# ANÁLISIS Y DISEÑO DE ALGORITMOS

## Algoritmos voraces

### Práctica 7 de laboratorio

**Fecha de entrega: Hasta el día 13 de abril, 23:55h.**

# Encaminamiento óptimo II

Una empresa que ofrece servicios de almacenamiento en la nube dispone de $n$ servidores (que también llamaremos nodos) donde aloja los archivos de sus clientes.

**Los nodos están conectados entre sí mediante un sistema de comunicaciones bidireccional que conforman un grafo conexo no necesariament completo**.[1]

Por cuestión de costes, solo una cantidad $m$ de servidores ($0 < m \leq n$) pueden estar conectados directamente con el exterior. Llamaremos "puerta de enlace" a un nodo de almacenamiento que dispone de conectividad exterior directa. Si un nodo no es puerta de enlace entonces se le asignará la que está más cerca (que será la puerta de enlace asociada a ese nodo sin conexión exterior directa). Cada archivo que un cliente solicita para su descarga se envía desde el nodo que lo aloja hasta su puerta de enlace asociada, teniendo que atravesar (si es el caso) todos los nodos intermedios que los separa.

Por otra parte, cada nodo $i$ está caracterizado por un parámetro $c_i \in \mathbb{R}^+$ que indica su capacidad de almacenamiento. Cuanto más alto es su valor, más archivos puede albergar el nodo.

Definimos *tráfico estimado* de una instalación como $\sum_{i=0}^{n-1} c_i \cdot d_{ij}$, donde $d_{ij} \in \mathbb{R}^+$ es la distancia más corta que separa el nodo $i$ de su puerta de enlace asociada (el nodo $j$). Si el nodo $i$ es, él mismo, puerta de enlace (es decir, $i = j$), entonces $d_{ij} = 0$ (por lo tanto, los nodos que son puerta de enlace no aumentan el tráfico estimado ya que no hacen uso del bus privado).

Se pretende seleccionar los $m$ nodos que actuarán de puerta de enlace hacia el exterior y, para el resto de los nodos, indicar cuáles serán sus puertas de enlace asociadas. Todo ello con el objetivo de minimizar el tráfico estimado de la instalación (para simplificar, supondremos que el valor de $m$ ha sido prefijado).

A diferencia del problema de la práctica anterior, en este caso puede haber más de un camino entre cualquier par de nodos (ya sea directo o a través de otros nodos). Esta circunstancia hace que no se cumpla la propiedad de *subestructura óptima* ya que el problema original no puede resolverse de manera eficaz a partir de las soluciones de subproblemas independientes. Por lo tanto, no puede aplicarse el esquema de *programación dinámica*. **En esta práctica se pide aplicar una estrategia voraz para encontrar una aproximación a la mejor ubicación posible de las puertas de enlace.**[2]

En cuanto a la forma de suministrar las distancias, para simplificar, el programa recibirá una matriz cuadrada $d$, $n \times n$, donde $d_{ii} = 0$ y $d_{ij} = d_{ji} \in R^+$ contiene la distancia del camino más

[1] En el grafo, los vértices son los nodos y las aristas los canales de comunicación entre nodos; aunque no todos los pares de vértices del grafo estén conectados por aristas, se garantiza que todos los vértices son alcanzables entre sí (grafo conexo).

[2] Debe tenerse en cuenta que el método voraz no garantiza, para este problema, la solución óptima; sin embargo puede emplearse para obtener una aproximación a dicha solución, es decir, un subóptimo.

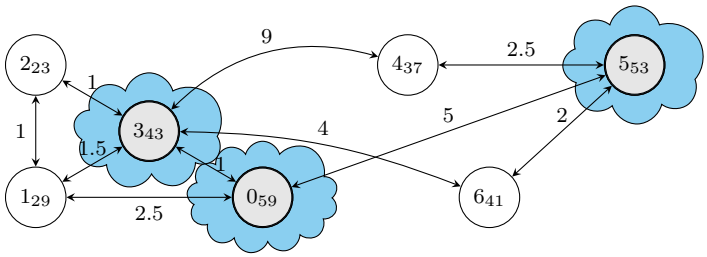


Figura 1: Posible ubicación de las tres puertas de enlace (en los vértices dibujados dentro de una nube). Las puertas de enlace asociadas a los nodos 1 y 2 son el nodo 3 (ya que es el nodo más cercano con conexión externa). La asociada a los nodos 4 y 6 es el nodo 5. El tráfico estimado asociado a esta solución viene dado por la expresión $29 \cdot 1{,}5 + 23 \cdot 1 + 37 \cdot 2{,}5 + 41 \cdot 2 = 241$. **Esta ubicación, que no es la mejor, está basada en el criterio voraz de colocar las puertas de enlace en los nodos con mayor capacidad de almacenamiento.**

corto entre el nodo $i$ y el nodo $j$, $\forall i, j : 0 \leq i, j \leq n-1$ (de esta manera no es relevante conocer cuál es el canal que produce el camino más corto o cuántos nodos intermedios se atraviesan).

Por otra parte, la capacidad de almacenamiento de cada nodo vendrá dada en un vector $c$ con $c_i \in \mathbb{R}^+$.

Para realizar este ejercicio, se implementarán los dos criterios voraces siguientes:

1. Colocar las $m$ puertas de enlace en los $n$ nodos con mayor capacidad de almacenamiento (esta es la solución mostrada en la figura 1).

