PRÁCTICA 2 – Esquema Greedy o Voraz

Instalación de fibra óptica

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 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 comunicaciones).
- **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 telecomunicaciones, la red de metro, etc. En este caso nos centraremos en una red de telecomunicaciones (con fibra óptica) que conecta todos los núcleos de población del país **EDAland**, y la necesidad que tiene la empresa **EDACom** para actualizar la red de telecomunicaciones entre cuidades con una nueva fibra óptica de reciente aparición en el mercado. Pero dada la actual crisis económica en que estamos inmersos, no se dispone de presupuesto suficiente para poder actualizar todos los posibles trazados de fibra óptica que hay en el país. Tras una dura negociación con los diferentes departamentos de la empresa y teniendo en cuenta el presupuesto disponible, se ha decidido que se instalarán con la nueva fibra óptica únicamente aquellos trazados que conecten grandes núcleos de población, y además, intentar dedicar el menor presupuesto posible en la consecución de la obra.
Por lo que, el CEO de EDACom está muy interesado en saber qué trazados hay que instalar con la nueva fibra óptica para conectar a todos los posibles grandes núcleos de población del EDAland con el menor coste posible. Dicho problema se ha planteado a **theBestSoft** (y ésta se lo ha encargado a **EDASoft**) para que lo resuelva de la mejor manera posible.

En primer lugar, en **EDASoft** se dispone de la **red de telecomunicaciones reducida** de EDAland (Figura 1) que conecta a los grandes núcleos de población (ciudades) del país (21 ciudades y 29 trazados), junto con la distancia en kilómetros entre dos ciudades. En base a ello, se puede estimar el coste en euros del tirado de fibra óptica entre dos ciudades, si cada km de fibra óptica cuesta 7.000€. Por ejemplo, calcular el coste de fibra entre Almería y Granada bastaría con calcular 173 × 7.000 = 1.211.000 euros. Esta red nos servirá para probar nuestros algoritmos y verificar su correcto funcionamiento.

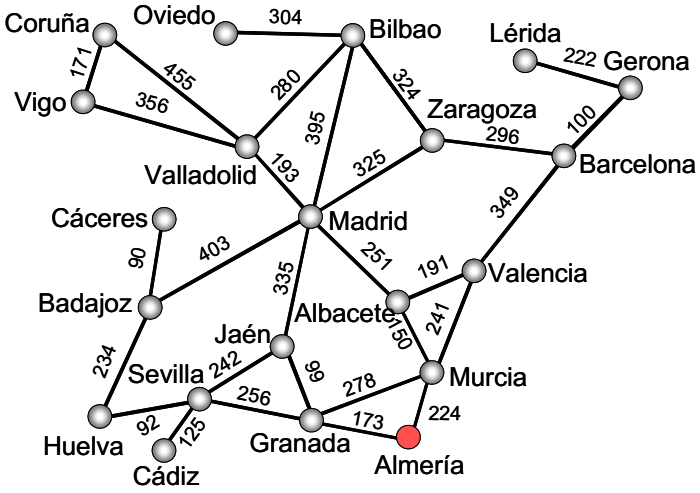

Figura 1. Red de telecomunicaciones reducida de EDAland.

Una vez comprobado el correcto funcionamiento de los algoritmos en la red de telecomunicaciones reducida. **EDASoft** ejecutará los algoritmos en la **red de telecomunicaciones completa** de EDALand (Figura 2) que dispone (1053 núcleos de población y 2017 posibles tramos en kilómetros conectando dos núcleos de población), y de esta manera se podría estimar el presupuesto total que costaría la obra global, que como ya sabemos debe satisfacer el menor posible (requerimiento principal de la EDACom).

Figura 2. Red de telecomunicaciones completa de EDAland.

