ANÁLISIS Y DISEÑO DE ALGORITMOS

Práctica 6 de laboratorio

Esta práctica ocupa dos semanas:

Semana 1: Versión recursiva sin almacén.
Semana 2: Práctica completa.

Entrega: Respectivamente, hasta los días 19 de marzo y 2 de abril, 23:55h.
A través de Moodle

El senderista perezoso

Un senderista está planificando una ruta de montaña haciendo uso de un plano de coordenadas $n \times m$. En cada posición (i, j) de dicho plano se representa, mediante un número natural, el grado de dificultad de visitar esa casilla. Cuanto más elevado es dicho valor más difícil es acceder a esa posición. Por ejemplo:

$\xrightarrow{(0,0)}$

1	3	5	1	1
2	4	6	3	1
1	2	9	7	1
9	1	7	1	9
1	3	7	5	1
8	1	2	2	1

$\xrightarrow{(5,4)}$

Se pide, aplicar el método *divide y vencerás* y su extensión a *programación dinámica* para obtener la dificultad del camino más favorable[1] que hay desde la casilla origen $(0, 0)$ hasta la casilla 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, los movimientos que llevan al exterior del mapa no son válidos.

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 (en este caso la solución puede no ser única) cuya dificultad coincida con la de la solución obtenida en cualquiera de los cuatro algoritmos enumerados (como es lógico, las cuatro soluciones coincidirán).

[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.

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

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

   caminante [-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. (debe seguirse este orden). No obstante, si se incorpora a la orden la opción [--ignore-recursive] se deberá excluir de la salida la solución recursiva sin almacén, 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 que deberá mostrarse. En cuanto al resto de opciones:

   - Si se hace uso de la opción [-t] se mostrará además la matriz obtenida en la versión iterativa (donde se almacenan los resultados intermedios).
   - Si se hace uso de la opción [-p] se mostrará además un camino cuya dificultad coincida con la solución obtenida y, a continuación, la dificultad de este camino obtenida de forma independiente, es decir, sumando los grados de dificultad de las casillas que lo componen.
   - Las opciones no son excluyentes entre sí, ninguna de ellas. Además pueden aparecer en cualquier orden.
   - Si se hace uso simultáneo de las opciones [-p] y [-t] (da igual el orden) se mostrará primero la tabla y a continuación el camino seguido de su dificultad según se ha descrito 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 mapa a resolver (véase siguiente apartado)

3. **Entrada al programa**

   El mapa 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.

   A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos, entre ellos está `1.map` y `2.map` utilizados a continuación para describir el formato de la salida.

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

   Sea el fichero `2.map`, se trata del mapa (6×5) mostrado anteriormente.

   En la página siguiente se muestra posibles formas de utilizar en la terminal la orden descrita y a continuación la salida que el programa debe mostrar. 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. Antes de mostrar la tabla iterativa o el camino se debe incorporar una línea vacía. No debe haber más líneas vacías, es decir, a lo sumo estas dos. La última línea debe terminar con un salto de línea (y sólo uno). En ningún caso debe añadirse texto o valores adicionales.

$caminante -f 2.map
```
Recursive: 13
Memoization: 13
Iterative: 13
Iterative (vectors): 13
```

$caminante --ignore-recursive -f 2.map
```
Memoization: 13
Iterative: 13
Iterative (vectors): 13
```

$caminante -t -f 2.map
```
Recursive: 13
Memoization: 13
Iterative: 13
Iterative (vectors): 13

Iterative table:
1 4 9 10 11
3 5 10 12 11
4 5 14 17 12
13 5 12 13 21
14 8 12 17 14
22 9 10 12 13
```

$caminante -p -f 2.map
```
Recursive: 13
Memoization: 13
Iterative: 13
Iterative (vectors): 13

A possible path:
(0,0)(1,0)(2,0)(3,1)(4,1)(5,2)(5,3)(5,4)
weight: 13
```

$caminante -p -t -f 2.map --ignore-recursive
```
Memoization: 13
Iterative: 13
Iterative (vectors): 13

Iterative table:
1 4 9 10 11
3 5 10 12 11
4 5 14 17 12
13 5 12 13 21
14 8 12 17 14
22 9 10 12 13

A possible path:
(0,0)(1,0)(2,0)(3,1)(4,1)(5,2)(5,3)(5,4)
weight: 13
```

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

$caminante -f 1.map
```
Recursive: 10
Memoization: 10
Iterative: 10
Iterative (vectors): 10
```

$caminante -f 1.map --ignore-recursive
```
Memoization: 10
Iterative: 10
Iterative (vectors): 10
```

$caminante -f 1.map -t
```
Recursive: 10
Memoization: 10
Iterative: 10
Iterative (vectors): 10

Iterative table:
10
```

$caminante -f 1.map -p
```
Recursive: 10
Memoization: 10
Iterative: 10
Iterative (vectors): 10

A possible path:
(0,0)
weight: 10
```

$caminante -t -p -f 1.map
```
Recursive: 10
Memoization: 10
Iterative: 10
Iterative (vectors): 10

Iterative table:
10

A possible path:
(0,0)
weight: 10
```

$caminante -f 1.map -t -a
```
ERROR: unknown option -a.
Usage:
caminante [-p] [-t] [--ignore-recursive] -f file
```

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

5. **Normas para la entrega.**

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) 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).

c) Se comprimirán en un archivo *.tar.gz* o equivalente cuyo nombre será el DNI (o NIE) del alumno. Por ejemplo: 123456789A.tgz

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

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

f) En la entrega correspondiente a la primera semana sólo es necesario que esté implementada la versión recursiva sin almacén. Sin embargo, la gestión de opciones del programa debe estar hecha en su totalidad. El resultado que se debe mostrar para los casos aún sin hacer es "¿?". Por ejemplo:

$caminante -t -p -f 1.map

```
Recursive: 10
Memoization: ¿?
Iterative: ¿?
Iterative (vectors): ¿?

Iterative table:
¿?

A possible path:
¿?
weight: ¿?
```