Pregunta 1

5 puntos Guardar respuesta

En una ciudad futurista, hay dos torres gemelas conectadas por una serie de puentes. Cada torre tiene varios pisos, y cada piso puede tener múltiples habitaciones. Los puentes conectan habitaciones específicas en diferentes pisos de las dos torres. Tu objetivo es encontrar un camino desde la entrada en la planta baja de la primera torre hasta la salida en el último piso de la segunda torre.

Consideraciones:

- Cada **torre** tiene (N) pisos, y cada piso tiene (M) habitaciones.
- Los **puentes** solo conectan habitaciones en diferentes pisos de las dos torres. No hay puentes entre habitaciones del mismo piso o de la misma torre.
- Movimientos Permitidos: Puedes moverte entre habitaciones adyacentes en el mismo piso de una torre, subir o bajar un piso en la misma torre, o cruzar un puente a la otra torre.
- No puedes moverte a una habitación que ya hayas visitado en el mismo intento de solución.
- Objetivo: Encontrar un camino desde la entrada en la planta baja de la primera torre hasta la salida en el último piso de la segunda torre.

El laberinto se puede representar como una matriz tridimensional donde cada celda ((i, j, k)) representa la habitación (k) en el piso (j) de la torre (i). Los puentes se pueden representar como una lista de tuplas ((i1, j1, k1, i2, j2, k2)) que indican que hay un puente entre la habitación ((i1, j1, k1)) y la habitación ((i2, j2, k2)).

ENTRADA

```
N = 3  # Número de pisos
M = 3  # Número de habitaciones por piso


# Matriz tridimensional representando las habitaciones
habitaciones = [
   [[0, 0, 0], [0, 0, 0], [0, 0, 0]],  # Torre 1
   [[0, 0, 0], [0, 0, 0], [0, 0, 0]]   # Torre 2
]
```

```
    [[0, 0, 0], [0, 0, 0], [0, 0, 0]]  # Torre 2
]

# Lista de puentes
puentes = [
    (0, 0, 1, 1, 1, 2),  # Puente entre (Torre 1, Piso 0, Habitación 1) y (Torre 2, Piso 1, Habitación 2)
    (0, 2, 0, 1, 2, 1)  # Puente entre (Torre 1, Piso 2, Habitación 0) y (Torre 2, Piso 2, Habitación 1)
]

# Posición de entrada y salida
entrada = (0, 0, 0)  # Torre 1, Piso 0, Habitación 0
salida = (1, 2, 2)  # Torre 2, Piso 2, Habitación 2
```

NOTA: Estos datos los puede considerar dentro de su programa.

SALIDA:

Camino encontrado: [(0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 1, 2), (0, 1, 1), (0, 1, 0), (0, 2, 0), (0, 2, 1), (0, 2, 2), (1, 2, 2)]

Defina una técnica y genere un algoritmo para explorar todas las posibles rutas desde la entrada hasta la salida, considerando tanto los movimientos dentro de las torres como los cruces de puentes.

Se solicita:

- Desarrollar un código en Python (.py o .ipynb) que permita generar un algoritmo eficiente para resolver el caso planteado.
- Considerar la entrada de datos a partir de un archivo de texto (no considerar ningún ingreso de datos por consola o de forma manual).
- Dibujar el grafo original y el grafo resultante que modelen el caso solicitado.
- Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.

Se tendrá en cuenta para la calificación el orden y la explicación de su solución.

Rúbrica de Calificación

Ítems	Excelente	Regular	Deficiente
Escribir un algoritmo en Python que implemente la entrada/salida de datos descrita, considerando la lectura de la entrada de datos dada.	**2.0**	**1.0**	**0**
Dibujar el grafo original y el grafo resultante.	**2.0**	**1.0**	**0**
Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.	**1.0**	**0.5**	**0**

Adjuntar archivo

Buscar en Archivos locales Buscar en la colección de contenido

Pregunta 2

Necesita calificación

Una estación espacial internacional ha sufrido un fallo catastrófico en su sistema de soporte vital. Los astronautas deben evacuar la estación y llegar a las cápsulas de escape antes de que se agote el oxígeno. La estación espacial está compuesta por múltiples módulos conectados por pasillos. Algunos pasillos están bloqueados debido a daños, y otros tienen diferentes niveles de dificultad para ser atravesados debido a la gravedad cero y otros obstáculos.

Consideraciones:

- La estación espacial se puede representar como un grafo no dirigido, donde cada nodo representa un módulo y cada arista representa un pasillo entre dos módulos.
- Algunos pasillos están bloqueados debido a los daños. Estos pasillos no pueden ser utilizados.
- Los astronautas deben evacuar desde su módulo actual hasta las cápsulas de escape.
- Algunos pasillos tienen diferentes niveles de dificultad para ser atravesados (por ejemplo, algunos pueden estar parcialmente bloqueados pero aún transitables).
- Nivel de dificultad para atravesar el pasillo (0 para pasillos bloqueados, 1 para pasillos fáciles, 2 para pasillos parcialmente bloqueados, etc.).
- Los astronautas tienen un tiempo limitado para llegar a las cápsulas de escape debido a la disminución del oxígeno.