Por otro lado, **EDACom**, después de analizar una ampliación de la red de telecomunicaciones reducida, ha decidido añadir 12 nuevos tirados de fibra óptica entre núcleos de población de EDAland marcados en azul y que se puede apreciar en la Figura 3 (21 ciudades y 41 trazados). En esta nueva red de telecomunicaciones reducida (cuyos valores de las aristas se corresponderían con kilómetros) y debido a que hay una centralita en cada núcleo de población (ciudad), la empresa ha encargado a su unidad de mantenimiento que realice una revisión general de las instalaciones de todas las centralitas que hay en la red. Para ello, **EDACom** ha encargado a **EDASoft** que le determine el **circuito con la distancia más corta** que debe seguir la unidad de mantenimiento, partiendo desde Almería (donde se encuentra la sede central de la empresa), visitando todos los núcleos de población (exactamente una vez) que tienen una centralita y regresar a la ciudad de partida (Almería).

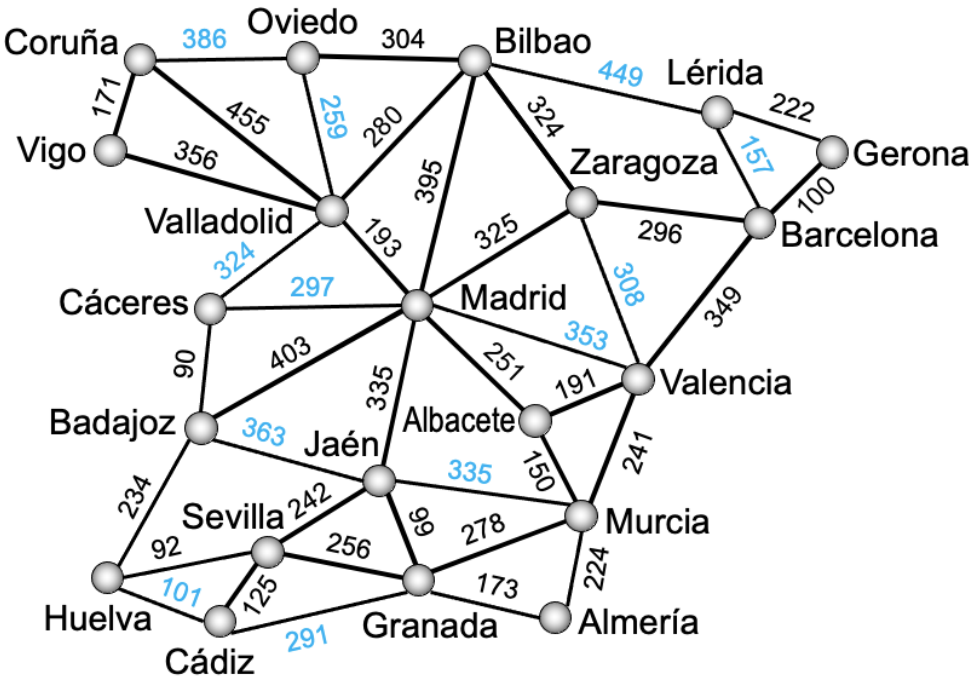

Figura 3. Nueva red de telecomunicaciones reducida de EDAland.

Este problema es conocido como el *problema del viajante de comercio* (TSP, Traveling Salesman Problem), y para resolver este problema se debe utilizar el algoritmo de **caminos simples** (basado en backtracking) estudiado en EDA I y que nos permitirá encontrar todos los posibles circuitos, empezando desde una ciudad origen, visitando todas las restantes ciudades exactamente una vez y volviendo a la ciudad origen, y quedarnos con el de menor distancia.

Trabajo a desarrollar

Deberá proponer e implementar las soluciones (algoritmos) con el esquema greedy (voraz) a los problemas planteados y que se exponen a continuación.

