INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

# Práctica 2

*Algoritmos básicos II*

**Unidad académica:** Análisis de Algoritmos
**Profesor a cargo:** Dra. Sandra Díaz Santiago
**Grupo:** 3CV1
**Realizada por:**


- Medina Juárez Jesús Booz
- Ríos Altamirano Alam Yael


**Sesión de laboratorio:** 28 de febrero del 2018
**Fecha de entrega:** 2 de marzo del 2018

# Diseño de algoritmos

## Ejercicio 1

| maximominimo(A[n]) | Costo | #Operaciones |
|---|---|---|
| res [2] := A[0] // arreglo de dos posiciones, ambas inician en A[0] | C1 | 1 |
| for i := 0 to n | C2 | n |
| if(A[n] < res[min]) | C3 | n |
| res[min] := A[n] | C4 | n |
| if(A[n] > res[max]) | C3 | n |
| res[max] := A[n] | C4 | n |
| return res | C5 | 1 |

Tiempo polinomial y notación asintótica

$T(n) = C_1 + (C_2 + 2C_3 + 2C_4)n + C_5$

$T(n) = a + bn$

$\boldsymbol{T(n) = O(n)}$

$Demostrado\ que\ existe\ c\ tal\ que\ \ a + bn \leq cn$

$suponiendo\ que\ c = a + b$

$a + bn \leq (a + b)n$

$Es\ cierto\ para\ n \geq 1$

## Ejercicio 2

Exponenciación modular rápida (logarítmica)

| exponenciacionModularRapida(a, m, n) | Costo | #Operaciones |
|---|---|---|
| res := 1 | C1 | 1 |
| a := a mod n | C2 | 1 |
| while (m>0) | C3 | $log_2 n$ |
| if (odd(m)) | C4 | $log_2 n$ |
| res = (res * a) mod n | C5 | $log_2 n$ |
| m = m >> 1 | C6 | $log_2 n$ |
| a = (a * a) mod n | C7 | $log_2 n$ |
| return res | C8 | 1 |

Tiempo polinomial y notación asintótica

$T(n) = C_1 + C_2 + (C_3 + C_4 + C_5 + C_6 + C_7)log_2 n$

$T(n) = a + b(log_2 n)$

$\boldsymbol{T(n) = O(log_2 n)}$

$Demostrado\ que\ existe\ c\ tal\ que\ \ a + b(log_2 n) \leq c(log_2 n)$

$suponiendo\ que\ c = a + b$

$a + b(log_2 n) \leq (a + b)log_2 n$

$Es\ cierto\ para\ n \geq 1$

## Ejercicio 3

| selectionsort(A[]) | Costo | #Operaciones |
|---|---|---|
| for i := 0 to n | C1 | n |
| min := I | C2 | n |
| for j := (i + 1) to n | C3 | $\sum n$ |
| if A[j] < A[i] | C4 | $\sum n$ |
| min := j | C5 | $\sum n$ |
| A[i],A[min] := A[min],A[i] | C6 | n |
| return A | C7 | 1 |

Tiempo polinomial y notación asintótica

$$T(n) = C_1 n + C_2 n + C_3\left[\frac{1}{2}n^2 + \frac{1}{2}n\right] + C_4\left[\frac{1}{2}n^2 + \frac{1}{2}n\right] + C_5\left[\frac{1}{2}n^2 + \frac{1}{2}n\right] + C_6 n + C_7$$

