ANÁLISIS Y DISEÑO DE ALGORITMOS

Práctica 6 de laboratorio

Esta práctica ocupa dos semanas:

Semana 1: Versión recursiva sin almacén.
Semana 2: Práctica completa.

Entrega: Respectivamente, hasta los días 18 de marzo y 1 de abril, 23:55h.
A través de Moodle

Asignación de coste mínimo

Una comarca compuesta de n aldeas comunicadas mediante una única carretera está planificando la instalación de g gasolineras ($0 < g \leq n$) para dar servicio a todos sus vehículos. De cada vehículo se conoce la aldea en la que está censado y para repostar deberá ir a la población más cercana que disponga de estación de servicio. Se pretende conocer en qué aldeas habría que ubicar las gasolineras para minimizar el total de las distancias que han de recorrer todos los vehículos para ir a la gasolinera más cercana.

Por ejemplo, supongamos que la comarca está compuesta de $n = 7$ aldeas cuyas distancias entre poblaciones adyacentes se representa en la figura 1. Para cada aldea, numeradas de 0 a 6, se muestra también el número de vehículos censados (en forma de subíndice). Si el número de gasolineras a instalar, valor preestablecido, fuera $g = 3$ y estas se ubicaran en las poblaciones 0, 3 y 5 (vértices sombreados en color gris), la suma total de distancias a recorrer vendría dada por la expresión $29 \cdot (1 + 1) + 23 \cdot 1 + 37 \cdot 2,5 + 41 \cdot 2 = 255,5$; sin embargo, esta no es la mejor ubicación de las estaciones de servicio puesto que si se instalaran en las aldeas 0, 2 y 5 se tendría una distancia menor ($29 \cdot 1 + 43 \cdot 1 + 37 \cdot 2,5 + 41 \cdot 2 = 246,5$), que corresponde en este caso a la mejor disposición posible.

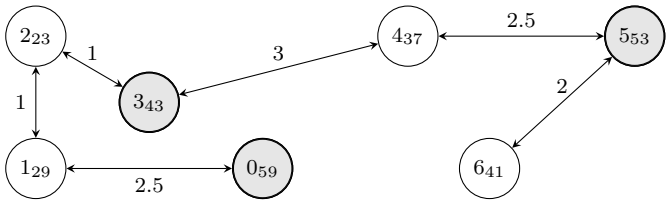

Figura 1: Posible ubicación de las tres gasolineras en los vértices sombreados en gris. Los habitantes de las aldeas 1 y 2 irían a repostar a la aldea 3 (ya que es donde está la gasolinera mas cercana), y los de las aldeas 4 y 6 irían a la 5. Los de las demás aldeas no se moverían ya que dispondrían de gasolinera en su aldea. **Esta ubicación de las gasolineras no es la mejor.**

Puesto que en este caso se puede establecer un orden entre las poblaciones, las distancias entre ellas vendrán dadas en un vector d donde $d_0 = 0$ y cada $d_i \in R^+$ contiene la distancia de la aldea i a la aldea origen (etiquetada con el número 0), por lo tanto se cumplirá $d_i > d_{i-1}, \forall i : 1 \leq i \leq n - 1$. Por otra parte, el número de vehículos censados en cada población vendrá dado en un vector v con $v_i \in N$.

La solución a este problema, al que llamaremos asignación de coste mínimo (mca), se puede obtener recursivamente de la siguiente manera:

- Si sólo hay que colocar una gasolinera ($g = 1$) la evaluación de mca(g, n) requiere calcular las n sumas (s_i) de las distancias recorridas por todos los vehículos a la ciudad i en la que

ubicamos la gasolinera y encontrar el mínimo de s_i (que corresponderá a la ciudad en la que hay que colocar finalmente la gasolinera). Llamando centroide$(0, n)$ a dicho mínimo tendremos que mca$(1, n)$ = centroide$(0, n)$. En otras palabras, centroide(a, n) es la distancia total recorrida por todos los vehículos si colocamos, en la mejor ubicación posible, una única gasolinera entre las ciudades a y $n - 1$.[1]

- Si hay más de una gasolinera ($g > 1$), supongamos que la primera de ellas da servicio a las últimas $n - a$ ciudades. La distancia total recorrida en este tramo es centroide(a, n). Se trata entonces de asignar las $g-1$ gasolineras restantes entre las primeras a ciudades (obsérvese que siempre tiene que haber al menos tantas ciudades como gasolineras, es decir, $g-1 \leq a \leq n-1$), en las que se recorrerá una distancia mca$(g - 1, a)$. La suma mínima de estas dos cantidades s_a = mca$(g - 1, a)$ + centroide(a, n) nos permite definir la expresión recursiva que buscamos.

