DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

ALGORÍTMICA

ETSIIT

Prácticas

ALGORITMOS GREEDY

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
UNIVERSIDAD DE GRANADA

Objetivos

El objetivo de esta práctica es desarrollar las habilidades para diseñar e implementar algoritmos mediante la técnica voraz. Para ello, se presentará al alumno un conjunto de problemas a solucionar mediante el uso de esta técnica. Cada problema deberá ser resuelto por grupos de alumnos (un problema por grupo).

1. El problema de asignación cuadrática (QAP)

Se dispone de N unidades y N localizaciones. Sean dos unidades *a* y *b* y dos localizaciones *i* y *j*, y dos matrices:

- $D_{NxN} = \{d_{ij}\}_{NxN}$ es una matriz cuadrada con las distancias entre la localización *i* y la localización *j* localizaciones i y j,
- $F_{NxN} = \{f_{ab}\}_{NxN}$ el flujo estimado de interacción desde la unidad a a la unidad b.

El problema del QAP consiste en encontrar una asignación de unidades a localizaciones, llamémosla p_{1xN}; donde *p(i)* = unidad asignada a la localización *i*, tal que se minimice el coste dado por la siguiente expresión:

$$p^* = \min_p \{H(p)\} = \min_p \{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f_{p(i)p(j)} d_{ij}\}$$

Se pide diseñar e implementar un algoritmo greedy que solucione este problema. Como salida, se deberá proporcionar la asignación entre unidades y localizaciones resultante del algoritmo junto con su coste.

2. El problema del recubrimiento de un grafo no dirigido

Consideremos un grafo no dirigido G = (V, E). Un conjunto U ⊆ V se dice que es un recubrimiento de G si cada arista en E incide en, al menos, un vértice o nodo de U. Es decir ∀(x, y) ∈ E, bien x ∈ U o y ∈ U . Un conjunto de nodos es un recubrimiento minimal de G si es un recubrimiento con el menor número posible de nodos.

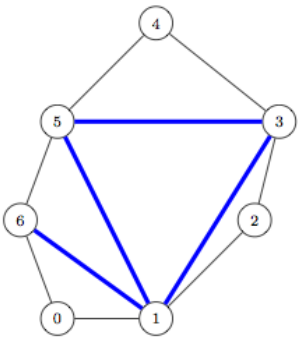
Se pide diseñar e implementar un algoritmo greedy que solucione este problema. Como salida, se deberá proporcionar el conjunto de nodos que forman el recubrimiento junto con el coste (número de nodos).

3. El problema de la triangulación mínima de polígonos

Un polígono P en el plano es un conjunto de n puntos (x_i, y_i) llamados vértices, y n segmentos de rectas $\{\alpha_1 , \alpha_2 , \ldots , \alpha_n \}$ llamados lados, verificando que:

1. Los puntos extremos de los lados son dos vértices distintos del polígono.
2. Cualquier vértice es el extremo de exactamente dos lados (distintos).

Un polígono del plano se dice convexo si, para cada dos puntos cualesquiera del interior del polígono, el segmento que los une está contenido en el mismo. Por la triangulación de un polígono plano entenderemos la división de éste en un conjunto de triángulos, con la restricción de que cada lado de un triángulo se reparta entre dos triángulos adyacentes (obviamente, sólo pertenece a uno si está en la frontera del polígono). Por ejemplo:



Diremos que una triangulación de un polígono convexo es mínima si la suma de los perímetros de los triángulos que la componen es mínima respecto a cualquier otra triangulación.

Se pide diseñar e implementar un algoritmo greedy que solucione el problema de encontrar la triangulación mínima de un polígono convexo. Como salida, se deberá proporcionar el conjunto de aristas que forman la triangulación junto con el coste (perímetro de cada triángulo generado).

4. El problema del embalaje (bin packing problem)

Se dispone de un número indefinido de embalajes o cajas (notamos S_i para la i-ésima caja), cada una de igual capacidad *V*, y un conjunto de *N* ítems a empaquetar (de tamaños $a_1, a_2, \ldots, a_n$). Si usamos la notación $|S_i|$ como el número de ítems contenidos en la caja *i*, y $S_i(j)$ al j-ésimo ítem empaquetado en la caja *i* ($1<=j<=|S_i|$), se desea:

Encontrar el mínimo número de cajas, y qué objetos deberían ir en cada una de ellas, con la siguiente restricción:

$$\forall i \sum_{j=1}^{|B_i|} a_{B_i(j)} \leq V$$

Es decir, que el volumen de todos los objetos contenidos en cada caja no supere la capacidad de cada caja.

