EI1022/MT1022 - Algoritmia

Entregable 1 - Fecha de entrega: lunes 16 de octubre de 2017

Trabajamos para una revista de puzles y nos han encargado generar un nuevo tipo de puzle de tipo 'laberinto', algo más difíciles. Nos piden que desarrollemos un programa que, dado un laberinto cualquiera generado mediante un MFSET, encuentre las dos celdas más alejadas entre sí. Utilizaremos dichas celdas como 'punto de partida' y 'punto de llegada' del laberinto que publicará la revista.

De los dos puntos que encuentra el programa, elegiremos como 'punto de partida' el que tenga menor número de fila. Si aún así persiste el empate deberá elegirse el de menor número de columna.

Se pide:

a) Un programa de línea de órdenes (`entregable1.py`) que reciba como parámetro el nombre de un fichero de texto conteniendo un laberinto en el formato que se especifica más adelante. Como resultado de su ejecución el programa mostrará tres líneas de texto por pantalla:

   - La primera línea contendrá dos enteros separados por un blanco: la fila y la columna de la celda en la que se debe colocar el punto de partida.
   - La segunda línea contendrá dos enteros separados por un blanco: la fila y la columna de la celda en la que se debe colocar el punto de llegada.
   - La tercera línea contendrá la longitud del camino más corto desde el punto de entrada al punto de llegada (medida como el número de aristas).

   El coste de ejecución del programa debe ser lineal con el tamaño del laberinto, entendido como el número de arcos del grafo correspondiente.

b) Memoria del entregable. Una memoria en la que se detallen los pasos seguidos y dificultades experimentadas, a la vez que se presente una solución para el problema en cuestión. La memoria debe incluir los costes espaciales y temporales de tu implementación. Las faltas de ortografía penalizan; una redacción descuidada penaliza.

c) Actas de las reuniones mantenidas hasta la entrega. Recordad que uno de vosotros será el secretario y se encargará de tomar nota en la/s reunión/es de trabajo. El cargo de secretario es rotativo: será una persona diferente para cada entregable.

d) Valoración personal. Cada miembro del grupo deberá escribir una breve valoración del trabajo realizado y de los resultados obtenidos.

Como respuesta a la tarea correspondiente en el aula virtual se enviará un fichero comprimido con todo (a, b, c y d). Además, los apartados b, c y d se entregarán también en papel, en clase, en la fecha indicada arriba.

Detalles de implementación:

- El formato de laberinto consistirá en un fichero de texto con la **situación de los muros** en cada celda, fila a fila. Por ejemplo:

```
wn,sn,n,ne,wn,sn,sn,ne,swn,ne
sw,sne,wse,sw,se,wn,ne,sw,ne,we
wn,sn,ne,wn,sn,se,sw,sn,se,we
we,swn,se,we,wn,sn,sn,sn,sn,e
sw,sn,sn,s,se,swn,sn,sn,sn,se
```

- Implementad una función `load_labyrinth`, que dado el nombre de un fichero conteniendo un laberinto, devuelva dicho laberinto como un grafo no dirigido. La función debe devolver el grafo del laberinto de tal forma que el vértice (0,0) –la entrada al laberinto– se corresponda con la primera celda de la primera línea del fichero. Es decir, la celda (r, c) limitará al sur con la celda (r+1, c), al norte con la celda (r-1, c), al este con la celda (r, c+1) y al oeste con la celda (r, c-1). En el ejemplo anterior la celda (0, 0) tiene paredes `'wn'` y la (rows-1, cols-1), paredes `'se'`.

- IMPORTANTE: Si una celda no tiene ninguna pared, la cadena representando la situación de sus muros será la cadena vacía. En el fichero de texto aparecerá como dos caracteres `','` seguidos. En Python es muy fácil de tratar si dividís cada línea con `split(',')`.

- Ejemplo de funcionamiento: Dado el laberinto anterior la salida debería ser

```
1 1
4 5
35
```

- Visualizar los laberintos puede ayudaros en la detección y corrección de bugs. Podéis utilizar la clase `LabyrinthViewer` que se presenta en el apartado *Extensión opcional* de este entregable. **IMPORTANTE**: El programa debe mostrar por pantalla EXÁCTAMENTE lo que se pide. No debe hacer ni mostrar nada más. Si leéis la extensión que se propone más adelante, veréis que la clase `LabyrinthViewer` sólo debe utilizarse si se llama al programa con el parámetro '-g'.

Algoritmo para encontrar las dos celdas más alejadas de un laberinto bien formado

1. Elige una celda al azar y encuentra la celda más alejada de ella, que llamaremos `'u'`. Dicha celda será uno de los puntos buscados. **IMPORTANTE**: dado que pueden haber varias soluciones, para que todos obtengamos la misma solución utilizaremos la celda (0,0) en vez de dejarla al azar y además, si hay empates en qué celda es la más lejana a la (0,0), elegiremos la de menor fila. Si continua el empate elegiremos la de menor columna.

