Laboratorio Nro. 5: Programación Dinámica

Isabela Muriel Roldán
Universidad Eafit
Medellín, Colombia
imurielr@eafit.edu.co

Mateo Flórez Restrepo
Universidad Eafit
Medellín, Colombia
mflorezr@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. En el ejercicio 1.2 se utiliza el algoritmo heldKarp, el cual busca recorrer todos los vertices de un grafo buscando el camino que vaya del nodo inicial al nodo inicial en la menor distancia posible.
2. Para resolver el problema del agente viajero existen varias soluciones, algunas eficientes y otras no tanto, algunos ejemplos de estas son: por fuerza bruta, aunque esta solución puede llegar a ser muy lenta para problemas muy grandes y puede llegar a tardar años en terminar, tambien puede ser resuelto por medio de la programación dinamica, con el algoritmo Held-Karp.
3. Para resolver la problemática planteada en el numeral 2.1 utilizamos el algoritmo Held-Karp y lo adaptamos al problema, ya que la situación que se explicaba se asemejaba mucho a la del agente viajero, y este algoritmo es una de las soluciones desarrolladas para este problema. El algoritmo en principio lo que hace es solicitar por consola los datos necesarios para el desarrollo problema, como las coordenadas de cada objeto, las cantidades, etc. Después crea un grafo de listas enlazadas según los datos recibidos y lo envía al algoritmo de Held-Karp donde será recorrido, sacando subconjuntos de nodos donde exista la distancia mínima y luego comparando las distancias entre estos subconjuntos y concatenándolos hasta que llega a la ruta óptima devolviendo la distancia mínima de recorrido total entre todas las evaluadas. Y este algoritmo se repite para cada uno de los escenarios que se hayan solicitado.
4. Complejidad.

```
public static void ejercicio2(){
    Scanner sc = new Scanner(System.in);                          //C1
    int n = sc.nextInt();                                         //C2
    for (int i = 0; i < n; i++) {                                 //C3*n
      int mundox= sc.nextInt();                                   //C4*n
      int mundoy= sc.nextInt();                                   //C5*n
      ArrayList<Pair> coord = new ArrayList<Pair>();              //C6*n
      coord.add(Pair.makePair(sc.nextInt(),sc.nextInt()));        //C7*n
      int dr = sc.nextInt();                                      //C8*n
      for (int j = 0; j < dr; j++){                               //C9*n*m
        int drx=sc.nextInt();                                     //C10*n*m
        int dry= sc.nextInt();                                    //C11*n*m
        if(drx <= mundox && dry <= mundoy)                        //C12*n*m
          coord.add(Pair.makePair(drx,dry));                      //C13*n*m
          else
```

```
            System.out.println("Fuera de rango");                    //C14*n*m

        }

        Digraph g = new DigraphAL(coord.size());                    //C15*n
        for (int k = 0; k < coord.size(); k++) {                    //C16*n*k
           for (int l = 0; l < coord.size(); l++) {                 //C17*n*k^2
              if(k != l) {
                 g.addArc(k,l,(Math.abs((int)coord.get(k).first - (int)coord.get(l).first)) +
Math.abs((int)coord.get(k).second - (int)coord.get(l).second));    //C18*n*k^2
              }
           }
        }
        int result= heldKarp(g);                                    //O(2^k)*n
       System.out.println("The shortest path has length: " + result);  //C19*n
     }
  }
```

Se encontraron varias complejidades en el algoritmo:

Para leer los datos:
T(n)= C*n*m
T(n)= O(C*n*m)
T(n)= O(n*m)

Para crear el grafo:
T(n)= C*n*k^2
T(n)= O(C*n*k^2)
T(n)= O(n*k^2)

Para hallar el costo mínimo:
T(n)= O(2^k)*n
T(n)= O(2^k)

5. **Explicación de las variables:** Para calcular la complejidad del algoritmo de la solución al punto 2.1 utilizamos varias variables:
n= Cantidad de veces que se repite el algoritmo por cada escenario
m= Número de coordenadas de desechos radioactivos que son solicitadas según la cantidad ingresada
k= Numero de nodos del grafo según la cantidad de objetos que están en el plano (en esta situación casi siempre es igual a m+1)

4) Simulacro de Parcial

1.

	-	C	A	L	L	E
-	0	1	2	3	4	5
C	1	0	1	2	3	4
A	2	1	0	1	2	3
S	3	2	1	1	2	3
A	4	3	2	2	2	3

	-	M	A	D	R	E
-	0	1	2	3	4	5
M	1	0	1	2	3	4
A	2	1	0	1	2	3
M	3	2	1	1	2	3
A	4	3	2	2	2	3

2. 2.1 T(n) = O(lx*ly)
2.2 Return table[lenx][leny]

3. 3.1. a
3.2. c

4. c

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co