Algoritmos y Estructura de Datos III

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico 2

Grupo 1

Integrante	LU	Correo electrónico
Candioti, Alejandro	784/13	amcandio@gmail.com
Maldonado, Kevin	018/14	maldonadokevin11@gmail.com
Shaurli, Tomas	671/10	tshaurli@gmail.com
Tripodi, Guido	843/10	guido.tripodi@hotmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Contents

1 Ejercicio 1

1.1 Descripción de problema

Estamos en el año 2048 y el Pabell on 0+infinito es todo un éxito. Los alumnos de Algoritmos III están contentos porque van a cursar este cuatrimestre en un aula que esta en el piso N, que es el más alto de todos. Con los avances de la ciencia y la tecnología, escaleras y ascensores han quedado obsoletos, y la forma de subir de un piso a otro es a traes de portales. El nuevo pabellon tiene P portales, cada uno de los cuales permite subir de un piso A a un piso mas alto B (para bajar de piso hay que tirarse con paracaıdas al piso 0 y luego volver a subir de ser necesario). Uno de los alumnos, que estaba cursando en el segundo cuatrimestre de 2015 y fue congelado por el metodo de criogenia, acaba de ser descongelado y no puede creer lo buenos que estan estos portales, algo que en su época no existía. Luego de completar todos los censos de estudiantes desde el año 2016 en adelante, este alumno quiere usar la mayor cantidad de portales posibles para llegar al piso N y asi seguir cursando Algoritmos III. Diseñar un algoritmo de complejidad O(N^2) para calcular la mayor cantidad de portales que puede utilizar el alumno para subir desde planta baja al piso N (sin tirarse nunca con paracaıdas). Se asegura que en toda instancia del problema es posible realizar el recorrido deseado, y que no hay más de un portal que comunique el mismo par de pisos.

1.2 Explicación de resolución del problema

1.3 Algoritmos

Algoritmo 1

```
1: function SOLVE(in cableSize : Integer, in stationDistances : List<Integer>)→ out res :
   Integer
2: end function
```

Complejidad: O(N^2)

1.4 Análisis de complejidades

1.5 Experimientos y conclusiones

1.5.1 Test

Por medio de los tests dados por la cátedra, desarrollamos nuestros tests, para corroborar que nuestro algoritmo era el indicado.

A continuación enunciaremos los tipos de casos diferentes que encontramos con mayor relevancia a la hora de realizar testeos sobre el algoritmo:

Unico portal que lleva a un unico piso N

Para este tipo de testeo mostraremos a continuación un ejemplo del mismo, exponiendo su respectivo resultado.

Con un:

$Piso$: 10

Obtuvimos el siguiente resultado:

Cantidad de portales usados : 1

Portales que pueden conectar todos los pisos desde el 0 hasta el N

Como ejemplo mostraremos, exponiendo su respectivo resultado, el siguiente testeo.

Con un secuencia de pisos:

Piso : 01020
Obtuvimos el siguiente resultado:

Cantidad de portales usados : 3

Ningún portal posible que conecte una secuencia para llegar al piso N

Como ejemplo mostraremos, exponiendo su respectivo resultado, el siguiente testeo.

Con un secuencia de pisos:

Piso : 04; 15; 26; 59
Obtuvimos el siguiente resultado:

Cantidad de portales usados : 0

1.5.2 Performance De Algoritmo y Gráfico

Por consiguiente, mostraremos buenos y malos casos para nuestro algoritmo, y a su vez, daremos el tiempo estimado según la complejidad del algoritmo calculada anteriormente.

Para llegar a dicha conclusión trabajamos con un total de 100 instancias y un n entre 1 y 1000000 obtuvimos que nuestro algoritmo finaliza lo solicitado demorando 184 milisegundos.

Para una mayor observacion desarrollamos el siguiente grafico con las instancias:

Si a esto lo dividimos por la complejidad propuesta obtenemos:

Para realizar esta división realizamos un promedio con el mismo input de aproximadamente 20 corridas tanto para la complejidad como para nuestro algoritmo y una vez calculado dicho promedio de ambas cosas realizamos la división para obtener resultados más consisos.

A continuación, adjuntamos una tabla con los considerados "mejor" caso que nos parecieron más relevantes
Dando un **promedio igual a**

Para llegar a dicha conclusión trabajamos con un total de 100 instancias y un n entre 1 y 1000000 obtuvimos que nuestro algoritmo finaliza lo solicitado demorando 224 milisegundos.

Si a esto lo dividimos por la complejidad propuesta obtenemos:

Para realizar esta experimentación nos parecio acorde, realizar un promedio con el mismo input de aproximadamente 20 corridas tanto para la complejidad como para nuestro algoritmo y una vez calculado dicho promedio de ambas cosas realizamos la división para obtener resultados más relevantes.

La información de los 10 datos mas relevantes referiendonos al peor caso fueron:
Dando un **promedio igual a**

Aquí, podemos observar como la cota de complejidad del algoritmo y la de dicho caso tienden al mismo valor con el paso del tiempo.

