Entregable 1
EI1022
Algoritmia

Mihaita Lupoiu
Pedro Segarra Cabedo
Sergio Varea Aguilella

Índice

a) Memoria del entregable3.....3
b) Actas de las reuniones.....6
c) Valoración personal.....9

a) Memoria del Entregable

1.- Resolución del entregable.

Como primer paso, una vez publicado el entregable, decidimos que cada uno se lo leyera e intentara hallar una posible solución para comentar antes de la reunión de los lunes. También habilitamos en DropBox un espacio compartido para ir subiendo las ideas y resoluciones sobre el mismo.

Todos habíamos resuelto individualmente el ejercicio de recorrido de grafos en anchura y en profundidad; al final el recorrido en anchura es el que nos ayudará en la resolución del entregable. También habíamos resuelto el problema de los amigos, y hallado una solución al movimiento del caballo en el ajedrez, aunque no sabíamos si era óptima.

A continuación empezamos a proponer ideas e implementamos la función que hemos llamado **`load_labyrinth`**. Con esta función recorremos un fichero de entrada (el laberinto) para crear el grafo del laberinto, y dependiendo donde tenga los muros (w,n,s,e), tendrá un nodo vecino u otro. Aquí la práctica del recorrido del caballo del ajedrez nos sirvió bastante para hallar rápidamente la función.

Una vez obtenido el grafo, nos centramos en obtener los caminos más cortos tanto desde la entrada como desde la salida, para ello creíamos que debíamos usar la biblioteca algoritmia con sus funciones y realizamos una función **movimientos**, que aunque realizaba su función,no conseguíamos obtener una solución óptima, ya que al pasarle el test con el laberinto de 200x200 se disparaba a 12 minutos.

```
def movimientos(grafo,vertice):
    return dict(BreadthFirstShortestPaths().one_to_all_backpointers(grafo,vertice))
```

Dedicamos mucho tiempo a probar también con métodos de la librería:

```
BreadthFirstShortestPaths().shortest_path(G, s, t)
ó BreadthFirstShortestPaths().distance(G, s, t)
```

Y luego a realizar lo siguiente: (NO es la solución realizada)

```
def sumamatrizes(diccionario1,diccionario2):
    maximo=0
    mini=max(diccionario1)

    for i in (diccionario1):
        if maximo < len(Backtracer(diccionario2).backtrace(i))+len(Backtracer(diccionario1).backtrace(i)):
            maximo = (len(Backtracer(diccionario2).backtrace(i))+len(Backtracer(diccionario1).backtrace(i)))
            mini = i
        else:
            if maximo == len(Backtracer(diccionario2).backtrace(i))+len(Backtracer(diccionario1).backtrace(i)):
                mini = min(i,mini)
    return mini
```

Que aunque obteníamos la solución, como hemos comentado antes, pasaba más de una vez por las aristas, y utilizábamos funciones de la librería que resolvía el entregable, pero que no entendíamos del todo.

Aquí cambiamos de estrategia de realización del entregable y empezamos a "pensar de otra forma";no obcecarnos en las librerías de algoritmia. Y así hallamos la solución.

Cambiamos de planteamiento y decidimos realizar un recorrido en anchura para la entrada y otro para la salida, basándonos en la implementación del **traverse**.

Funciones realizadas:

```
def visitor (fnt,dst):
    return dst

def dictAnchura (grafo,inicial):
    visitados=set()
    queue=[]
    visitados.add(inicial)
    queue.append((inicial,0))
    movimiento=0
    diccionario={}
    diccionario[(inicial)] =0

    while(len(queue)>0):
        u=queue.pop()
        for v in grafo.succs(u[0]):
            movimiento=u[1]+1 //conteo del recorrido
            if v not in visitados:
                queue.append((v,movimiento))
                visitados.add(v)
                a,b = visitor(u,v)
                diccionario[(a,b)] = movimiento
    return diccionario
```

A la función **dictAnchura**, le pasamos como argumento el grafo del laberinto y un punto inicial (el punto de entrada y el punto de salida). Esta función realiza un recorrido en anchura utilizando un conjunto de visitados para llevar un control de los nodos visitados y una cola para realizar el recorrido en anchura.

Una vez entramos en el bucle vamos recorriendo el grafo en primero en anchura y almacenando en un diccionario los resultados, es decir, para cada nodo(clave) la distancia desde el nodo inicial que le pasamos a la función hasta ese nodo, por ejemplo si le pasamos a la función el punto inicial(0,0) almacenará en el diccionario para el nodo (0,1): **diccionario{(0,1):1}**.

Cuando la ejecución salga de los bucles la función devolverá el diccionario con el recorrido en anchura.

