ANÁLISIS Y DISEÑO DE ALGORITMOS

Vuelta atrás

Práctica 8 de laboratorio

Entrega: Hasta el domingo 19 de mayo, 23:55h. A través de Moodle

Camino de coste mínimo III

`[ Seguimos con el problema de la obtención del camino de coste mínimo pero en este caso se amplía el conjunto de movimientos válidos. La salida del programa también cambia. ]`

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 (en este caso, el 0 no se incluye entre los posibles valores), el grado de dificultad de visitar esa casilla. Cuanto más elevado es dicho valor más difícil es acceder a esa posición. Por ejemplo:

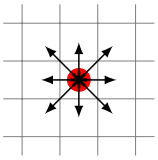
Por ejemplo:

$\xrightarrow{(0,0)}$

4	10	37	10	20	1	1
6	11	4	8	34	30	10
8	26	9	46	32	4	31
28	27	9	42	1	42	7
4	4	29	22	3	38	8
6	10	10	7	22	9	33
3	1	15	37	58	5	55
1	5	26	4	11	56	3
1	8	13	7	9	3	8

$\xrightarrow{(8,6)}$

Un mapa 9 x 7



Camino de coste mínimo con ocho movimientos: Se permite el desplazamiento a cualquier casilla colindante.

En esta nueva práctica se pide aplicar el método de la vuelta atrás (*backtracking*) para obtener la dificultad del camino más favorable[1] que partiendo del origen $(0, 0)$ conduce al destino $(n-1, m-1)$ suponiendo que desde una posición cualquiera (i, j) puede visitarse una casilla cualquiera de entre sus ocho adyacentes (esto es un cambio respecto de las prácticas anteriores de este curso). El punto de partida es la casilla $(0, 0)$ y el de llegada la casilla $(n - 1, m - 1)$.[2]

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

   La función que, siguiendo una estrategia de vuelta atrás, encuentra el camino más corto se debe llamar `mcp_bt`.

[1] 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] Nótese que podría haber más de un camino con la misma dificultad mínima. En tal caso, se pide que se obtenga uno cualquiera de ellos.

Para valorar esta práctica, es imprescindible que esa función esté en el código (con ese mismo nombre).

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

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

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

   `mcp_bt` *`[-p] [--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 de ello 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".
   - Las opciones `-p` y `--p2D` se utilizarán para indicar el camino encontrado; solo se diferencian en la forma de mostrarlo, como se explica más adelante.
   - Las opciones podrán aparecer en cualquier orden y no son excluyentes entre sí.
   - 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 advirtiendo de ello según se acaba de describir.

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

   Si no se hace uso de las opciones `-p` o `--p2D`, el programa mostrará **(únicamente)** los siguientes tres resultados, cada uno en una línea distinta:

   a) Línea 1: La dificultad del camino de salida más corto.

   b) Línea 2: Una estadística sobre los nodos del árbol de búsqueda de vuelta atrás (*backtracking*) que consiste en los siguientes cinco valores:[3]

   $n_{\textbf{visit}}$**:** Número de nodos visitados, es decir nodos que se han considerado tanto para ser explorados, descartados por no ser factibles, o descartados por no ser prometedores.

   $n_{\textbf{explored}}$**:** Número de nodos explorados, es decir, aquellos para los que se hace la llamada recursiva.

   $n_{\textbf{leaf}}$**:** Número de nodos hoja que se visitan.

   $n_{\textbf{unfeasible}}$**:** Número de nodos descartados por no ser factibles.

   $n_{\textbf{not-promising}}$**:** Número de nodos descartados por no ser prometedores.

   El carácter separador debe ser un espacio en blanco y deben mostrarse en ese orden.

[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.

c) Línea 3: El tiempo de CPU, expresado en milisegundos, que se ha consumido para resolver el problema. Si se utiliza un sub-óptimo de partida o se hace una preparación previa de los datos, el tiempo necesario para ello también cuenta en este cómputo.

