Algoritmos y Estructuras de Datos
Tercer Parcial

Integrantes:

- Luna, Lihué Leandro
- Merino, Mateo
- Bonino, Francisco Ignacio

1) Para resolver el primer enunciado, tomamos el código del algoritmo de Dijkstra proporcionado en clase, y no hubo mayores complicaciones con la matriz de adyacencia.

2) Para encontrar el camino Hamiltoniano más barato, investigando acerca de teoría de grafos en internet, nos encontramos con un método que aplica la Multiplicación Latina de matrices, siendo esta de la siguiente forma:

Multiplicación Latina

Ejemplo:

Asociamos a la red una matriz latina:

- Si existe el vértice (i, j) en la coordenada correspondiente de la matriz colocamos los 2 vértices.
- En caso contrario pondremos un 0.

$$L_1 = \begin{pmatrix} 0 & AB & 0 & 0 & 0 & AF \\ 0 & 0 & 0 & BD & BE & 0 \\ 0 & CB & 0 & 0 & 0 & CF \\ DA & 0 & DC & 0 & 0 & 0 \\ EA & 0 & EC & ED & 0 & EF \\ 0 & 0 & 0 & FD & FE & 0 \end{pmatrix}$$

Obtendremos L_1^* eliminando de L_1 los vértices origen:

$$L_1^* = \begin{pmatrix} 0 & B & 0 & 0 & 0 & F \\ 0 & 0 & 0 & D & E & 0 \\ 0 & B & 0 & 0 & 0 & F \\ A & 0 & C & 0 & 0 & 0 \\ A & 0 & C & D & 0 & F \\ 0 & 0 & 0 & D & E & 0 \end{pmatrix}$$

El proceso de obtención de los caminos hamiltonianos de longitud i consiste en obtener $L_i = L_{i-1} \times L_1^*$ multiplicando como en el cálculo matricial y en un cierto sentido:

- Pondremos un 0 en la posición (i, j) si $\forall k = 1, \ldots, n$ en los productos del elemento (i, k) por (k, j) al menos uno de ellos es 0.
- Si las dos celdas contienen secuencias, asociaremos 0 si la última letra de la celda (i, k) coincide con la primera de la celda (k, j) o si las dos secuencias tienen alguna letra repetida.
- Si al realizar estos productos, asociamos a cada secuencia contenida en la celda (i, k) una o más de la celda (k, j) sin que se repitan las letras, entonces llevaremos las secuencias encadenadamente a la celda (i, j).

Licesio J. Rodríguez-Aragón — Métodos Cuantitativos Org. Ind. – 44 / 47

Primeramente, se construyeron dos matrices que cumplen con los requisitos enunciados (L1 y L1*) por el algoritmo de Multiplicación Latina.
Paralelamente, se construyó una matriz de objetos tipo lista que contienen objetos tipo string, en la que cada celda de la matriz es una lista que guardará las secuencias admitidas para el producto realizado para esa posición.
En una función realizamos el producto de la matriz L1 por L1*, almacenando el resultado en la matriz de listas, y luego volcándolo a la matriz L1, para llamar a la función multiplicadora nuevamente con L1* y L1 actualizada.
La función multiplicar opera como si fuese una multiplicación normal de matrices con la diferencia de que en vez de multiplicar los elementos, concatena los strings.
Se debe multiplicar el elemento de la matriz L1* por todos los elementos de la lista de la posición correspondiente en la matriz L1, asegurando así que cada secuencia posible está siendo tenida en cuenta.
Si en algún punto una secuencia que era admitida pasa a tener un caracter repetido, la misma se borra de la lista de la matriz, y se trabaja con el resto.
Este proceso ha de realizarse n-2 veces, siendo ‘n’ la cantidad de vértices del grafo. En este caso, al tener un grafo de 12 vértices, debemos realizar esta multiplicación 10 veces.
Finalmente, en una posición específica de la matriz resultante se encuentra una lista que contiene todos los caminos Hamiltonianos posibles del grafo. Procedimos a calcular sus precios y determinar el camino más barato, mostrándolo en pantalla indicando su costo y la secuencia de ciudades recorridas.
La función que obtiene el precio de un camino, toma el string de la secuencia y la analiza caracter a caracter, acumulando el precio del viaje entre ciudad y ciudad utilizando como referencia la matriz de adyacencia.
Se calcula el precio del primer camino encontrado, y se lo almacena en una variable entera. Ésta se usa como referencia para el cálculo de las siguientes secuencias.
Si dado un precio mínimo, al calcular el costo de otro camino, éste ya se supera, se interrumpe el cálculo del viaje y se sigue con la siguiente secuencia.
Si se halla un camino más barato, el precio del mismo se almacena en la variable de referencia.

3) Para el tercer enunciado se reutiliza el recurso de la multiplicación latina pero con el fin de hallar caminos de 10 ciudades sin repetir. Con la matriz de listas obtenida, se llama a la función que devuelve el precio para que muestre en pantalla el camino más barato por celda. Hecho esto, se toma el camino más económico y se lo muestra en pantalla.