Informática

Facultad de Ciencias Matemáticas
Curso 2020-2021
Examen final

Pregunta 1

Una agencia de viajes desea recomendar la visita de monumentos a los turistas. Los monumentos tienen un nombre y una ubicación (la estación de metro más cercana). También se conoce el tiempo en segundos que un turista tarda en visitar dicho monumento.

1. **[2 puntos]** Crear una clase llamada `Monumento` para representar monumentos de la ciudad de Madrid. Por ejemplo, el monumento "Reloj" ubicado en la estación "Sol" con duración de visita 15 minutos, se representa textualmente como:

   ```
   (Reloj - 15 - Sol)
   ```

   Crear una clase `Itinerario` que permita representar una colección de monumentos diferentes a visitar en un cierto orden. Los objetos de la clase `Itinerario` se crean a partir de una lista de monumentos. La representación textual de los objetos de la clase `Itinerario` tiene el siguiente aspecto:

   ```
   [Museo del Prado, Reloj, Torre Madrid]
   ```

   La clase `Itinerario` ha de implementar la función `calcular_tiempo`. Dicha función calcula el tiempo mínimo empleado en la realización del itinerario (en el orden en el que aparecen los monumentos). El tiempo mínimo de desplazamiento entre monumentos será el tiempo mínimo de desplazamiento entre las estaciones asociadas a dichos monumentos (realizando como mucho 1 transbordo en cada desplazamiento). Por tanto, la función `calcular_tiempo` necesitará como parámetro de entrada un objeto de tipo `Metro`. Por supuesto, se ha de tener en cuenta el tiempo que el turista invierte en la visita del monumento. Si el itinerario no se puede realizar, la función devuelve `None`.

b) **[1 punto]** Implementar una función estática llamada `calcular_visita_corta`, que dada una lista de itinerarios (objetos de tipo `Itinerario`), un objeto de tipo `Metro` y la estación más cercana al hotel del turista, devuelva el itinerario que el turista pueda realizar en el menor tiempo posible. Por supuesto, será necesario tener en cuenta que el turista invierte tiempo en desplazarse desde la estación de metro de su hotel hasta el primer monumento. Si no es posible realizar ninguno de los itinerarios de la lista, la función devuelve `None`.

c) **[2.5 punto]** Implementar una función `generar_itinerarios` que reciba una lista de monumentos y devuelva la lista de todos los posibles itinerarios formados por 3 monumentos cualesquiera de la lista.

Pregunta 2

Una matriz de números enteros **kasi konstante** es una matriz con una enorme cantidad de elementos cuyo valor es un cierto **k**. Cada matriz kasi konstante A se puede representar de forma reducida mediante una tupla R_A de cuatro elementos.

$$R_A = (m, k, f, c)$$

donde m es una matriz de tres columnas (la llamaremos compacta), k es el valor por defecto, y f, c son el número de filas y columnas respectivamente de la matriz kasi konstante A. Para cada elemento $a_{ij} \in A$ con $a_{ij} \neq k$, la matriz compacta m tendrá una fila de la forma $[a_{ij}, i, j]$. Además, **las filas de la matriz compacta m están ordenadas lexicográficamente en relación a sus dos últimos elementos**. Por ejemplo, la matriz kasi constante siguiente:

$$A = \begin{bmatrix} 1 & 1 & 3 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 6 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

se representa de forma reducida mediante la tupla:

$$R_A = \left(\begin{bmatrix} 3 & 0 & 2 \\ 6 & 2 & 1 \\ 0 & 2 & 2 \\ 0 & 4 & 4 \end{bmatrix}, \quad 1, \quad 5, \quad 5 \right)$$

donde el valor por defecto es 1, el número de filas es 5 y el número de columnas es 5.

Las operaciones sobre matrices kasi konstantes pueden realizarse empleando sus representaciones reducidas. Tened cuidado con **todas** las condiciones de la representación de las matrices kasi konstantes.

1. **[2 puntos]** Diseña y escribe la función `sum_reduced`. Dicha función recibe las representaciones reducidas R_{A_1}, R_{A_2} con

   $$R_{A_i} = (m_i, k_i, f_i, c_i)$$

   de dos matrices kasi konstantes A_1, A_2, y devuelve la representación reducida de la suma de A_1 y A_2 de la forma $(m, k_1 + k_2, f_1, c_1)$. Si A_1 y A_2 no se pueden sumar, la función devolverá `None`.

   ```
   >>> m1 = ([[4, 0, 0],[2, 0, 2], [0, 1 ,0],[4, 1, 2], [7, 2, 2]], 1, 7, 7)
   >>> m2 = ([[-1, 0, 0],[1, 0, 1], [3, 0, 2], [5, 1, 2] ], 2, 7, 7)
   >>> print(sum_reduced(m1, m2))

   ([[2, 0, 1], [5, 0, 2], [2, 1, 0], [9, 1, 2], [9, 2, 2]], 3, 7, 7)
   ```

2. **[2.5 puntos]** Diseña y escribe la función `traspose`. Dicha función recibe la representación reducida R_A de una matriz kasi konstante A, y devuelve la representación reducida de su traspuesta A^t.

Se valorará la eficiencia de las soluciones planteadas, la claridad del código, la ausencia de código repetido, etc.

Duración del examen: 2.5 horas.

Anexo. Resumen práctica Metro

La clase `Line` representa líneas de metro. Tiene los siguientes métodos:

- `contains_station(e)`: devuelve el valor `True` si la línea contiene una estación de nombre `e` y `False` en caso contrario.
- `cost_origin2destination(start, finish)`: calcula la cantidad de segundos que se tarda en recorrer el trayecto en la línea desde la estación `start` hasta la estación `finish`. Si alguna de las estaciones `start` o `finish`no pertenece a la línea, entonces el método genera una excepción de la clase `LineException` con el mensaje adecuado.

La clase `Metro` representa una red de metro. Entre sus métodos se encuentran:

- `add_line(line_name, line)`: añadir a la red de metro la línea `line` con el nombre `line_name`.
- `add_connections(line_name)`: se encarga de actualizar la información de los transbordos a partir de las estaciones de la línea indicada.
- `load_metro(file_name)`: es un método estático que recibe el nombre `file_name` de un fichero con los datos de la red de metro y devuelve un objeto de la clase `Metro`.
- `get_connections(e, line_name)`: devuelve una lista con los nombres de las líneas de la red con las que podemos conectar haciendo transbordo desde la estación `e` de la línea cuyo nombre es `line_name`. Si la línea `line_name` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Análogamente, si la estación `e` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado.
- `cost_origin2destination_transfer(start, finish)`: calcula la cantidad mínima de segundos que se tarda en realizar un trayecto que comienza en la estación `start` y termina en la estación `finish` realizando como mucho un transbordo. Si alguna de las estaciones `start` o `finish` no pertenece a la red de metro, entonces el método genera una excepción de la clase `MetroException` con el mensaje adecuado. Si es imposible desplazarse desde una estación a otra haciendo como máximo un transbordo, el método `cost_origin2destination_transfer()` devuelve `None`.