A partir de la cuarta línea de la salida se mostrarán los resultados de las opciones `-p` o `--p2D`, que servirán para indicar el camino encontrado, de la siguiente forma:

- Opción `--p2D`: Si se hace uso de esta opción (y solo en este caso), se añadirá, a partir de la cuarta línea de la salida del programa, un camino cuya dificultad coincide con el valor mostrado 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 de vuelta atrás.)
- Opción `-p`: En este caso, el camino se muestra codificado mediante una secuencia (continua y sin espacios) de dígitos, cada uno de los cuales representa el movimiento que de forma consecutiva hay que hacer para llegar al destino. Se seguirá esta codificación numérica para cada desplazamiento: 1-norte; 2-noreste; 3-este; 4-sureste; 5-sur; 6-suroeste; 7-oeste; 8-noroeste.

  La secuencia debe comenzar y terminar, respectivamente, por los caracteres "`<`" y "`>`", sin dejar espacios en blanco y se mostrará en la línea inmediatamente siguiente a cualquier salida previa.

  Siguiendo esta codificación, un ejemplo de un camino de dificultad 82 es `<44564324545>`.[4] Notar que el tamaño de la secuencia es una unidad inferior al número de casillas que componen el camino. Por lo tanto, si el camino está compuesto por una única casilla, la secuencia a mostrar es "`<>`".

En el caso de que se utilicen ambas opciones (`-p` y `--p2D`), se mostrará primero el resultado que corresponde a `--p2D`; a continuación, en la siguiente línea pero sin dejar líneas vacías, el de `-p`.

Al finalizar la salida del programa no deben quedar líneas vacías.

4. Entrada al programa

De la misma manera que en las prácticas anteriores, 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 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.

[4] Esta secuencia corresponde a un camino de salida de longitud mínima para el mapa utilizado como ejemplo al comienzo de este enunciado.

5. **Ejemplos de ejecución**

A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos `???.map`, junto con posibles soluciones `???.map.sol_bt` para el caso de que se haga uso de las opciones `-p2D` y `-p`.[5] Es importante tener en cuenta que el hecho de que el programa realizado obtenga la salida correcta para todos estos ejemplos, no es garantía de que la obtenga para otros.

A continuación se muestran posibles formas de utilizar en la terminal la orden descrita y la salida que el programa debe mostrar (en los ejemplos, las salidas `cout` y `cerr` están dirigidas a la terminal —siguiendo el comportamiento predeterminado—).

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.

En ningún caso debe añadirse texto o valores adicionales y se debe seguir el orden especificado.

```
~$mcp_bt -f 002.map
82
7369 948 27 1439 4982
0.051


~$mcp_bt -f 002.map -p
82
7369 948 27 1439 4982
0.051
<44564324545>


~$mcp_bt --p2D -f 002.map
82
7369 948 27 1439 4982
0.051
x......
.x.....
..x....
..x....
.x..x..
..xx.x.
.....x.
......x
......x
82
```

```
~$mcp_bt -p -f 002.map --p2D
82
7369 948 27 1439 4982
0.051
x......
.x.....
..x....
..x....
.x..x..
..xx.x.
.....x.
......x
......x
82
<44564324545>


~$mcp_bt -f 001.map -p  --p2D
10
1 1 1 0 0
0.024
x
10
<>
```

[5] Nótese que las soluciones pueden no ser únicas y por lo tanto, cada una en particular puede no coincidir con la publicada; no por ello es incorrecta.

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

En la evaluación de este trabajo se tendrá en cuenta la calidad, claridad y organización del código fuente, las estructuras de datos utilizadas y los mecanismos de poda implementados para minorar la cantidad de nodos explorados o reducir el tiempo de respuesta.

Así pues, ya que el tiempo de respuesta también se valora, es muy recomendable que incluyas en el archivo `makefile`, la opción del compilador `-O3` (optimizador de código) y que quites cualquier directiva, como `-g`, que pueda incluir, en el programa, información para poder llevar a cabo tareas de depuración en tiempo de ejecución, que lo ralentiza.

7. **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 seguir la estrategia de vuelta atrás y debe funcionar correctamente, tanto en la solución obtenida como en las dos formas de mostrar un camino que corresponde a esa solución, con los ejemplos básicos como son {`001 002 003 004`}`.map`.

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

c) Se debe entregar únicamente los ficheros `mcp_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 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*.[6] 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.

[6] 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`).