Título:

INFORME TRABAJO PRÁCTICO ESPECIAL 2

Nivel y Área:

72.34 - Estructuras de Datos y Algoritmos

2º Cuatrimestre 2017

Comisión: S - **Carrera:** Informática - **Grupo:** Nº1

Fecha: 10 de noviembre de 2017

Alumnos Expositores:

- AQUILI, Alejo Ezequiel
- BASSANI, Santiago
- LO COCO, Juan Pablo
- PINILLA, Lautaro Joaquín
- SANGUINETI ARENA, Francisco Javier

Trabajo Práctico Especial 2

Algoritmos y estructuras principales utilizadas:

El objetivo del trabajo práctico aquí expuesto fue resolver la problemática de escoger la mejor combinación de vuelos para conectar dos aeropuertos.

El proyecto se puede dividir a grandes rasgos en las siguientes partes.
Por un lado, tenemos la estructura genérica de grafo que subyace a todo el proyecto la cual contiene todos los métodos genéricos que un grafo dirigido puede tener.

Por otro lado tenemos la implementación propia del ya mencionado grafo genérico que usamos en el proyecto. En el mismo tomamos como que los nodos son aeropuertos que contienen una locación en el mapa (latitud y longitud) y un nombre de 3 siglas. Las aristas , en cambio, son los vuelos que contienen la información propia del vuelo más los atributos genéricos de una arista (peso , nodo origen y nodo destino), esto fue hecho así ya que permite reutilizar la estructura de grafo para otros usos.

Finalmente, utilizamos una clase administradora del grafo que genera el programa, un parser con el cual trabaja la shell y una clase Day que es utilizada para el manejo del tiempo. No utilizamos clases Time provistas por otras librerías debido a que no era fácil adaptarlas para nuestra implementación.

Decisiones importantes:

Algunas decisiones importantes que tuvimos que tomar a la hora de llevar a cabo la realización del proyecto fueron:

Estructura para el guardado de nodos adyacentes.Para este problemas optamos por un hashmap que use como key a los nodos y tenga como values listas ordenadas de aristas que conectan ese nodo con el actual. esto nos permite acceder a cada nodo en O(1) y con el uso de la lista ordenada obtenemos la mejor arista para ese camino en O(1) .

Luego tuvimos que sobrepasar las complicaciones que nos trae la búsqueda del camino con menor tiempo total. En este caso priorizamos la robustez del sistema para soportar una gran cantidad de aeropuertos y vuelos , haciéndolo iterativo , modificando el algoritmo de dijkstra para aceptar multi arista.

Para realizar el algoritmo de la vuelta al mundo se optó por un back-Tracking recursivo, en lugar de uno iterativo, dado que era mucho más fácil la implementación. Sin embargo hay que avisar que este algoritmo en un grafo con muchos nodos genera Overflow dado a la enorme cantidad de llamadas recursivas que se generan, siendo el peor caso uno donde se tomen todos los vuelos. En un algoritmo iterativo no daria Overflow. Para realizar una optimización del mismo se decidió implementar una clase Solution donde guarda el mejor camino con su costo, y permite comparar con el camino actual, realizando una poda que baja la complejidad temporal. Otra decisión importante en este algoritmo es que retorna que hay una vuelta al mundo siempre que no se repitan vuelos. Por ende, aunque haya una vuelta al mundo, podría retornar que no existe o un resultado que no sea el mejor.

Dificultades:

Para representar el problema decidimos implementar la estructura de grafos con listas de adyacencias vista en clase. Pero las mismas resultaron no ser muy útiles debido a que nuestro grafo se presentaba como multiartista (dado que varios vuelos puede conectar los mismos dos aeropuertos). En base a este problema decidimos usar una nueva implementación que utiliza un hashMap interno para referenciar a todo nodo vecino (key) y cómo value usamos dos Sorted list (estructuras creadas por nosotros para ordenar las aristas según un comparador) en las cuales los vuelos se organizaban según su duración de vuelo o su precio. de este modo pudimos convertir el grafo en simplemente conexo ya que solo utilizamos la arista de menor precio o de menor tiempo de vuelo y así pudimos implementar el algoritmo de Dijkstra visto en clase modificado a nuestro problema.

A al hora de pensar el algoritmo de tiempo total resulta evidente que el mismo no puede reducirse en una sola aristas dado que no siempre el vuelo con menos horas era el mejor ya que puede salir muy tarde a comparación de otros. Debido a esto tuvimos que tomar la decisión de modificar el algoritmo de búsqueda para que lleve un recuento del tiempo (medido en min respecto de las 00:00hs del lunes) en que se encuentra al momento de llegar a un aeropuerto y de esa forma iteraria por todas las aristas que conectan dos aeropuertos para elegir la que menos horas agrega a este tiempo desde la salida del origen.

Conclusiones:

Concluimos que aunque la complejidad de los algoritmos deterministas para este tipo de problemas donde subyace una estructura de grafo , son de alta complejidad es posible hacer ciertas modificaciones estructurales para bajar su orden y evitar recaer en el uso de algoritmos no deterministas para encontrar subóptimos del problema.