ANÁLISIS Y DISEÑO DE ALGORITMOS

Método voraz

Práctica 7 de laboratorio

Entrega: Hasta el domingo 28 de abril, 23:55h. A través de Moodle

Camino de coste mínimo II

`[El enunciado del problema es idéntico al de la práctica anterior; la diferencia está en la forma de abordarlo y en la solución obtenida, que en este caso puede que no sea la óptima.]`

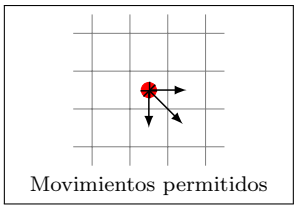
Un senderista está planificando una ruta de montaña haciendo uso de un mapa de coordenadas $n \times m$. En cada posición (i, j) de dicho mapa se representa, mediante un número natural -incluido el 0-, el grado de dificultad de visitar esa casilla. Cuanto más elevado es dicho valor más difícil es acceder a esa posición. Por ejemplo:

$\xrightarrow{(0,0)}$

1	3	5	1	1
2	4	6	3	1
1	2	9	7	1
9	1	7	1	9
1	3	7	5	1
8	1	2	2	1

$\xrightarrow{(5,4)}$

Un mapa de coordenadas 6×5



Se pide, aplicar el método voraz[1] para tratar de obtener la dificultad del camino más favorable[2] que partiendo del origen $(0, 0)$ conduce al destino $(n - 1, m - 1)$ asumiendo solo tres posibles movimientos desde una casilla cualquiera (i, j):

1. derecha: $(i, j + 1)$,
2. abajo: $(i + 1, j)$,
3. abajo y derecha (diagonal): $(i + 1, j + 1)$.

Como es evidente, los movimientos que llevan al exterior del mapa no son válidos.

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

   La función que, siguiendo una estrategia voraz, decide las casillas que componen el camino se debe llamar `mcp_greedy`. Para que esta práctica sea evaluada, es condición necesaria que esté en el código entregado.

   Se deja total libertad para añadir los parámetros que se consideren, y para elegir el tipo de datos de retorno, las estructuras de datos, así como las librerías, los elementos del lenguaje, etc.

[1] Debe tenerse en cuenta que el método voraz no garantiza, para este problema, encontrar la solución óptima.

[2] Definimos la dificultad de un camino como la suma de los grados de dificultad de las casillas que lo componen. Por lo tanto, la dificultad de un camino compuesto por una única casilla (origen y destino coinciden), es la dificultad de esa casilla.

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

   El programa a realizar se debe llamar `mcp_greedy`. La orden tendrá la siguiente sintaxis:

   `mcp_greedy` *`[--p2D]`* `-f` *`fichero_entrada`*

   - La opción `-f`, la única de uso obligado, se utiliza para suministrar el nombre del fichero donde está la instancia del problema a resolver. En el caso de que no se suministre o se produzca algún tipo de error al tratar de abrirlo (no existe, sin permisos para abrirlo, etc.) se advertirá con un mensaje de error. No es necesario controlar posibles errores en el contenido del fichero de entrada ya que siempre se ajustará fielmente al formato establecido, que se describe en el apartado "Entrada al programa".
   - La opción `--p2D` se utilizará para mostrar el camino propuesto.
   - En el caso de que se haga uso de la orden con una sintaxis distinta a la descrita se emitirá un mensaje de error advirtiendo de ello y a continuación se mostrará la sintaxis correcta. Se deben considerar en este caso únicamente las opciones inexistentes; la duplicidad de opciones puede ser ignorada y tratada por tanto como si se hubiera escrito solo una vez. No es necesario indicar todos los errores sintácticos que pueda contener la orden, basta con hacerlo solo con el primer error que se detecte.
   - Para todos los mensajes de error, incluso aquel que informa del uso apropiado de la orden, debe utilizarse la salida estándar de error, es decir, `cerr`.
   - Ante cualquier circunstancia de error en las opciones, el programa deberá terminar advirtiéndolo según se acaba de describir.

3. **Salida del programa y descripción de las opciones:**

   La salida del programa será siempre a la salida estándar,[3] es decir, a la terminal (si no se redirige) y consistirá, en primer lugar y en la primera línea, en sendas dificultades del mejor camino encontrado siguiendo un criterio voraz que recorre el mapa de dos formas distintas:

   a) Avanzando desde el origen hasta el destino.

   b) Retrocediendo desde el destino hasta el origen.

