

Práctica 3

Problema 1

Detección de Minas Subterráneas

Archivo de entrada: `minas.in` **Archivo de salida:** `minas.out`

Trabajas para una empresa de exploración geológica que utiliza sensores de alta resolución para detectar yacimientos subterráneos. Tu tarea es desarrollar un programa que analice un mapa digitalizado del terreno y localice posibles "minas" (áreas de interés geológico) basándose en los datos de densidad del subsuelo.

Cada elemento de una matriz representa el valor de densidad de un punto específico en el terreno. Estos valores oscilan entre 0 y 50. Una mina se considera presente en una celda (i,j) si la densidad en esa celda es inusualmente alta en comparación con sus vecinos.

Condición para considerar una mina: Una mina se encuentra en la posición (i,j) si se cumple la siguiente condición:

$$(MD(i,j) + \text{promedio de las 8 intensidades vecinas}) > 40.0$$

Donde $MD(i,j)$ es el valor de densidad de la matriz digitalizada en la posición (i,j). Las "8 intensidades vecinas" se refieren a las celdas directamente adyacentes (arriba, abajo, izquierda, derecha y las cuatro diagonales). Si una celda no tiene 8 vecinos (está en un borde o esquina), se consideran solo los vecinos existentes para calcular el promedio.

Entrada (minas.in):

La primera línea contendrá dos enteros separados por un espacio, representando el número de filas (F) y columnas (C) de la matriz, respectivamente. Ambos valores serán menores que 10 ($F < 10$, $C < 10$).

Las siguientes F líneas contendrán C enteros cada una, separados por un espacio, que representan los valores de densidad de cada posición de la matriz.

Ejemplo del archivo de entrada:

```
5 7
10 12 15 20 25 22 18
8 11 14 30 40 35 20
9 13 16 28 38 32 21
7 10 13 25 30 28 19
5 8 11 18 22 20 15
```

Salida (minas.out):

La salida debe ser una representación de la matriz donde cada posición

(i,j) que contenga una mina se marque con un asterisco (*), y las posiciones donde no haya mina se marquen con un espacio en blanco ().

La matriz resultante debe estar rodeada por un borde que indique las coordenadas de cada fila y columna, comenzando desde la coordenada 1. Cada elemento de la matriz (asterisco o espacio en blanco) debe estar separado por un espacio en blanco.

Ejemplo del archivo de salida (correspondiente al ejemplo de entrada):

```
 1 2 3 4 5 6 7
1
2      * *
3      *
4
5
 1 2 3 4 5 6 7
```

Problema 2: Análisis de Red de Transporte Urbano

Archivo de entrada: `red_transporte.txt` **Archivo de salida:** `rutas_optimas.txt`

Trabajas para el departamento de planificación de transporte de una ciudad. Te han solicitado desarrollar un programa que analice una red de transporte público (metros, autobuses) y calcule las rutas óptimas entre diferentes estaciones. La red se representa como un **grafo dirigido y ponderado**, donde las estaciones son nodos y las rutas entre ellas son aristas con un costo asociado (tiempo de viaje, tarifa, etc.).

Representación del Grafo:

- **Nodos (Estaciones):** Cada estación se identifica con un ID numérico único (entero).
- **Aristas (Rutas):** Una arista desde la estación A a la estación B significa que hay una ruta directa. Cada arista tiene un **peso** asociado, que es el "costo" (ej. tiempo en minutos) de viajar de A a B. Las rutas pueden ser unidireccionales (grafo dirigido).

Funcionalidades del Sistema:

1. Carga de la Red de Transporte:

- El programa debe leer la configuración de la red desde el archivo `red_transporte.txt`.
- Este archivo tendrá el siguiente formato:
 - La primera línea contendrá dos enteros: `N` (número total de estaciones) y `M` (número total de rutas).
 - Las siguientes `M` líneas representarán las rutas, cada una con tres enteros: `Estacion_Origen Estacion_Destino Costo`.

2. Implementación del Grafo:

- Deberás representar el grafo en memoria utilizando una **lista de adyacencia**. Esto significa que para cada estación, mantendrás una lista de todas las estaciones a las que se puede llegar directamente desde ella, junto con el costo asociado.
- Cada lista de adyacencia (para cada nodo) debe ser una **lista enlazada dinámica** de nodos adyacentes (`struct Adyacente` con `id_destino`, `costo`, `siguiente_adyacente`).

3. Algoritmo de Búsqueda de Ruta Óptima:

- Implementar el algoritmo de **Dijkstra** para encontrar la ruta de menor costo (tiempo) entre dos estaciones dadas.
- El algoritmo de Dijkstra requiere el uso de estructuras de datos auxiliares (como una cola de prioridad, que en C se puede simular con un arreglo o lista ordenada, o una min-heap si buscas mayor optimización).

Menú Interactivo (Front-End en Consola):

1. **Mostrar Red (Opcional):**
 - Permitir al usuario ver una representación de la red (ej: listar cada estación y sus conexiones directas con costos).
2. **Calcular Ruta Óptima (Opción Principal):**
 - Pedir al usuario el ID de la "Estación de Origen" y el ID de la "Estación de Destino".
 - Ejecutar el algoritmo de Dijkstra para encontrar la ruta más corta.
 - Mostrar la **ruta completa** (secuencia de estaciones) y el **costo total** de esa ruta.
 - Si no hay ruta posible, indicarlo.
3. **Salir:** Terminar el programa y liberar toda la memoria.

Ejemplo del archivo de entrada (`red_transporte.txt`):

```
5 7
1 2 10
1 3 30
2 4 20
3 4 10
3 5 40
4 5 15
5 1 5
```

(Esto representa 5 estaciones y 7 rutas. Por ejemplo, de la estación 1 a la 2 hay un costo de 10).

Ejemplo de Interacción y Salida Esperada (Para "Calcular Ruta Óptima"):

```
Ingresa Estación de Origen: 1
Ingresa Estación de Destino: 5
```

```
La ruta optima de 1 a 5 es:
Estación 1 -> Estación 3 (Costo: 30)
Estación 3 -> Estación 4 (Costo: 10)
Estación 4 -> Estación 5 (Costo: 15)
Costo total de la ruta: 55
```

(O la ruta podría ser 1 -> 2 (10) -> 4 (20) -> 5 (15) = 45) (El algoritmo debe encontrar la más corta: 1 -> 2 (10), 2 -> 4 (20), 4 -> 5 (15) = 45)

NOTAS:

- **Guarde su práctica en una carpeta con:**
- **Primer-Apellido-4utilmos-Dígitos-de-su-Cedula**
- **Comprímalo en un .rar o .zip.**
- **Deposite su práctica en el aula virtual.**
- **No se repetirán prácticas. Es su responsabilidad que su práctica se deposite correctamente en la dirección dada.**
- **Práctica de código compartido tienen 0 Puntos.**
- **El código debe de ser defendido en la hora pautada con el docente.**