En esta función aunque hay dos bucles el coste es constante, debido a que el bucle **for** como máximo se realizará 4 veces (O(4)) y el **while** se realizara V veces(O(V)), siendo V número de vértices. Por lo tanto el coste de esta función es lineal O(4V)=O(V)

Por último la función:

```
def colocaTesoro(M1,M2):
    maximo = 0
    posicionMin = inicialSalida
    for i in M1:
        if maximo < M1[i] + M2[i]:
            maximo = M1[i] + M2[i]
            tesoro= i
        else:
            if  maximo == M1[i] + M2[i]:
                tesoro = min(posicionMin,i)
    return tesoro
```

colocaTesoro le pasamos como argumento los dos diccionarios obtenidos a partir de la función **dictAnchura**, y nos devuelve la posición donde debemos colocar el tesoro, para cumplir que la suma del camino desde la entrada al tesoro y desde la salida al tesoro sea la distancia máxima y en caso de empate coja la posición mas cerca de la entrada.

El coste de esta función también es lineal, ya que el bucle recorre uno de los diccionarios y busca utilizando la clave en el otro diccionario y comprueba si la suma de los dos valores de la clave es mayor al máximo que teníamos, y si es así lo guardamos hallando la posición ideal para dejar el tesoro.

De esta manera resolvimos el entregable1, ahora adjuntamos las actas de las reuniones mantenidas y la valoración personal.

b) Actas de las reuniones.

Acta nº1.-

Reunión celebrada el día 7 de Octubre de 2013 a las 8.00 en cabina sala de estudio.

Asistentes:
- Mihaita Lupoiu (secretario)
- Pedro Segarra Cabedo(responsable edición)
- Sergio Varea Aguilella(moderador)

Orden del día

1.- Puesta en común del desarrollo del entregable1.
2.- Resolución de entrada de datos.
3.- Acercamiento a resolución gráfica.
4.- Dudas sobre excesivo tiempo de laberinto 200x200

Desarrollo de la Sesión

1.- Puesta en común del desarrollo del entregable1

Como hemos compartido el ejercicio entregable1 en una carpeta en DropBox, y hemos ido realizando modificaciones sobre el mismo, venimos a comentar las pruebas que se han probado en casa y las conclusiones que hemos ido viendo.
Sergio Varea ha dedicado varias horas en la realización de diferentes funciones en el entregable1, Pedro Segarra ha dedicado tiempo en modificación de código de los diferentes funciones, y en traer una posible solución visual del algoritmo. Mihai ha dedicado algunas horas en la prueba y codificación de la parte de movimientos del laberinto y devolución del grafo además de buscar una estructura de datos que sea rápida a la hora de posicionarse en el laberinto.

2.- Resolución de entrada de datos.

Después de comentar las funciones del algoritmo, se prueba a ejecutar el fichero de pruebas que debe superar el entregable para que se pueda obtener una nota de 4.
Aquí nos damos cuenta que debemos cambiar la entrada de datos, y la función debe recibir como argumento el nombre del laberinto para poderse ejecutar correctamente. Una vez corregido la entrada de datos, efectuamos la prueba del entregable.

3.- Acercamiento a resolución gráfica.

Adaptamos el ejemplo del visualizado del laberinto a nuestro entregable.

4.- Dudas sobre excesivo tiempo de laberinto 200x200.

Después de depurar un poco el código, todos los test se han pasado correctamente. Aunque vemos que con un tamaño del laberinto 200x200 cuesta unos 12 minutos su ejecución. Sergio, irá a tutorías a preguntar si estamos realizando algo incorrectamente.

Levantamos la sesión a las 10.47.

Tiempo dedicado a la reunión 2 horas y 47 minutos.
Tiempo dedicado al entregable (fuera de la reunión)
Mihai 4 horas.
Pedro Segarra 4 horas.
Sergio Varea 6 horas.

Castellón a 7 de Octubre de 2013.

Fdo. Mihaita Lupoiu.

Acta nº 2

Reunión celebrada el día 13 de Octubre de 2013 a las 8.30 en cabina sala de estudio.

Asistentes:
- Mihaita Lupoiu (secretario)
- Pedro Segarra Cabedo(responsable edición)
- Sergio Varea Aguilella(moderador)

Orden del día

1.- Puesta en común del desarrollo del entregable1.
2.- Cambio de planteamiento del problema.
3.- Consultar de nuevo en tutorías.

Desarrollo de la Sesión

1.- Puesta en común del desarrollo del entregable1

Sergio después de ir a tutorías, había explicado a Pedro y a Mihai los comentarios del profesor sobre el entregable, expuso que el entregable estaría aprobado con esta versión pero con nota baja.