2. Comenzar colocando una puerta de enlace en cada nodo (el tráfico estimado asociado sería por lo tanto 0) y a continuación ir quitando, una a una, las $n - m$ puertas de enlace que hacen que el tráfico estimado de la nueva configuración (con una puerta de enlace menos) aumente menos. Si se siguiera este criterio con el grafo de la figura 1 las puertas de enlace se ubicarían en los nodos 0, 1 y 5; de lo que resulta un tráfico estimado con valor 240,5.

El criterio 2 produce, en general, mejores soluciones al aproximarse más al valor óptimo que para el problema del ejemplo corresponde con la ubicación 3, 4 y 5, de lo que resulta un tráfico estimado igual a 207,5.[3] (Nótese que ninguno de los dos criterios de selección mencionados ha sido capaz de encontrar esta disposición de las puertas de enlace).

- **Nombres, en el código fuente, de algunas funciones importantes.**

  Para una correcta identificación en el código de las dos heurísticas voraces mencionadas en el apartado anterior, deberán ser nombradas de manera preestablecida.

  Siguiendo el mismo orden en el que se han enumerado, los nombres de las funciones de `C++` deben ser: `met_greedy_1` y `met_greedy_2`, respectivamente.

  Se deja total libertad para añadir los parámetros que se consideren y también para escoger el tipo de datos de retorno.

  Si esas dos funciones no están (o están con otro nombre) entonces se considerará que no han sido desarrolladas, con la consiguiente merma en la calificación del trabajo.

[3] Este valor ha sido obtenido con un método de búsqueda exhaustiva como es *Backtracking* (siguiente práctica).

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

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

  `met_greedy -f` *`fichero_entrada`*

  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 $m$, en este orden.
  - Línea 2: Capacidad de almacenamiento de cada nodo: Una lista de $n$ números reales positivos incluyendo el 0.
  - Línea 3 y $n-1$ líneas siguientes: Una matriz simétrica de números reales positivos que contiene el camino más corto entre cualquier par de nodos según se ha descrito anteriormente.

  Por tanto, el fichero contendrá $n+2$ 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.

  El programa debe ser robusto en cuanto al uso incorrecto de la sintaxis de la orden. En este sentido, debe informar también del primer error que se detecte (siguiendo las directrices de la práctica anterior). No es necesario controlar posibles errores en el fichero de entrada ya que este siempre se ajustará fielmente al formato establecido.

  A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos y sus soluciones; entre ellos está `07n03m.p`, utilizado como ejemplo en este enunciado

- **Salida del programa, formato de salida y ejemplo de ejecución.**

  La salida del programa consistirá, en primer lugar y en la primera línea: el valor del tráfico estimado de las soluciones obtenidas mediante cada una de las dos heurísticas mencionadas. Ambos valores se mostrarán en la misma línea, separados por un espacio en blanco y siguiendo el orden en el que se han discutido en este enunciado.

  En la segunda línea de la salida, se mostrará la relación de puertas de enlace asociadas a cada uno de los nodos (*tabla de encaminamiento*) considerando la mejor heurística de las dos.[4] Los $n$ valores numéricos deben estar separados mediante un espacio en blanco; el último terminará con un salto de línea.

  En la tercera línea deberá mostrarse el mínimo tráfico estimado calculado a partir de la tabla de encaminamiento mostrada en la línea anterior.

  Ejemplo de ejecución:

```
$met_greedy -f 07n03m.p
241 240.5
0 1 1 0 5 5 5
240.5
```

[4] Se trata de escoger, de entre las dos alternativas voraces, la que obtenga un tráfico estimado menor. Debe tenerse en cuenta que aunque normalmente el criterio 2 produce mejores resultados, no siempre será así.

- **Normas para la entrega.**

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

**Es imprescindible ceñirse al formato y texto de salida descrito, incluso en lo que se refiere a los saltos de línea o carácter separador, que en todos los casos es el espacio en blanco. No debe añadirse texto o valores adicionales. Si estos requisitos no se cumplen, es posible que falle una de las fases de la evaluación de este trabajo, que se hace de manera automática y en tal caso la práctica tampoco será evaluada.**

1. El programa debe realizarse en un único archivo fuente con nombre `met_greedy.cc`.
2. Se debe entregar únicamente los ficheros `met_greedy.cc` y `makefile` (para generar el archivo ejecutable a través de la orden `make` sin añadir nada más). **Sigue escrupulosamente los nombres de ficheros y funciones que se citan en este enunciado. Solo hay que entregar esos dos archivos (en ningún caso se entregarán archivos de *test*).**
3. Es imprescindible que no presente errores ni de compilación ni de interpretación (según corresponda), en los ordenadores del laboratorio asignado y en el sistema operativo *GNU/Linux*.[5] Se tratará de evitar también cualquier tipo de aviso (*warning*).
4. 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).
5. 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 válida esta forma de nombrar el archivo.**
6. En el archivo comprimido **no debe haber subcarpetas**, es decir, al extraer sus archivos estos deben quedar guardados en la misma carpeta donde está el archivo que los contiene.
7. La práctica hay que subirla a *Moodle* respetando las fechas expuestas en el encabezado de este enunciado.

[5] Si trabajas con tu propio ordenador o con otro sistema operativo asegúrate de que este requisito se cumple (puedes comprobarlo haciendo uso del compilador *online* de `https://godbolt.org`).