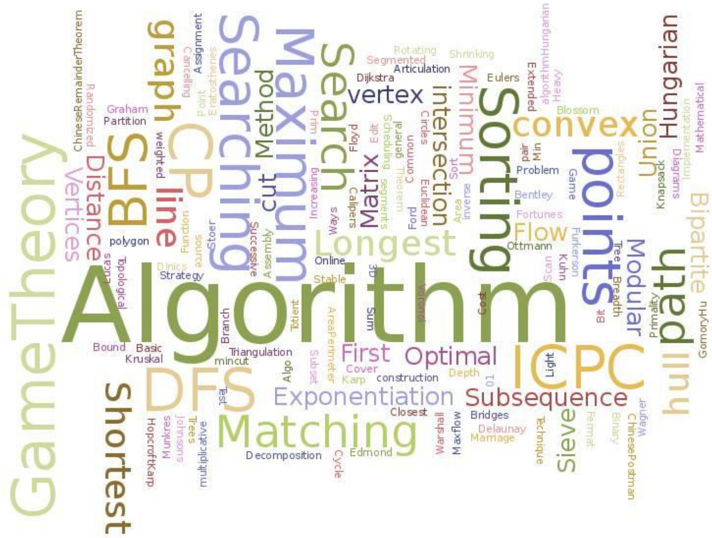
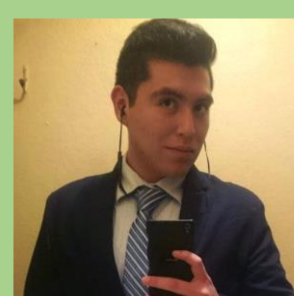
Análisis de algoritmos





Vargas Romero Erick Efraín

Grupo: 3CM2

Prof.: Franco Martínez Edgardo

Tarea 04: Análisis de algoritmos no recursivos

1

```
for(i=1;i<n;i++)
        for(j=0;j<n-1;j++)
        {
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
        }
```

n – 2 (for j)

n – 2 (for i)

Tenemos como resultado:

$$(n-2)(n-2) = n^2 - 4n + 4$$

$$\therefore O(n^2)$$

2

```
polinomio=0;
for(i=0;i<=n;i++)
{
        polinomio=polinomio*z + A[n-i];
}
```

n + 1

Tenemos como resultado:

$$n + 1$$

$$\therefore O(n)$$

3

```
for i = 1 to n do
        for j = 1 to n do
                C[i,j] = 0;
                for k = 1 to n do
                        C[i,j] = C[i,j] + A[i,k]*B[k,j];
```

n

n

n

Tenemos como resultado:

$$(n)(n\,(n)) = n^3$$

$$\therefore O(n^3)$$

4

```
anterior = 1;
actual = 1;
while (n>2)
{
   aux = anterior + actual;
   anterior = actual;
   actual = aux;
   n = n - 1;
}
```

n-2

Tenemos como resultado:

$$(n-2) = n$$

$$\therefore O(n)$$

5

```
for (i = n - 1,j=0;i>=0;i--,j++)
            s2[j]= s[i];

for (i = 0,i<n;i++)
            s[i]= s2[i];
```

n

n

Tenemos como resultado:

$$n + n = 2n$$

$$\therefore O(n)$$

9

```
func Producto2Mayores(A,n)
if(A[1] > A[2])                    ← 1
            mayor1 = A[1];
            mayor2 = A[2];
else                               ← 1
            mayor1 = A[2];
            mayor2 = A[1];
i = 3;

while(i<=n)                        ← n-3
            if(A[i] > mayor1)
                        mayor2 = mayor1;
                        mayor1 = A[i];
            else if (A[i] > mayor2)
                        mayor2 = A[i];
            i = i + 1;

return = mayor1 * mayor2;
fin
```

