ANÁLISIS Y DISEÑO DE ALGORITMOS

Ramificación y poda

Práctica Final

Entrega: Hasta el domingo 1 de junio, 23:55h. A través de Moodle

Camino de coste mínimo y IV

`[ El enunciado del problema es idéntico a la práctica 8, lo que cambia es la estrategia de resolución y algunos datos a mostrar. En esta práctica, además, hay que entregar una memoria que explique el trabajo realizado.]`

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:

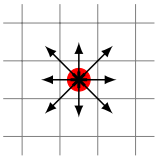
Por ejemplo:

$\xrightarrow{(0,0)}$

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 ramificación y poda (*branch & bound*) 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 $(n - 1, m - 1)$.

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_bb`.

   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.

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

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

`maze_bb` *`[-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 de ello 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 se explica más adelante.
- 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 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 advirtiendo de ello según se acaba de describir.

3. **Salida del programa y descripción de las opciones**

`[ La salida, en todas sus variantes, es casi igual a la práctica 8: solo cambia la línea 2, en la que se muestran 8 valores estadísticos en lugar de 5. ]`

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:

- 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].
- Línea 2: Una cuantificación de los distintos tipos de nodos encontrados durante el proceso de búsqueda de la solución óptima. Se mostrará los siguientes ocho valores, **en este orden** y separados mediante un espacio en blanco[3]:

  $n_{\textbf{visit}}$**:** Número de nodos visitados, es decir nodos que se han considerado tanto para ser explorados, descartados por no ser factibles, o descartados por no ser prometedores.

  $n_{\textbf{explored}}$**:** Número de nodos explorados, es decir, aquellos que se introducen en la *lista de nodos vivos*.

  $n_{\textbf{leaf}}$**:** Número de nodos hoja que se visitan.

  $n_{\textbf{unfeasible}}$**:** Número de nodos descartados por no ser factibles.

  $n_{\textbf{not-promising}}$**:** Número de nodos descartados por no ser prometedores.

[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.

$n_{\textbf{promising-but-discarded}}$: Número de nodos que fueron prometedores pero que finalmente se descartaron.

$n_{\textbf{best-solution-updated-from-leafs}}$: Número de veces que la mejor solución hasta el momento se ha actualizado a partir de un nodo completado.

$n_{\textbf{best-solution-updated-from-pessimistic-bound}}$: Número de veces que la mejor solución hasta el momento se ha actualizado a partir de la cota pesimista de un nodo sin completar.

- 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 (nada cambia con respecto a la práctica anterior):

- 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. 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`.

Para mostrar el camino en cualquiera de sus formas, no debe utilizarse ningún carácter adicional a lo estrictamente necesario para cumplir con el formato descrito; tampoco espacios en blanco o cualquier otro delimitador, ni siquiera al comienzo o al final de una línea. Tras la salida del programa no deben quedar líneas vacías. Tras la última línea debe haber un salto de línea.

4. Entrada al programa

De la misma manera que en las prácticas anteriores, 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:

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

- 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 (y siguientes) fila del laberinto, 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.

5. **Ejemplos de ejecución**

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

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

   A continuación se muestra una posible forma de utilizar en la terminal, la orden descrita y la salida que el programa debe mostrar, que siempre debe realizarse mediante `cout` o `cerr`, según corresponda.

```
~$maze_bb -p -f 02-bb.maze --p2D
13
121 15 0 80 26 0 0 1
0.009
**01101
10*1001
10*0010
00*0101
1*00*01
10**0*0
10000*0
010110*
010111*
<345564324545>
```

```
~$maze_bb -f 00-bb.maze --p2D -p
0
1 0 0 0 0 0 0 0
0.001
0
<0>
```

[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.

6. **Documentación del trabajo realizado.**
   Se debe entregar una **memoria**, en formato PDF, con nombre `memoria.pdf`, que describa las estructuras de datos empleadas así como la/s estrategia/s de búsqueda, los mecanismos de poda y las cotas optimistas y pesimistas utilizadas.

   Incluirá también un análisis del comportamiento de distintas estrategias[6] de búsqueda combinándolas, si se considera apropiado, con distintos mecanismos de poda. Para ello, se realizará un cuadro comparativo que contenga, para cada caso estudiado, el número de nodos que se añaden a la lista de nodos vivos y el tiempo de respuesta del algoritmo (en las diapositivas de teoría se muestran cuadros comparativos que pueden servirte de guía).

   La primera página de la memoria contendrá el nombre del autor y su DNI. El resto se estructurará en seis apartados de la siguiente manera: de

