



Instituto Politécnico Nacional
Escuela Superior de Cómputo

PRÁCTICA 1: Demostración experimental de la complejidad temporal de un algoritmo.

Análisis de algoritmos
3CV12

Reyes Rodríguez Enrique Abdiel
Chávez Hernandez Juan Diego

abykings1@gmail.com
jdiegohdez0233@gmail.com

Septiembre 2021

Resumen: En el siguiente trabajo se presenta la determinación experimental de la complejidad temporal de un algoritmo, llevando a cabo el análisis de 2 algoritmos.

Palabras clave: C, algoritmo, análisis, complejidad, tiempo.

1. Introducción

La planeación de un algoritmo es de lo más importante al momento de solucionar problemas, básicamente es la idea de solución del problema. Sin una buena planeación se puede llegar a fallos lógicos o de eficiencia, se debe analizar a profundidad todos los casos. Después de analizar el algoritmo es necesario traducir este algoritmo a instrucciones.

Como objetivo de la práctica está el demostrar que el análisis de la complejidad temporal de un algoritmo, nos da diferentes complejidades según el tipo de solución.

2. Conceptos básicos

Un algoritmo es una serie de pasos organizados, que describe el proceso que se debe seguir, para dar solución a un problema en específico. Como tal pensar en un algoritmo, es pensar la forma en la que podemos resolver un problema paso a paso, si no se resuelve un paso del problema no se puede pasar al siguiente. [1]

La complejidad temporal es quien juzga la duración del tiempo de ejecución del algoritmo, pero nos es imposible calcular el tiempo utilizado. Dado que la complejidad temporal del algoritmo es directamente proporcional al número de ejecuciones del algoritmo, la complejidad temporal del algoritmo se puede obtener de acuerdo con el número de ejecuciones del algoritmo. El número de ejecuciones del algoritmo también se llama Frecuencia de tiempo Denotado como $T(n)$. Donde n es el tamaño del problema, el número de operaciones básicas realizadas en el algoritmo es una función de n , dando $T(n)$. [2]

El Algoritmo Euclides es un procedimiento para calcular el máximo común divisor (m.c.d.) de dos números. Euclides fue un matemático griego que recopiló varios datos en una obra llamada Elementos. Esta obra es considerada como uno de los pilares de las matemáticas, y Euclides el "padre de la geometría".

Aunque la palabra algoritmo nos hace pensar en cálculos complejos resueltos por ordenadores, en nuestro caso el cálculo es mucho más sencillo. Solo hace falta seguir los siguientes pasos.[3]

1. Se divide el número mayor entre el menor.
2. Si la división es exacta, el divisor es el m.c.d.

3. Si la división no es exacta, dividimos el divisor entre el resto obtenido y continuamos de esta forma hasta obtener una división exacta. El m.c.d. es el último divisor.

Análisis priori significa antes o previo a algo; usándose en el campo del conocimiento para hacer referencia a la información que no procede del resultado de una experiencia, siendo anterior a ella. Los conocimientos “a priori” aparecen en ciertos casos, como verdades indubitables, por ser conocimientos puramente racionales, lógicos e innatos.[4]

Análisis a posteriori es una expresión del latín que significa ‘posteriormente’ o ‘con posterioridad’. Literalmente, puede traducirse como ‘de lo posterior’ o ‘por lo que viene después’. A posteriori, en este sentido, hace alusión a examinar o considerar un hecho después de que ha ocurrido. Por ejemplo: “Los análisis de las elecciones se harán a posteriori, no en este momento”. [5]

Los pseudocódigos de los algoritmos a analizar son los siguientes:

Ejercicio 1

```
w -> 1
d -> 0
for i -> n-1, i >= 0, i-- do
  and -> a[i] & b[i]
  pow -> (and * w)
  d -> d + pow
  w -> w * 2
return d
```