Como Sergio comunicó que el entregable no era del todo adecuado, Pedro – que tenía un poco más de tiempo para hacer pruebas – realizó unos cambios. Y que mostrará hoy.

2.- Cambio del planteamiento del problema

La función de entrada del laberinto es correcta, lee correctamente el laberinto. El problema está en el cálculo del recorrido que visita más de una vez las aristas, cosa que no debería hacerse.

Pedro trae unas pruebas realizadas del funcionamiento y consigue recorrer el laberinto con un recorrido en anchura, guardando un contador de las casillas visitadas sobre una matriz. Lo único que sucede que cuando un vértice tiene más de un sucesor, el contador no recoge adecuadamente el conteo de las casillas.

De la idea de Pedro, Sergio, con la ayuda de Mihai sabe como solucionarlo porque le ha pasado, una cosa parecida en el ejercicio del ajedrez (movimientos del caballo).

Modificando el recorrido en anchura con el "truco" del ajedrez, la ejecución del entregable se realiza correctamente para todos los test. Ya no hay un retardo de 12 minutos al ejecutar la matriz de 200 x200.

3.- Consultar de nuevo en tutorías

Los tres nos acercamos a los despachos de los profesores por si nos pueden atender, aunque no sea el horario de tutorías, pero encontramos que el profesor está impartiendo una clase en un despacho adyacente, y no queremos molestar. Sergio tiene horario libre después de clase del martes y se acercará a comentarle los cambios del entregable.

4.- Realización de la memoria.

Pedro será el primero en realizar las actas y la memoria del entregable. Cómo coincidimos en clase de Algoritmia, en prácticas y en algunas otras asignaturas vamos a seguir comentando la respuesta del profesor sobre el entregable, y ver si la solución que se ha dado es correcta.

Levantamos la sesión a las 10.43.

Tiempo dedicado a la reunión 2 horas y 13 minutos.
Tiempo dedicado al entregable (fuera de la reunión)
Mihaita Lupoiu 2 horas.
Pedro Segarra 4 horas.
Sergio Varea 2 horas.

Castellón a 14 de Octubre de 2013.

Fdo. Mihaita Lupoiu.

c) Valoración personal.

Mihaita Lupoiu: " *La primera gran dificultad que he encontrado al iniciar este trabajo ha sido darme cuenta de que realmente no sabía como empezar. Me sentía perdido porque no entendía lo que se pedía en el enunciado, hasta que después de leerlo unas cuantas veces por fin empecé a comprender algunas de las cosas.*

Otras de las dificultades que he experimentado ha sido poder realizar un código óptimo, ya que el concepto de coste no lo tenía del todo claro.

El trabajo en general, cuando se encuentra la solución, no parece tan complicado, pero tampoco tan fácil. Creo que el ejercicio tiene como fin que entendamos muy bien como funcionan los grafos y lo generalicemos para todo tipo de problemas. El entregable ha sido entretenido, aunque también ha traído dolores de cabeza."

Pedro Segarra: *"Tras leer el enunciado del entregable, no sabía como empezar a resolverlo. Habíamos visto la práctica del movimiento del caballo e intuía que la solución podría estar relacionada con algunas partes de esta práctica.*

Al comentarlo con los compañeros empezamos a buscar una posible solución que fué rechazada porque el coste se disparaba; lo supimos al ir a tutorías. Intentamos otra solución que pasaba los test, pero en el laberinto de 200x200, se quedaba en espera unos 12 minutos; aquí me frustré bastante porque pensaba que no daríamos con una solución adecuada.

Tuve que dibujar el laberinto de ejemplo – 5x10 - para entender como funcionaba, y así realizar un recorrido en anchura. Del dibujo vi que podíamos recorrer las celdas del laberinto e ir contándolas, pero cuando había más de un sucesor el conteo fallaba y no conseguía resolverlo, pero al comentarlo con Sergio y Mihaita hallamos una solución que consideramos adecuada.

Después de solucionar algunos errores, pasarle el paquete de test, y ver que pasaba todas las pruebas respiré tranquilo.

La principal dificultad que he tenido es encontrar una idea feliz para resolverlo, de hecho si no hubiera trabajado en grupo no hubiera resuelto el problema."

Sergio Varea: *"La mayor dificultad que he encontrado en este entregable ha sido la de entender lo que se nos pedía, y sobre todo el cálculo de coste, debido a que este concepto no lo tenía muy claro y por lo tanto no sabía si el algoritmo que tenía era óptimo o no.*

Una vez solucionado me ha parecido muy útil, ahora entiendo mejor como funcionan los grafos y como recorrerlos; además me ha parecido muy entretenido ya que es un algoritmo que le ves que tiene una función y que sirve para resolver un problema interesante."