Analisis de Algoritmos

MS705167

8/27/2015

Angel de Jesus Bañuelos Sahagun

Tarea Parte 1.

¿A qué orden de complejidad temporal y espacial pertenecen los algoritmos conocidos que resuelven los siguientes problemas? Justificar su respuesta.

NOTA: Se agrega clase, donde se realizó el análisis Homework1.java.

1. Determinación de igualdad de dos cadenas de texto.
    a. Temporal: $g(N) = 3N + 5 \epsilon O(N^2)$
        i. Debido a que se compara elemento por elemento y en cuanto se encuentre uno diferente se termina.
    b. Espacial: $2N + 1$
        i. Debido a que son dos arreglos de caracteres y un contador.
    c. Ver: isStringEquals
2. Cálculo de la mediana en una lista desordenada de números enteros.
    a. Temporal: $g(N) = n^2 + 3N + 11 \epsilon O(N \log N)$
        i. Debido a que se uso el ordenamiento secuencial que es cuadrático.
    b. Espacial: $N + 4$
        i. Un arreglo, 4 variables que apoyan.
    c. Ver: medianCalculation

3. Multiplicación de matrices.
    a. Temporal: $g(N) = 6N^3 + 6N + 6 \epsilon O(N^3)$
        i. Se hace el uso de 3 ciclos anidados
    b. Espacial: $3N^2 + 3$
        i. La información es guardada en 3 matrices.
    c. Ver: matrixMultiplication

4. Conteo de los números primos en el rango [a … b].
    a. Temporal: $g(N) = N^2 + 5N + 7 \epsilon O(N \log N)$
        i. U
    b. Espacial: 5
        i. Solo se usan variables contadores y banderas.
    c. Ver: countPrimeNumbersBetween

5. Encontrar el número de veces en que se tiene que dividir un número entero entre 3 hasta llegar a la unidad.
    a. Temporal: $g(N) = 4N + 5$
        i. Uno solo ciclo varias operaciones.

b. Espacial: 4
    i. Se dos contadores. Y otras dos variables.
c. Ver: countNumberDivided

6. Encontrar el número de agrupaciones de N dígitos (iguales o diferentes) que sumados no sean mayores a M. Considerar por ejemplo: $\{0, 1, 2, 3\} \neq \{1, 0, 2, 3\}$

7. Registrar todas los pares (a, b) de números enteros (de 1 a N) que satisfagan la desigualdad: $\cos(a) \cdot \sin(b) \leq b / 2a$.

Tarea 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: $O(\lg n)$.

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.

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.

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.
11. 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.
12. Con la ecuación mostrada, demuestre: $g(N) \in O(\lg N)$.

[PILÓN]

13. Identifique el mejor caso y su complejidad g(N).
14. ¿ Se cumple $g(N) \in \Omega(\lg N)$?

```java
package ada.session1;

import com.utils.Utils;
import com.utils.sort.Selection;

/**
 *
 * @author Angel.Sahagun
 */
public class Homework1 {

    public static boolean isStringEquals(String objToCompare1, String objToCompare2) { // Temp = 3N + 5 , Espacial = 2N + 1
        char[] objToCompareChar1 = objToCompare1.toCharArray(); // 1
        char[] objToCompareChar2 = objToCompare2.toCharArray(); // 1
        for (int i = 0; i < objToCompareChar1.length; i++) { // 1 + (N+1) + N
            if (objToCompareChar1[i] != objToCompareChar2[i]) {// N(3)
                return false;
            }
        }
        return true;
    }

    public static int medianCalculation(int[] array) {
        if (!Utils.isSorted(array)) { // 5N + 3
            Selection.sort(array); //
            int medianIndex = (array.length) / 2;
            return array[medianIndex];
        }
        return 0;
    }

    public static void main(String[] args) {
        if (isStringEquals("Angel", "Angel")) {
            System.out.println("Strings are identical");
        } else {
            System.out.println("Strings are no equals ");
        }
        int[] array = Utils.createArray(15, 3, 30); ///{9,5,4,8,3,1,6,9,7,5,2};
        Utils.printArray(array);
        System.out.println("Median = " + medianCalculation(array));
        Utils.printArray(array);
        int[][] matrixA = new int[3][3];
        int[][] matrixB = new int[1][3];
        // filling MatrixA
        matrixA[0][0] = 1;
        matrixA[1][0] = 2;
        matrixA[2][0] = 3;
        matrixA[0][1] = 4;
        matrixA[1][1] = 5;
        matrixA[2][1] = 6;
        matrixA[0][2] = 7;
        matrixA[1][2] = 8;
        matrixA[2][2] = 9;
```

```java
    // Filling MatrixB
    matrixB[0][0] = 1;
    matrixB[0][1] = 3;
    matrixB[0][2] = 9;

    // multiplicating
    int[][] matrixReutlt = matrixMultiplication(matrixA, matrixB);

    for (int i = 0; i < matrixReutlt.length; i++) {
        for (int j = 0; j < matrixReutlt[0].length; j++) {
            System.out.println("Matrix [" + i + "][" + j + "] =" + matrixReutlt[i][j]);
        }
    }
    System.out.println(" Count of prime numbers between (" + 1 + "," + 1000 + "): " + countPrimeNumbersBetween(1, 1000));

    System.out.println(" Count 3: " + countNumberDivided(243, 3));

    System.out.println(" Greatest Common Divisor: " + greatestCommonDivisor(5, 8));

}

public static int[][] matrixMultiplication(int[][] matrixA, int[][] matrixB) { // 6N^3 + 6N + 6

    int[][] matrixReutlt = new int[matrixA.length][matrixA[0].length];
    for (int i = 0; i < matrixA.length; i++) // 2N + 2
    {
        for (int j = 0; j < matrixB.length; j++) // 2N  + 2
        {
            for (int k = 0; k < matrixA.length; k++) // 2N + 2
            {
                matrixReutlt[i][k] = matrixA[i][k] * matrixB[j][k] + matrixReutlt[i][k]; // 6N^3

            }
        }
    }
    return matrixReutlt;
}

public static int countPrimeNumbersBetween(int start, int end) { // N^2 + 5N + 7
    int counter = 0; // 1
    for (int i = start; i < end; i++) { // 1 + (N + 1) + N
        if (i != 1 && isPrime(i)) { // N^2 + 3N + 4
            counter++; //
        }
    }
    return counter;
}

public static boolean isPrime(int i) { // N^2 + 2N + 4
    boolean isPrime = true; // 1
    for (int j = 2; j < i; j++) { // 1 + N + 1 + N
        if (i % j == 0) { // N^2
            isPrime = false; // 1
            break;
        }
    }
    return isPrime;
}
```

```java
    public static int countNumberDivided(int num, int divNum) { // Temporal: 4N + 5, Espacial: 4
        int counter = 0; // 1
        if (num % divNum == 0) { // 1
            int aux = num / divNum;  // 1
            counter++; // 1
            while (aux > 1) { // N
                if (aux % divNum == 0) { // N
                    aux = aux / divNum; //N
                    counter++; // N
                }else{
                    counter = 0; // 1
                    break;
                }
            }
        }
        return counter;
    }

    /**
     * Algoritmo de Euclides
     *
     * @param a
     * @param b
     * @return
     */
    public static int greatestCommonDivisor(int a, int b) {
        int r = 0, div = 0;
        r = a % b;
        div++;
        while (r != 0) {
            a = b;
            b = r;
            r = a % b;
            div++;
        }
        System.out.println("el maximo comun divisor es:" + b);
        System.out.println("el numero de divisiones fue:" + div);
        return div;

    }

}
```