En esta práctica se pide, aplicar el método *divide y vencerás* y su extensión a *programación dinámica* para obtener la distancia mínima que recorrerán todos los vehículos según la mejor ubicación posible de las gasolineras.

Para resolver este ejercicio se debe implementar los siguientes algoritmos:[2]

1. Recursivo sin almacén (ineficiente)
2. Recursivo con almacén (memoización)
3. Iterativo con almacén que hace uso de una tabla para almacenar los resultados intermedios.
4. Iterativo con almacén con complejidad espacial mejorada.
5. Por último deberá obtenerse, a partir de alguna de las tablas de almacén (memoización o versión iterativa), el emplazamiento idóneo para las g gasolineras y a continuación, la gasolinera (identificada por el número de aldea en la que está instalada) en la que deben repostar los vehículos de cada población.[3]

- **Nombre del programa, opciones y sintaxis de la orden.**

  El programa a realizar se debe llamar **mca**. La orden tendrá la siguiente sintaxis:

  mca [-t] [--ignore_recursive] -f `fichero_entrada`

  En el siguiente apartado se describe el significado de las opciones.

- **Salida del programa y descripción de las opciones:**

  En todas las formas posibles de utilizar la mencionada orden, la salida del programa (véase los ejemplos a continuación) será, en primer lugar, la solución obtenida mediante los cuatros algoritmos anteriormente citados: recursivo sin almacén, memoización, iterativo con tabla e iterativo con complejidad espacial mejorada. (debe seguirse este orden). No obstante, si se incorpora a la orden la opción [--ignore_recursive] se deberá excluir de la salida la solución recursiva sin almacén, por su elevado coste computacional (evidentemente el programa tampoco deberá hacer la llamada a esta función); en este caso la solución de los otros tres algoritmos sí que deberá mostrarse. En segundo lugar, deberá indicarse el emplazamiento idóneo de las gasolineras seguido de la aldea en la que debe repostar cada población y, para terminar, el coste asociado a esta asignación.[4] En cuanto al resto de opciones:

  - Si se hace uso de la opción [-t] se mostrará además la matriz obtenida en la versión iterativa (donde se almacenan los resultados intermedios).

[1] Para facilitar la obtención de los mejores emplazamientos la función centroid(a, n) debería devolver no sólo la distancia mínima, sino también el índice de la ciudad en la que se debe colocar la gasolinera.

[2] Como es lógico, la solución de los algoritmos que muestran el coste mínimo deben coincidir.

[3] En estos casos la solución puede no ser única.

[4] Como es lógico, este coste coincidirá con el obtenido en cualquiera de los algoritmos pedidos pero servirá como comprobación de que la asignación es compatible con la solución mostrada con dichos algoritmos.

- Las opciones no son excluyentes entre sí, ninguna de ellas. Además pueden aparecer en cualquier orden.
- En cualquiera de las formas posibles de utilizar la orden, si se incorpora la opción [--ignore_recursive] se debe excluir (únicamente) la solución recursiva sin almacén.
- La opción -f, la única de uso obligado, se utiliza para suministrar el nombre del fichero donde está el mapa a resolver (véase siguiente apartado)

- **Entrada al programa**

  El problema a resolver se suministrará codificado en un fichero de texto cuyo nombre se recogerá a través de la opción -f. Su formato y contenido será:

  - Línea 1 del fichero: Valores n y g, en este orden.
  - Línea 2: Censo de vehículos: Una lista de n números enteros mayores que cero.
  - Línea 3: Distancias con respecto a la aldea origen: una lista de n números reales mayores que cero, excepto el primero que siempre será 0.

  Por tanto, el fichero contendrá 3 líneas que finalizarán con un salto de línea, salvo en todo caso, la última línea. El carácter separador entre números siempre será el espacio en blanco.

