ANÁLISIS Y DISEÑO DE ALGORITMOS

Vuelta atrás

Práctica 8 de laboratorio

Fecha de entrega: Hasta el día 15 de mayo, 23:55h.

Encaminamiento óptimo III

`[El enunciado del problema es idéntico al de la práctica anterior (algoritmos voraces). La diferencia está en la forma de abordarlo y en la solución que se obtiene, que en este caso debe ser la mejor ubicación posible de las puertas de enlace, es decir, la solución óptima. La salida del programa también ha cambiado: en lugar de la tabla de encaminamiento se pide la relación de nodos que actuarán de puerta de enlace]`

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 necesariamente 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).

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 $d_{ij} = d_{ji} \in \mathbb{R}^+$ contiene la distancia del camino más corto entre en nodo i y el nodo j, $\forall i, j : 0 \leq i, j \leq n-1$ (de esta manera no es relevante conocer

[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).

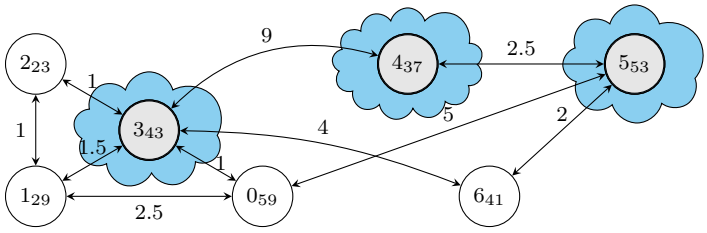

Figura 1: Posible ubicación de las tres puertas de enlace en los nodos dibujados sobre una nube. Los archivos de los nodos 0, 1 y 2 irían al nodo 3 (ya que es la puerta de enlace más cercana), y los del nodo 6 irían a la puerta de enlace ubicada en el nodo 5. El tráfico estimado asociado a esta solución viene dado por la expresión $59 \cdot 1 + 29 \cdot 1{,}5 + 23 \cdot 1 + 41 \cdot 2 = 207{,}5$. **Esta ubicación de las puertas de enlace es la mejor.**

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 $c_i \in \mathbb{R}$.

En esta nueva práctica se pide aplicar el método de la vuelta atrás (*backtracking*) para obtener la mejor ubicación posible de las puertas de enlace.

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

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

  `met_bt -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.**

  El programa mostrará los siguientes cuatro resultados, cada uno en una línea distinta:

1. Línea 1: Tráfico estimado que se obtiene de la ubicación escogida para las puertas de enlace.
2. Línea 2: La relación de los m nodos que han sido seleccionados para actuar de puerta de enlace, separados por un espacio en blanco.[2]
3. Línea 3: Una estadística sobre los nodos del árbol de búsqueda de vuelta atrás (*backtracking*) que consiste en los siguientes cinco valores:[3] (1) Número de nodos visitados; (2) número de nodos explorados; (3) número de nodos hoja que se visitan; (4) número de nodos descartados por no ser factibles y (5) número de nodos descartados por no ser prometedores. El carácter separador debe ser un espacio en blanco y deben mostrarse en ese orden.
4. Línea 4: El tiempo de CPU, expresado en milisegundos, que se ha consumido para resolver el problema. Si se utiliza un subóptimo de partida o se hace una preparación previa de los datos, el tiempo necesario para ello también cuenta en este cómputo.

Por ejemplo:

```
$met_bt -f 07n03m.p
207.5
4 3 5
39 19 10 0 10
0
```

Es imprescindible ceñirse al formato 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). **En ningún caso debe añadirse texto o valores adicionales.**

- **Sobre la evaluación de esta práctica.**

  Además del uso apropiado de la estrategia y de la obtención del resultado correcto, en esta práctica se tendrá en cuenta también el tiempo de CPU que se requiere para obtener la solución. En este sentido puede ser relevante:

  1. Forma de expandir el árbol de estados posibles.
  2. Uso de un subóptimo de partida lo más ajustado posible (cota pesimista del estado inicial).
  3. Uso de técnicas de poda basadas en la mejor solución hasta el momento (cotas optimistas).

  Es muy recomendable que en el archivo `makefile` incluyas la opción del compilador `-O3` (optimizador de código).

[2] No importa el orden en el que se muestren los nodos seleccionados. Por otra parte, la selección que produce un tráfico estimado mínimo puede no ser única; en estos casos, da igual cuál se muestre.

[3] La magnitud de cada uno de estos valores depende de cada implementación particular por lo que no tienen por qué coincidir con los publicados (de hecho, lo normal es que no coincidan), incluso algunos de ellos podrían ser 0.

- **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_bt.cc`.
2. Se debe entregar únicamente los ficheros `met_bt.cc` y `makefile` (para generar el archivo ejecutable a través de la orden `make` sin añadir nada más). **Sigue escrupulosamente los nombres de ficheros 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.[4] Se tratará de evitar también cualquier tipo de *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 la fecha expuesta en el encabezado de este enunciado.

[4] 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`).