TAREA 1

Análisis de algoritmos

David Alfonso Velasco Sedano

19 de Agosto del 2017

Parte 1

¿A qué orden de complejidad temporal y espacial pertenecen los algoritmos conocidos que resuelven los siguientes problemas? Justificar su respuesta brevemente (como las diapositivas anteriores).

1. Definir si dos cadenas de texto son iguales
    a. Temporal: Lineal. Simplemente tenemos que hacer un recorrido por las 2 cadenas y hacer comparación en el mismo elemento.
    b. Espacial: Lineal. Es 2N, dependiendo del largo de las cadenas.

2. Cálculo de la mediana en una lista desordenada de números enteros.
    a. Temporal: Semi-Lineal. Asumiendo que usamos el algoritmo de heapsort para ordenar la lista de elementos. De ahí hacemos un cálculo para obtener la mediana (es constante la última parte)
    b. Espacial: Lineal, dependiendo de la lista de elementos

3. Multiplicación de matrices.
    a. Temporal: Cúbica, ya que se ocupan 3 ciclos anidados para exitosamente hacer la operación.
    b. Espacial: Cuadrática, estamos utilizando y almacenando matrices de mxn.

4. Conteo de los números primos en el rango [a ... b].
    a. Temporal: Cuadrática, necesitamos 2 ciclos. Uno para hacer el conteo normal, y otra para ver cuáles otros números primos hemos almacenado y dividir el número actual por ellos.
    b. Espacial: Lineal, necesitamos un arreglo o una lista para almacenar los elementos que estemos encontrando.

5. Encontrar el número de veces en que se tiene que dividir un número entero entre 7 hasta llegar a la unidad.
    a. Temporal: Constante, se puede utilizar una fórmula para obtener el valor
    b. Espacial: Constante, a lo máximo necesitamos una variable para almacenar los valores

6. Encontrar el número de agrupaciones de N dígitos (iguales o diferentes) que sumados no sean mayores a un valor M. Considerar que $\{0, 1, 2, 3\} \neq \{1, 0, 2, 3\}$
    a. Temporal: Cuadrática, ya que tenemos que recorrer los 2 arreglos. Es posible que uno de los 2 arreglos sea recorrido de manera anidad con respecto al otro, para saber las combinaciones ideales
    b. Espacial: Lineal, sería 2N ya que tenemos 2 arreglos de números

7. Registrar todos los pares (a, b) de números enteros (de 1 a N) que satisfagan la desigualdad: cos(a) * sin(b) ≤ b / 2ª
    a. Temporal: Cuadrático, se tiene que recorrer b veces a para encontrar las combinaciones
    b. Espacial: Cuadrática, tendremos una serie de arreglos para guardar los pares encontrados

Parte 2

Algoritmo de Euclides: calcula el máximo común divisor de dos números enteros A, B

- Primer algoritmo interesante de la historia.
- Complejidad temporal y espacial en el peor caso: O(lgn).
    - A y B son dos números consecutivos de la serie de Fibonacci.
- Comprobar de forma práctica (a posteriori) tal complejidad

8. Implementarlo en su lenguaje de programación favorito (no más de 4 líneas de código). Suponer A > B.

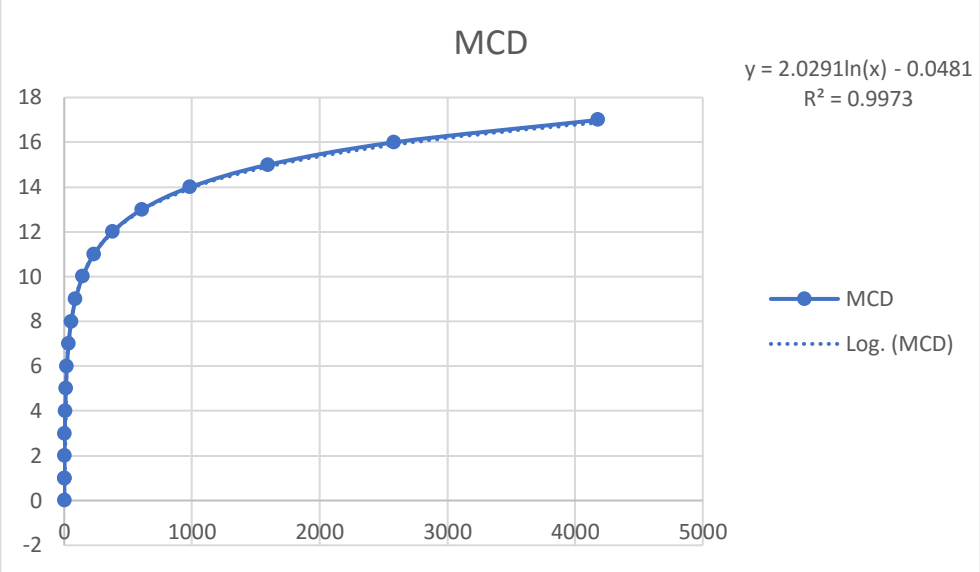
```
public static int mcd(int a, int b) {
        if(b <= 0)
                return a;
        return mcd(b, a%b);
}
```