Se pide diseñar e implementar un algoritmo greedy que solucione el problema. Como salida, se deberá proporcionar el conjunto de objetos contenidos en cada caja y el coste de la solución (número de cajas usadas).

5. El problema del mural

Se dispone de una fotografía *F* dividida en *NxM* regiones donde cada región tiene un tamaño de *kxk* píxeles, y un total de *N*M* fotografías p_i más pequeñas $\{p_i, 1<=i<=N*M\}$ de tamaño *kxk* píxeles. Cada región de *F*, *F(x,y)*, tiene asignado un color entre 0 y 255. A su vez, cada fotografía pequeña p_i también tiene asignado un color entre 0 y 255. El problema consiste en realizar una asignación de cada fotografía p_i a cada región de un mural *F'* de tamaño *NxM*, *F'(x,y),* de modo que no se repitan asignaciones de la misma fotografía p_i a dos puntos diferentes de *F'* y donde la composición del mural $F'(x,y)= p_i$ sea lo más similar posible a la fotografía original *F*; es decir:

Encontrar una asignación $F'(x,y)= p_i$ tal que se minimice la siguiente función de coste:

$$\sum_{x=1}^{N} \sum_{y=1}^{M} \left| F'(x,y) - F(x,y) \right|$$

Sujeto a la restricción:

$$F'(x_1,y_1) \neq F'(x_2,y_2) \forall (x_1,y_1),(x_2,y_2)$$

Se pide diseñar e implementar un algoritmo greedy que solucione el problema con el mínimo coste posible. Como salida, se deberá proporcionar la asignación de la matriz F' a la imagen correspondiente junto con su coste.

6. Enfoque Greedy para resolver los ejercicios

En la plataforma online docente de la asignatura se proporciona al alumno material de apoyo a la explicación de los ejercicios, así como instancias de problemas dentro de estas explicaciones, que ayuden a su resolución.

7. Tareas a realizar

Se presentará una memoria de prácticas conteniendo lo siguiente:

1. Se debe incluir un análisis del problema.
2. Diseño de la solución, describiendo cada una de las componentes de la misma relacionándola con las componentes de un algoritmo greedy.
3. Esqueleto del algoritmo greedy (pseudocódigo) que soluciona el problema, explicando cómo se ha incorporado cada una de las componentes greedy.
4. Explicación del funcionamiento del algoritmo, sobre un caso de ejemplo pequeño propuesto por los estudiantes.
5. Enunciado de un problema o caso real donde se pueda utilizar el algoritmo greedy para solucionarlo (distinto de los comentados en clase).
6. Cálculo del orden de eficiencia teórico del algoritmo.
7. Instrucciones sobre cómo compilar y ejecutar el código de la práctica.

8. Condiciones de entrega

Se deberá entregar, a través de la plataforma online, un fichero ZIP con la siguiente información:

- La solución de los ejercicios planteados en la práctica deberá presentarse en una **memoria de prácticas en formato PDF que contenga, al menos, la respuesta a los apartados requeridos en las tareas a realizar**.
- Se deberá presentar, además, el código fuente de la solución al problema (sólo código fuente, no se aceptarán ejecutables).

- **No se admitirán ficheros .zip que contengan programa ejecutables.**

Adicionalmente:

- En horario de clase, los grupos expondrán su solución ante el resto de la clase. Para ello, se remitirá, mediante la plataforma online docente, un PDF con la presentación, adaptada para su exposición en 5 minutos. Dicha entrega se realizará con posterioridad a la entrega de la memoria, previo aviso por el profesor a través de mensajes por la plataforma docente.

9. Evaluación

Criterios de evaluación:

- La calificación de la práctica será un valor comprendido entre 0 y 10.
- La solución del alumno resuelve correctamente el ejercicio (el algoritmo hace aquello para lo que es diseñado de forma eficiente, limpia y óptima, sin errores, 40% del valor de la pregunta).
- Análisis del problema (20%).
- Explicación y diseño de la solución, con ejemplos (40%).

