UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Universidad del Perú, Decana de América

Facultad de Ingeniería de Sistemas e Informática

INFORME: IMPLEMENTACIÓN DE LA FUNCIÓN DE ACKERMANN

CURSO: Análisis y diseño de algoritmos

DOCENTE: Chávez Soto, Jorge Luis

ESTUDIANTE : Diaz Chambi, Leslie Yadira

SECCIÓN: 3

LIMA – PERÚ

2025

I. INTRODUCCIÓN

Este proyecto desarrolla una implementación de la función de Ackermann utilizando tanto métodos recursivos como iterativos. Incluye análisis de complejidad temporal y espacial, comparaciones de rendimiento entre implementaciones y recomendaciones sobre límites prácticos de cálculo.

II. MARCO TEÓRICO

2.1 Definición de la función de Ackermann

La función de Ackermann A(m,n) se define recursivamente como:

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

2.2 Propiedades matemáticas

- **Función total:** Está definida para todos los números naturales.
- **Estrictamente creciente:** A(m,n) < A(m,n+1) y A(m,n) < A(m+1,n)
- **No primitiva recursiva:** No puede expresarse mediante bucles anidados de longitud fija.
- **Crecimiento extremo:** Crece más rápido que cualquier función primitiva recursiva.

2.3 Interpretación de los primeros valores

- **A(0,n) = n + 1:** Función sucesor
- **A(1,n) = n + 2:** Suma con constante
- **A(2,n) = 2n + 3:** Multiplicación
- **A(3,n) = 2^(n+3) - 3:** Exponenciación
- **A(4,n):** Tetración (torre de exponentes)

III. DISEÑO E IMPLEMENTACIÓN

El sistema se estructura en cuatro clases principales:

1. **AckermannMain (Clase principal)**

   Propósito: Interfaz de usuario interactiva

   Funcionalidades:

- Entrada de datos con validación
- Ejecución de ambas implementaciones
- Medición de tiempos de ejecución
- Advertencias de seguridad para valores grandes
- Análisis automático de complejidad

2. **AckermannCalculadora (Motor de cálculo)**

   Propósito: Implementaciones de la función

   Métodos principales:

- ackermannRecursivo(m, n): Implementación recursiva tradicional
- ackermannIterativo(m, n): Versión iterativa usando pila
- ackermannOptimizado(m, n): Casos especiales optimizados
- calcularAckermann(m, n): Selector automático de método

3. **AnalizadorComplejidad (Análisis teórico)**

   Propósito: Análisis de complejidad y rendimiento

   Funcionalidades:

- Cálculo de complejidad temporal y espacial
- Estimaciones de tiempo de ejecución
- Recomendaciones de uso
- Explicaciones educativas

4. **DemoAckermann (Casos de Prueba)**

   Propósito: Demostración y validación

   Características:

- Casos de prueba predefinidos
- Comparación de implementaciones
- Benchmarks de rendimiento
- Tabla de valores conocidos

IV. ANÁLISIS DE COMPLEJIDAD

Complejidad temporal

m	Función equivalente	Complejidad	Descripción
0	$n + 1$	$O(1)$	Función sucesora
1	$n + 2$	$O(1)$	Suma
2	$2n + 3$	$O(n)$	Multiplicación
3	$2^{n+3} - 3$	$O(2^{n})$	Exponenciación
4	$2 \uparrow\uparrow (n + 3)$	$O(2 \uparrow\uparrow n)$	Tetración
≥ 5	No primitiva recursiva	No acotable	Hiperoperaciones

Complejidad Espacial

- **Recursiva:** O(A(m,n)) en el peor caso debido a la pila de llamadas.
- **Iterativa:** O(A(m,n)) debido al uso de pila explícita.
- **Optimizada:** O(1) para casos base (m ≤ 1)

Número de operaciones: Para valores específicos:

- A(3,4): ~125 operaciones
- A(3,10): ~8,189 operaciones
- A(4,1): ~65,533 operaciones
- A(4,2): Más de 10^19000 operaciones (prácticamente incalculable)

V. RESULTADOS Y PRUEBAS

Casos de Prueba Ejecutados

m	n	Resultado	Tiempo (ms)	Método
0	5	6	< 1	Optimizado
1	5	7	< 1	Optimizado
2	5	13	< 1	Optimizado
3	4	125	< 10	Iterativo
3	10	8189	< 100	Iterativo

4	0	13	< 1	Optimizado
4	1	65533	< 50	Iterativo

Comparación de implementaciones

Para A(3,4):

- Recursiva: 15 ms, 125 llamadas recursivas
- Iterativa: 2 ms, 45 operaciones
- Resultado: Idéntico en ambos casos

Para A(3,10):

- Recursiva: 1,200 ms, 8,189 llamadas recursivas
- Iterativa: 15 ms, 350 operaciones
- Ventaja: Iterativa 80x más rápida

Límites identificados

Límites prácticos recomendados:

- $m \leq 2$: Cualquier n práctico
- $m = 3$: $n \leq 15$ (factible)
- $m = 4$, $n \leq 1$: Límite absoluto recomendado
- $m = 4$, $n \geq 2$: Prácticamente incalculable
- $m \geq 5$: No recomendado

VI. CONCLUSIONES

1. La implementación iterativa supera significativamente a la recursiva en términos de velocidad y uso de memoria para valores moderados.

2. Existen límites prácticos absolutos más allá de los cuales el cálculo se vuelve inviable independientemente de la implementación.

VII. RECOMENDACIONES

Para uso práctico

- Utilizar $m \leq 3$ y $n \leq 15$ para cálculos interactivos
- Emplear la versión iterativa para mejor rendimiento
- Implementar timeouts para evitar cálculos excesivamente largos

Para uso educativo.

- Comenzar con casos simples ($m \leq 2$)
- Demostrar el crecimiento progresivo con ejemplos.
- Enfatizar las limitaciones computacionales reales