# CC3001: Algoritmos y Estructuras de Datos
# Control 1

Prof: Nelson Baloian
Aux: José M. Saavedra R.

14 de abril, 2010
Tiempo: 1 hora 45 minutos

## 1. Pregunta 1: Ciclos e Invariantes (2 puntos)

Para obtener un arreglo $C$ ordenado a partir de otros dos $A[0,\ldots,m]$, $B[0,\ldots,n]$ ya ordenados (ver Figura 1), se puede usar el siguiente invariante:

$$\begin{array}{ll} i+j=k & \\ C[l] \leq C[l+1] & l=0\ldots k-2 \\ A[i] \geq C[l] & l=0\ldots k-1 \\ B[j] \geq C[l] & l=0\ldots k-1 \end{array}$$

donde, $0 \leq i < m$, $0 \leq j < n$, $0 \leq k < m+n$.

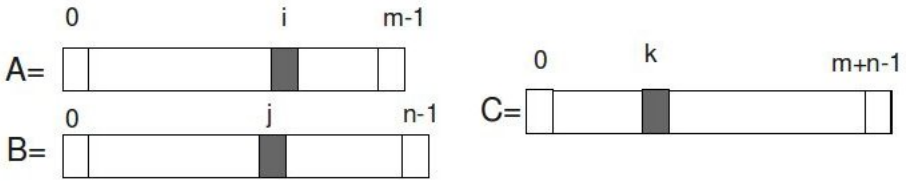


Figura 1: .

Se pide diseñar un algoritmo basado en un proceso iterativo (un ciclo) que cumpla el invariante, definiendo los siguientes puntos:

1. Las condiciones iniciales (antes del ciclo) para que se cumpla el invariante.
2. La condición de término del ciclo.
3. Cómo avanzar hasta la condición de término (quiebre y restitución del invariante).

## 2. Pregunta 2: Recurrencias (2 puntos)

1. Resuelva la siguiente recurrencia:

$$T(n) = 6T(n-1) - 4T(n-2)$$

2. Considere el procedimiento *proc* , que tiene como entrada un arreglo $A$ y los índices $i$, $j$, que indican la posición inicial y final de los elementos que se procesarán, respectivamente. Inicialmente, el procedimento es llamado mediante *proc(A, 0, n-1)*, donde $n$ es el tamaño de $A$.

```
procedimiento proc(A: array, i: entero, j:entero)
    n=j-i+1
    si (n>1)
        x=n/sqrt(n)
        para k=0  hasta x-1
            proc(A, k*x, (k+1)*x-1)
        fin_para
        ordenarPorSeleccion(A,i,j)
    fin_si
fin_procedimiento
```

Se pide:

*a*) Escribir la ecuación de recurrencia correspondiente al procedimiento *proc*.

*b*) Resuelver la recurrencia.

*c*) Indicar la complejidad del algoritmo.

# 3. Pregunta 3: Diagramas de Estado (2 puntos)

Suponga que ha sido contratado por una empresa dedicada al desarrollo de software para sistemas de información geográfica. Su primera tarea es implementar un programa que valide expresiones que indican ubicaciones geográficas representadas por **Latitud** y **Longitud**. Según lo requerido, el programa debe recibir como entrada una cadena que puede representar una latitud o una longitud, con sus correspondiente grados(g), minutos(m) y segundos(s). La entradas correctas siguen el siguiente formato:

**Para la latitud**:

`Lat:[-]<Entero>g<Entero>m<Entero>s`

**Para la longitud**:

`Lon:[-]<Entero>g<Entero>m<Entero>s`

En la notación dada:

- [−] indica que el signo − es opcional.
- <Entero> indica que en esa posición se requiere un número entero.

Se pide:

1. Diseñar un digrama de estados que permita validar la sintaxis correspondiente a una latitud o longitud.
2. Implementar una función booleana que determine la validez de la entrada.
3. Desde el punto de vista semántico, los grados de una latitud pueden variar desde −90 hasta 90, mientras que los grados en las longitudes varían entre −180 y 180. Así mismo, los minutos y segundos varían entre 0 y 59. Por lo tanto, se le pide agregar estas restricciones semánticas en el método de validación anterior.