Análisis de Algoritmos, Sem: 2018-1, 3CV2 Práctica 1,
24-Agosto-2017

Práctica 1: Determinación experimental de la complejidad de un algoritmo

Martínez Berumen Luis Daniel
Escuela Superior de Cómputo
Instituto Politécnico Nacional
dany.berumen@gmail.com

En el presente trabajose analizará, por medio de la experimentación, la complejidad de un algoritmo dado, proponiendo una función g(n) tal que S $\in$ O(g(n)) y g(n) sea mínima, en el sentido de que si S $\in$ O(h(n)), entonces g(n) $\in$ O(h(n)). A través de la observación se compará el comportamiento entre el tiempo de ejecución y un número de veces que se ejecuta una función.

Palabras Clave

- Algoritmo
- Complejidad
- Experimentacion
- Eficiencia

1. Introducción

En los tiempos actuales es importante la manera en la que resolvemos los trabajos diariamente, por medio de diferentes acciones que pueden llegar a ser repetitivas se logran resolver la mayoria de los problemas que enfrentamos diariamente. Es por eso que es importante el estudio de los algoritmos empleados día con día para así determinar los que nos permitan resolver nuestro trabajo de la manera más optima posible. Al igual que nuestro trabajo, los algoritmos que usamos pueden ser empleados para automatizar diferentes aspectos de nuestras actividades cotidianas, o hasta trabajos de grandes industrias. Empleando el algoritmo que mejor se adecue al trabajo a realizar, se lograra tener tiempos menores, lo que se podria traducir en ahorros en produccion y aumento en la eficiencia.

2. Conceptos Básicos

Es necesario definir algunos terminos para la comprension de el analicis efectuado a continuacion, tales como θ, O y Ω.
θ(n):
Sea g(n) una función. Se define θ (g(n)) como:

$\theta(\text{g(n)}) = \{f(n) \quad | \quad \exists c1, c2 > 0 \quad \& \quad n_0 > 0 \quad | \quad \forall n >= n_0 \quad 0 <= c1g(n) <= f(n) <= c2g(n)\}$

O(n):
Sea g(n) una función, O(n) se define como:

$$\text{O(n)}=\{f(n) \quad | \quad \exists c > 0 \quad \& \quad n_0 > 0 \quad | \quad f(n) <= Cg(n) \quad \forall n >= n_0\}$$

Ω(n):
Sea g(n) una función. Se define Ω (g(n)) como:

$$\Omega(\text{g(n)}) =\{f(n) \quad | \quad \exists c > 0 \quad \& \quad n_0 > 0 \quad | \quad 0 <= cg(n) <= f(n) \quad \forall n >= n_0\}$$

Se muestra experimentalmente el tiempo computacional de dos algoritmos y un tercero de manera teórica, el primer algoritmo de desarrollo experimental consiste en sumar dos números enteros en notación binariaconsiderando que: Dos arreglos unidimensionales A de tamaño n y B de tamaño m con k = $\log_2(n)$ y t = $\log 2(m)$ almacenarán los números a sumar. La suma se almacenará en un arreglo C. El propósito es obtener el tiempo que se tarda en realizar dicha operación con un tamaño r que se le asigna al arreglo.

- 0+0=0
- 0+1=1
- 1+0=1
- 1+1=0 y se acarrea un 1 que se sumara a la siguiente posicion
- Con esto podemos aclarar 4 casos:
- cuando la suma da 0: se agrega un 0 al arreglo C
- cuando la suma da 1: se agrega un 1 al arreglo C
- cuando la suma da 2: se agrega un 0 al arreglo C y el acarreo se hace 1
- cuando la suma da 3: se agrega un 1 al arreglo C y el acarreo se hace 1

El segundo ejercicio trata de la implementacion del algoritmo de Euclides para encontrar el maximo comun divisor de dos números enteros positivos m y n. Con el finn deobtener el tiempo de ejecución cuando se mande a llamar la función de Euclides, pero con la condición que los números a los cueles se obtendrá el m.c.d

Para el algoritmo teórico se encontrará la complejidad de un programa de ordenamiento.

