Examen Extraordinario de Programación
Curso 2022-2023
Parte 1

Alimentando más Leones

Regresamos al zoológico de 26, donde los trabajadores se han topado con un nuevo problema. Un visitante (de un tal pueblo Macondo) ha traído la enfermedad del imsomnio y contagiado a los leones con ella. Esto representa Un problema para los pobres trabajadores que se encargan de alimentar a las bestias, pues los carritos que utilizan para entregarles la carne tienen poca seguridad y no es tarea fácil acercarse a un león despierto y, además, malhumorado por falta de sueño. La buena noticia es que Larry, el director del zoológico ha hecho un gran descubrimiento. Se dio cuenta de que cada león tiene una cantidad de comida favorita, y si se le alimenta todos los días con exactamente esa cantidad, el animal se mantiene feliz y se abstiene de utilizar empleados como merienda. Por desgracia, cada carrito, además de contar con gasolina limitada y poca protección, también tiene una cantidad de comida fija que puede transportar. Por tanto Larry necesita una nueva planeación de rutas para llevar a cada león su comida requerida y, al mismo tiempo, optimizar la distancia recorrida por los carritos para prevenir el derroche de combustible.

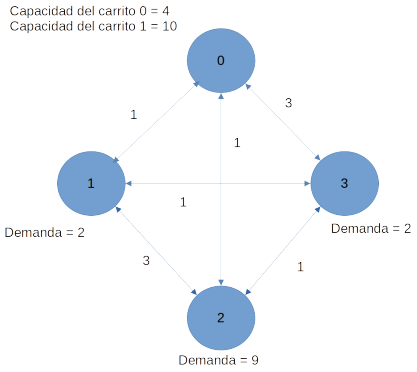
Se cuenta con n trabajadores y m posiciones por las que estos se desplazan. El depósito de la carne está representado por 0, mientras que la posición de cada león se representa con un entero entre 1 y m - 1. Luego de alimentar sus respectivos leones, cada trabajador debe regresar al depósito. Los leones nunca se mueven de sus respectivas posiciones. Se conoce la distancia entre cada par de leones, así como la distancia entre cada león y el depósito. Para cada león se conoce su demanda de comida y para cada carrito su capacidad de carga.

Su objetivo es encontrar una distribución de n rutas (una por cada trabajador) de forma que cada león sea visitado por exactamente un trabajador, que la capacidad de carga de cada trabajador sea mayor o igual a la suma de las demandas de cada león que este haya visitado y que la suma de las distancias recorridas por todos los carritos sea mínima. Cualquier ruta en la que la suma de las demandas de los leones visitados supere a la capacidad de su trabajador es inválida. Note que, de ser necesario, un trabajador puede no salir del depósito (o sea, quedarse en el depósito sin visitar a ningún león).

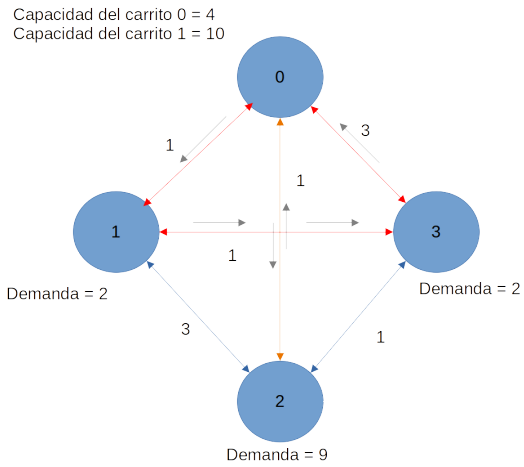
Ejemplo

A continuación se presenta un ejemplo de problema con tres leones (con sus demandas), dos trabajadores (con sus capacidades), y su respeciva matriz de distancia y representación visual.

	0	***1***	***2***	***3***
0	0	1	1	3
1	1	0	3	1
2	1	3	0	1
3	3	1	1	0



Una posible solución del problema es la siguiente:



En este caso, un trabajador (el trabajador 0) se desplaza hacia 1, luego hacia 3 y regresa a 0 (camino [1,3,0]). Mientras que el otro (el trabajador 1) se desplaza hacia 2 y regresa a 0 (camino [2,0]). Esta solución tiene costo 7 y cumple con las restricciones de capacidad. El trabajador 0 utiliza 4 de su capacidad (dos por el león 1 y dos por el león 3), mientras que el trabajador 1 utiliza 9 de su capacidad (nueve por el león 2).

Note que de haber utilizado un solo carrito para hacer el recorrido [1,3,2,0] mientras el otro se queda en el depósito, se hubiera obtenido una solución de costo 4. Sin embargo, esta no es una solución válida pues ni el carrito 0, ni el carrito 1 tienen suficiente capacidad para satisfacer la suma de las demandas de los tres leones (2+2+9).

Implementación

Para solucionar el problema cuenta con la siguiente clase, dentro de Utils.cs:

```
1 public class Map
2 {
3     private int[,] Distances { get; set; }
4     public int M {get; private set; }
5     public int[] Demand {get; private set;}
```

```
6
7       public Map(int[,] distances, int[] demand)
8       {
9           Distances = distances;
10          M = Distances.GetLength(0);
11          Demand = demand;
12
13      }
14
15      public int this[int i, int j]
16      {
17          get => Distances[i,j];
18      }
19  }
```

La propiedad `Distances` representa la matriz y `M` la cantidad dotal de leones del mapa (más el depósito). El array Demand, de tamaño M, contiene la demanda de cada león. La posición 0 siempre tendrá valor 0 pues el depósito no demanda comida. Un mapa no puede ser modificado una vez construido. Al indexar dos enteros en una instancia de la clase Map se obtiene la distancia de los dos leones (o depósito) representados por esos enteros.

Usted debe implementar el siguiente método dentro de la clase Solution, en el archivo Solution.cs:

```
1  public static int Solve(Map map, int[] capacities)
2  {
3      throw new NotImplementedException();
4  }
```

El método *Solve* retorna el costo de la planeación de rutas óptima. El array de enteros *capacities* tiene tamaño igual a la cantidad de trabajadores (o carritos, es lo mismo). En cada posición tiene la capacidad de su correspondiente carrito.

Consideraciones

- Todos los leones son representados por números enteros entre 1 y *m* - 1.
- En caso de no existir solución, debe retornar el valor int.MaxValue.
- Cada león puede ser visitado una sola vez y sólo se puede regresar al dpósito luego de finalizar cada ruta.

En el archivo `Program.cs` se muestra un ejemplo, sin embargo, usted puede (y debe) crear sus propios casos para poner a prueba su solución.