Ejercicio 2

```
while n != 0 do
  r <- m mod n
  m <- n
  n <- r
return m
```

3. Experimentación y resultados

Ejercicio 1

Para generar la entrada n , se realizo por medio de numeros aleatorios, limitandolos solo a 0 y 1, para poder realizar la operación logica AND. La salida a graficar fue el tamaño de el arreglo y el numero de pasos que nos dio la ejecución del programa. Graficando estos, obtenemos una grafica lineal, que corresponde con el analisis a priori, como se muestra en el anexo.

```
abdiel@dell:~/coding/alg$ gcc and.c && ./a.out

out:0, size: 0, count: 11
0
1
out:0, size: 1, count: 19
1 1
0 0
out:0, size: 2, count: 27
1 0 0
0 0 0
out:0, size: 3, count: 35
0 1 0 0
0 1 0 1
out:4, size: 4, count: 43
1 0 0 0 1
1 0 1 0 0
out:16, size: 5, count: 51
0 0 1 1 1 1
1 1 0 0 1 0
out:2, size: 6, count: 59
0 1 1 0 0 1 1
0 0 0 0 1 0 1
out:1, size: 7, count: 67
0 0 0 0 0 1 1 1
0 0 1 0 1 1 0 0
out:4, size: 8, count: 75
1 0 0 0 0 0 1 0 0
1 0 0 0 1 0 0 1 0
out:256, size: 9, count: 83
1 0 1 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1
out:512, size: 10, count: 91
0 0 1 1 1 1 0 1 0 1 1
1 1 1 1 1 1 1 1 1 1
out:491, size: 11, count: 99
1 0 1 1 0 0 1 1 1 0 1 0
1 0 0 1 1 0 0 1 1 0 0 0
out:2328, size: 12, count: 107
1 1 1 0 0 1 1 1 1 0 1 1
1 0 0 1 0 1 0 0 1 1 0 1 0
out:4250, size: 13, count: 115
1 0 0 1 0 1 0 1 1 0 0 1 1 1
0 1 1 0 1 1 0 1 0 0 0 1 0
```

Figura 1: Muestra el arreglo a analizar, junto con su salida, tamaño y número de pasos en la ejecución.

n	contador
0	11
1	19
2	27
3	35
4	43
5	51
6	59
7	67
8	75
9	83
10	91
11	99
12	107
13	115
14	123
15	131
16	139
17	147
18	155
19	163
20	171

Cuadro 1: Estos son 21 resultados obtenidos del algoritmo del ejercicio 1, n es la entrada, siendo el tamaño del array y contador, que es el numero de pasos ejecutados por el algoritmo.

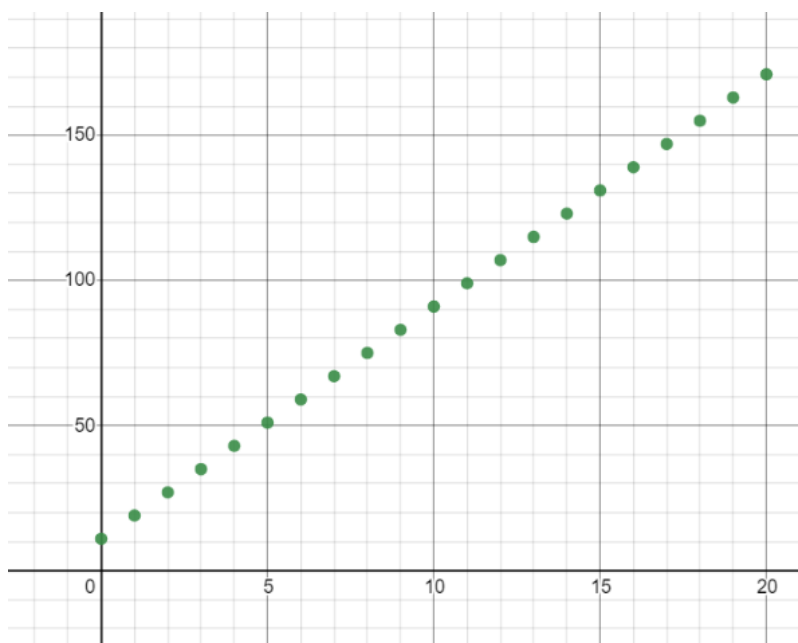
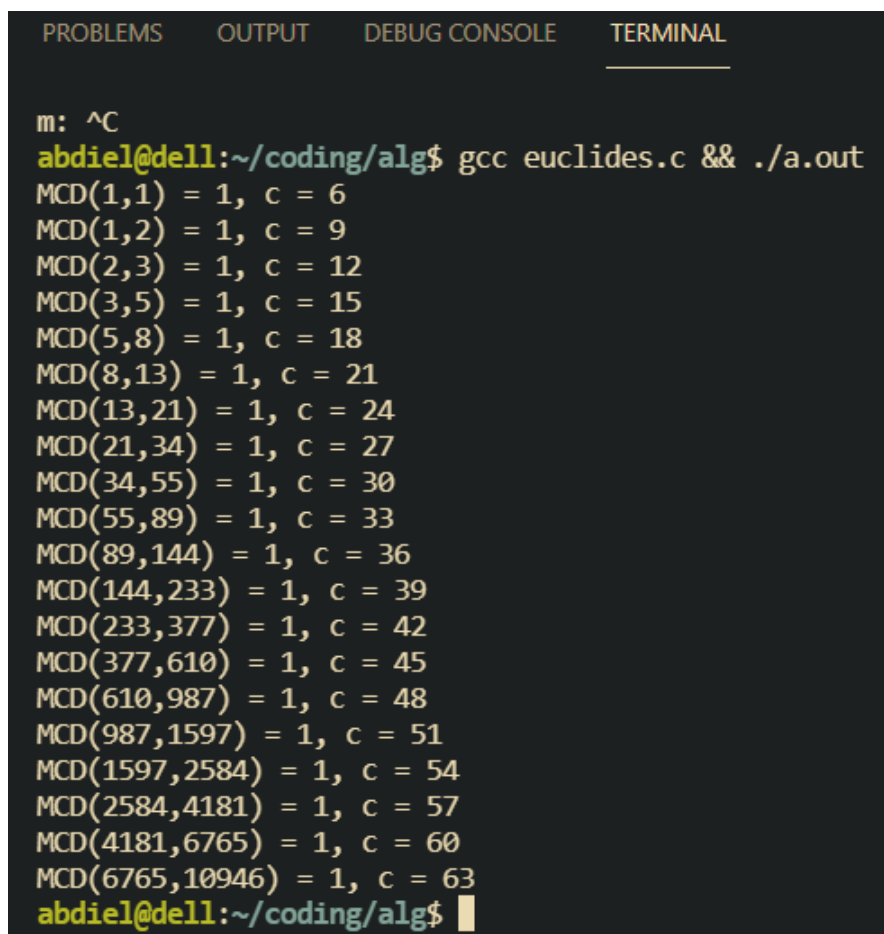


