ESTRUCTURA DE DATOS 2
Código ST0247

Laboratorio Nro. 4
Algoritmos Voraces

Miguel Ángel Martínez Florez
Universidad Eafit
Medellín, Colombia
mamartinef@eafit.edu.co

Pablo Micolta López
Universidad Eafit
Medellín, Colombia
pmicoltal@eafit.edu.co

Daniel Ricardo Palacios Diego
Universidad Eafit
Medellín, Colombia
drpalaciod@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 nosotros utilizamos una serie de diccionarios para el código, el primero contiene la idea asociada a la ruta**;** segundo contiene la id asociada al número de creación del nodo, y el tercero es el inverso del segundo. En el algoritmo en general, se utilizo un arreglo para representar las ciudades que componían a la ruta; además se recorrió el grafo a partir del primer nodo, en este asumimos que este es el nodo te partida, y otro arreglo para decir cual, entre todos los nodos, aunque no estén en la ruta original hemos visitado. Luego realizamos tres cosas sobre los sucesores del nodo que estamos revisando:

1) Que no sea el mismo nodo que estamos revisando.
2) Que no lo hayamos visitado aún.
3) Que el peso del nodo que estamos revisando sea el menor entre todos los sucesores. Después de esto, actualizamos el valor de la mínima distancia y añadimos el nuevo sucesor a la ruta, para que después sumemos esa distancia mínima a la distancia total.

Luego, sumamos esa distancia mínima, añadiendo el nodo que estamos revisando a la lista de las que ya visitamos, luego pasamos al siguiente nodo para que finalmente cuando lleguemos al ultimo nodo añadamos la distancia a la distancia total desde ultimo nodo hasta el nodo inicial.

3.2 No siempre va a dar la solución mas eficiente, ya que, al solo revisar la distancia menor en cada nodo individual en lugar de a gran escala como lo haríamos en backtraking, entonces la respuesta se va a saltar algunos casos donde de un nodo hay dos nodos que desde la instancia del padre uno tiene una ruta mejor, pero a gran escala la otra tenía una ruta que pasaba por otro nodo que era mas eficiente y finalizaba en el mismo nodo.

Las condiciones para que el algoritmo de una solución tiene que existir al menos una o más rutas hacia el origen para poder hacer la conexión, y esa conexión tiene que estar al final de una ruta eficiente porque si no, no hay forma de detener al algoritmo antes de que entre a esos “callejones sin salida”.

3.3 La solución se podría dejar tal cual esta, ya que, la única diferencia seria encontrar las id de las calles o un remplazo para ellas. El objetivo de este algoritmo seria pasar por todos los putos donde haya un domicilio, no únicamente porque sería imposible en algunos casos, y no se pasa por todos porque sería ineficiente a nivel de negocio, ya que el repartidor estaría haciendo recorridos innecesarios intentando llegar a cada calle de Colombia. Además, la distancia entre cada nodo se mediría en metros, ya que, si se mide esta distancia con algo que no sea una medida espacial como vendría a ser el viento, por ejemplo, el problema que hace que los algoritmos voraces puedan ser inexactos, es decir, son incapaces de medir cosas a un nivel global, solo su siguiente paso se vería amplificado.

3.4 para la estructura de datos se eligió usar un arreglo, por su fácil acceso y ordenamiento de este y como ya tenemos un tamaño predilecto del mismo, no nos tendríamos que preocupar por añadir nuevos espacios. El método recibe dos arreglos, uno por las rutas de la mañana y el otro por las rutas de la tarde; luego la cantidad de conductores se dividió en dos, una mitad para cada horario. Después, se hace un ciclo en el cual se recorre el arreglo ordenado de la duración de rutas de la mañana las cuales se suman en una variable propia para las rutas de la mañana, y se repite este proceso con las rutas de la tarde. Además, en caso de que el número de conductores fuese impar, se le añade un conductor extra a uno de los dos horarios en base a cuál tubo menor cantidad de horas extra. Finalmente, se calculan la cantidad el extra pagar y se suman ambos resultados.

3.5 la complejidad del algoritmo es $O(n\log_2(n))$, a causa de la función sort, que fue usada para organizar los arreglos.

3.6 n es la cantidad de rutas que hay.

4) ***Simulacro de Parcial***

4.1 *i + 1; }*
4.2 (min > adjacencyMatrix[element][i])
4.3 Opcional
4.4 .
Línea 10 = Temp = (temp % 2) + min;
Línea 11= return temp ;

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.6

Linea 6 X[i]==i+1

Linea 8 res = res+1

Linea 9 last = i

La salida es 4

5) Lectura recomendada (opcional)

Mapa conceptual

6) Trabajo en Equipo y Progreso Gradual (Opcional)

6.1 *Actas de reunión*
6.2 *El reporte de cambios en el código*
6.3 *El reporte de cambios del informe de laboratorio*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473