2 Ejercicio 2

2.1 Descripción de problema

El Pabellón 0+infinito acaba de reabrir sus puertas con la novedad de que ahora tiene P portales que son bidireccionales; asimismo, los paracaıdas fueron eliminados por considerarse inseguros. Cada piso del renovado pabellón consta de un pasillo de L metros de longitud, y cada portal permite viajar entre posiciones específicas de los pasillos de dos pisos. Más concretamente, cada portal puede describirse por medio de cuatro enteros no negativos A, DA, B y DB, los cuales indican que el portal comunica el piso A, a DA metros del comienzo del pasillo de ese piso, con el piso B, a DB metros del comienzo del pasillo de ese piso. Los alumnos, acostrumbrados a los portales que solo permitían subir, están un poco confundidos al poder utilizar un mismo portal tanto para subir como para bajar entre dos pisos, o incluso para moverse entre posiciones diferentes dentro del pasillo de un mismo piso. Todos los alumnos de Algoritmos III quieren llegar primero a la clase, que es en un aula que está al final del piso N (el máas alto del pabellón).
Diseñar un algoritmo de complejidad O(NL + P) para calcular la mínima cantidad de segundos que se necesitan para llegar del comienzo del pasillo del piso 0 al final del pasillo del piso N, suponiendo que recorrer un metro requiere 1 segundo, y utilizar cualquier portal requiere 2 segundos (en cualquiera de las dos direcciones posibles). Se asegura que en toda instancia del problema es posible realizar el recorrido deseado, y que no hay máas de un portal que comunique las mismas posiciones del mismo par de pisos. No obstante, puede haber más de un portal que comunique el mismo par de pisos, y portales que comuniquen posiciones diferentes dentro del pasillo de un mismo piso.

2.2 Explicación de resolución del problema

2.3 Algoritmos

Algoritmo 2 A MEDIAS

1: **function** SOLVE(**in** *list*: `List<Integer>`)→ **out** *res*: `List<Integer>`
2: **end function**

Complejidad: O(NL + P)

2.4 Análisis de complejidades

2.5 Experimientos y conclusiones

2.5.1 Test

Por medio de los tests dados por la cátedra, desarrollamos nuestros tests, para corroborar que nuestro algoritmo era el indicado.

Dichos tests los hemos catalogado y diferenciado en distintos casos, que enunciaremos a continuación:

Los portales alcanzan para cubrir todo el camino sin necesidad de caminar por los pasillos

Para este tipo de testeo mostraremos a continuación un ejemplo del mismo, exponiendo su respectivo resultado.

Necesidad de caminar por todo el pasillo para llegar hasta los portales

Como ejemplo mostraremos, exponiendo su respectivo resultado, el siguiente testeo.

2.5.2 Performance De Algoritmo y Gráfico

Acorde a lo solicitado, mostraremos los mejores y peores casos para nuestro algoritmo, y además, daremos el tiempo estimado según la complejidad del algoritmo calculada anteriormente.

Para una mayor observacion desarrollamos el siguiente grafico con las instancias:

Y dividiendo por la complejidad de nuestro algoritmo llegamos a:

Para realizar esta experimentación nos parecio prudente, realizar un promedio con el mismo input (n entre 1 y 1001000) de aproximadamente 20 corridas tanto para la complejidad como para nuestro algoritmo y una vez calculado dicho promedio de ambas cosas realizamos la división para obtener resultados más relevantes.

Se puede observar, como luego de realizar la división por la complejidad cuando el n aumenta el valor tiende a 0.

A continuación mostraremos una tabla con los 10 datos de medición mas relevantes y mostraremos un promedio de la totalidad de las instancias probadas.

Promedio final de todas las instancias:

Verificando el peor caso, llegamos a la conclusión que el tipo de caso en el que resulta menos beneficioso trabajar con nuestro algoritmo será cuando ambas ramas se encuentran desordenadas.

Y dividiendo por la complejidad propuesta llegamos a:
Para realizar esta experimentación nos parecio acorde, realizar un promedio con el mismo input de aproximadamente 20 corridas tanto para la complejidad como para nuestro algoritmo y una vez calculado dicho promedio de ambas cosas realizamos la división para obtener resultados más relevantes.

Se puede observar que a pesar de tardar varios milisegundos este tipo de caso, al dividir por vuestra complejidad es propenso a tender a 0 quedando comparativamente por encima del mejor caso.

A continuación mostraremos una tabla de valores de lass ultimas 10 instancias y mostraremos el promedio total conseguido .

Promedio total conseguido:

Se puede observar como el peor caso presenta un promedio mayor que el mejor caso, concluyendo lo que enunciamos inicialmente.

3 Ejercicio 3

3.1 Descripción de problema

El Pabellon 0+infinito, nuevamente remodelado, ahora tiene un diseño basado en un conjunto de M pasillos de distintas longitudes con intersecciones en donde se unen dos o mas pasillos. Es ası que puede modelarse como un grafo con pesos en los ejes, donde cada eje es un pasillo (de peso igual a la longitud del pasillo), y cada vértices es una intersección o un extremo donde termina un pasillo sin unirse con ningun otro. El decano junto con el director del Departamento de Computacion, están preocupados porque talvez existen ciclos en dicho grafo, lo que podría perjudicar a los alumnos al hacer que se pierdan buscando las aulas. Por tal motivo el decano decidió clausurar los pasillos que sea necesario de manera tal que no queden ciclos en el grafo que representa al pabellon. El problema es que cuanto más largo es un pasillo, más costoso es clausurarlo. Diseñar un algoritmo de complejidad O(M log M) para calcular la mínima suma posible de las longitudes de los pasillos que deberían ser clausurados (eventualmente ninguno) para que no existan ciclos formados por tres o mas pasillos en el grafo que representa al pabellón. Se asegura que en toda instancia del problema el grafo que representa al pabellon es conexo.

3.2 Explicación de resolución del problema

3.3 Algoritmos

A continuación se detalla el pseudo-código de la parte principal del algoritmo:

Algoritmo 3 Calculate

```
1: function CALCULATE(in currentIdx: Integer)
2: end function
```

Complejidad: O(M log M)

3.4 Análisis de complejidades