Figura 2: Muestra el arreglo a analizar, junto con su salida, tamaño y número de pasos en la ejecución.

Ejercicio 2

Ya ejecutado el código, se probó con diferentes métodos de entrada. El primero fue por el usuario, donde se daban los 2 números y se realizaba el algoritmo. En el segundo fué por medio de la serie fibonacci, la cual da como resultado el peor caso para el algoritmo de Euclides, dando como resultado una gráfica logarítmica.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
m: ^C
abdiel@dell:~/coding/alg$ gcc euclides.c && ./a.out
MCD(1,1) = 1, c = 6
MCD(1,2) = 1, c = 9
MCD(2,3) = 1, c = 12
MCD(3,5) = 1, c = 15
MCD(5,8) = 1, c = 18
MCD(8,13) = 1, c = 21
MCD(13,21) = 1, c = 24
MCD(21,34) = 1, c = 27
MCD(34,55) = 1, c = 30
MCD(55,89) = 1, c = 33
MCD(89,144) = 1, c = 36
MCD(144,233) = 1, c = 39
MCD(233,377) = 1, c = 42
MCD(377,610) = 1, c = 45
MCD(610,987) = 1, c = 48
MCD(987,1597) = 1, c = 51
MCD(1597,2584) = 1, c = 54
MCD(2584,4181) = 1, c = 57
MCD(4181,6765) = 1, c = 60
MCD(6765,10946) = 1, c = 63
abdiel@dell:~/coding/alg$
```

Figura 3: Esta es la salida de el código por consola, viendo que ninguna de las entradas da un resultado mayor a uno siendo todos casos no favorables.

n	contador
10	8
20	13
50	18
100	13
500	13
800	28
1000	18
2000	13
4500	18
6000	18
10000	23
10200	18
10000	13
10001	18
10001	13

Cuadro 2: Estos son los resultados de un experimento aleatorio, en la parte izquierda tenemos a el numero más grande de cual se hizo el algoritmo y el contador, que es el número de pasos que se contabilizaron en la ejecución.

n	contador
1	8
2	13
3	18
5	23
8	28
13	33
21	38
34	43
55	48
89	53
144	58
233	63
377	68
610	73
987	78
1597	83
2584	88
4181	93
6765	98
10946	103

Cuadro 3: Estos son los resultados del experimento con la sucesión Fibonacci como entrada, siendo esta, el peor caso para dicho algoritmo por lo cual se observa como evoluciona el contador cuando incrementa n.

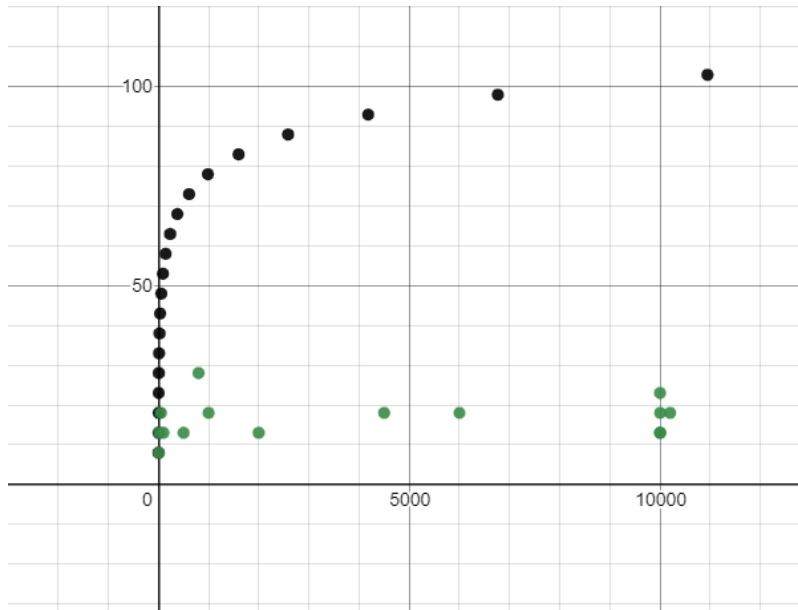


Figura 4: Los puntos de color negro representan los resultados obtenidos con la sucesión Fibonacci y los verdes por datos al azar. Se observa como están dentro de los límites marcados por el peor caso.

4. Conclusiones



Enrique Abdiel Reyes Rodríguez:

Empezando por los problemas, desde el inicio se tuvo que estudiar un poco de lenguaje C, ya que se tenían problemas con los apuntadores que después, sirvieron para el contador de pasos. Las implementaciones de los algoritmos a código no fueron mucho problema, se empezó haciendo soluciones un poco más toscas y de ahí refinarlas para que quedaran lo más eficientes posible. Lo que me sorprendió fue el resultado de la gráfica de el segundo ejercicio ya que la gráfica estaba muy bien descrita, tuve que investigar por qué la sucesión Fibonacci es el peor caso para el algoritmo de Euclides. Para el primer ejercicio, dado que sólo teníamos un ciclo for, supuse que era lineal, y con el análisis a posteriori lo comprobé.



Juan Diego Chávez Hernández:

En el desarrollo de la practica podemos observar que el uso de apuntadores es fundamental para poder generar soluciones de los dos ejercicios, dichos apuntadores ayudaron al conteo de los pasos. Cabe mencionar que en el primer ejercicio se realiza el análisis a posteriori y se comprueba que es lineal, en el segundo ejercicio se utiliza dos métodos de entrada uno otorgada por el usuario y la otra por la serie del Fibonacci , se obtiene una grafica logarítmica lo cual fue algo nuevo e interesante de ver.

5. Anexo

Ejercicio 1

Este es el análisis a priori del ejercicio 1:

