ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2022-10. Proyecto – PARTE 1
Entrega: viernes, Abril 1 de 2022 6:00PM (por Bloque Neon)

0 OBJETIVOS

- Diseñar soluciones computacionales para problemas.
- Estimar costos de las soluciones planteadas.
- Implementar soluciones.

Se premiarán las mejores soluciones y se castigarán las peores, en cuanto a eficiencia en tiempo y espacio.

1 CONDICIONES GENERALES

El proyecto se divide en tres partes independientes entre sí. Este documento describe la PARTE I. Cada parte contiene un problema a resolver mediante soluciones implementadas en `Java` o `Python`.

Para cada problema se pide:
- Descripción de la solución.
- Análisis temporal y espacial.
- Una implementación en Java o Python

2 DESCRIPCIÓN DEL PROBLEMA

A Torre de teletransportación

Un estudiante de DALGO se propone descubrir la ruta optima que lleva al último cuarto de una torre laberinto constituida por una grilla de $n \times m$. Cada fila de la grilla representa uno de los pisos i de la torre ($1 \leq i \leq n$); y cada piso de la torre esta divido en m cuartos. En un determinado cuarto (i, j) de la torre puede existir un portal de teleportación que comunica con cuartos de pisos de superiores. Un portal le permite viajar de un cuarto (x_{start}, y_{start}) a otro cuarto (x_{end}, y_{end}) pero no en el sentido contrario. Se garantiza que $x_{start} < x_{end}$. El estudiante en un punto dado de la grilla puede moverse a la izquierda, derecha o tomar el portal si existe en dicho cuarto. Nótese que la única forma de moverse a pisos superiores es haciendo uso de los portales.

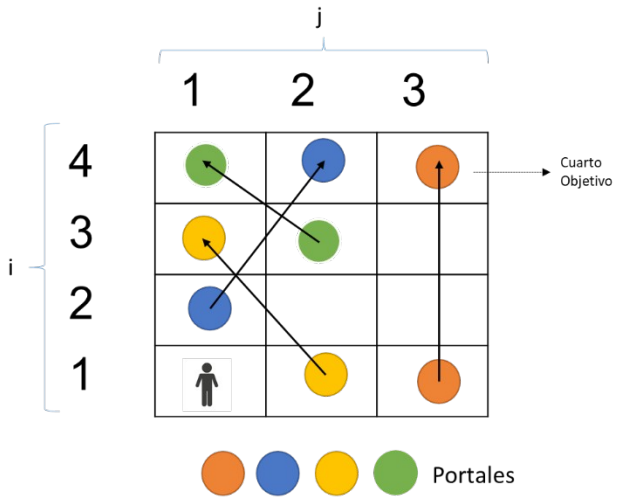

Fig 1. Torre de 4 x 3.

En cada piso el estudiante se tiene que enfrentar un conjunto de desafíos que le van mermando constantemente su energía. El estudiante solo puede moverse horizontalmente (i.e. izquierda o derecha) un cuarto a la vez. Cada movimiento en un determinado piso le consume e_i de su energía ($e_i \geq 0$), en consecuencia. Los movimientos usando portales no le consumen ninguna energía.

Problema

Dado que el estudiante inicia en la posición (1,1) el problema es encontrar la ruta que consume menos energía y le permite llegar al cuarto (n, m) (si existe).

Ejemplo:

Suponga la torre de la Fig.1 y adicionalmente los valores de perdida de energía de cada piso ($e_1 = 2, e_2 = 1, e_3 = 3, e_4 = 0$). Para esta torre existen dos rutas directas (i.e. sin repetir la visita a un cuarto).

Nota: en rojo movimientos de teleportación.

Ruta 1: (1,1)-(1,2)-(3,1)-(3,2)-(4-1)-(4,2)-(4,3)

Energía Ruta 1:

$e_1 + e_3 + e_4 + e_4 = 2 + 3 = 5$

Ruta 2: (1,1)-(1,2)-(1,3)-(4-3)

Energía Ruta 2:

$2e_1 = 4$

La ruta 2 es la de menor consumo de energía.

3 ENTRADA Y SALIDA DE DATOS

En todas las soluciones que se presenten, la lectura de los datos de entrada se hace por la entrada estándar; así mismo, la escritura de los resultados se hace por la salida estándar.

Puede suponer que ninguna línea de entrada tiene espacios al principio o al final, y que los datos que se listan en cada línea están separados por exactamente un espacio.

A continuación, se establecen parámetros que definen su tamaño y formato de lectura de los datos, tanto de entrada como de salida.

Descripción de la entrada

La primera línea de entrada especifica el número de cados de prueba que contiene el archivo. El programa debe terminar su ejecución, una vez termine de resolver la cantidad de casos de prueba dados por este número.

Cada caso contiene varias líneas. La primera línea de cada caso contiene 3 enteros $n\left(0 \le n \le 10^3\right), m\left(0 \le m \le 10^3\right), p\left(0 \le p < nm\right)$ n es el número de pisos, m es el número de cuartos y $p$es el número de portales de teletransportación que existen en la torre.

