ANÁLISIS Y DISEÑO DE ALGORITMOS

Práctica Final de laboratorio

Periodo de entrega: del 11 al 14 de mayo de 2015
(en la sesión de laboratorio que corresponde a cada alumno)

El problema del laberinto (IV)

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:

$$\xrightarrow{(1,1)} \begin{array}{ccccc} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array} \xrightarrow{(6,5)}$$

Se pide, aplicar el **método *Ramificación Poda*** para encontrar el camino de longitud mínima[1] que conduzca a la salida del laberinto. El punto de partida es la casilla $(1, 1)$ y el de llegada la casilla (n, m). Sólo son validos tres tipos de 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.

Entrada al programa:

Una vez más, la entrada al programa se realiza mediante un fichero de texto suministrado a través de la línea de comando. Sin embargo, a diferencia de la práctica anterior en este fichero podría haber codificado más de un laberinto (véanse los ficheros de test publicados en el Campus Virtual).

La forma de codificar cada uno de los laberintos es la misma que ya se ha utilizado en prácticas anteriores.

[1] Se entiende por longitud del camino el número de casillas que lo componen.

Salida del programa:

El programa mostrará una línea por cada laberinto que se encuentre en el fichero: contendrá la longitud del camino mínimo encontrado (sin valor si no existe camino de salida); el número de iteraciones efectuadas por el bucle que explora cada uno de los elementos de la "lista de nodos vivos" y el tiempo en segundos que tarda en procesar cada uno de los laberintos del fichero de entrada (haciendo uso de la función `tiempo()` que ya se empleó en las primeras prácticas)

Por ejemplo, si ejecutamos `maze mazes-1.txt`, escribirá:

```
Maze 0: Lenght best path= 16. Iterations= 1. Time= 0
Maze 1: Lenght best path= 28. Iterations= 45046732. Time= 22.6614
Maze 2: Lenght best path= 21. Iterations= 1. Time= 0
Maze 3: Lenght best path= 30. Iterations= 25956. Time= 0.032002
Maze 4: Lenght best path= 31. Iterations= 13. Time= 0
Maze 5: Lenght best path= 20. Iterations= 28. Time= 0
Maze 6: Lenght best path= 20. Iterations= 1. Time= 0
Maze 7: Lenght best path= 21. Iterations= 3. Time= 0
Maze 8: Lenght best path= 34. Iterations= 1565. Time= 0.004
Maze 9: Lenght best path= 41. Iterations= 3329 Time= 0.012001
```

Como es lógico, las iteraciones y los tiempos de proceso no tienen porqué coincidir ya que dependen, entre otras cosas, de mecanismos de poda y estrategias de búsqueda utilizados.

Puntuación de la práctica:

De 0 a 1 punto sobre la nota final de la asignatura. Para valorar esta práctica es imprescindible que obtenga la solución correcta de alguno de los ficheros de test publicados.

Se valorará especialmente:

1. La eficiencia de las estructuras de datos empleadas, se recomienda el uso de la *Standard Template Library* de C++.
2. La función (o funciones) de cota superior e inferior utilizadas.
3. El uso de algún heurístico voraz como subóptimo de partida.
4. La realización de un estudio comparativo, en cuanto a eficiencia, de distintas estrategias de búsqueda y de la necesidad de utilizar un subóptimo de partida y funciones de cota: tiempo de ejecución, número de nodos explorados, etc.
5. La documentación que explique y justifique las estructuras de datos utilizadas y los mecanismos de poda empleados.
6. La defensa, ante el profesor, del trabajo realizado.