3. Experimentación y Resultados

3.1. Suma Binaria

Suma (A,B,n[1,...,17],c)
Entrada: i[1,..., 17]
Salida: Tiempo computacional de n[1,...,17]
1. for i ← 1 to i ← 17 do
2. n ← potencia(2,i)
3. suma(A,B,n,c)
4. print(c)

-Mostrar la grafica para la función S que muestre tiempo vs r con r = m = n (considere diversos valores de r).

Arreglo	Veces que entro
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15

Figura 1

Cuya grafica es es:

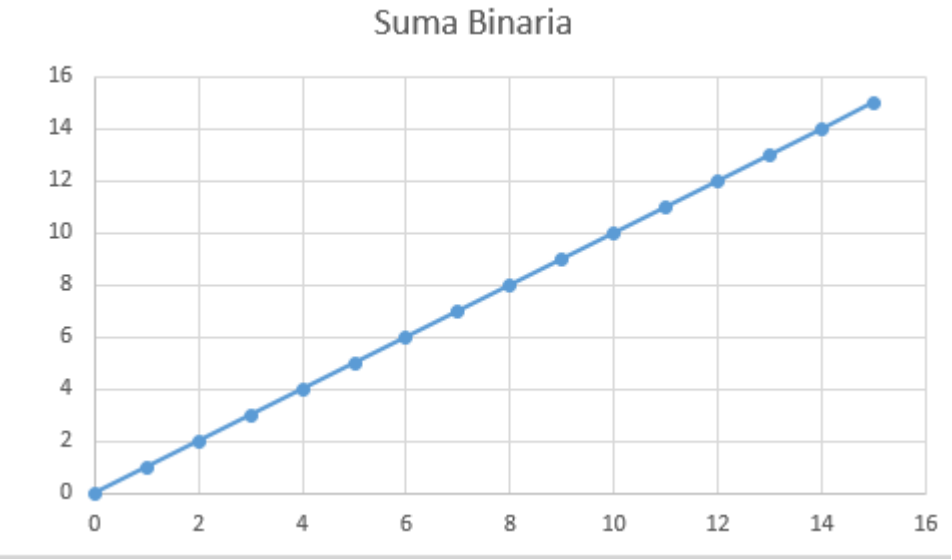

Figura 2

-Proponer una función g(n) tal que S ∈ O(g(n)) y g(n) sea mínima, en el sentido de que si S ∈ O(h(n)), entonces g(n) ∈ O(h(n)).

Por los resultados obtenidos en la última gráfica proponemos que la complejidad del algoritmo S es de tipo lineal, y se propone la función O(n).

-Conclusiones

En este experimento se pudo comprobar el como se efectua una suma binaria, aunque en cuestiones de tiempo, no marco diferencia al meter grandes arreglos, supongo que el algoritmo empleado fue una buena solucion, veo que los tamaños no importan mucho en este problema o no con la computadora donde fueron programados.

3.2. Euclides

1.- Pseudocódigo Algoritmo de Euclides

Euclides (A,B,c]
Entrada: fibonacci(n),fibonacci(n-1) con $1 <= n < 50$
Salida: Tiempo computacional de $1 <= n < 50$
1. for i ← 1 to i ← 79 do
2. **a ← fibonacci(i)**
3. **b ← fibonacci(i+1)**
4. **e ← euclides(b,a,c)**
5. **print(c)**

-Mostrar diversas gráficas para la función Euclides que muestre tiempo vs diferentes valores de m y n.