$$T(n) = an^2 + bn + c$$

$$\boldsymbol{T(n) = O(n^2)}$$

$$Demostrado\ si\ existe\ d\ tal\ que\ \ an^2 + bn + c \leq dn^2$$

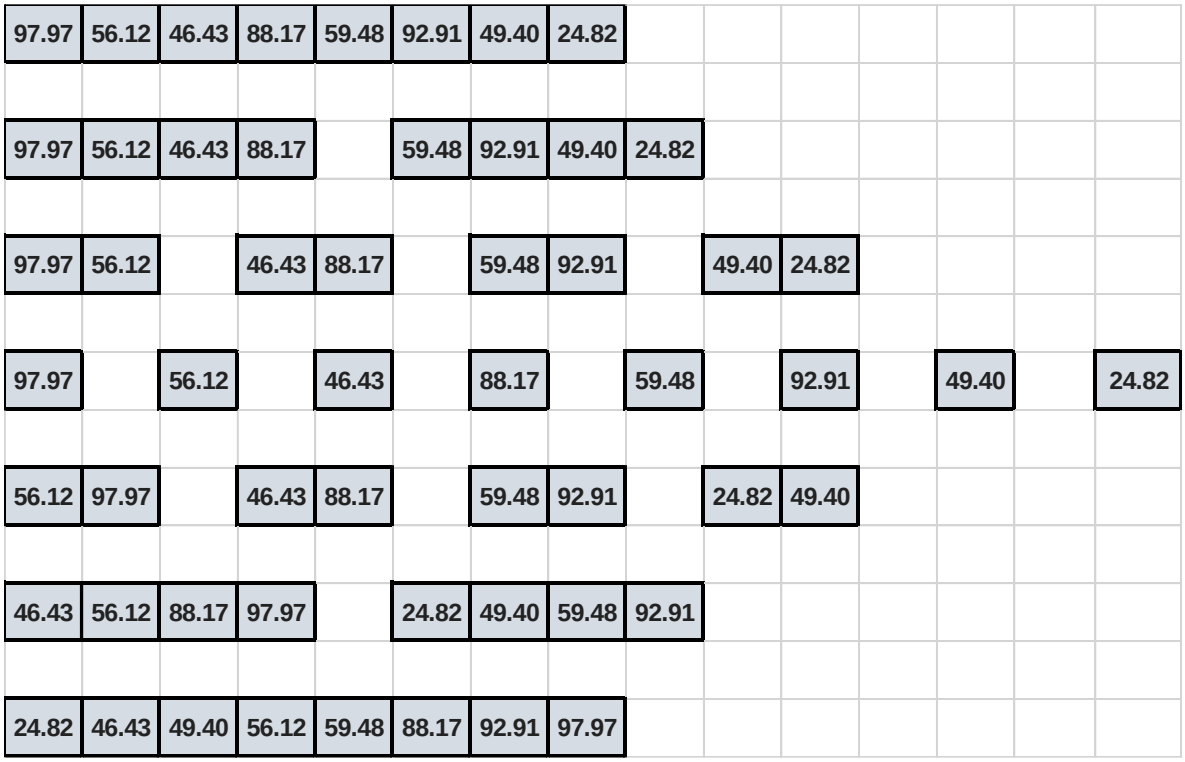
$$Suponiendo\ d = a + b + c$$

$$an^2 + bn + c \leq (a + b + c)n^2$$

$$Es\ cierto\ para\ n \geq 1$$

## Ejercicio 4

Ejemplo de aplicación de algoritmo merge sort para el ordenamiento de alturas de ramas de árboles



# Ejercicios de programación

## Ejercicio 1

Selección de máximo y mínimo implementada en Python 2.7

```
maxmin = {'max':A[0],'min':A[0]}
for i in range(0,len(A)):
    if A[i] < maxmin['min']:
        maxmin['min'] = A[i]
    if A[i] > maxmin['max']:
        maxmin['max'] = A[i]
```

```
 RESTART: D:\ESCUELA\Sext
ractica\maxmin.py
lista [50, 34, 2, 0, -5]
{'max': 50, 'min': -5}
 RESTART: D:\ESCUELA\Se
ractica\maxmin.py
lista [-1, 0, 0, 0, 1]
{'max': 1, 'min': -1}
```

```
 RESTART: D:\ESCUELA\SextoSemestre\Alg
ractica\maxmin.py
lista [54, 24, 1212, 312131, -123125]
{'max': 312131, 'min': -123125}
```

## Ejercicio 2

Exponenciación modular rápida implementada en Python 2.7

```
def exponenciacion(a,m,n):
    res = 1
    a = a % n
    while m > 0:
        if m%2 != 0:
            res = res*a % n
        m = m>>1
        a = a*a % n
    return res
```

```
 RESTART: D:\ESCUELA\SextoS
ractica\expmodularrapida.py
a: 4 m: 5 n: 6
4
 >>>
  RESTART: D:\ESCUELA\SextoS
 ractica\expmodularrapida.py
 a: 5 m: 5 n: 2
 1
```

```
>>>
 RESTART: D:\ESCUELA\SextoS
ractica\expmodularrapida.py
a: 5 m: 2 n: 2
1
```

## Ejercicio 3

Selection sort implementado en Python 2.7

```
for i in range(0,len(A)):
    min = i
    for j in range(i+1,len(A)):
        if A[j] < A[i]:
            min = j
    A[i],A[min] = A[min],A[i]
```

```
 RESTART: D:\ESCUELA\Sextc
ractica\selectionSort.py
[4, 67, 23, 14, 1]
[1, 4, 14, 23, 67]
>>>
```

```
 RESTART: D:\ESCUELA\Sext
ractica\selectionSort.py
[-5, 67, 43, 14, 1]
[-5, 1, 14, 43, 67]
```

```
 RESTART: D:\ESCUELA\Sext
ractica\selectionSort.py
[50, 34, 2, 0, -5]
[-5, 0, 2, 34, 50]
```

## Ejercicio 4

Merge sort implementado en Python 2.7

```
def mergesort(A):
    n = len(A)
    if n == 1:
        return A
    Ap = mergesort(A[:n/2])
    App = mergesort(A[n/2:])
    return merge(Ap,App)
```

```
def merge(L,R):
    Re = []
    nL = len(L)
    nR = len(R)
    while nL > 0 and nR > 0:
        if L[len(L) - nL] < R[len(R) - nR]:
            Re.append(L[len(L) - nL])
            nL -= 1
        else:
            Re.append(R[len(R) - nR])
            nR -= 1
    if nL > 0:
        Re += L[-nL:]
    elif nR > 0:
        Re += R[-nR:]
    return Re
```

```
 RESTART: D:\ESCUELA\SextoSemestre\Algoritmos\Practicas\Prime
actica\mergesort.py
[13, 23, 1, 232, 8] [1, 8, 13, 23, 232]
>>>
 RESTART: D:\ESCUELA\SextoSemestre\Algoritmos\Practicas\Prime
actica\mergesort.py
[13, 23, 1, 232, 8, 0, 2, 134] [0, 1, 2, 8, 13, 23, 134, 232]
>>>
```

```
 RESTART: D:\ESCUEL
actica\mergesort.py
[1, 0] [0, 1]
>>>
```

```
 RESTART: D:\ESCUELA\SextoSemestre\Algo
actica\mergesort.py
[1, 0, -1, 20, -20] [-20, -1, 0, 1, 20]
>>>
```