La siguiente tabla muestra la interconexión entre módulos:

Módulo	Módulo	Dificultad	Módulo	Dificultad	Módulo	Dificultad
A	B	1	C	2		
B	A	1	D	1	E	2
C	A	2	F	1		
D	B	1	G	1		
E	B	2	H	1		
F	C	1	Cápsula de Escape	2		
G	D	1				
H	E	1				
Cápsula de Escape	F	2				

Por ejemplo, para primera fila de la tabla: El Módulo A se conecta con el Módulo B y la dificultad de esta conexión es 1. El Módulo A se conecta con el Módulo C y la dificultad de esta conexión es 2.

Se debe representar esta tabla de módulos en el archivo **modulos.txt**, el cual será la entrada para el programa.

ENTRADA (archivo de texto, el cual debe generarse)

modulos.txt

SALIDA (consola)

['Módulo A', 'Módulo C', 'Módulo F', 'Cápsula de Escape']

Encontrar la ruta más corta y segura desde el módulo actual de los astronautas hasta las cápsulas de escape, considerando los pasillos bloqueados y las diferentes dificultades de los pasillos.

Se solicita:

- Desarrollar un código en Python (.py o .ipynb) que permita generar un algoritmo eficiente para resolver el caso planteado.
- Considerar la entrada de datos a partir de un archivo de texto (no considerar ningún ingreso de datos por consola o de forma manual).
- Dibujar el grafo original y el grafo resultante que modelen el caso solicitado.
- Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.

Se tendrá en cuenta para la calificación el orden y la explicación de su solución.

Rúbrica de Calificación

Ítems	Excelente	Regular	Deficiente
Escribir un algoritmo en Python que implemente la entrada/salida de datos descrita, considerando la lectura de la entrada a partir del archivo de texto.	**2.0**	**1.0**	**0**
Dibujar el grafo original y/o el grafo resultante.	**2.0**	**1.0**	**0**
Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.	**1.0**	**0.5**	**0**

Pregunta 3

5 puntos Guardar respuesta

Un almacén tiene múltiples estantes donde se almacenan paquetes, y un robot autónomo necesita planificar la ruta más eficiente para recoger varios paquetes y devolverlos a la estación de carga. El objetivo es minimizar el tiempo total de recorrido desde la estación de carga hasta los estantes y viceversa.

Consideraciones:

- Hay 5 estantes en el almacén.
- Hay una estación de carga desde donde comienza y termina el robot.
- Las conexiones entre los estantes y la estación tienen tiempos de recorrido conocidos.
- Se utilizará la distancia euclidiana (distancia entre dos puntos) para estimar el costo restante.

almacen.txt

```
6
C: (0, 0)
1: (1, 2)
2: (3, 1)
3: (4, 4)
4: (2, 5)
5: (5, 3)
C <-> 1 (2)
C <-> 2 (3)
C <-> 3 (5)
C <-> 4 (4)
C <-> 5 (6)
1 <-> 2 (1)
1 <-> 3 (4)
2 <-> 4 (2)
3 <-> 5 (3)
```

Donde la primera línea del archivo (6) representa la cantidad de líneas que contienen las coordenadas de la estación de carga y los estantes. Es decir, en el archivo anterior hay una estación de carga (C) y 5 estantes (1,2,3,4,5). Así, en la línea dos del archivo, se encuentra la coordenada de la estación de carga: (0,0); en la línea tres del archivo, se encuentra la coordenada del estante uno: (1,2); en la línea cuatro del archivo, se encuentra la coordenada del estante dos (3,1) y así hasta el último estante (estante cinco).

Donde la primera línea del archivo (6) representa la cantidad de líneas que contienen las coordenadas de la estación de carga y los estantes. Es decir, en el archivo anterior hay una estación de carga (C) y 5 estantes (1,2,3,4,5). Así, en la línea dos del archivo, se encuentra la coordenada de la estación de carga: (0,0); en la línea tres del archivo, se encuentra la coordenada del estante uno: (1,2); en la línea cuatro del archivo, se encuentra la coordenada del estante dos (3,1) y así hasta el último estante (estante cinco).

Luego, las siguientes líneas del archivo representan las conexiones (tiempos de recorrido) entre la estación de carga y los estantes y las conexiones entre los estantes; por ejemplo; C <-> 1 (2), representa que la conexión de la estación de carga(C) al estante uno (1) toma un tiempo de 2.

