FUNDAMENTOS DE PROGRAMACIÓN – ST0242
Práctica Final
2024-1

Objetivo General
Poner en práctica los conceptos vistos en el curso, principalmente matrices de referencias a objetos.

Objetivos Específicos
- Poner en práctica los conceptos de matrices.
- Poner en práctica los conceptos de lectura desde un archivo de texto.
- Poner en práctica los conceptos de documentación utilizando JavaDoc.

Qué hay que hacer

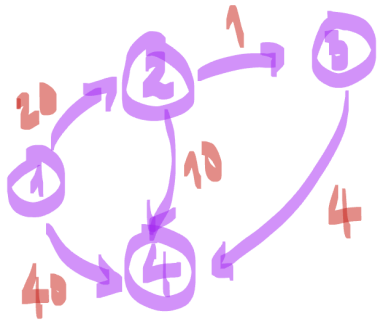
Cada grupo (máximo 2 personas) va a implementar un programa que va a encontrar la distancia más corta entre pares de nodos de un grafo que se lee de un archivo tipo texto.

Además de calcular la distancia más corta, el programa debe imprimir los nodos que se visitan en esa ruta más corta.

La explicación del algoritmo es tomada del texto “Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. Fourth. Edition. MIT press.”

En la siguiente explicación, se asume que $w_{k,j}$ es la entrada k, j de la matriz de adyacencia W, que contiene los pesos o las distancias *directas* entra cada par de nodos. El valor $w_{k,j}$, sería entonces la distancia para ir directamente del nodo k hasta el nodo j.

Por ejemplo, dado el siguiente grafo:



La matriz de adyacencia sería la siguiente:

0	20	∞	40

∞	0	1	10
∞	∞	0	4
∞	∞	∞	0

En el caso de nuestro proyecto, asumiremos que el grafo *no* es dirigido. En un grafo dirigido, el peso para ir del nodo *i* hasta el nodo *j* es el mismo que para ir del nodo *j* hasta el nodo *i*. El grafo se dibuja *sin* flechas y la matriz es simétrica respecto a la diagonal principal.

Explicación del algoritmo, de acuerdo con el texto de Cormen:

A recursive solution to the all-pairs shortest-paths problem

Now, let $l_{ij}^{(r)}$ be the minimum weight of any path from vertex i to vertex j that contains at most r edges. When $r = 0$, there is a shortest path from i to j with no edges if and only if $i = j$, yielding

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \ , \\ \infty & \text{if } i \neq j \ . \end{cases} \tag{23.2}$$

For $r \geq 1$, one way to achieve a minimum-weight path from i to j with at most r edges is by taking a path containing at most $r-1$ edges, so that $l_{ij}^{(r)} = l_{ij}^{(r-1)}$. Another way is by taking a path of at most $r-1$ edges from i to some vertex k and then taking the edge (k, j), so that $l_{ij}^{(r)} = l_{ik}^{(r-1)} + w(k, j)$. Therefore, to examine paths from i to j consisting of at most r edges, try all possible predecessors k of j, giving the recursive definition

$$\begin{aligned} l_{ij}^{(r)} &= \min \left\{ l_{ij}^{(r-1)}, \ \min \left\{ l_{ik}^{(r-1)} + w_{kj} : 1 \leq k \leq n \right\} \right\} \\ &= \min \left\{ l_{ik}^{(r-1)} + w_{kj} : 1 \leq k \leq n \right\} \ . \end{aligned} \tag{23.3}$$

The last equality follows from the observation that $w_{jj} = 0$ for all j.

Computing the shortest-path weights bottom up

Taking as input the matrix $W = (w_{ij})$, let's see how to compute a series of matrices $L^{(0)}, L^{(1)}, \ldots, L^{(n-1)}$, where $L^{(r)} = \left(l_{ij}^{(r)}\right)$ for $r = 0, 1, \ldots, n-1$. The initial matrix is $L^{(0)}$ given by equation (23.2). The final matrix $L^{(n-1)}$ contains the actual shortest-path weights.

```
EXTEND-SHORTEST-PATHS(L^(r-1), W, L^(r), n)
1  // Assume that the elements of L^(r) are initialized to ∞.
2  for i = 1 to n
3      for j = 1 to n
4          for k = 1 to n
5              l_ij^(r) = min{l_ij^(r), l_ik^(r-1) + w_kj}
```

Adicionalmente, para encontrar los nodos que se visitan en la ruta más corta, se puede utilizar el siguiente método:

- Ya sabemos la longitud más corta.
- ¿Cuál es el camino más corto?
  - Inicializar la matriz PI así:

```
for(int f = 1; f <= numV; f++) {
    for(int c = 1; c <= numV; c++) {
        if(f == c) {
            pi[f][f] = f;
        } else {
            pi[f][c] = -1;
        }
    }
}
```

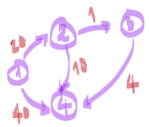
  - Y si en la matriz L^n se obtiene una distancia más corta para ir del nodo *f* al nodo *c*, pasando por el nodo *k*, se actualiza pi[f][c] = k. Asi:

```
if(distK < l[i][f][c]) {
    l[i][f][c] = distK;
    pi[f][c] = k;
}
```

- Ejemplo:

```
Matriz de predecesores
1 1 2 3
-1 2 2 3
-1 -1 3 3
-1 -1 -1 4
```

- En el mismo grafo:



- Para ir del nodo 1 al 4, se van obteniendo los predecesores:
  - Del 1 al 4, el último predecesor es el 3
  - Del 1 al 3, el último predecesor es el 2
  - Del 1 al 2, el último predecesor es el 1
  - Y así ya se llegó al nodo origen (1)

Cómo entregar el proyecto:

1. Se debe subir a Interactiva lo siguiente:
   a. El código fuente, en un proyecto de BlueJ. **El código debe estar comentado usando JavaDoc. Se debe documentar cada clase y cada método**. En el siguiente link se puede consultar cómo documentar el código utilizando JavaDoc:

https://www.youtube.com/watch?v=ez55fdwUjlg

   b. Un video corto (menos de 3 minutos) demostrando el funcionamiento del programa.

2. Algunos grupos van a sustentar la práctica con el profesor la fecha definida. Todos los participantes del grupo que sustenta deben dar cuenta del software que se entrega.

Fechas:

- Entrega en el buzón de Interactiva: viernes 8 de noviembre antes de las 23:30.
- Sustentación: se realizará en época de finales.

NOTA:

Si los dos integrantes del grupo tienen la nota del curso por encima de 4, no tienen que sustentar. Simplemente envían el código y el video correspondiente por Interactiva.

Código de Honor:

- **NO se debe mirar, y menos copiar, el código de otro grupo. Si se descubre copia, tanto el grupo que copia como el que se deja copiar tendrán 0 en la nota.**