TC2017 – Análisis y Diseño de Algoritmos **Tarea #3**

Ing. Luis Humberto González G

Nombre: Solución *Matricula:*

Entrega un reporte de solución por escrito a cada uno de los siguientes problemas.

1. *(40 puntos)* Analizar cada uno de los siguientes segmentos de pseudocódigo, e indicar cuál es la complejidad del algoritmo representado. Suponer siempre que la "instrucción" es una operación básica.

CASO A:

```
for (j = 1; j <= n ; j = j * 2)
    instrucción;
```

$4\log_2 n + 7$

CASO B:

```
for (j = 1; j <= n ; j++)
  for (k = 0; k < n/5; k++)
    instrucción;
```

$\frac{4}{5}n^2 + 6n + 3$

CASO C:

```
for (j = 1; j <= n ; j++)
{ k = n;
  while (k >= 1)
   { instrucción;
     k /= 2;
   }
}
```

$4n\log_2(n) + 10n + 3$

CASO D:

```
j = n;
while ( j > 0)
  { for ( k = j; k<=n; k = k*2)
        instrucción;
    j /= 2;
  }
```

$2(\log_2 n)^2 + 12\log_2 n + 13$

$\cancel{2(\log_2 n)^2 + 12\log_2 n + 13}$

2. *(20 puntos)* A continuación se muestran 2 casos en los que para cada uno se muestran 2 algoritmos. Ambos algoritmos en cada caso, sirven para resolver el mismo problema.
   a) Identifica cuál es el problema que está resolviéndose en cada caso.
   b) Haz un análisis de la complejidad de cada algoritmo.
   c) Indica para cada caso, cuál es el algoritmo que más convendría elegir para la implementación de la solución al problema, justificando tu respuesta.

CASO A:

Algoritmo A1:

```
s = 0;
for (int x = 1; x<=n; x++)
  for (int y = 1; y<=n; y++)
    if (x == y) then
       s += a[x,y];
```

Suma Diag. Princ

$4n^2 + 7n + 4$

Algoritmo A2:

```
s = 0;
for (int j = 1; j<=n; j++)
  s += a[j,j];
```

$4n + 4$

CASO B:

Algoritmo B1:

```
r = 1;
for (int j=1; j<=n; j++)
  r = r * x
```

4n

X

4n + 4

Algoritmo B2:

```
r = 1;
while (n > 0)
{ if (n % 2 != 0) r *= x;
  x *= x;
  n /= 2;
}
```

0

$7 \log_2 n + 10$

3. *(40 puntos)* Desarrolla un algoritmo que dado un arrelo **A** que contiene **n** distintos enteros positivos, generé un arreglo de dos dimensiones **B**, en donde en la posición **B[i][j] = B[j][i] = A[i]+A[i+1]+…+A[j-1]+A[j].** Se calificará eficiencia.

```
for (int i=ϕ; i<N; i++)
{
    B[i][i] = A[i];
    for (j=i+1; j<N; j++)
    {
        B[j][i] = B[i][j] = B[i][j-1]+A[j].
    }
}
```

1/3

Análisis y Diseño de Algoritmos | ***Tarea #5***

Ing. Luis Humberto González G

Nombre: | *Matricula:*

1) *(10 puntos) Contesta las preguntas en base al* siguiente algoritmo

```
s = 0
for (int i=1; i<=n; i++)
  s = s + i * i
return s
```

a) ¿Qué realiza el algoritmo? $\sum_{i=1}^{n} i^2$.

b) ¿Cuál es la operación básica? s = s + i*i

c) ¿Cuántas veces se realiza la operación básica? n

d) ¿Cuál es la complejidad del algoritmo? 4n+5

e) ¿Cuál es el orden del algoritmo? O(n)

2) *(40 puntos)* ¿Cuál es el orden de cada uno de los siguentes algoritmos?

a) // Entrada: Matriz A[0..n-1, 0..n-1] de números reales.

```
for (int i=0; i<= n-2; i++)
  for (int j=i+1; j<n; j++)
   for (int k=i; k<n; k++)
     A[i,k] = A[j,k] –A[i,k] * A[j,i] / A[i,i]
```

$O(n^3)$

b) //Entrada: Un entero positivo (n)

```
int Q(int n){
  if (n==1)
    return 1
  return n;
}
```

O(1)

c) //Entrada: Un entero positivo (n)

```
int P(int n){
  int acum = 0;
  if (n==0)
     return 0
  else
    if (n % 2 == 0)
         for (int i=1; i<n; i*=2)
              acum +=I;
else
       return n;
}
```