La segunda línea contiene una lista de n enteros que corresponden a los valores de consumo de energía e_i $(0 \le e_i \le 10^3)$. Las siguientes p líneas contienen la descripción de los portales. Cada portal es descrito por 4 enteros $x_{start}, y_{start}, x_{end}, y_{end}$.

$1 \le x_{start} < x_{end} \le n$

$1 \le y_{start} \le m$

$1 \le y_{end} \le m$

Portales: se puede mover hacia abajo,
pero NO se puede mover a la izquierda.

Descripción de la salida

Para cada caso de prueba, imprimir:

1. NO EXISTE, si no existe ninguna ruta.
2. El costo de la ruta óptima (i.e. menor energía gastada) para llegar al cuarto (n, m).

Ejemplo de entrada / salida

Para el ejemplo de la Fig 1.

Entrada	Salida
1 4 3 4 2 1 3 0 1 2 3 1 1 3 4 3 2 1 4 2 3 2 4 1	4

Cinco casos de prueba:

Entrada	Salida
5 4 3 4 2 1 3 0 1 2 3 1 1 3 4 3 2 1 4 2 3 2 4 1 5 3 3 5 17 8 1 4 1 3 3 3 3 1 5 2 3 2 5 1 6 3 3 5 17 8 1 4 2 1 3 3 3 3 1 5 2 3 2 5 1 5 3 1 5 17 8 1 4 1 3 5 3 5 5 5 3 2 3 7 5 3 5 4 2 2 2 5 4 4 4 5 2 1 2 4 2 3 3 5 2	4 26 NO EXISTE 10 32

Nota: Se van a diseñar casos de prueba para valores de n y m muchos más grandes y dentro de los valores establecidos en el enunciado. Los casos mostrados en este documento son demostrativos de la estructura de entrada/salida esperada.

4 COMPRENSIÓN DE PROBLEMAS ALGORITMICOS

A continuación, se presentan un conjunto de escenarios hipotéticos que cambian el problema original. Para cada escenario debe contestar[1]: (i) que nuevos retos presupone este nuevo escenario -si aplica-?, y (ii) que cambios -si aplica- le tendría que realizar a su solución para que se adapte a este nuevo escenario?

ESCENARIO 1: Suponga ahora que los portales son bidireccionales.

ESCENARIO 2: Se le pide ahora calcular el número de rutas que existen.

ESCENARIO 3: Se le pide ahora calcular la ruta optima en la cual el estudiante puede visitar todos los cuartos y finaliza en (n,m).

Nota: Los escenarios son independientes entre sí.

5 ENTREGABLES

El proyecto puede desarrollarse por grupos de hasta tres estudiantes de la misma sección. La entrega se hace por bloque neon (una sola entrega por grupo de trabajo).

El grupo debe entregar, por bloque neon, un archivo de nombre `proyectoDalgoP1.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP1`, comprimida en formato `.zip`, dentro de la cual hay archivos fuente de soluciones propuestas y archivos que documentan cada una de las soluciones.

5.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* o en Python

Para el problema:

- Entregar un archivo de código fuente en *Java* (`.java`) o python (`.py`) con su código fuente de la solución que se presenta.
- Incluir como encabezado de cada archivo fuente un comentario que identifique el (los) autor(es) de la solución.
- Denominar `ProblemaP1.java` o `ProblemaP1.py` el archivo de la solución que se presente.

[1] NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* o Spyder durante el desarrollo del proyecto, la entrega requiere incluir solo un archivo por cada solución. El archivo debe poderse compilar y ejecutar independientemente (sin depender de ninguna estructura de directorios, librerías no estándar, etc.).

5.2 Archivos que documentan la solución propuesta

La solución al problema debe acompañarse de un archivo de máximo 3 páginas que la documente, con extensión `.pdf`. El nombre del archivo debe ser el mismo del código correspondiente (`ProblemaP1.pdf`).

Un archivo de documentación debe contener los siguientes elementos:

0 *Identificación*
Nombre de autor(es)
Identificación de autor(es)

1 *Algoritmo de solución*
Explicación del algoritmo elegido. Si hubo alternativas de implantación diferentes, explicar por qué se escogió la que se implementó.
Deseable:
Anotación (contexto, pre-, poscondición, …) para cada subrutina o método que se use.

2 *Análisis de complejidades espacial y temporal*
Cálculo de complejidades y explicación de estas. Debe realizarse un análisis para cada solución entregada.

3 *Respuestas a los escenarios de comprensión de problemas algorítmicos.*
Respuesta a las preguntas establecidas en cada escenario. NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

Téngase en cuenta que los análisis de 2 tienen sentido en la medida que la explicación de 1 sea clara y correcta. No se está exigiendo formalismo a ultranza, pero sí que, como aplicación de lo estudiado en el curso, se pueda describir un algoritmo de manera correcta y comprensible.

No describa un algoritmo con código GCL a menos que lo considere necesario para explicarlo con claridad. Y, si lo hace, asegúrese de incluir aserciones explicativas, fáciles de leer y de comprender.