10. Motivación adicional y puesta en práctica de las soluciones

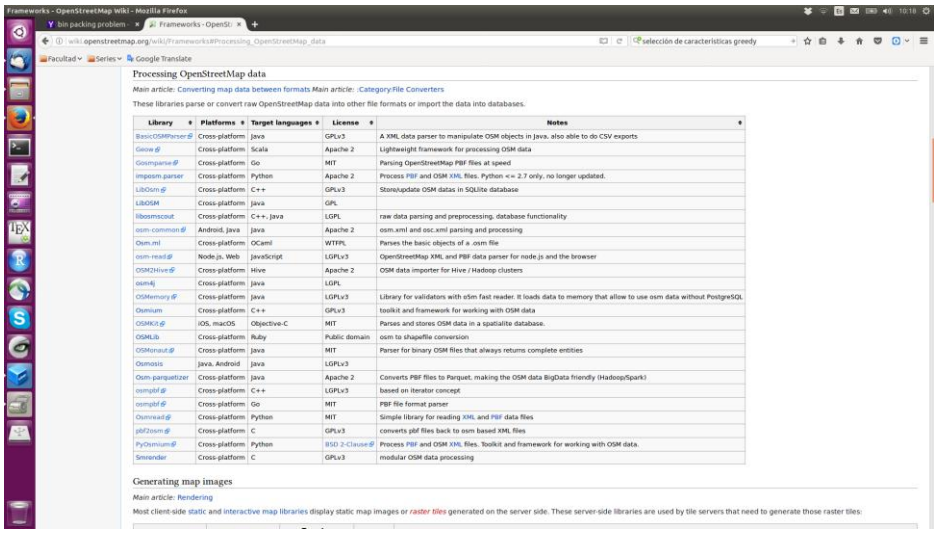
Aclaración: La información contenida en este apartado no es necesario incorporarla a ninguna de las soluciones elaboradas.

La asignatura se centra en el análisis y diseño de algoritmos, y se estudian diferentes técnicas tanto para optimizar la eficiencia de los algoritmos como para saber diseñar soluciones a problemas de optimización. No obstante, estos algoritmos suelen ser una pequeña parte de otros grandes sistemas de información y de cálculo científico, donde se incorporan para dar una solución real a los problemas de una empresa.

Por ejemplo, el problema 4 es un problema al que se enfrentan multitud de empresas de transporte y de embalaje como **Amazon**, **eBay**, etc., para minimizar el coste tanto del contenedor de los ítems que se compran o transportan, como del espacio que es ocupado por cada contenedor a la hora de ser transportado.

En particular, los problemas 1 a 3 suelen ser problemas ampliamente utilizados en tareas de planificación y logística de empresas medianas, grandes y multinacionales, y muchas veces conllevan el análisis geográfico de diferentes sedes o almacenes de la empresa, localizaciones de clientes, etc. Para incorporar las soluciones de los algoritmos en una aplicación de gestión o de planificación de recursos, en ocasiones (cada vez más frecuente) es conveniente dotar a dicha aplicación de geo-referenciación de datos. Existen diferentes APIs de programación para acceso libre a datos geográficos (posiciones, direcciones, etc.), donde algunas de las más famosas son **GoogleMaps** y **OpenStreetMap**.

En particular, **OpenStreetMap** (http://wiki.openstreetmap.org/wiki/Frameworks) contiene varias APIs para C++ que permiten acceder a localizaciones (latitud, longitud) o direcciones reales, calcular rutas, etc.:



Página de frameworks de desarrollo para OpenStreetMaps

Ejemplo de mapa mostrado con Mapnik, usando la biblioteca OSM (fuente: http://wiki.openstreetmap.org/wiki/Mapnik)

Aunque **el estudio de estos sistemas queda fuera de los objetivos específicos de la asignatura**, se anima al estudiante a comprobar la viabilidad de incluir información real geo-referenciada a través de alguna de las APIs para C++ o Java (según el lenguaje escogido) dentro del código de sus soluciones, y estudiar la complejidad adicional que, más allá del diseño de algoritmos, supone la resolución de un caso real.

La no realización de esta tarea no será evaluada negativamente en la práctica, dado que supone únicamente un ejemplo para motivar al alumno en su tarea, y es una línea de trabajo para desarrollo de habilidades adicionales del alumno que no son objeto de esta asignatura y cuya carga de trabajo supera, en promedio, lo esperado para cada práctica.