   Debe seguirse ese orden y deben mostrarse separados por un único espacio en blanco, con un único salto de línea inmediatamente después del segundo valor.

   Si se hace uso de la opción `--p2D` (y solo en este caso), se añadirá, a partir de la segunda línea de la salida del programa, un camino cuya dificultad coincida con el menor de los dos valores mostrados en la primera línea. Para ello, se utilizará un formato de dos dimensiones, mostrando una matriz, de la misma dimensión que el mapa de entrada, que contendrá el carácter 'x' para todas las casillas que componen el camino y el carácter '.' para las demás. No debe haber ningún carácter separador entre todos los caracteres de cada fila (es decir, todos los elementos de una misma fila se mostrarán juntos). Al final de cada fila de la matriz debe haber un único salto de línea. A continuación de esta matriz con la que se representa el camino, en la siguiente línea, debe mostrarse su dificultad calculada, exclusivamente, a partir de las celdas que lo componen. (Se trata de comprobar que la dificultad de este camino coincide con el valor devuelto por el algoritmo voraz.)

   Por lo tanto, el formato de la salida que corresponde a la opción `--p2D` es exactamente el mismo que el de la práctica anterior.

[3] Véanse los ejemplos de ficheros de *test* suministrados, junto con las soluciones, para conocer los detalles de la sintaxis de salida.

Al final de cada línea de la salida, sea cual sea, debe haber un único salto de línea, sin espacios en blanco ni tabuladores delante (por lo tanto, al finalizar el programa no se deben mostrar líneas visualmente vacías).

4. **Entrada al programa**

   El mapa $n \times m$ 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 separados mediante un único espacio en blanco.
   - Línea 2 (y siguientes): m números naturales -incluido el 0- que componen la dificultad de la primera fila (y siguientes) del mapa, separados mediante un único espacio en blanco

   por tanto, el fichero contendrá $n + 1$ líneas que finalizarán con un salto de línea, salvo en todo caso, la última línea.

5. **Formato de salida. Ejemplos de ejecución.**

   A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos (`??.map`), junto con las soluciones (`??.map.sol_greedy`) para el caso de que se haga uso de la opción `--p2D`. Es importante tener en cuenta que el hecho de que el programa realizado obtenga la misma salida que la publicada para todos estos ejemplos, no es garantía de que su funcionamiento sea el correcto. Por la otra parte, puede ocurrir que aún mostrándose una salida distinta (incluso en lo que se refiere a la dificultad del camino), ésta sea correcta al tratarse también de una composición voraz en cuanto a las decisiones tomadas. **Es decir, la salida de cada implementación particular no tiene porqué coincidir con la publicada, pero tiene que ser voraz.**

   Es imprescindible ceñirse al formato y texto de salida mostrados, incluso en lo que se refiere a los saltos de línea o carácter separador, que en todos los casos es el espacio en blanco. En ningún caso debe añadirse texto o valores adicionales.

   De entre los ejemplos de entrada publicados, está el fichero `02.map` cuyo contenido es el laberinto (6×5) utilizado en la presentación de este problema. El siguiente sería un ejemplo de ejecución con este fichero:

```
~$mcp_greedy --p2D -f 02.map
14 15
x....
x....
.x...
.x...
.x...
..xxx
14
```

6. **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. En ningún caso debe añadirse texto o valores adicionales. Ten en cuenta que los ficheros de entrada pueden estar nombrados de cualquier otra forma. 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.

a) El programa debe funcionar correctamente, tanto en los valores mostrados como en el camino, con los ejemplos básicos como son `01.map` y `02.map`.

b) El programa debe realizarse en un único archivo fuente con nombre `mcp_greedy.cc`.

c) Se debe entregar únicamente los ficheros `mcp_greedy.cc` y `makefile` (para generar el archivo ejecutable, `mcp_greedy`, 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*).**

d) 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*.[4] Se tratará de evitar también cualquier tipo de aviso (*warning*).

e) 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).

f) 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.**

g) 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.

h) La práctica hay que subirla a *Moodle* respetando las fechas expuestas 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 mediante el compilador *online* de `https://godbolt.org` seleccionando la versión "`x86-64 gcc 9.5`").