Logotipo

Descripción generada automáticamente con confianza mediaLogotipo

Descripción generada automáticamenteINSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ANÁLISIS Y DISEÑO DE ALGORITMOS

PRÁCTICA 03

Rodrigo García Mayorga

2012630554

# Introducción

El código presentado genera todas las posibles combinaciones de letras asociadas a una secuencia de dígitos, emulando el funcionamiento de un teclado de teléfono antiguo. En estos teclados, cada número del 2 al 9 corresponde a un conjunto específico de letras (por ejemplo, el 2 se asocia con "abc" y el 7 con "pqrs").

**Funcionamiento del Código**

1. **Mapeo de Letras:**  
   Se define un arreglo mapaTeclado donde cada posición (del 0 al 9) contiene las letras correspondientes al dígito en un teclado telefónico. Los índices 0 y 1 están vacíos ya que no tienen letras asignadas.
2. **Función Recursiva:**  
   La función generarCombinaciones se encarga de construir las combinaciones letra por letra:
   * Si se ha alcanzado el último índice, se imprime la combinación actual.
   * De lo contrario, se obtiene el dígito actual, se validan los rangos permitidos (2-9) y se recorre cada letra correspondiente a ese dígito.
   * Se llama recursivamente con el siguiente índice, generando todas las combinaciones posibles.
3. **Ejecución Principal:**  
   En la función main, se solicita al usuario una cadena de dígitos válidos, se inicializa el array actual para almacenar la combinación en construcción y se llama a la función recursiva para empezar el proceso.

**Ejemplo**

Si el usuario ingresa "23", las posibles combinaciones generadas serían:

* ad, ae, af
* bd, be, bf
* cd, ce, cf

**Relación con la Complejidad Algorítmica**

La complejidad temporal del algoritmo es **exponencial**, específicamente O(4n)O(4^n)O(4n), donde n es la longitud de la cadena de dígitos. Esto se debe a que cada dígito puede generar hasta 4 posibles letras, y estas combinaciones se multiplican recursivamente en cada nivel de profundidad.

En cuanto a la complejidad espacial, el uso de memoria es **lineal**, O(n)O(n)O(n), ya que el array actual y la profundidad de la pila de llamadas recursivas dependen directamente de la longitud de la entrada.

El código, aunque eficiente en memoria, puede volverse ineficiente para secuencias largas debido a su crecimiento exponencial en tiempo de ejecución.

# Capturas

Texto

El contenido generado por IA puede ser incorrecto.

## Enlace del código (github):

[https://github.com/LowisN/ADA\_Practica2.git](https://github.com/LowisN/ADA_Practica2)