1. Uno de los algoritmos determinará el árbol de recubrimiento mínimo, partiendo de una ciudad (por ejemplo, Almería) e irá escogiendo la arista de menor coste para tirar la fibra óptica entre las ciudades que queden disponibles, manteniendo conexa la subred que se está construyendo (éste se conoce como algoritmo de **Prim**). Para este caso, se implementarán dos variantes del algoritmo, una utilizando una **cola de prioridad** y la otra **sin ella**. Estos algoritmos se probarán sobre las redes de telecomunicaciones reducida y completa de EDAland.
2. Otro algoritmo que resuelve el mismo problema (árbol de recubrimiento de coste mínimo), pero en este caso se seleccionará siempre la arista de menor coste entre todas las aristas restantes, aunque la subred o subredes resultantes no generen un grafo conexo (algoritmo de **Kruskal**). Este algoritmo debe implementarse de forma que su eficiencia sea la mejor y para ello deberá utilizar una **cola de prioridad**. Además, este algoritmo se probará también sobre las redes de telecomunicaciones reducida y completa de EDAland.
3. Como ya hemos indicado, para el caso de la nueva red de telecomunicaciones reducida (Figura 3) se debe implementar también un algoritmo que resuelva el problema planteado (revisión de la red por la unidad de mantenimiento). Éste consiste en encontrar **el circuito con la distancia más corta** (problema del viajante de comercio utilizando caminos simples) que debe seguir la unidad de mantenimiento, partiendo desde Almería, y visitando todas las ciudades que tiene una centralita exactamente una vez y regresar a Almería. Este algoritmo se probará su correcto funcionamiento únicamente en la nueva red de telecomunicaciones reducida.
4. Generar grandes redes aleatorias (grafos no orientados, valorados positivamente y conexos) de diferentes tamaños. Por ejemplo, podemos fijar un número vértices (n = 500, 1000, 1500, 2000 y 2500) y variando el número aristas (m) de tal manera que el grafo quede siempre conexo (**generador de redes sintéticas**). Finalmente, sobre estas nuevas redes comprobar el funcionamiento de ambos algoritmos (Prim y Kruskal, utilizando colas de prioridad) y mostrar su comportamiento (gráfica) en base al *tiempo total de ejecución*.
5. **Obligatorio para los grupos de 3 estudiantes (optativo para los grupos de 2)**. Realizar la representación gráfica de las soluciones (árboles (ARCM) y circuito) de los apartados anteriores utilizando **Graphviz**, sabiendo que están los archivos de datos disponibles en el repositorio GitHub de esta práctica. Además, el **generador de redes sintéticas** (que no será necesario representar) podría generar una red que conecte **todos con todos** los ***n*** nodos con $\frac{n\cdot(n-1)}{2}$ aristas para comprobar el correcto funcionamiento del algoritmo implementado para resolver el problema del viajante de comercio (TSP) con la técnica algorítmica greedy, destacando el error cometido al utilizar la heurística con respecto a la solución exacta (que utiliza *caminos simples*). Probar este caso para valores de **n** no muy grandes (por ejemplo 10, 50, 100, 250, 500 y 1000).

Con ánimo de probar las dos estructuras de datos que conocemos para representar los grafos, los algoritmos que resuelven (Prim y Kruskal) se utilizará una *lista de adyacencia* (o **mapa de adyacencia** con un mapa de mapas (visto en EDA I con `TreeMap`)) y para el otro algoritmo (calcular uno de los circuitos, TSP con heurística greedy) se utilizará una **matriz de adyacencia** (obligatorio solo para los grupos de 3).

Para generar esas redes sintéticas (no reales), deberá implementar un **generador de redes aleatorias** (grafos no orientados, valorados positivamente y conexos). En el que dados un número de vértices (**n**) y un número de aristas (**m**) 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 sobre redes sintéticas mucho más grandes que las de EDAland.

```
0 // 0 = no dirigido, 1 = dirigido
n // número de vértices del grafo
1 // vértice 1
2 // vértice 2
…
n // vértice n
m // número de aristas del grafo
1 2 125.0 // v_i v_j distancia_ij
// así hasta completar m entradas/aristas
```

Además, para poder resolver los problemas planteados, los archivos de datos están disponibles en el repositorio GitHub de esta práctica, en la carpeta `dataset`:

- **`graphEDAland.txt`** ⇒ Archivo con la red de telecomunicaciones reducida de EDAland (Figura 1).
- **`graphEDAlandNew.txt`** ⇒ Archivo con la nueva red de telecomunicaciones reducida de EDAland (Figura 3).
- **`graphEDAlandLarge.txt`** ⇒ Archivo con la red de telecomunicaciones completa de EDAland (Figura 2).
- **`graphEDAlandVertices.txt`** ⇒ Archivo con los vértices y sus coordenadas (`x`, `y`) para representar gráficamente la red de telecomunicaciones reducida de EDAland.

