Práctica Individual 1

Problema Número 2

Análisis y Diseño de Datos y Algoritmos / Estructura de Datos y Algoritmos

Grado de Software

Curso 2º

Luis Candelario Luna (luisitiluna@hotmail.com)

Tutor: María Luisa Parody

Índice

1 Enunciado

Dada una matriz de enteros mayores o iguales a cero de dimensión NxN con N potencia de 2, se dice que esa matriz es “equisuma” si los elementos de las 4 submatrices que la componen suman lo mismo y a su vez cada submatriz es también equisuma. Además, una matriz de 2x2 siempre se considera equisuma. La siguiente matriz es equisuma ya que sus cuatro submatrices suman 6 y además esas submatrices son de tamaño 2x2 y por tanto también lo son. Observe también que para el citado ejemplo la suma de todos sus elementos es 24.

1.1 Cuestiones a resolver

1. Defina la función recursiva equisuma. Esta función devolverá un valor entero que represente la suma de todos sus elementos. En caso de que no sea equisuma devolverá el valor -1.

2. ¿Se trata de recursividad simple o múltiple? ¿Final o no final?

3. Implemente la función recursiva definida en el apartado 1 empleando el lenguaje de programación que le ha indicado su profesor.

4. Indique el tamaño del problema, T(n) y la complejidad del algoritmo.

1.2 Lenguaje de programación asignado

El lenguaje de programación asignado es C y el entorno de programación usado es Eclipse.

2. Seguimiento

2.1 Control de seguimiento y tutorías

El alumno debe listar y detallar las visitas a tutoría

Fecha	Profesor	Detalles y dudas resueltas
25/10/2016	María Luisa Parody	La duda se resolvió durante el seguimiento ya que el algoritmo utilizado era demasiado extenso complicándose así su implementación en C.

2.2 Portafolio

Como vemos en el ejercicio el caso base sería una matriz 2x2 por lo tanto únicamente bastará con sumar los elementos de la matriz uno a uno y así obtener la suma de sus elementos.

En el caso de que nos den una matriz de tamaño n =4 o n= 8 (siempre que

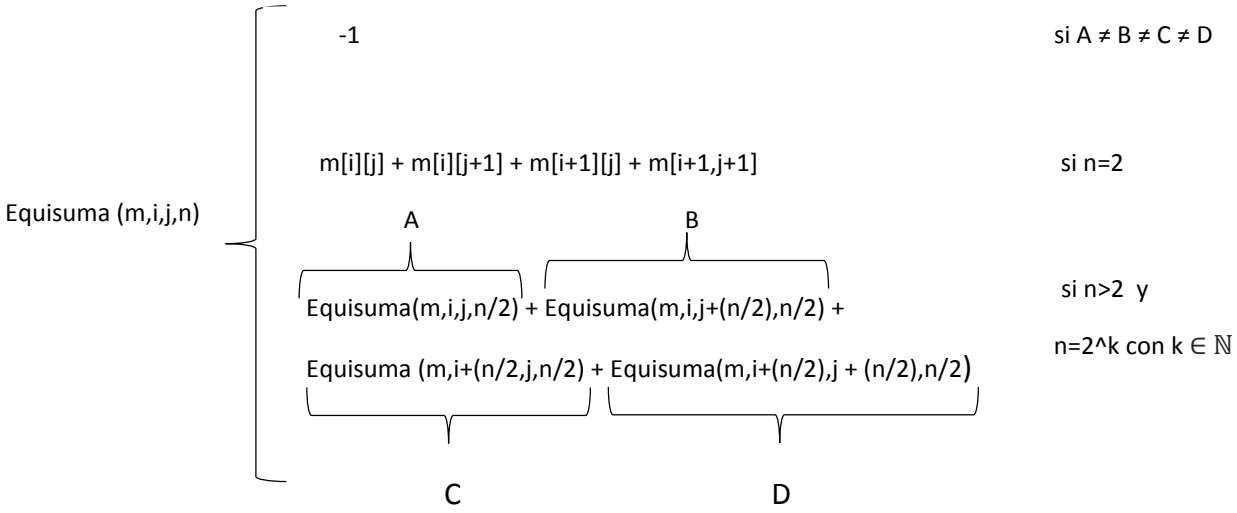
n=2^k con k ∈ ℕ) ésta se subdividirá en n/2 sucesivamente hasta que lleguemos al caso base.

El problema que tuve inicialmente se basaba en que no había pensado en añadir un parámetro más en la llamada (posteriormente incluí "n" como parámetro) por lo tanto se me complicaba mucho su implementación en C.

Una vez añadido, su valor debía dividirse por 2 en cada llamada recursiva con el fin de ir reduciendo la matriz hasta llegar al caso base. De esta forma se simplificó el problema y pude realizarlo sin problemas.

3. Implementación

1. La función recursiva es la siguiente:



2. Se trata de una recursividad múltiple ya que como podemos apreciar en la función recursiva tiene más de una llamada a sí mismo, en nuestro caso tiene 4 llamadas.

Además la función recursiva es no final ya que se trata de una recursividad múltiple.