ENTRADA (archivo de texto, el cual debe generarse)

almacen.txt

SALIDA (consola)

```
Ruta óptima desde C hasta 1: C -> 1
Ruta óptima desde 1 hasta 2: 1 -> 2
Ruta óptima desde 2 hasta 3: 2 -> 1 -> 3
Ruta óptima desde 3 hasta 4: 3 -> 1 -> 2 -> 4
Ruta óptima desde 4 hasta 5: 4 -> 2 -> 1 -> 3 -> 5
Ruta total final: C -> 1 -> 2 -> 1 -> 3 -> 1 -> 2 -> 4 -> 2 -> 1 -> 3 -> C
```

Generar las rutas óptimas encontradas desde la estación de carga hasta cada uno de los estantes y mostrar la ruta total que incluye todas las entregas y el regreso a la estación.

Se solicita:

- Desarrollar un código en Python (.py o .ipynb) que permita generar un algoritmo eficiente para resolver el caso planteado.
- Considerar la entrada de datos a partir de un archivo de texto (no considerar ningún ingreso de datos por consola o de forma manual).
- Dibujar el grafo original y el grafo resultante que modelen el caso solicitado.
- Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.

Se tendrá en cuenta para la calificación el orden y la explicación de su solución.

Rúbrica de Calificación

Ítems	Excelente	Regular	Deficiente
Escribir un algoritmo en Python que implemente la entrada/salida de datos descrita, considerando la lectura de la entrada a partir del archivo de texto.	**2.0**	**1.0**	**0**
Dibujar el grafo original y el grafo resultante.	**2.0**	**1.0**	**0**
Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.	**1.0**	**0.5**	**0**

Adjuntar archivo

Buscar en Archivos locales | Buscar en la colección de contenido

Pregunta 4

5 puntos Guardar respuesta

En un sistema de control de versiones, como GitVer, las ramas pueden tener dependencias entre sí. Por ejemplo, una rama puede ser creada a partir de otra, o una rama puede requerir cambios que se han realizado en otra. Estas relaciones pueden ser representadas como un grafo dirigido donde se representen las ramas del repositorio y las dependencias entre ramas (una rama depende de otra)

El objetivo es descubrir grupos de ramas que están interrelacionadas y pueden ser gestionadas juntas.

Generar un algoritmo que lea el archivo de texto y que permita optimizar la gestión y fusión de ramas en el desarrollo del software y el control d versiones. Este algoritmo permitirá descubrir grupos de ramas que están interrelacionadas y pueden ser gestionadas conjuntamente. Esto es crucial para optimizar la fusión y resolución de conflictos entre ramas en el desarrollo del software. Además, esta metodología puede escalarse a repositorios más grandes y complejos con múltiples ramas y dependencias.

ENTRADA (archivo de texto, el cual debe generarse):

A continuación, se muestra algunas ramas y sus interrelaciones, las cuales se encuentran en el archivo de texto:

ramas.txt:

Rama A depende de Rama B
Rama G depende de Rama H
Rama F depende de Rama D
Rama B depende de Rama C
Rama E depende de Rama F
Rama C depende de Rama A
Rama B depende de Rama D
Rama H depende de Rama I
Rama D depende de Rama E
Rama I depende de Rama G

SALIDA (consola):

[[H, I, G], [B, C, A], [E, F, D]]

SALIDA (consola):

[[H, I, G], [B, C, A], [E, F, D]]

Se solicita:

- Desarrollar un código en Python (.py o .ipynb) que permita generar un algoritmo eficiente para resolver el caso planteado.
- Considerar la entrada de datos a partir del archivo de texto (no considerar ningún ingreso de datos por consola o de forma manual).
- Dibujar el grafo original y el grafo resultante que modelen el caso solicitado.
- Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.

Se tendrá en cuenta para la calificación el orden y la explicación de su solución.

Rúbrica de Calificación

Ítems	Excelente	Regular	Deficiente
Escribir un algoritmo en Python que implemente la entrada/salida de datos descrita, considerando la lectura de la entrada a partir del archivo de texto.	**2.0**	**1.0**	**0**
Dibujar el grafo original y/o el grafo resultante.	**2.0**	**1.0**	**0**
Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.	**1.0**	**0.5**	**0**

Adjuntar archivo

Buscar en Archivos locales | Buscar en la colección de contenido

Pregunta 3

Necesita calificac

Un equipo de rescate está intentando salvar a un grupo de personas atrapadas en una ciudad subterránea después de un terremoto. Se be modelar cada ubicación de la ciudad (por ejemplo, una habitación, un túnel, una plaza) y el camino entre dos ubicaciones. Algunas ubicaciones están bloqueadas debido a los escombros, y el equipo de rescate debe encontrar la ruta más eficiente para llegar a las personas atrapadas.