Para desarrollar la práctica 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.

- **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 sintéticos no orientados, valorados positivamente y conexos), fijando un número vértices (por ejemplo 500, 1000, 1500, 2000 y 2500) 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. Caso especial, es para las redes (conectando todos con todos) que se utilizarán para resolver el problema del viajante de comercio (TSP) con heurística greedy (para grupos de 3).

Entregas

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

- En dicho repositorio en la carpeta de fuentes `src/main/java`, crear un nuevo paquete llamado `org.eda2.practica02`, para el **código fuente** de la práctica. Además, en esta carpeta deben de incluirse los archivos de datos con las redes de telecomunicaciones reales y sintéticas.
- En la carpeta `docs`, dedicada a la **documentación**, crear una subcarpeta `practica02` para guardar toda la documentación (documento en `pdf` y los fuentes utilizados para su creación (por ejemplo, `.docx`)).
- **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 o en C++, que resuelva todo lo planteado en la práctica.
- **Juegos de prueba** que consideréis oportunos incluir para justificar que todo funciona correctamente (fundamentalmente para los casos de las redes más pequeñas o reducidas). En este caso los juegos de prueba deben estar en la carpeta de fuentes llamada `src/test/java` y dentro de ella en un paquete llamado `org.eda2.practica02`.

La **memoria de la práctica** a entregar debe ser breve, clara y estar bien escrita, adaptándose en lo posible a la norma UNE 157001:2014 “Criterios generales para la elaboración de proyectos”. Ésta debe incluir las siguientes secciones:

- Un apartado **Objetivo** con un estudio teórico del método algorítmico utilizado en esta práctica (greedy). Además, en esta sección deberá exponerse claramente ***qué miembro* del grupo ha actuado como *líder* y *qué tareas ha realizado cada miembro del grupo***. Es muy importante que todos los miembros dominen la práctica en su conjunto. Se puede incorporar la hoja de datos indicada para tal fin (`PR2_Tareas_a_repartir.xlsx`).
- Un apartado **Antecedentes** en el que se explique el motivo que lleva a realizar esta memoria y se enumeran los aspectos necesarios, en su caso, de las alternativas contempladas y la solución final adoptada.

- Una sección para cada uno de **apartados propuestos** a desarrollar en esta práctica (estudio de la implementación, estudio teórico y estudio experimental). Para el **estudio experimental**, el correcto funcionamiento de los algoritmos debe de justificarse con la correspondiente captura de pantalla en la que se pueda apreciar la salida correcta. Hemos de remarcar que deben incluirse los apartados en el mismo orden en el que se han expuesto. El apartado **estudio experimental** de la memoria recogerá los resultados finales de lo realizado.
- Se incluirá también un **anexo** con el diseño del código implementado, con diagramas de clases y cualquier otro diagrama que estiméis necesario incluir, **no introducir ningún código en este anexo**. Además, añadir en esta sección una lista de los archivos fuente que componen la práctica y una breve descripción del contenido de cada uno.
- En el apartado de **estudio experimental** o en un **anexo**, se expondrán los cálculos realizados en el estudio experimental, con la explicación del método seguido para la toma de datos y gráficos con el resultado obtenido (datos de muestras y ecuación seleccionada). Cuando se elija una curva para explicar el orden de un algoritmo (por ejemplo, O(n)), se deberá añadir el estadístico de la varianza explicada R^2, de forma que este valor (que varía entre 0 y 1) debería ser superior a 0.95 para que se acepte la solución propuesta. Valores por debajo 0.90 indican que el proceso de toma de muestras o el de elaboración de la hipótesis, ha sido incorrecto y debería llevar a su revisión. Cada muestra debería ser una media de varias medidas (al menos 10) de tiempo de ejecución y el tiempo medido debería ser, al menos, cercano al segundo para evitar interferencias del sistema operativo.
- 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: 21 de Abril de 2024