Tenemos como resultado:

$$2 + n - 3 = n - 1$$

$$\therefore O(n)$$

10

```
func OrdenamientoIntercambio(a,n)
for (i=0; i<n-1; i++)              ← n-1
            for (int j=i+1; j<n;j++)   ← Σ_{i=1}^{n} i
                        if (a[ j ] < a[ i ])
                        {
                                    temp=a[ i ];
                                    a[ i ]=a[ j ];
                                    a[ j ]=temp;
                        }
fin
```

Tenemos como resultado:

$$n \sum_{i=1}^{n} i \approx n^2$$

$$\therefore O(n^2)$$

```
11
func MaximoComunDivisor(m, n)
{
        a=max(n,m);
        b=min(n,m);
        residuo=1;
        mientras (residuo > 0)          <-- log_b a
        {
                residuo=a mod b;
                a=b;                    } 3
                b=residuo;
        }
        MaximoComunDivisor=a;
        return MaximoComunDivisor;
}
```

Tenemos como resultado:

$$(\log_b(a))3 = 3\log_b(a)$$

$$\therefore O(\log_b(a))$$

```
12
Procedimiento BurbujaOptimizada(A,n)
        cambios = "No"
        i=0
        Mientras i< n-1 && cambios != "No" hacer   <-- n-1
                cambios = "No"
                Para j=0 hasta (n-2)-i hacer       <-- n-2
                        Si(A[i] < A[j]) hacer
                                aux = A[j]
                                A[j] = A[i]
                                A[i] = aux          } 4
                                cambios = "Si"
                        FinSi
                FinPara
                 i= i+1
        FinMientras
fin Procedimiento
```

Tenemos como resultado:

$$(n-1)\big((n-2)(4)\big) =$$

$$4n^2 - 12n + 8$$

$$\therefore O(n^2)$$

```
13
Procedimiento BurbujaSimple(A,n)
    sum_{i=0}^{n-2} i   para i=0 hasta n-2 hacer   <-- n-2
                        para j=0 hasta (n-2)-i hacer
                                si (A[j]>A[j+1]) entonces
                                        aux = A[j]
                                        A[j] = A[j+1]
                                        A[j+1] = aux
                                fin si
                        fin para
                fin para
fin Procedimiento
```

Tenemos como resultado:

$$\left((n-2)\left(\sum_{i=0}^{n-2} i\right)\right) \approx$$

$$(n-2)(n-2)$$

$$n^2 - 4n + 4$$

$$\therefore O(n^2)$$

```
14
Procedimiento Ordena(a,b,c)
{
    if(a>b)                          ← 1
        if(a>c)                      ← 1
            if(b>c)                  ← 1
                salida(a,b,c);
            else                     ← 1
                salida(a,c,b);
        else                         ← 1
            salida(c,a,b);
    else                             ← 1
        if(b>c)                      ← 1
            if(a>c)                  ← 1
                salida(b,a,c);
            else                     ← 1
                salida(b,c,a);
        else                         ← 1
            salida(c,b,a);
}
```

3 = 1 + 1 + (1 + 1 + 1) + 1 + (1 + 1 + 1)

Obtenemos

3

$\therefore O(c)$

```
15
Procedimiento Seleccion(A,n)
    para k=0 hasta n-2 hacer                          ← n-2
        p=k
        para i=k+1 hasta n-1 hacer                    ← Σ_{i=1}^{n-1} i
            si A[i]<A[p] entonces                     ← 1
                p=i
            fin si
        fin para
        temp = A[p]                                   ← 3
        A[p] = A[k]
        A[k] = temp
    fin para
fin Procedimiento
```

Tenemos como resultado:

$$\left((n-2)\left(3+\sum_{i=1}^{n-1} i\right)\right) \approx$$

$$(n-2)(n+2)$$

$$n^2-4$$

$$\therefore O(n^2)$$

Conclusión

Si hacemos una comparación con la tarea 02 podemos notar que no requerimos de hacer un análisis tan detallado ya que tendremos una muy buena aproximación del algoritmo tomando solo el orden 0