Examen Final de Programación
Curso 2022-2023
Parte 1

Alimentando Leones

Por algún extraño motivo, en los últimos años el zoológico de 26 se ha llenado de leones. Esta es una buena noticia para el público que disfruta viendo a los animales, pero es una mala noticia para los trabajadores que tienen que alimentarlos. Para mayor desgracia de los pobres empleados, los carritos con los que se mueven alrededor de la zona de leones no tienen muchas medidas de seguridad (lo cual presenta un problema a la hora de acercarse a las bestias) y además casi siempre tienen poca gasolina. Por suerte, los leones de La Habana (nuevamente, por algún extraño motivo) nunca se mueven de sus lugares y siempre duermen a la misma hora. Larry, el director del zoológico quiere aprovechar esto para planificar recorridos con los que se gaste la menor cantidad de combustible posible y además se alimente a cada león sólo cuando esté dormido. Su tarea consiste en ayudar planeando las rutas de los recorridos.

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

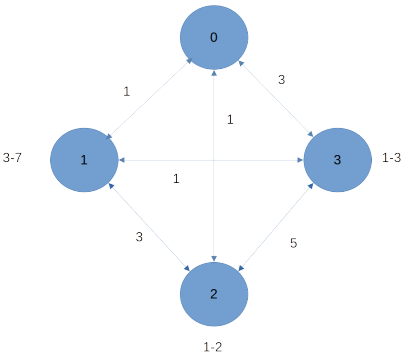
posición de cada león se representa por un entero entre 1 y m - 1. Luego de alimentar sus respectivos leones, cada trabajador debe regresar al depósito. Se conoce la distancia entre cada par de leones, así como la distancia entre cada león y el depósito. Además, para cada león se conoce el momento en que se duerme y el momento en que despierta.

Su objetivo es encontrar una distribución de n rutas (una por cada trabajador) de forma que cada león sea visitado por un trabajador durante su horario de sueño y la suma de las distancias recorridas por todos los carritos sea mínima. Cualquier ruta en la que algún león sea visitado fuera de su horario de sueño es inválida. Note que, de ser necesario, un trabajador puede no salir del depósito y eso se considera también una ruta. Un trabajador puede dejar de moverse sólo cuando haya alcanzado el final de su ruta y recorrer una unidad de distancia implica el paso de exactamente una unidad de tiempo. Por tanto, a lo largo del viaje de un trabajador por su ruta, el tiempo transcurrido coincide con la distancia recorrida.

Ejemplo

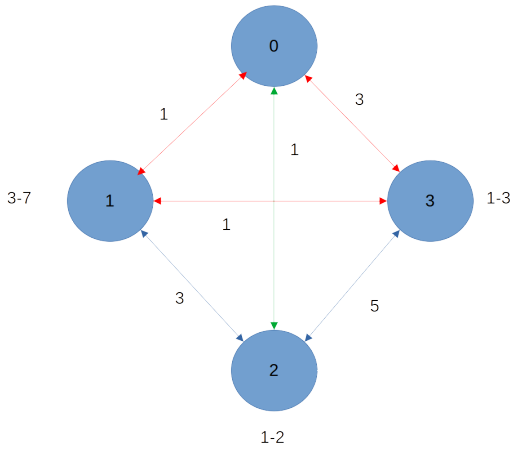
A continuación se presenta un ejemplo de problema con tres leones y dos trabajadores con su respeciva matriz de distancia y representación visual.

	0	***1***	***2***	***3***
0	0	1	1	3
1	1	0	3	1
2	1	3	0	5
3	3	1	5	0



Una posible solución del problema es la siguiente:

En este caso, un trabajador se desplaza hacia 3 (tiempo = 3), luego hacia 1 (tiempo = 4) y regresa a 0 (tiempo = 5) (camino [3,1,0]). Mientras que el



otro se desplaza hacia 2 (tiempo = 1) y regresa a 0 (tiempo = 2) (camino [2,0]). Esta solución tiene costo 7.

Note que la solución en la que uno toma el camino [1,3,0] y el otro [2,0] tiene también costo 7, sin embargo, esta no es una solución válida pues se estaría visitando al león 1 en tiempo = 1, antes de su tiempo de sueño (de 3 a 7).

Implementación

Para solucionar el problema cuenta con la siguiente clase:

```
    class Map
{
    private int[,] Distances { get; set; }
    public int M {get; private set; }
    public int[] Start {get; private set;}
    public int[] End {get; private set;}

    public Map(int[,] distances, int[] start, int[] end)
    {
        Distances = distances;
        N = Distances.GetLength(0);
        Start = start;
```

```
13          End = end;
14      }
15
16      public int this[int i, int j]
17      {
18          get => Distances[i,j];
19      }
20
21      public bool IsOnTime(int node, int time)
22      {
23          return Start[node] <= time && End[node] >= time;
24      }
25  }
```

La propiedad `Distances` representa la matriz y `M` la cantidad dotal de lones del mapa (más el depósito). Los arrays `Start` y `End` almacenan horarios en que cada león (entre 1 y m-1) se duerme o se despierta respectivamente. 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. El método IsOnTime recibe un entero que representa a un león y un entero que representa un momento del tiempo. Retorna un valor booleano en dependencia de si el león está o no dormido durante ese momento

Usted debe implementar el siguiente método:

```
1  int Solve(Map map, int n)
2  {
3      throw new NotImplementedException();
4  }
```

El método *Solve* retorna el costo de la planeación de rutas óptima. El entero n reprsenta la cantidad de trabajadores del zoológico y por tanto la cantidad de rutas a planificar.

Consideraciones

- Todos los leones son representados por números enteros entre 1 y m - 1.
- `Start` y `End` siempre tendrán tamaño n y valor 0 en su primera posición (el depósito no duerme ni despierta).
- 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 el dpósito al 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.