2. Busca la celda más alejada de `'u'`, que llamaremos `'v'`. Dicha celda será el otro punto buscado. Si hay empates en la celda más lejana, elegiremos la de menor fila. Si continua el empate elegiremos la de menor columna.

3. Dados `'u'` y `'v'`, decide cuál de ellos es el punto de entrada y el de salida siguiendo la restricción del enunciado.

Leer parámetros de la línea de órdenes

Es necesario que implementes tu programa para que lea sus parámetros a través de la línea de órdenes. Veamos cómo se hace mediante un pequeño programa de ejemplo. Crea un fichero `demolineaordenes.py` con el siguiente contenido:

```
import sys
print(sys.argv)
```

Si lo ejecutas desde la línea de órdenes así:

```
python3 demolineaordenes.py 123 -g hola
```

mostrará por pantalla la lista: `['demolineaordenes.py', '123', '-g', 'hola']`, es decir `sys.argv` es una lista de cadenas con lo argumentos de la línea de órdenes.

Prueba automática del programa

En el aula virtual puedes descargarte el paquete **'test-entregable1.zip'**. Este paquete contiene todo lo necesario para comprobar la implementación realizada en el fichero **'entregable1.py'**. Este programa ejecuta vuestro programa sobre varios laberintos y compara la salida obtenida con la con la salida correcta. Copia tu programa `entregable.py` al directorio donde se encuentra `comprueba.py`. Ve a ese directorio y ejecuta la orden:

```
./comprueba.py -a 'python3 entregable1.py' pruebas
```

Tened en cuenta que, para que la línea anterior funcione, `python` debe tener en su ruta de búsqueda la biblioteca `algoritmia` (y, si la utilizas, `easycanvas`). En Linux u OS X es suficiente con ejecutar esta orden (poniendo las rutas que correspondan):

```
export PYTHONPATH=/.../algoritmia/src:/.../easycanvas/src/
```

No se podrá obtener una nota superior a 4 en el entregable si no se superan todas las pruebas del paquete. Tras la entrega se realizarán pruebas adicionales cuyos resultados se tendrán en cuenta para la calificación del entregable.

Extensión opcional

Modificar el programa 'entregable1.py' para que, si recibe un segundo parámetro con el valor '-g', muestre gráficamente el laberinto junto con los puntos de entrada y salida y el camino más corto que los conecta.

Para facilitar la realización de esta extensión podéis utilizar la clase `LabyrinthViewer` que se encuentra disponible en el aula virtual. Esta clase nos permite visualizar nuestros laberintos (grafos) utilizándo `EasyCanvas` (necesitáis instalarlo al igual que hicisteis con la biblioteca 'algoritmia').

La clase `LabyrinthViewer` tiene un método `add_path` que permite dibujar caminos en el laberinto y dos métodos `set_input_point` y `set_output_point` que permiten especificar los puntos de entrada y salida respectivamente. El fichero `demo_labyrinthviewer.py` contiene un ejemplo de utilización de la clase `LabyrinthViewer`.

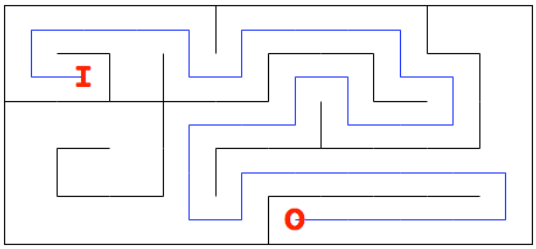
Ejemplo de utilización:

```
lv = LabyrinthViewer(graph, canvas_width=1000, canvas_height=600, margin=10)

pos_input = (0,0)
pos_output = (3,1)
path = [(0, 0), (1, 0), ...]

lv.set_input_point(pos_input)
lv.set_output_point(pos_output)
lv.add_path(path,'blue')

lv.run()
```

Por ejemplo, para el laberinto de nuestro ejemplo:

```
wn,sn,n,ne,wn,sn,sn,ne,swn,ne
sw,sne,wse,sw,se,wn,ne,sw,ne,we
wn,sn,ne,wn,sn,se,sw,sn,se,we
we,swn,se,we,wn,sn,sn,sn,sn,e
sw,sn,sn,s,se,swn,sn,sn,sn,se
```

obtendríamos esta representación gráfica de la solución ('I': punto de partida, 'O': punto de llegada):



Fecha de entrega al aula virtual: lunes 16 de octubre de 2017.

Fecha de entrega de la memoria: lunes 16 de octubre de 2017, en clase.

No os quedéis colgados. Recordad que hay tutorías.