  A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos y sus soluciones, entre ellos está `0.problem` y `1.problem` utilizados a continuación para describir el formato de la salida.

- **Formato de salida. Ejemplos de ejecución.**

  Sea el fichero 1.problem, se trata del problema utilizado como ejemplo anteriormente.

  A continuación se muestra posibles formas de utilizar en la terminal la orden descrita y la salida que el programa debe mostrar. Es imprescindible ceñirse al formato y texto de salida mostrado, incluso en lo que se refiere a los saltos de línea o carácter separador, que en todos los casos es el espacio en blanco (aunque no hay problema si hay más de uno). La última línea debe terminar con un salto de línea (y sólo uno). **Debe respetarse los textos y formatos mostrados y en ningún caso debe añadirse texto o valores adicionales.**

```
$mca -f 1.problem
Recursive: 246.5
Memoization: 246.5
Iterative (table): 246.5
Iterative (vector): 246.5
Emplacements: 0 2 5
Assignment: 0 2 2 2 5 5 5
Assignment cost: 246.5
$
$mca -t --ignore_recursive -f 1.problem
Memoization: 246.5
Iterative (table): 246.5
Iterative (vector): 246.5
Emplacements: 0 2 5
Assignment: 0 2 2 2 5 5 5
Assignment cost: 246.5
Iterative table:
     0   -     -
  72.5   0     -
   153  2      0
 256.5  72    23
 426.5 192    72
   749 349 164.5
1056.5 431 246.5
```

```
$mca -f 0.problem -t --ignore_recursive -t
Memoization: 0
Iterative (table): 0
Iterative (vector): 0
Emplacements: 0
Assignment: 0
Assignment cost: 0
Iterative table:
0
$
$mca -f -t
ERROR: can't open file: -t.
Usage:
mca  [-t] [--ignore_recursive] -f file
$
$mca -f 0.problem -t -a
ERROR: unknown option -a.
Usage:
mca  [-t] [--ignore_recursive] -f file
$
$mca -f mifichero -t
ERROR: can't open file: mifichero.
Usage:
mca  [-t] [--ignore_recursive] -f file
```

Normas para la entrega.

> **ATENCIÓN: Estas normas son de obligado cumplimiento para que esta práctica sea evaluada.**

1. Se debe entregar únicamente el código fuente, con nombre `mca.cc`, y el fichero `makefile`. No hay que entregar nada más, en ningún caso se entregarán ficheros de test.
2. Es imprescindible que no presente errores ni de compilación ni de interpretación (según corresponda), en los ordenadores del laboratorio asignado.[5] Se tratará de evitar también cualquier tipo de *warning*.
3. Todos los ficheros que se entregan deben contener el nombre del autor y su DNI (o NIE) en su primera línea (entre comentarios apropiados según el tipo de archivo).
4. Se comprimirán en un archivo *.tar.gz* cuyo nombre será el DNI del alumno, compuesto de 8 dígitos y una letra (o NIE, compuesto de una letra seguida de 7 dígitos y otra letra). Por ejemplo: 12345678A.tar.gz o X1234567A.tar.gz. **Solo se admite este formato de compresión y solo es válido esta forma de nombrar el archivo.**
5. En el archivo comprimido **no debe existir subcarpetas**, es decir, al extraer sus archivos estos deben quedar guardados en la misma carpeta donde está el archivo que los contiene.
6. La práctica hay que subirla a *Moodle* respetando las fechas expuestas en el encabezado de este enunciado.
7. En la entrega correspondiente a la primera sesión sólo es necesario que esté implementada la versión recursiva sin almacén. Sin embargo, la gestión de opciones del programa debe estar hecha en su totalidad. El resultado que se debe mostrar para los casos aún sin hacer es "¿?". Por ejemplo:

```
$mca -t -f 1.problem
Recursive: 246.5
Memoization: ¿?
Iterative (table): ¿?
Iterative (vector): ¿?
Emplacements: ¿?
Assignment: ¿?
Assignment cost: ¿?
Iterative table:
¿?
```

[5] Si trabajas con tu propio ordenador o con otro sistema operativo asegúrate de que este requisito se cumple.