9. Contar el número de divisiones que toma el cálculo GCD(A, B), donde A = Fibonacci(n), B = Fibonacci(n – 1), para n = 2 hasta 16, y reportarlo en una tabla.

n	A	B	# Div
1	0	-	
2	1	0	0
3	1	1	1
4	2	1	1
5	3	2	2
6	5	3	3
7	8	5	4
8	13	8	5
9	21	13	6
10	34	21	7
11	55	34	8
12	89	55	9
13	144	89	10
14	233	144	11
15	377	233	12
16	610	377	13

17	987	610	14
18	1597	987	15
19	2584	1597	16
20	4181	2584	17

10. Apoyado de Excel, crear una gráfica de dispersión (ó XY) tomando A como las abscisas y el conteo de divisiones de GCD(A, B) como las ordenadas. Sobre los datos de la gráfica, agrega una línea de tendencia (trendline). El tipo de tendencia debe ser logarítmica. Seleccionar la opción Presentar ecuación en el gráfico.



11. Con la ecuación mostrada, demuestre: g(N) ∈ O(lg N)

g(N) **ϵ** O(f(N))
g(N) = 2ln(N)-0.04
f(N) = lg(N)
2ln(N)-0.04 ≤ Klg(N)
(2ln(N)-0.04) / lg(N) ≤ K
N = 2
K ≥ (2ln(2)-0.04) / lg(2)
K ≥ (2(.69)-0.04) / 1
K ≥ (1.38-0.04)
K ≥ 1.35

N	2ln(N)-.04	Klg(N) | K = 1.35*	Klg(N) | K = 2.35
1	-0.04	0.00	0.00
2	1.35	1.39	2.35
3	2.16	2.20	3.72

4	2.73	2.77	4.69
5	3.18	3.22	5.45
6	3.54	3.58	6.07
7	3.85	3.89	6.59
8	4.12	4.16	7.04
9	4.35	4.39	7.44
10	4.57	4.61	7.79

g(N) pertence a O(lgN) con valores de K igual o mayor a 1.35.
*Los números en está columna son ligeramente más grandes que de la función g(N). Esto es por el hecho de que redondeé el valor de K para que sea fácil de enseñar.

12. Identifique el mejor caso y su complejidad g(N)
El mejor caso es cuando 'a' es divisible entre 'b' haciendo que a % b = 0 desde el inicio. Haciendo la complejidad constante (g(N) = 1).

13. De la respuesta a la pregunta anterior ¿Se cumple g(N) $\in$ Ω(lg N)? Justifica

g(N) **ϵ** Ω(f(N))
g(N) = 1
f(N) = lg(N)
1 ≥ Klg(N)
(1 / lg(N) ≥ K
N = 2
K ≤ (1 / lg(2))
K ≤ 1 / 1
K ≤ 1

Se cumple en los escenarios donde K sea menor o igual a 1. En el caso de K = 1, se cumple cuando N está en el rango de 1 a 2.

N	1	Klg(N) | K = 1	Klg(N) | K = 0.5
1	1.00	0.00	0.00
2	1.00	1.00	0.50
3	1.00	1.58	0.79
4	1.00	2.00	1.00
5	1.00	2.32	1.16

Referencias

- 15 julio de 2017, Algoritmo de Euclides, http://tinyurl.com/y7wlfemw
- 16 agosto de 2017, Sorting algorithm, http://tinyurl.com/9kpsx8b