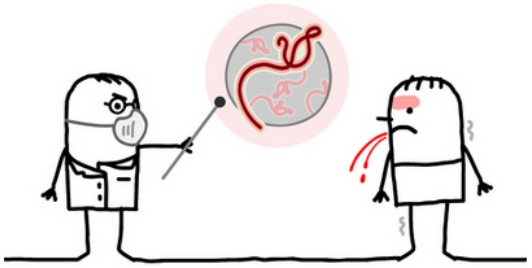
```
1 def ciudadUnida(N,M,mat):
2     # codigo aqui
```

S2 - Contagio de ébola - Intermedio

S2 - Contagio de ébola - Intermedio

Ficheros requeridos: Ebola.py (Descargar)
Tipo de trabajo: Individual

Dada la matriz binaria N * M , 1 representa a la persona sana y 0 representa a un paciente afectado por Ebola. La tarea es verificar el tiempo mínimo requerido para que todas las personas se vean afectadas. Un paciente en la celda [i, j] afecta a una persona en la celda [i, j-1], [i, j + 1] [i + 1, j] e [i-1, j] en un segundo.



Nota: Habrá al menos un paciente

Entrada:

1. A la función se le pasa una matriz que representa las personas infectadas.

2. Se le pasan como parámetros N y M que son las filas y columnas

Restricciones:
1. 1 <= N, M <= 100
2. 0 <= A [i] [j] <= 1

Resultado: Para cada caso de prueba, imprima el tiempo mínimo requerido para todas las personas afectadas por COVID19

Entrada

Primer caso de prueba:
1 0
1 0
Segundo caso de prueba:
1 1 1
1 0 1
1 1 1

Salida

1
2

Explicación: Caso de prueba 2: Después del primer segundo, la matriz se verá como {{1, 0, 1}, {0, 0, 0}, {1, 0, 1}}. Después de dos segundos, la matriz se verá como {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}.

Nota:

- Recuerde que el ejercicio es totalmente INDIVIDUAL.
- El ejercicio debe solucionarse con GRAFOS
- La matriz A, el número N y M se pasan como parámetros de la función

Ficheros requeridos

Ebola.py

```
1 def Ebola(A,N,M):
2     # codigo aqui
3
```

S2 - Toboganes y escaleras - Intermedio

S2 - Toboganes y escaleras - Intermedio

Ficheros requeridos: toboganesyesc.py (Descargar)
Tipo de trabajo: Individual

- **Toboganes y escaleras**

Dado un tablero de toboganes y escaleras, debes encontrar el número mínimo de lanzamientos de dados necesarios para llegar al destino o la última celda desde la fuente o la primera celda. Básicamente, el jugador tiene control total sobre el resultado del lanzamiento de dados y quiere averiguar el número mínimo de lanzamientos necesarios para llegar a la última celda.

Si el jugador llega a una celda que es la base de una escalera, el jugador tiene que subir esa escalera y si llega a una celda que es la entrada del tobogán, tiene que bajar hasta el final del tobogán sin un lanzamiento de dados.

La entrada está representada por dos cosas, primero es 'N' que es el número de celdas en el tablero dado, segundo es una matriz 'move [0... N-1]' de tamaño N. Una entrada de movimiento [i] es -1 si no hay serpiente ni escalera desde i, de lo contrario, el movimiento [i] contiene el índice de la celda de destino para la serpiente o la escalera en i.

Dada la matriz move y N como parámetros, debes retornar el número mínimo de lanzamientos necesarios para llegar a la última celda desde la primer celda (en este caso donde está el conejo)

Nota:

- Recuerde que el ejercicio es totalmente INDIVIDUAL.
- El ejercicio debe solucionarse con RECORRIDOS DE GRAFOS
- La matriz move y el N se pasan como parámetros a la función que debes crear.

Ficheros requeridos

toboganesyesc.py

```
1 def toboganes(move,N):
2     # codigo aqui
```