Para para los primeros 79 valores de Fibonacci:

```
Para:[ 1 ] , [ 1 ]El contador es:  1
Para:[ 1 ] , [ 2 ]El contador es:  2
Para:[ 2 ] , [ 3 ]El contador es:  3
Para:[ 3 ] , [ 5 ]El contador es:  4
Para:[ 5 ] , [ 8 ]El contador es:  5
Para:[ 8 ] , [ 13 ]El contador es:  6
Para:[ 13 ] , [ 21 ]El contador es:  7
Para:[ 21 ] , [ 34 ]El contador es:  8
Para:[ 34 ] , [ 55 ]El contador es:  9
Para:[ 55 ] , [ 89 ]El contador es:  10
Para:[ 89 ] , [ 144 ]El contador es:  11
Para:[ 144 ] , [ 233 ]El contador es:  12
Para:[ 233 ] , [ 377 ]El contador es:  13
Para:[ 377 ] , [ 610 ]El contador es:  14
Para:[ 610 ] , [ 987 ]El contador es:  15
Para:[ 987 ] , [ 1597 ]El contador es:  16
Para:[ 1597 ] , [ 2584 ]El contador es:  17
Para:[ 2584 ] , [ 4181 ]El contador es:  18
Para:[ 4181 ] , [ 6765 ]El contador es:  19
Para:[ 6765 ] , [ 10946 ]El contador es:  20
Para:[ 10946 ] , [ 17711 ]El contador es:  21
Para:[ 17711 ] , [ 28657 ]El contador es:  22
Para:[ 28657 ] , [ 46368 ]El contador es:  23
Para:[ 46368 ] , [ 75025 ]El contador es:  24
Para:[ 75025 ] , [ 121393 ]El contador es:  25
Para:[ 121393 ] , [ 196418 ]El contador es:  26
Para:[ 196418 ] , [ 317811 ]El contador es:  27
Para:[ 317811 ] , [ 514229 ]El contador es:  28
Para:[ 514229 ] , [ 832040 ]El contador es:  29
Para:[ 832040 ] , [ 1346269 ]El contador es:  30
Para:[ 1346269 ] , [ 2178309 ]El contador es:  31
Para:[ 2178309 ] , [ 3524578 ]El contador es:  32
Para:[ 3524578 ] , [ 5702887 ]El contador es:  33
Para:[ 5702887 ] , [ 9227465 ]El contador es:  34
Para:[ 9227465 ] , [ 14930352 ]El contador es:  35
```

```
Para:[ 14930352 ] , [ 24157817 ]El contador es:  36
Para:[ 24157817 ] , [ 39088169 ]El contador es:  37
Para:[ 39088169 ] , [ 63245986 ]El contador es:  38
Para:[ 63245986 ] , [ 102334155 ]El contador es:  39
Para:[ 102334155 ] , [ 165580141 ]El contador es:  40
Para:[ 165580141 ] , [ 267914296 ]El contador es:  41
Para:[ 267914296 ] , [ 433494437 ]El contador es:  42
Para:[ 433494437 ] , [ 701408733 ]El contador es:  43
Para:[ 701408733 ] , [ 1134903170 ]El contador es:  44
Para:[ 1134903170 ] , [ 1836311903 ]El contador es:  45
Para:[ 1836311903 ] , [ 2971215073 ]El contador es:  46
Para:[ 2971215073 ] , [ 4807526976 ]El contador es:  47
Para:[ 4807526976 ] , [ 7778742049 ]El contador es:  48
Para:[ 7778742049 ] , [ 12586269025 ]El contador es:  49
Para:[ 12586269025 ] , [ 20365011074 ]El contador es:  50
Para:[ 20365011074 ] , [ 32951280099 ]El contador es:  51
Para:[ 32951280099 ] , [ 53316291173 ]El contador es:  52
Para:[ 53316291173 ] , [ 86267571272 ]El contador es:  53
Para:[ 86267571272 ] , [ 139583862445 ]El contador es:  54
Para:[ 139583862445 ] , [ 225851433717 ]El contador es:  55
Para:[ 225851433717 ] , [ 365435296162 ]El contador es:  56
Para:[ 365435296162 ] , [ 591286729879 ]El contador es:  57
Para:[ 591286729879 ] , [ 956722026041 ]El contador es:  58
Para:[ 956722026041 ] , [ 1548008755920 ]El contador es:  59
Para:[ 1548008755920 ] , [ 2504730781961 ]El contador es:  60
Para:[ 2504730781961 ] , [ 4052739537881 ]El contador es:  61
Para:[ 4052739537881 ] , [ 6557470319842 ]El contador es:  62
Para:[ 6557470319842 ] , [ 10610209857723 ]El contador es:  63
Para:[ 10610209857723 ] , [ 17167680177565 ]El contador es:  64
Para:[ 17167680177565 ] , [ 27777890035288 ]El contador es:  65
Para:[ 27777890035288 ] , [ 44945570212853 ]El contador es:  66
Para:[ 44945570212853 ] , [ 72723460248141 ]El contador es:  67
Para:[ 72723460248141 ] , [ 117669030460994 ]El contador es:  68
Para:[ 117669030460994 ] , [ 190392490709135 ]El contador es:  69
Para:[ 190392490709135 ] , [ 308061521170129 ]El contador es:  70
Para:[ 806515533049393 ] , [ 1304969544928657 ]El contador es:  73
Para:[ 1304969544928657 ] , [ 2111485077978050 ]El contador es:  74
Para:[ 2111485077978050 ] , [ 3416454622906707 ]El contador es:  75
Para:[ 3416454622906707 ] , [ 5527939700884757 ]El contador es:  76
Para:[ 5527939700884757 ] , [ 8944394323791464 ]El contador es:  77
Para:[ 8944394323791464 ] , [ 14472334024676221 ]El contador es:  78
```

