ANÁLISIS Y DISEÑO DE ALGORITMOS

Programación dinámica

Práctica 6 de laboratorio

Esta práctica ocupa dos sesiones:

Sesión 1: Versión recursiva sin almacén y gestión de argumentos.
Sesión 2: Práctica completa.

Entrega: Respectivamente, hasta los días 31 de marzo y 7 de abril, 23:55h.
A través de Moodle

El problema del laberinto

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:

$$\xrightarrow{(0,0)} \begin{matrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{matrix} \xrightarrow{(5,4)}$$

Se pide aplicar el método *divide y vencerás* y su extensión a *programación dinámica* para calcular el número de caminos distintos[1] que hay desde el origen $(0, 0)$ hasta el destino $(n - 1, m - 1)$ asumiendo que sólo son válidos 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.

Para resolver este ejercicio se debe implementar los siguientes algoritmos:

1. Recursivo sin almacén (ineficiente)
2. Recursivo con almacén (memoización)
3. Iterativo con almacén (tabla)
4. Iterativo con almacén (vector —versión con complejidad espacial mejorada—)
5. Además deberá mostrarse un camino de salida cualquiera.

[1] Dos caminos, con el mismo origen y destino, son distintos si varían en al menos una de las casillas intermedias que lo componen.

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

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

   maze [-t] [-p] [--ignore-recursive] -f `fichero_entrada`

   En el siguiente apartado se describe el significado de las opciones.

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

   En todas las formas posibles de utilizar la mencionada orden, la salida del programa (véase los ejemplos a continuación) será, en primer lugar, la solución obtenida mediante los cuatros algoritmos anteriormente citados: recursivo sin almacén, memoización, iterativo con tabla e iterativo con complejidad espacial mejorada[2] (debe seguirse este orden). No obstante, si se incorpora a la orden la opción [`--ignore-recursive`] se deberá excluir de la salida del programa la solución recursiva sin almacén.[3] En cuanto al resto de opciones:

   - Si se hace uso de la opción [-t] se mostrará, a continuación de lo descrito anteriormente, las tablas que almacenan los resultados intermedios en las versiones memoización e iterativa con matriz (en este orden). Asimismo, para los subproblemas que no han sido resueltos en la versión recursiva se mostrará el carácter guión ('-').
   - Si se hace uso de la opción [-p] se mostrará además un camino cualquiera que conduzca a la casilla $(n-1, m-1)$ desde la casilla $(0, 0)$. Si no existe ningún camino se imprimirá el literal **"NO EXIT"**.
   - Las opciones no son excluyentes entre sí, ninguna de ellas. Además podrán aparecer en cualquier orden.
   - Si se hace uso simultáneo de las opciones [-p] y [-t] (da igual el orden) se mostrará primero las tablas y a continuación el camino, siguiendo las pautas descritas anteriormente.
   - En cualquiera de las formas posibles de utilizar la orden, si se incorpora la opción [--ignore-recursive] se debe excluir (únicamente) la solución recursiva sin almacén.
   - La opción -f, la única de uso obligado, se utiliza para suministrar el nombre del fichero donde está el laberinto a resolver (véase siguiente apartado). En el caso de que se suministre un fichero inexistente 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.
   - 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. Considerar en este caso únicamente las opciones inexistentes; la duplicidad de opciones puede ser ignorada y tratada por tanto como si se hubiera escrito sólo una vez. No es necesario indicar todos los errores sintácticos que pueda contener la orden, basta con hacerlo sólo con el primero que se detecte.

3. **Entrada de datos al programa. Formato del fichero de entrada.**

   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.

[2] En la versión iterativa con complejidad espacial mejorada puede utilizarse, en lugar de una matriz, o bien dos vectores o bien un único vector y una variable adicional, lo que se prefiera, lo importante es que se reduzca la complejidad espacial de $\Theta(n \times m)$ a $\Theta(m)$.

[3] Por su elevado coste computacional (evidentemente el programa tampoco deberá hacer la llamada a esta función). En este caso la solución de los otros tres algoritmos sí deberá mostrarse.

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

A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos (??.maze), junto con las soluciones para el caso de que se haga uso de todas las opciones (??.maze.sol). Entre ellos está `00.maze` y `02.maze` utilizados a continuación para describir el formato de la salida.