```

1      w -> 1
2      d -> 0
3      for i -> n-1, i >= 0, i-- do
4          and -> a[i] & b[i]
5          pow -> (and * w)
6          d -> d + pow
7          w -> w * 2
8      return d

```

Cost	Times
C1	1
C2	1
C3	n+1
C4	n
C5	n
C6	n
C7	n
C8	1

Cuadro 4: Tabla relacion costo y número de repeticiones

$$\begin{aligned}
 T(n) &= C1 + C2 + C3(n+1) + C4(n) + C5(n) + C6(n) + C7(n) + C8 \\
 &= C1 + C2 + C3n + C3 + n(C1 + C4 + C5 + C6 + C7) + C8 \\
 &= n(C3 + C4 + C5 + C6 + C7) + C1 + C2 + C3 + C8 \quad (1)
 \end{aligned}$$

En este caso, el mejor caso, es el mismo que el peor caso, por lo tanto usamos θ

$$T(n) \in \theta(n) \quad (2)$$

Ejercicio 2

De el analisis a posteriori podemos deducir que:

$$T(n) = O(\log|n|) \quad (3)$$

Ya que los resultados estuvieron debajo del peor caso.

Referencias

- [1] (s.f.). Recuperado el 06 de Septiembre de 2021, de <https://programmerclick.com/article/89671544092/>
- [2] (s.f.). Recuperado el 06 de Septiembre de 2021, de <https://es.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm>
- [3] (s.f.). Recuperado el 06 de Septiembre de 2021, de <https://deconceptos.com/general/a-priori>
- [4] (s.f.). Recuperado el 06 de Septiembre de 2021, de <https://www.significados.com/a-posteriori/>
- [5] Flores, L. A. (s.f.). UAEM. Recuperado el 06 de Septiembre de 2021, de <https://www.uaeh.edu.mx/scige/boletin/prepa4/n10/e1.html>