Figura 3

La gráfica es:

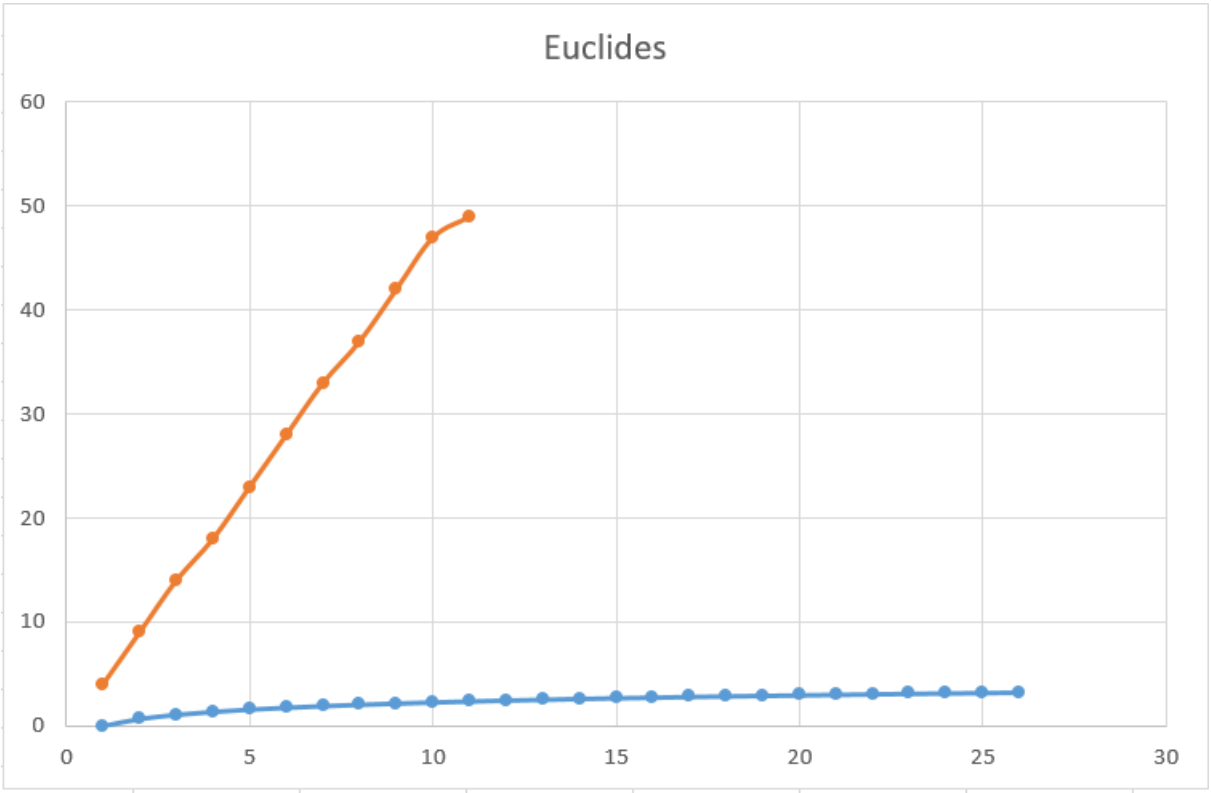

Figura 4

-Proponer una función g(n) tal que Euclides $\in$ O(g(n)) y g(n) sea mínima, en el sentido de que si Euclides $\in$ O(h(n)), entonces g(n) $\in$ O(h(n)).
Por los resultados obtenidos en la última gráfica proponemos que la complejidad del algoritmo Euclides es de tipo logarítmica y propones la función O($\log_2(n)$) ya que O($\log_2(n)$) pasa ser cota superior cumpliendo así la definición de O($\log_2(n)$).

-Conclusiones

No creia que el algoritmo de Euclides fuera de tipo logaritmico, la cuestion de los tiempos es algo comnplicado de medir, al menos para mi nivel de programacion actual, para solucuionar esto, implemente un contador en las veces que entraba a la funcion, siendo esta la mejor forma para lograr contabilizar el tiempo computable.

4. Conclusión General

Como lo mencione al principio del trabajo, los algoritmos son uina herramienta que ocupamos todos en nuestro dia a dia, por eso es importante encontrar la manera de analizar estos algoritmos y determinar si son la mejor solucion a nuestras necesidades, con este comienzo vemos que el analisis de algoritmos no es cosa de gracia, es un tema bastante complejo que requiere de mucha atencion.

5. Anexos

Select-Sort

Select-Sort(A[0...n-1])	**Costo**	**#Pasos**
