ANÁLISIS Y DISEÑO DE ALGORITMOS

Vuelta atrás

Práctica 8 de laboratorio

Entrega: Hasta el domingo 18 de mayo, 23:55h. A través de Moodle

El problema del laberinto III

`[ Seguimos con el problema del laberinto codificado en un fichero de la misma manera que en las prácticas anteriores, pero en este caso se amplía el conjunto de movimientos válidos (problema general). La salida del programa también cambia.]`

Se dispone de una cuadrícula $n \times m$ de valores $\{0, 1\}$ que representa un laberinto. Un valor 0 en una casilla cualquiera de la cuadrícula indica una posición inaccesible; por el contrario, con el valor 1 se simbolizan las casillas accesibles. Por ejemplo:

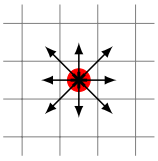
Por ejemplo:

$\xrightarrow{(0,0)}$

1	1	0	1	1	0	1
1	0	1	1	0	0	1
1	0	1	0	0	1	0
0	0	1	0	1	0	1
1	1	0	0	1	0	1
1	0	1	1	0	1	0
1	0	0	0	0	1	0
0	1	0	1	1	0	1
0	1	0	1	1	1	1

$\xrightarrow{(8,6)}$

Un laberinto 9 x 7



El laberinto con ocho movimientos: Se permite visitar cualquier casilla colindante siempre que sea accesible.

En esta nueva práctica se pide aplicar el método de la vuelta atrás (*backtracking*) para obtener la longitud[1] del camino más corto (y un camino con esa longitud) que conduzca a la salida del laberinto suponiendo que desde una posición cualquiera (i, j) puede visitarse una casilla cualquiera de entre sus ocho adyacentes siempre que sea accesible. El punto de partida es la casilla $(0, 0)$ y el de llegada la casilla $(n - 1, m - 1)$.

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

  La función que, siguiendo una estrategia de vuelta atrás, encuentra el camino más corto (o resuelve que el laberinto no tiene salida) se debe llamar `maze_bt`.

  Si esa función no está en el código (o está con otro nombre) entonces la entrega no se dará por válida.

  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.

[1] Definimos *longitud del camino* como el número de casillas que lo componen. Un camino con origen y destino en la misma casilla tiene longitud 1.

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

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

  `maze_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 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 más adelante se explica.
  - 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 advirtiéndolo 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.

- **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:

  1. Línea 1: La longitud del camino de salida más corto. En el caso de que el laberinto no tenga salida se mostrará el valor 0[2].
  2. Línea 2: 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.
  3. 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:

[2] El caso de un laberinto sin entrada (casilla $(0, 0)$ cerrada) se considera también un laberinto sin salida.

[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.

- Opción `--p2D`: El camino se muestra superpuesto al laberinto de entrada, mediante el carácter '*'. Las demás casillas se dejarán con su valor original. No debe haber ningún carácter separador entre todos los valores 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. (hasta aquí, exactamente igual que en la práctica anterior). No obstante, si el laberinto no tiene salida, se mostrará únicamente el valor 0 (en este caso no hay que mostrar el laberinto).
- 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 a la salida del laberinto. 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.

  Siguiendo esta codificación, un ejemplo de un camino de coste 13 es <345564324545>[4]. Notar que el tamaño de la secuencia es una unidad inferior a la longitud del camino. Por lo tanto, si el camino esta compuesto por una única casilla, la secuencia a mostrar es <>.

  Si el laberinto no tiene salida (o su entrada está cerrada) se mostrará <0> y nada más.

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`.

- **Entrada al programa**

  El laberinto $n \times m$ 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 valores {0,1} que componen la primera fila (y siguientes) del laberinto, separados mediante un único espacio en blanco.

  Por lo tanto, el fichero contendrá $n + 1$ líneas que finalizarán, todas ellas, con un salto de línea.

- **Ejemplos de ejecución**

  A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos (??-bt.maze), junto con posibles soluciones (??-bt.maze.sol_bt) para el caso de que se haga uso de las opciones `-p2D` y `-p`[5]. Es importante advertir 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.

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

[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.

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

```
$maze_bt -f 02-bt.maze
13
496 64 2 323 110
0.013

$maze_bt -f 02-bt.maze -p
13
496 64 2 323 110
0.013
<345564324545>

$maze_bt --p2D -f 02-bt.maze
13
496 64 2 323 110
0.013
**01101
10*1001
10*0010
00*0101
1*00*01
10**0*0
10000*0
010110*
010111*
```

```
$maze_bt -p -f 02-bt.maze --p2D
13
496 64 2 323 110
0.013
**01101
10*1001
10*0010
00*0101
1*00*01
10**0*0
10000*0
010110*
010111*
<345564324545>

$maze_bt -f 01-bt.maze -p  --p2D
1
1 1 1 0 0
0.003
*
<>

$maze_bt -f 06-bt.maze -p  --p2D
0
1 0 0 0 0
0.001
0
<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 procedimiento de vuelta atrásdebe estar implementado en la función `maze_bt`. Puede ser una función o un método de clase, es indiferente.
2. El programa debe realizarse en un único archivo fuente con nombre `maze_bt.cc`.
3. Se debe entregar únicamente los ficheros `maze_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).**
4. 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*).
5. 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).
6. 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.**
7. 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.
8. 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`).