**Solución al laboratorio N°03**

## Algoritmo Ackermann

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ y } n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \text{ y } n > 0 \end{cases}$$

Este algoritmo crece más que una función exponencial debido a cada llamada recursiva genera otra dentro de sí misma, y esa a su vez genera otra, y así sucesivamente.

## Código del algoritmo Ackermann iterativo

```java
public int ackermannIterativo(int m, int n) {
    if (m < 0 || n < 0) {
        throw new IllegalArgumentException( s:"m y n deben ser no negativos.");
    }

    Stack<Integer> stack = new Stack<>();
    stack.push( item:m);

    // Valor actual de n. Cuando la pila este vacia, n será el resultado final.
    int nActual = n;

    while (!stack.isEmpty()) {
        int mActual = stack.pop();

        if (mActual == 0) {
            // Caso base: A(0, N) = N + 1
            nActual = nActual + 1;

        } else if (nActual == 0) {
            // Caso A(m, 0) = A(m-1, 1)
            stack.push(mActual - 1);
            nActual = 1;
        } else {
            // Caso A(m, n) = A(m-1, A(m, n-1)) donde m > 0, n > 0
            stack.push(mActual - 1);
            stack.push( item:mActual);
            nActual = nActual - 1;
        }
    }

    return nActual;
}
```

## Función tiempo

Dado que en cada iteración, el algoritmo saca un valor de la pila y puede agregar uno o dos nuevos valores. Debido a esto la pila puede crecer, y el número total de iteraciones depende de cuántos elementos se agregan a la pila.
Por lo que el número total de iteraciones del ciclo 'mientras' es igual al número total de llamadas recursivas en Ackermann.

$$T(m,n) = \text{Número de interaciones del ciclo 'mientras'} \;=\; \Theta(A(m,n))$$

**Notación asintotica**

Dado que el algoritmo crece más que una función exponencial, entonces, se tiene la asintótica $O(n^n)$ ya que su tiempo de ejecución se multiplica por una constante cada vez que aumenta el tamaño de entrada n.

$$T(m, n) \notin O(c^n) \text{ para nigún } c > 0$$

Esto significa que el tiempo de ejecución del algoritmo Ackermann crece aún más rápido que cualquier función exponencial, entonces, no existe un valor de c que sirva como cota superior para dicho crecimiento.