3. Código:

Declaraciones.h

```
#ifndef DECLARACIONES_H_
#define DECLARACIONES_H_

#define tamanyoMatrix 4

int main(void);
int equisuma(int m[tamanyoMatrix][tamanyoMatrix]);
```

```
int equisumaRecursiva(int m[tamanyoMatrix][tamanyoMatrix], int i, int
j, int n);


#endif
```

Ejercicio.c

```
#include <stdio.h>
#include <stdlib.h>
#include "declaraciones.h"

int main(void) {


      int m [tamanyoMatrix][tamanyoMatrix] =
{{1,3,3,0},{2,0,0,3},{4,0,1,2},{1,1,1,2}};

      int i;
      int j;

      puts("La matriz m es:");
       for ( i = 0; i < tamanyoMatrix; i++ ) {
            for ( j = 0; j <tamanyoMatrix; j++ ) {
               printf("a[%d][%d] = %d\n", i,j, m[i][j] );
            }
       }

       int res = equisuma(m);
       printf("La suma de los elementos de la matriz es %d",res);


       return 0;
}

int equisuma(int m[tamanyoMatrix][tamanyoMatrix]){
      return equisumaRecursiva(m,0,0,tamanyoMatrix);    // <--
Importante cambiar valor aqui
}

int equisumaRecursiva(int m[tamanyoMatrix][tamanyoMatrix], int i, int
j, int n){

      int res = -1;
            int a=0;
            int b=0;
            int c=0;
            int d=0;

            if(n==2){
                  res = m[i][j] + m[i][j+1] + m[i+1][j] + m[i+1][j+1];
            }else if(n>2){
```

```
                a = equisumaRecursiva(m,i,j,n/2);
                b = equisumaRecursiva(m,i,j+(n/2),n/2);
                c = equisumaRecursiva(m,i+(n/2),j,n/2);
                d = equisumaRecursiva(m,i+(n/2),j+(n/2),n/2);

                if(a==b && a==c && a==d && a!= -1){
                        res = a+b+c+d;
                }
        }


        return res;
}
```

3 Análisis y complejidad

El tamaño del ejercicio sería (n/2) ^2 ya que si introducimos una matriz de 4x4 el método realizaría 4 llamadas recursivas. De la misma forma si introducimos una de 8x8 el método realizaría 4 llamadas recursivas y a su vez cada una de ellas tendría 4 llamadas recursivas.

En cuanto al tiempo, cada función tiene 4 llamadas recursivas y cada una de ellas divide el tamaño por 2 por lo tanto:

T(n) = 4T(n/2) + K (Es una constante porque no tiene bucles, etc)

La complejidad del ejercicio sería:

$$\theta\left(n^{log_b a}\right) si \quad a > b^d$$

Ya que a= 4; b=2 y d=1

5 Pruebas y traza

En el caso que nos plantea el ejercicio al ejecutar el código adjuntado en el apartado 3.3 nos aparece en la consola lo siguiente, por lo tanto podemos comprobar que el algoritmo utilizado funciona.

```
 La matriz m es:
a[0][0] = 1
a[0][1] = 3
a[0][2] = 3
```

```
a[0][3] = 0
a[1][0] = 2
a[1][1] = 0
a[1][2] = 0
a[1][3] = 3
a[2][0] = 4
a[2][1] = 0
a[2][2] = 1
a[2][3] = 2
a[3][0] = 1
a[3][1] = 1
a[3][2] = 1
a[3][3] = 2

La suma de los elementos de la matriz es 24
```

Además podemos comprobar que si no cumple las condiciones la consola nos muestra el valor -1 para indicarnos que no es equisuma la matriz introducida.

```
La matriz m es:
a[0][0] = 1
a[0][1] = 3
a[0][2] = -3
a[0][3] = 0
a[1][0] = 2
a[1][1] = 0
a[1][2] = 0
a[1][3] = 3
a[2][0] = 4
a[2][1] = 0
a[2][2] = 1
a[2][3] = 2
a[3][0] = 1
a[3][1] = 1
a[3][2] = 1
a[3][3] = 2

La suma de los elementos de la matriz es -1
```

Por último para probar el caso base bastará con añadir una matriz de tamaño 2 y ejecutar el código como se muestra a continuación (hay que cambiar el valor del `#define tamanyoMatrix 4` contenido en *declaraciones.h* por del `#define tamanyoMatrix 2`)

```
int main(void) {

	int m [2][2] = {{2,2},{2,2}};
```

```
    int i;
      int j;

      puts("La matriz m es:");
       for ( i = 0; i < tamanyoMatrix; i++ ) {
            for ( j = 0; j < tamanyoMatrix; j++ ) {
               printf("a[%d][%d] = %d\n", i,j, m[i][j] );
            }
       }

       int res = equisuma(m);
       printf("La suma de los elementos de la matriz es %d",res);


       return 0;
}

int equisuma(int m[tamanyoMatrix][tamanyoMatrix]){
      return equisumaRecursiva(m,0,0,2);
}
```

Consola:

```
La matriz m es:
a[0][0] = 2
a[0][1] = 2
a[1][0] = 2
a[1][1] = 2
La suma de los elementos de la matriz es 8
```