$O(\log_2 n)$

2/3

d) **//Entrada: Un entero positivo (n)**

```
int a=0;
int b=n;
for (int i=1; i<= 2*n; i++) {
        a++;
        b+=a;
        c*=(a+b);
}
b=c+a;
```

$O(n)$

e) **//Entrada: Un entero positivo (n)**

```
int acum=1;
for (int i=1; i<=n; i++)
   for (int j=i;j<=n; j++)
      acum+=(i*j);
```

$O(n^2)$

f) **//Entrada: Un entero positivo (n)**

```
int b=1;
j = n;
while (j>=0) {
    b++;
    j--;
}
```

$O(n)$

g) **//Entrada: Un entero positivo (n)**

```
int acum=1;
for (int i=1; i<=n; i+=2)
   for (int j=i;j<=n; j++)
      acum+=(i*j);
```

$O(n^2)$

h) **//Entrada: Un entero positivo (n)**

```
int acum=1;
for (int i=1; i<=n; i*=2)
   for (int j=i;j<=n; j+=2)
      acum+=(i*j);
```

$O(n \log_2 n)$

3) *(50 puntos)* Escribe un algoritmo que dado un arreglo que contiene enteros positivos, regrese la suma de los enteros impares contenidos en el arreglo.

a) Realiza el algoritmo en forma iterativa, ¿Cuál es el orden del algoritmo?

b) Realiza el algoritmos en forma recursiva ¿Cuál es el orden del algoritmo?

3) a)

```
int acum=0
for (int iA=0; iA < n ; iA++)
{
    if ( arr[iA] %2 != 0)
        acum += arr[iA];

}
return acum;
```

b)

```
int sumaImpar(int n)
{   if(n==1)
        return (arr[0]%2!=0? arr[0]:0);

    retur arr[n-1]%2!=0?arr[n-1]:0 +
          sumaImpar(n-1);
}
```

1/2

Análisis y Diseño de Algoritmos | **Tarea #6**

Ing. Luis Humberto González G

Nombre: Solución *Matricula:*

1) *(50 puntos)* Soluciona las siguientes ecuaciones recursivas, llegando a su forma cerrada.

   a. $T(n) = T(n/2)+1 \quad n>1;\ T(1) = 1$ — $T(n) = \log_2 n + 1$

   b. $T(n) = 3T(n-1) \quad n>1;\ T(1) = 4$ — $4 \cdot 3^{n-1}$ ó $4/3\ 3^n$

   c. $T(n) = T(n/2) + n \quad n>1;\ T(1) = 1$ — $2n - 1$

   d. $T(n) = 3T(n/4)+2 \quad n>1;\ T(1) = 2$ — $3n^{\log_2 3} - 1$

   e. $T(n) = T(n-2)+1 \quad n>2;\ T(2) = 1$ — $n/2$

2) Escribe un algoritmo recursivo que dado una matriz cuadrada de ***nxn***, que contiene enteros positivos, regrese la cantidad de casillas con valor mayor a 100,

   a) *(30 puntos)* Realiza el algoritmo recursivo.

   b) (10 puntos) ¿Cuál sería la formula recursiva del tiempo de ejecución?

   c) (10 puntos) Encuentra la solución de la formula recursiva del inciso b.

a)

```
int suma(int ri, rf, ci, cf)
{   if(ri==rf && ci==cf)
        return (mat[ri][cf] > 100 ? 1 : 0);
    int rm = (ri+rf)/2;
    int cm (ci+cf)/2;
    return suma(ri, rm, ci, cm) + suma(ri, rm, cm+1, cf)
        + suma(rm+1, rf, ci, cm) + suma(rm+1, rf, cm+1, cf)
}
```

b $$T(n) \begin{cases} 1 & n=1 \\ 4T(n/2)+1 & n>1 \end{cases}$$

Tarea 6 2/2

c)

$$T(n)=4T(n/2)+1$$

$$n=2^k$$

$$T(2^k)=4T(2^{k-1})+1$$

$$T(2^k)=4(4T(2^{k-1})+1)+1$$

$$T(2^k)=4^2T(2^{k-1})+4^1+4^0$$

$$T(2^k)=4^kT(\cancel{2^{k-k}}\,1)+4^{k-1}+\dots+4^1+4^0$$

$$T(2^k)=\sum_{i=0}^{k}4^i$$

$$T(2^k)=\frac{4^{k+1}-1}{4-1}$$

$$T(2^k)=\frac{4\cdot 4^k-1}{3}$$

$$T(n)=\frac{4\cdot 4^{\log_2 n}-1}{3}$$

$$T(n)=\frac{4\cdot n^{\log_2 4}-1}{3}$$

$$\boxed{T(n)=\frac{4n^2-1}{3}}$$