ANÁLISIS Y DISEÑO DE ALGORITMOS

Método voraz

Práctica 7 de laboratorio

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

El problema del laberinto 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 o que no encuentre camino de salida aunque exista.]`

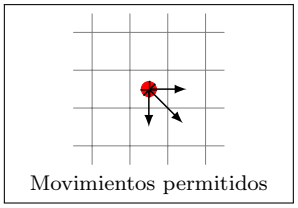
Se dispone de una cuadrícula $n \times m$ de valores $\{0, 1\}$ que representa un laberinto. Un valor 0 en una casilla cualquiera de la cuadrícula indica una posición inaccesible; por el contrario, con el valor 1 se simbolizan las casillas accesibles. Por ejemplo:

$\xrightarrow{(0,0)}$

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

$\xrightarrow{(5,4)}$

Un laberinto 6 x 5



Se pide, aplicar el método voraz[1] para tratar de obtener la longitud[2] del camino más corto 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, tampoco son válidos los movimientos que llevan al exterior del laberinto ni los que conducen a casillas inaccesibles.

- **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 `maze_greedy`.

  Si esa función no está en el código (o está con otro nombre) entonces se considerará que no ha sido desarrollada, con la consiguiente merma en la calificación de esta práctica.

[1] Debe tenerse en cuenta que el método voraz no garantiza, para este problema, encontrar la solución óptima; ni siquiera encontrar un camino de salida aunque exista.

[2] 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.

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.

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

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

  `maze_greedy` *`[-p]`* `-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 `-p` se utilizará para mostrar el camino propuesto.
  - Ambas opciones podrán aparecer en cualquier orden.
  - 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.

- **Salida del programa y descripción de las opciones:**

  Si no se hace uso de la opción `-p`, el programa mostrará, en la primera línea de la salida, la longitud del mejor camino encontrado siguiendo un criterio voraz de decisiones. En el caso de que este criterio no conduzca a la salida (o que no exista camino de salida) se mostrará el valor 0.

  Si se hace uso de la opción `-p`, se añadirá a la salida anterior (a partir de la segunda línea), el camino recorrido 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, deberá mostrarse el camino recorrido aunque no se haya podido completar el camino hasta la salida (véanse los ejemplos publicados).

- **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.

A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos (`??.maze`), junto con posibles soluciones (`??.maze.sol_greedy`) para el caso de que se haga uso de la opción `-p`.[3] Es importante saber 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.

- **Formato de salida. Ejemplos de ejecución.**

  De entre los ejemplos de entrada publicados, está el fichero 02.maze cuyo contenido es el laberinto (6×5) utilizado en la presentación de este problema, que no tiene solución voraz; y el fichero 11.maze, cuyo contenido es un laberinto (6×6) que tiene solución voraz.

  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 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.

```
$maze_greedy -f 02.maze
0

$maze_greedy -p -f 02.maze
0
*1001
1*111
01*00
110**
11100
00011


$maze_greedy -f 11.maze
8


$maze_greedy -f 11.maze -p
8
*01111
*01111
*00001
1*1111
11*111
111***
```

```
*ALGUNOS EJEMPLOS DE SITUACIONES DE ERROR*

$maze_greedy -f
ERROR: missing filename.
Usage:
maze_greedy  [-p] -f file


$maze_greedy
Usage:
maze  [-p]  -f file


$maze_greedy -f -f
ERROR: can't open file: -f.
Usage:
maze_greedy  [-p] -f file


$maze_greedy -f 0.problem -p -a -b
ERROR: unknown option -a.
Usage:
maze_greedy  [-p] -f file


$maze_greedy -p -f anonexistentfile
ERROR: can't open file: anonexistentfile.
Usage:
maze_greedy  [-p] -f file
```

[3] 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.

- **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 `maze_greedy.cc`.
2. Se debe entregar únicamente los ficheros `maze_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*.[4] 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.

[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`).