PRÁCTICA 2 – Esquema Algorítmico Greedy o Voraz: Pavimentación de caminos

Tabla de contenidos

Estructuras de datos y Algoritmos II. Universidad de Almería

Version **curso.2022**

Objetivos

- Construir soluciones a un problema utilizando el método algorítmico greedy (o voraz).
- Comparar los algoritmos implementados tanto cualitativa como cuantitativamente.
- Realizar el análisis de la eficiencia de las soluciones aportadas, y una comparativa tanto desde el punto de vista teórico como práctico.

Requerimientos

Para superar esta práctica se debe realizar lo siguiente:

- Recordar (de EDA I, el mapa de adyacencias) las estructuras de datos para implementar una red (grafo no dirigido) y su uso para la resolución de problemas.
- Conocer cómo implementar el esquema greedy sobre grafos para responder a una necesidad concreta (en este caso, resolver un problema relativo a una red de caminos).
- Evaluar los algoritmos greedy implementados, utilizando redes reales y redes

generadas aleatoriamente.

Enunciado del problema

Una de las principales aplicaciones prácticas de los grafos son las **redes**, por ejemplo: la red de carreteras, la red de ferrocarriles, las redes de comunicaciones, etc. En este caso tenemos una red de caminos que conectan todos los núcleos de población del país **EDAland**, y la necesidad que tiene el Ministerio de Infraestructuras y transportes de ese país es la de pavimentarlos. Pero dada la actual crisis económica en que estamos inmersos, no se dispone de presupuesto suficiente para poder pavimentar todos los posibles caminos del país. Tras una dura negociación con el ministerio y teniendo en cuenta el reducido presupuesto disponible, se ha decidido que se pavimentarán únicamente aquellos caminos para poder ir de un núcleo de población a otro siguiendo una ruta pavimentada, y además, se dedique el menor presupuesto posible en la realización de la obra. Otro aspecto a considerar es que a los habitantes del país no les importa recorrer mucha distancia, pues ahora se mueven en bicicleta (dado el elevado precio del combustible) que es muy beneficioso para la salud y el medioambiente. **El Ministro está muy interesado en saber qué caminos hay que pavimentar para conectar a todos los posibles núcleos de población del país con el menor presupuesto posible.**

En un primer lugar se dispone de la **red reducida de caminos** de EDAland (Figura 1) que conecta a los grandes núcleos de población (ciudades) del país (21 ciudades y 29 caminos), junto con la estimación en miles de euros de lo que costaría pavimentar el trayecto entre dos ciudades (misma red de EDA I). Por ejemplo, pavimentar el camino que conecta Almería con Granada constaría 173.000 euros. Esta red nos servirá para probar nuestros algoritmos greedy y verificar su correcto funcionamiento.

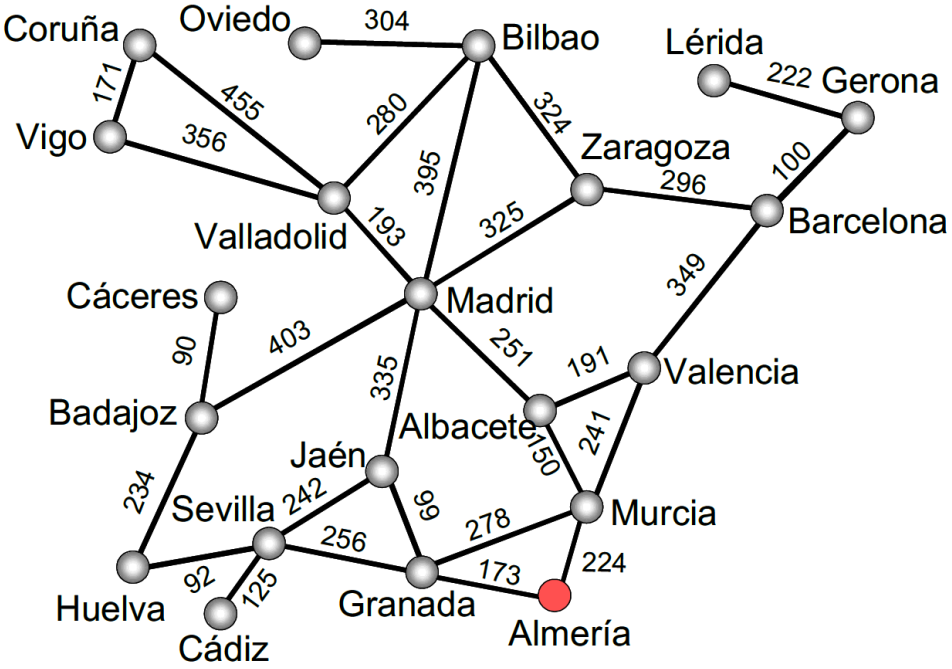

Figura 1. Red reducida de caminos de EDAland

Una vez comprobado el correcto funcionamiento de nuestros algoritmos greedy en la red reducida de caminos, los ejecutaremos en la **red nacional de caminos** de EDALand que dispone el Ministerio (1053 núcleos de población y 2017 caminos con el coste estimado de pavimentarlos en miles de euros) (Figura 2), y de esta manera ya se tendría el presupuesto total que costaría la obra global, que como ya sabemos debe ser el menor posible (requerimiento principal del Ministerio).