1.- **for j ← 0 to n-2 do**	**C1**	**n**
2.- **k ← j**	**C2**	**n-1**
3.- **for i ← j+1 to n-1 do**	**C3**	$\sum_{i=0}^{n-1} t_i$
4.- **if A[i] < A[k] then**	**C4**	$\sum_{i=0}^{n-1} (t_i - 1)$
5.- **k ← i**	**C5**	$\sum_{i=0}^{n-1} R_i$
6.- **intercambia(A[j],A[k])**	**C6**	**n-1**

El peor de los casos se presenta cuando el arreglo esta ordenado en forma decreciente por lo que:
$t_i = n - i$ **y**
$R_i = n - i - 1 = t_i - 1$

Por lo tanto tenemos que:
T(n) = C1n + C2(n-1) + C3($\sum_{i=0}^{n-1}(n-i)$) **+ C4**($\sum_{i=0}^{n-1}(n-i-1)$) **+ C5**($\sum_{i=0}^{n-1}(n-i-1)$) **+ C6(n-1)**

T(n) = C1n + C2n -C2 + C3($\sum_{i=0}^{n-1} n$) **- C3**($\sum_{i=0}^{n-1} i$) **+ C4**($\sum_{i=0}^{n-1} n$) **- C4**($\sum_{i=0}^{n-1} i$) **- C4**($\sum_{i=0}^{n-1} 1$) **+ C5**($\sum_{i=0}^{n-1} n$) **-C5** ($\sum_{i=0}^{n-1} i$) **- C5**($\sum_{i=0}^{n-1} 1$) **+ C6n -C6**

T(n) = (C1+C2+C6)n + (C3+C4+C5)(n^2) **- (C3+C4+C5)**($\sum_{i=1}^{n}(i-1)$) **- (C4+C5)n -(C2+C6)**

T(n) = (C3+C4+C5)n^2 **-(C3+C4+C5)**($\frac{n(n+1)}{2}$ **- n) +(C1+C2-C4-C5+C6)n - (C2+C6)**

T(n) = ($\frac{C3+C4+C5}{2}$)n^2 **+ (C1+C2+C3+C6)n - (C2+C6)**

$\therefore T(n) \quad \in \quad O(n^2)$

6. Bibliografía

- **Brassard, G. (1997). Fundamentos de Algoritmia. España: Ed. Prentice Hall. ISBN**