4. **Formato de salida. Ejemplos de ejecución.**

Sea el fichero 02.maze cuyo contenido es el laberinto (6×5) mostrado anteriormente y el fichero 00.maze cuyo contenido es un laberinto (1×1) con la casilla origen (y también destino) inaccesible.

A continuación se muestra posibles formas de utilizar la orden descrita y la salida que el programa debe mostrar en la terminal. 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. Si fuera el caso, antes de mostrar sendos almacenes de resultados parciales o el camino se debe incorporar una línea vacía. No debe haber más líneas vacías, es decir, a lo sumo estas tres. La última línea debe terminar con un salto de línea (y sólo uno). En ningún caso debe añadirse texto o valores adicionales.

$maze -f 02.maze
```
Recursive: 8
Memoization: 8
Iterative: 8
Iterative (vector): 8
```

$maze -p --ignore-recursive -f 02.maze
```
Memoization: 8
Iterative: 8
Iterative (vector): 8

A possible path:
(0,0)(1,0)(2,1)(3,1)(4,2)(5,3)(5,4)
```

$maze -t --ignore-recursive -p -f 02.maze
```
Memoization: 8
Iterative: 8
Iterative (vector): 8

Memoization table:
1 1 - - -
1 3 - - -
0 4 - - -
0 4 0 - -
0 4 8 0 0
- - 0 8 8

Iterative table:
1  1  0  0  0
1  3  4  4  4
0  4 11  0  0
0  4  0 11 11
0  4  8  0  0
0  0  0  8  8

A possible path:
(0,0)(1,0)(2,1)(3,1)(4,2)(5,3)(5,4)
```

$maze -f -t -p
```
ERROR: can't open file: -t.
Usage:
maze [-p] [-t] [--ignore-recursive] -f file
```

$maze -f 00.maze
```
Recursive: 0
Memoization: 0
Iterative: 0
Iterative (vector): 0
```

$maze -p -f 00.maze --ignore-recursive -p
```
Memoization: 0
Iterative: 0
Iterative (vector): 0

A possible path:
NO EXIT
```

$maze -f 00.maze -t -t -t
```
Recursive: 0
Memoization: 0
Iterative: 0
Iterative (vector): 0

Memoization table:
0

Iterative table:
0
```

$maze -f 00.maze -t -a
```
ERROR: unknown option -a.
Usage:
maze [-p] [-t] [--ignore-recursive] -f file
```

$maze -f milaberinto -t -p
```
ERROR: can't open file: milaberinto.
Usage:
maze [-p] [-t] [--ignore-recursive] -f file
```

$maze -a -b -p -f esteficheronoexiste
```
ERROR: unknown option -a.
Usage:
maze [-p] [-t] [--ignore-recursive] -f file
```

Normas para la entrega.

ATENCIÓN: Estas normas son de obligado cumplimiento para que esta práctica sea evaluada.

a) Se debe entregar el código fuente y un *makefile* para obtener el ejecutable. No hay que entregar nada más, en ningún caso se entregarán ficheros de test.

b) 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*).

c) Todos los ficheros que se entregan deben contener el nombre del autor y su DNI (o NIE) en su primera línea (entre comentarios para que no afecte a la compilación).

d) 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.**

e) En el archivo comprimido **no deben existir subcarpetas**, es decir, al extraer sus archivos estos deben quedar guardados en la misma carpeta donde está el archivo que los contiene.

f) La práctica hay que subirla a *Moodle* respetando las fechas expuestas en el encabezado de este enunciado.

g) En la entrega correspondiente a la primera sesión sólo es necesario que esté implementada la versión recursiva sin almacén (aunque no hay inconveniente si se realiza algo más). Sin embargo, la gestión de opciones del programa debe estar hecha en su totalidad. El resultado que se debe mostrar, en cualquiera de las dos entregas, para los casos aún sin hacer es "¿?". Por ejemplo:

```
$maze -t -p -f 02.maze
Recursive: 8
Memoization: ¿?
Iterative: ¿?
Iterative (vector): ¿?

Memoization table:
¿?

Iterative table:
¿?

A possible path:
¿?
```

[4] Si trabajas con tu propio ordenador o con otro sistema operativo asegúrate de que este requisito se cumple.