Figura 2. Red nacional de caminos de EDAland

Trabajo a desarrollar

Deberá proponer e implementar dos soluciones (algoritmos) con el esquema greedy (voraz) al problema planteado. Un algoritmo escogerá la arista de menor coste para pavimentar entre las que queden disponibles, manteniendo conexa la subred que se está construyendo. Y el otro, seleccionará la arista de menor coste entre todas las restantes, aunque la subred resultante no sea conexa. Para el primer caso, se implementarán dos variantes del algoritmo, utilizando una **cola de prioridad y sin ella**.

Además, deberá implementar un **generador de redes aleatorias** (grafos no orientados, valorados positivamente y conexos). En el que dados un número de vértices y un número de aristas válido, generará una red aleatoria en un archivo de texto en disco (siguiendo el mismo formato que las redes reales) para luego poder cargarlo y ejecutar los algoritmos greedy sobre redes mucho más grandes.

```
<strong>0</strong> // no dirigido
<strong>n</strong> // número de vértices
<strong>1</strong> // vértice 1
<strong>2</strong> // vértice 2
<strong>...</strong>
<strong>n</strong> // vértice n
<strong>m</strong> // número de aristas
<strong>1 2 25.0</strong> // vi vj costeij
// así hasta completar m entradas/aristas
```

Para ello deberá realizar los siguientes apartados:

- **Estudio de la implementación**: Explicar los detalles más importantes de la implementación, tanto de las estructuras de datos utilizadas para almacenar la red, como de los algoritmos greedy implementados. El código debe de estar razonablemente bien documentado (JavaDoc).
- **Estudio teórico**: Estudiar los tiempos de ejecución de los algoritmos implementados, en función del número de núcleos de población (vértices) y del número de caminos (aristas). Comparar también los algoritmos propuestos, teniendo en cuenta las características de la red (grafo) y las técnicas de implementación elegidas. Responder de forma justificada a las siguientes preguntas: **(1)** ¿el resultado de la ejecución de cada algoritmo es único?. **(2)** ¿el resultado de la ejecución de los dos algoritmos debe ser el mismo?, ¿por qué?. **(3)** si el peso de las aristas fuese la distancia entre dos ciudades, con la estructura resultante, ¿podemos determinar el camino mínimo entre dos pares de ciudades cualquiera?.
- **Estudio experimental**: Validación de los algoritmos greedy implementados sobre las redes reales (EDAland) proporcionadas. Para ello, se deberán obtener y comparar los tiempos de ejecución de los algoritmos implementados. Se contrastarán los resultados teóricos y los experimentales, comprobando si los experimentales confirman los teóricos previamente analizados. Se justificarán los experimentos realizados, y en caso de discrepancia entre la teoría y los experimentos se debe intentar buscar una explicación razonada. Además, se generarán **redes aleatorias** (grafos no orientados, valorados positivamente y conexos), fijando un número vértices (por ejemplo 5000, 10000, 15000 y 20000) y variando el número aristas de tal manera que el grafo quede siempre conexo (**generador de redes aleatorias**). Sobre estas nuevas redes, se volverán a probar los algoritmos greedy implementados para poder compararlos con un número mayor de vértices y aristas.

Entregas

Se ha de entregar, en fecha, un repositorio público de GitHub (mismo repositorio para todas las prácticas de EDA II) con toda la documentación y código fuente requerido en la práctica:

- En dicho repositorio crear una nueva carpeta llamada `practica_2`, donde creéis dos subcarpetas una para la documentación, `docs` y otra para el código fuente `sources`.
- Memoria que explique todo lo que habéis realizado en la práctica. La memoria deberá tener el formato que se indica a continuación. Si se desea, también se podrá realizar una presentación de la práctica.
- Código fuente de la aplicación, desarrollada en JAVA, que resuelva todo lo planteado en la práctica. Recordad que tendréis que medir tiempos de ejecución de vuestras soluciones por lo que deberéis incluir las órdenes necesarias para ello en el código fuente.
- Juegos de prueba que consideréis oportunos para asegurarnos de que todo funciona correctamente.

La **memoria** de práctica a entregar debe ser breve, clara y estar bien escrita. Ésta debe incluir las siguientes secciones:

- Una breve **introducción** con un estudio teórico del método algorítmico utilizado en esta práctica (greedy).
- Una sección para cada uno de **apartados propuestos** a desarrollar en esta práctica (estudio de la implementación, estudio teórico y estudio experimental). Hemos de remarcar que deben incluirse los apartados en el mismo orden en el que se han expuesto.
- Se incluirá también un anexo con el diseño del código implementado (no incluir código), junto con una lista de los archivos fuente y una breve descripción del contenido de cada uno.
- Es importante incluir siempre las **fuentes bibliográficas** utilizadas (web, libros, artículos, etc.) y hacer referencia a ellas en el documento.

Evaluación

Cada apartado se evaluará independientemente, aunque es condición necesaria para aprobar la práctica que los programas implementados funcionen correctamente.

- La implementación junto con la documentación del código se valorará sobre un 40%
- El estudio de la implementación se valorará sobre un 10%
- El estudio teórico se valorará sobre un 15%
- El estudio experimental se valorará sobre un 35%

Se penalizará no entregar el apartado de introducción teórico o una mala presentación de la memoria.

Se podrá requerir la defensa del código y de la memoria por parte de profesor.

Fecha de entrega

Fecha de entrega: **17 de Abril de 2020**