   1. **Estructuras de datos**
      1.1 **Nodo**
      En este apartado se describirá brevemente el contenido del nodo, es decir, las variables que lo componen y su cometido.
      1.2 **Lista de nodos vivos**
      Breve descripción de la estructura de datos utilizada y si procede, qué criterios se han analizado para extraer el nodo más prometedor (El programa debe utilizar el que se considere más rápido pero todos los demás se mencionarán también en este apartado).
   2. **Mecanismos de poda**
      2.1. **Poda de nodos no factibles**
      Se deberá indicdear cómo se descartan los nodos que no son factibles; si no procede este tipo de poda debe indicarse el motivo.
      2.2. **Poda de nodos no prometedores**
      Se deberá indicar cómo se decide que un nodo no es prometedor; si no procede debe indicarse el motivo.
   3. **Cotas pesimistas y optimistas**
      Se describirá en qué consisten dichas cotas, distinguiéndose, en el caso de la pesimista, entre el nodo inicial y los demás nodos.
      3.1 **Cota pesimista inicial (inicialización).**
      3.2 **Cota pesimista del resto de nodos.**
      3.3 **Cota optimista.**
   4. **Otros medios empleados para acelerar la búsqueda**
      En este apartado se citará cualquier otro mecanismo utilizado con el objetivo de acelerar la búsqueda o cualquier otro aspecto que se quiera reflejar en la memoria y que no se adapta a ninguno de los apartados anteriores.
   5. **Estudio comparativo de distintas estrategias de búsqueda**
      En este apartado se incluirá el cuadro comparativo mencionado anteriormente junto con una breve discusión de los resultados obtenidos.
   6. **Tiempos de ejecución**.
      Se mostrará en este apartado el tiempo de proceso, en milisegundos, requerido para resolver los problemas de test que se relacionan a continuación, escogidos de entre los

[6] Según se ha visto en las clases de teoría, se entiende por estrategia de búsqueda, una forma concreta de priorizar las extracciones de la lista de nodos vivos; *Backtraking* no sirve como estrategia de búsqueda.

publicados. El tiempo de respuesta no debe ser superior a 15 segundos (15.000 milisegundos). Si alguno de ellos no ha podido ser resuelto (de forma correcta) o su tiempo excede ese límite, se escribirá el carácter '?' en lugar del valor numérico. Por ejemplo[7]:

Fichero de test	Tiempo (ms)
100-bb.maze	0.49
200-bb.maze	1.00
300-bb.maze	6.75
500-bb.maze	11.83
700-bb.maze	22.80
900-bb.maze	39.01
k01-bb.maze	30.56
k02-bb.maze	?
k03-bb.maze	207.90
k05-bb.maze	632.52
k10-bb.maze	?

IMPORTANTE: Todos los apartados (salvo el último) se complementarán con el correspondiente fragmento de código que corrobore el trabajo descrito. Si algunos de los apartados no ha sido realizado se pondrá únicamente la frase "NO IMPLEMENTADO".

De los distintos elementos que puede contener un algoritmo de ramificación y poda, sólo se evaluarán los que estén reflejados en esta memoria. Por lo tanto, todo lo que no esté en dicho documento se entenderá que no ha sido realizado.

7. Sobre la evaluación de esta práctica.

En la evaluación de este trabajo se tendrá en cuenta la calidad, claridad y organización del código fuente, las estructuras de datos utilizadas, el análisis de distintas estrategias de búsqueda y los mecanismos de poda implementados para minorar la cantidad de nodos explorados o reducir el tiempo de respuesta.

No obstante, para que esta práctica sea evaluada es imprescindible que se siga una estrategia de ramificación y poda y que el programa resuelva algunos ejemplos básicos, donde el tiempo de ejecución es irrelevante.

Otro aspecto muy relevante en la nota final será el tiempo de respuesta[8], por lo tanto, aunque se hayan implementado distintas estrategias de búsqueda, podas y funciones de cota, etc, **el programa que se entrega debe funcionar con la configuración que se entienda más rápida** (en la memoria se deberá hacer mención a cualquier otro trabajo realizado que, por ese motivo, no se haya incluido en la versión final del programa, aunque si este es el caso, también debe aparecer comentado en el código para que se tenga en cuenta y se valore).

Así pues, ya que el tiempo de respuesta también se valora, es muy recomendable que incluyas en el archivo `makefile`, la opción del compilador `-O3` (optimizador de código) y que quites cualquier directiva, como `-g`, que añade al programa el código necesario para poder llevar a cabo tareas de depuración en tiempo de ejecución, que lo ralentiza.

[7] Tiempos en milisegundos con dos decimales. Se han obtenido en los ordenadores de las aulas de la EPS-IV.

[8] El tiempo se tomará desde que se lanza el programa en la terminal, hasta que termina; por lo tanto, la entrada/salida también computa en este cálculo.

8. **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.

a) El programa debe seguir el esquema de ramificación y poda y debe funcionar correctamente, tanto en la solución obtenida como en las dos formas de mostrar un camino que corresponde a esa solución, con los ejemplos básicos como son {`01 02 03 04 05`}`-bb.maze`.

b) El programa debe realizarse en un único archivo fuente con nombre `maze_bb.cc`.

c) Se debe entregar únicamente los ficheros `memoria.pdf`; `maze_bb.cc` y `makefile` (para generar el archivo ejecutable a través de la orden `make` sin añadir nada más). **Sigue escrupulosamente los nombres de ficheros que se citan en este enunciado. Solo hay que entregar esos tres archivos (en ningún caso se entregarán archivos de *test*).**

d) Es imprescindible que no presente errores ni de compilación ni de interpretación (según corresponda), en los ordenadores del laboratorio asignado.[9] Se tratará de evitar también cualquier tipo de *warning*.

e) 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).

f) 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.**

g) 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.

h) La práctica hay que subirla a *Moodle* respetando la fecha expuesta en el encabezado de este enunciado.

[9] Si trabajas con tu propio ordenador o con otro sistema operativo asegúrate de que este requisito se cumple (puedes comprobarlo con la versión "`x86-64 gcc 9.5`"del compilador *online* de `https://godbolt.org`).