Tomar en cuenta:

- El rescate comienza en una ubicación específica, designada como el nodo de inicio.
- Algunas ubicaciones están bloqueadas y no se pueden atravesar.
- Algunas ubicaciones contienen personas atrapadas que necesitan ser rescatadas.

Por ejemplo

El equipo de rescate cuenta con información de la siguiente distribución:

- La ubicación 1 se encuentra conectada a la ubicación 2 y a la ubicación 3.
- La ubicación 2 está bloqueada y se encuentra conectada a la ubicación 4.
- La ubicación 3 tiene personas atrapadas y se encuentra conectada a la ubicación 5.
- La ubicación 4 no se encuentra conectada a otra ubicación.
- La ubicación 5 tiene personas atrapadas.

La interpretación anterior se puede representar en el archivo **ubicacion.txt**, cuyo contenido es el siguiente:

Ubicacion(1)

Conectada a: Ubicación 2, Ubicación 3

Ubicacion(2, bloqueada)

Conectada a: Ubicación 4

Ubicacion(3, personas_atrapadas)

Conectada a: Ubicación 5

Ubicacion(4)

Conectada a:

Ubicacion(5, personas_atrapadas)

Conectada a:

ENTRADA (archivo de texto, el cual debe generarse)

ubicacion.txt

SALIDA (consola)

Rutas posibles: [[1, 3], [1, 3, 5]]

Personas rescatadas en las ubicaciones: [3, 5]

Desarrollar un algoritmo para encontrar todas las rutas posibles desde el punto de entrada (Ubicación 1) hasta las ubicaciones de las personas atrapadas, evitando las áreas bloqueadas, así como indicar las ubicaciones en las cuales se llegan rescatar personas.

Se solicita:

- Desarrollar un código en Python (.py o .ipynb) que permita generar un algoritmo eficiente para resolver el caso planteado.
- Considerar la entrada de datos a partir de un archivo de texto (no considerar ningún ingreso de datos por consola o de forma manual).
- Dibujar el grafo original y el grafo resultante que modelen el caso solicitado.
- Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.

Se tendrá en cuenta para la calificación el orden y la explicación de su solución.

Rúbrica de Calificación

Ítems	Excelente	Regular	Deficiente
Escribir un algoritmo en Python que implemente la entrada/salida de datos descrita, considerando la lectura de la entrada a partir del archivo de texto.	**2.0**	**1.0**	**0**
Dibujar el grafo original y el grafo resultante.	**2.0**	**1.0**	**0**
Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.	**1.0**	**0.5**	**0**

Pregunta 4

Necesita calificación

En una ciudad, las estaciones de transporte (como trenes, autobuses o metro) pueden ser representadas como un grafo dirigido. El objetivo es descubrir grupos de estaciones que están interconectadas y pueden ser alcanzadas entre sí sin salir del sistema de transporte.

Dado un grafo dirigido que representa las estaciones y sus rutas directas entre estaciones (un viaje directo desde una estación a otra), necesitamos implementar un algoritmo que nos permitirá optimizar la planificación de rutas y mejorar la eficiencia del sistema de transporte. El algoritmo nos debe ayudará a descubrir grupos de estaciones que están interconectadas y pueden ser alcanzadas sin salir del sistema. Esto es crucial para optimizar la planificación de rutas y mejorar la eficiencia del transporte público.

ENTRADA (archivo de texto, el cual debe generarse):

A continuación, se muestra algunas interconexiones entre estaciones, las cuales se encuentran en el archivo de texto estaciones.txt:

estaciones.txt:

Estación A se conecta con B
Estación D se conecta con E
Estación B se conecta con C
Estación G se conecta con H
Estación B se conecta con D
Estación C se conecta con A
Estación E se conecta con F
Estación F se conecta con D

SALIDA (consola):

Recomendaciones dadas por el algoritmo:

Subsistema 1: [G]
Subsistema 2: [H]
Subsistema 3: [B, C, A]
Subsistema 4: [E, F, D]

Se solicita:

- Desarrollar un código en Python (.py o .ipynb) que permita generar un algoritmo eficiente para resolver el caso planteado.
- Considerar la entrada de datos a partir del archivo de texto (no considerar ningún ingreso de datos por consola o de forma manual).
- Dibujar el grafo original y el grafo resultante que modelen el caso solicitado.
- Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.

Se tendrá en cuenta para la calificación el orden y la explicación de su solución.

Rúbrica de Calificación

Ítems	Excelente	Regular	Deficiente
Escribir un algoritmo en Python que implemente la entrada/salida de datos descrita, considerando la lectura de la entrada a partir del archivo de texto.	2.0	1.0	0
Dibujar el grafo original y/o el grafo resultante.	2.0	1.0	0
Sustentar que técnica y/o tipo de algoritmo ha utilizado para implementar la solución.	1.0	0.5	0