

# Actividad Integradora 1

JORGE EMILIANO POMAR MENDOZA |  
A01709338  
ELIUTH BALDERAS NERI | A01703315

# KMP

---

El algoritmo KMP es una técnica de búsqueda de subcadenas la cuál utiliza información previa para agilizar la búsqueda de un patrón dentro de una cadena. Todo ello se logra mediante la creación de una tabla de valores basada en el patrón, esto permite identificar dónde podría ocurrir la siguiente ocurrencia sin tener que volver a analizar toda la cadena cada que se haga este proceso

# EJEMPLO

String: a b a b c a b c a b a b d  
          |  
          i

Pattern: a b a b d

LPS: 0 0 1 2 0  
      j

SCALER  
Topics

## Tabla LPS

String: A B C D E F G H

Pattern:

D E F

E == E?

Yes.

Check next index.

SCALER  
Topics

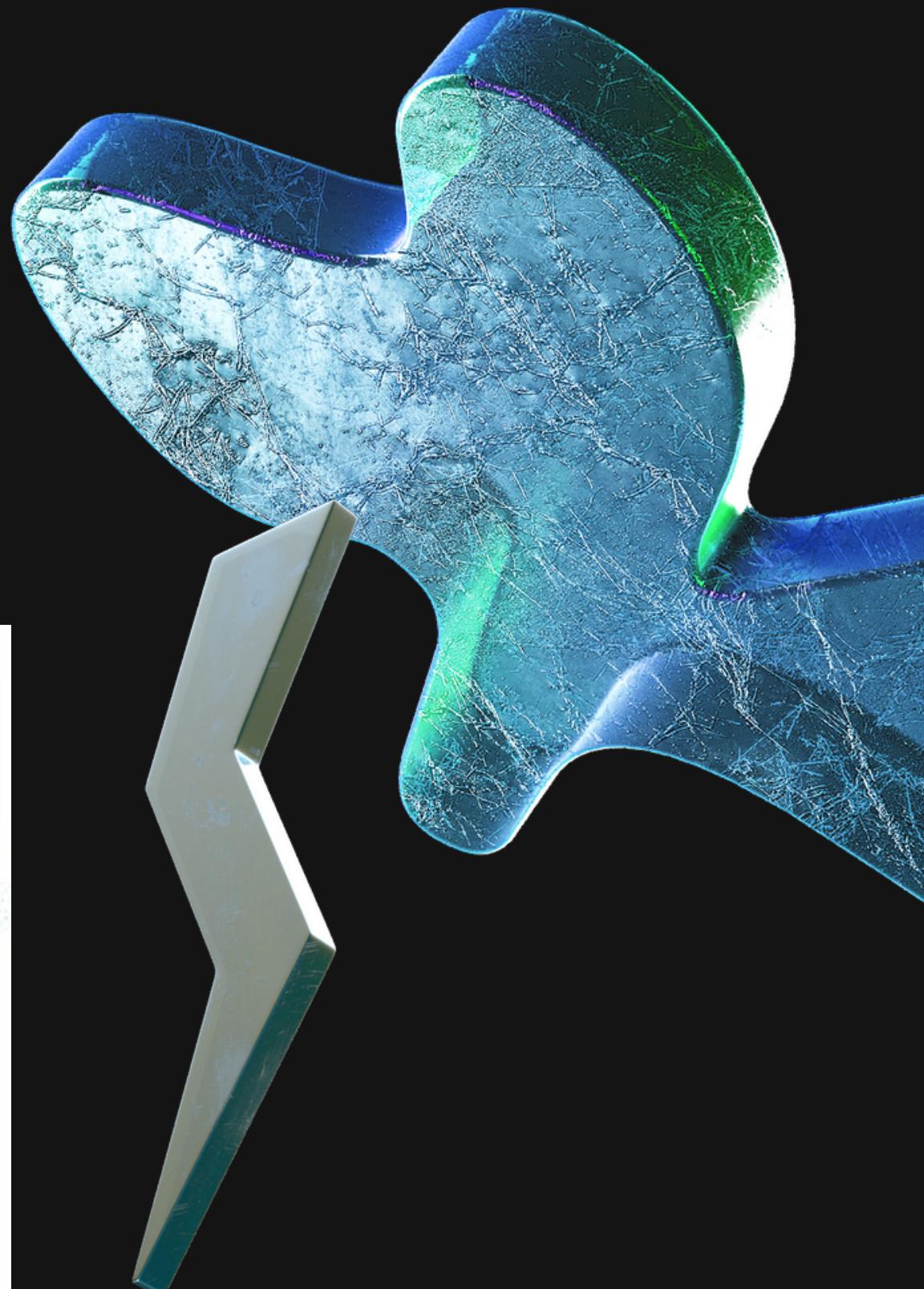
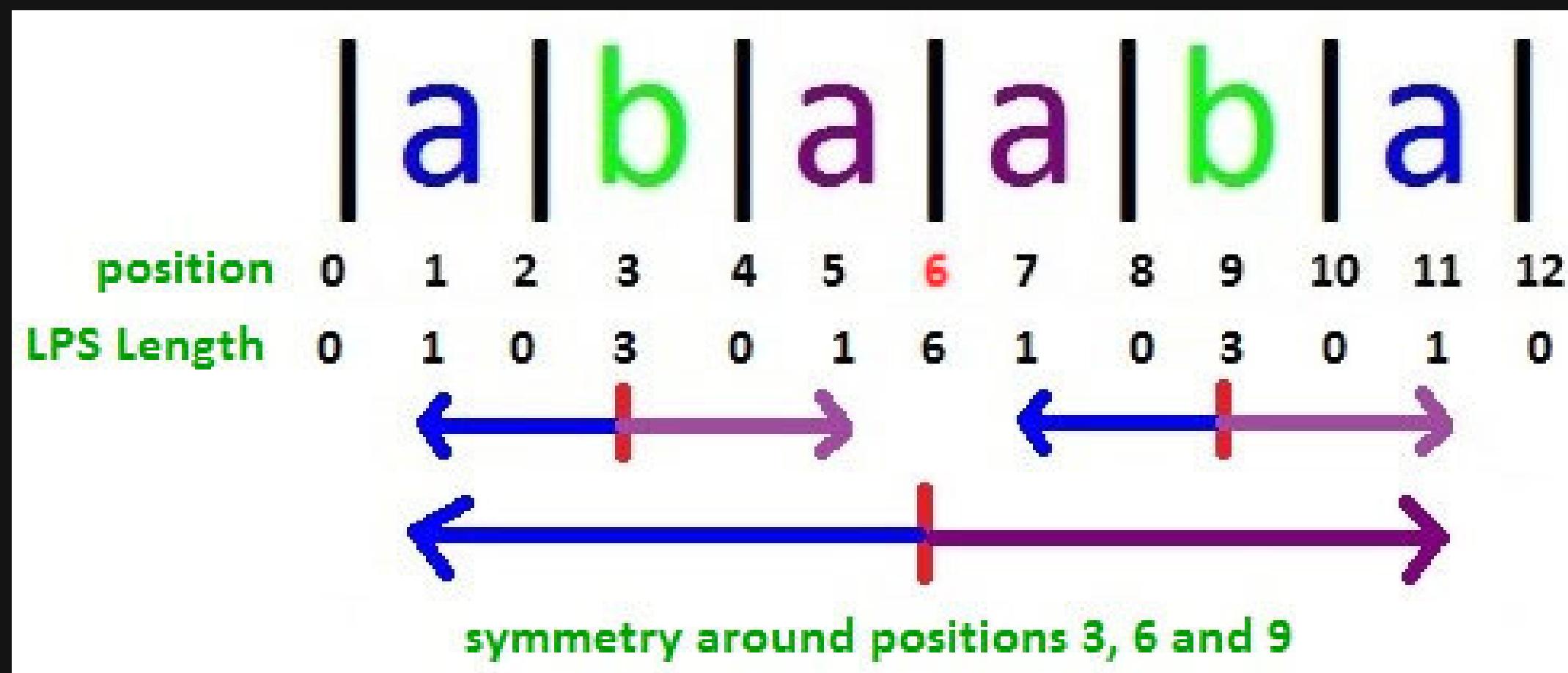
## Implementación KMP

# Algoritmo de Manacher

---

El algoritmo de Manacher sirve particularmente para encontrar el palíndromo más largo en una cadena. Primero se asegura que el arreglo sea impar o asimétrico \$#@. Después define un radio o un espacio que contiene un centro (c) y un índice (r) que indica el ultimo carácter más largo hasta el momento del palíndromo.

# EJEMPLO



# Fuerza bruta

---

El método de fuerza bruta consiste en explorar todas las posibles combinaciones para encontrar una solución. En este algoritmo se examina cada elemento hasta dar con la solución que se busca o el elemnto buscado.

# EJEMPLO

Texto: 

a	n	a	l	i	s	i	s	d	e	a	l	g	o	r	i	t	m	o	s
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

×

Patrón: 

a	l	g	o
---	---	---	---

×

a	l	g	o
---	---	---	---

×

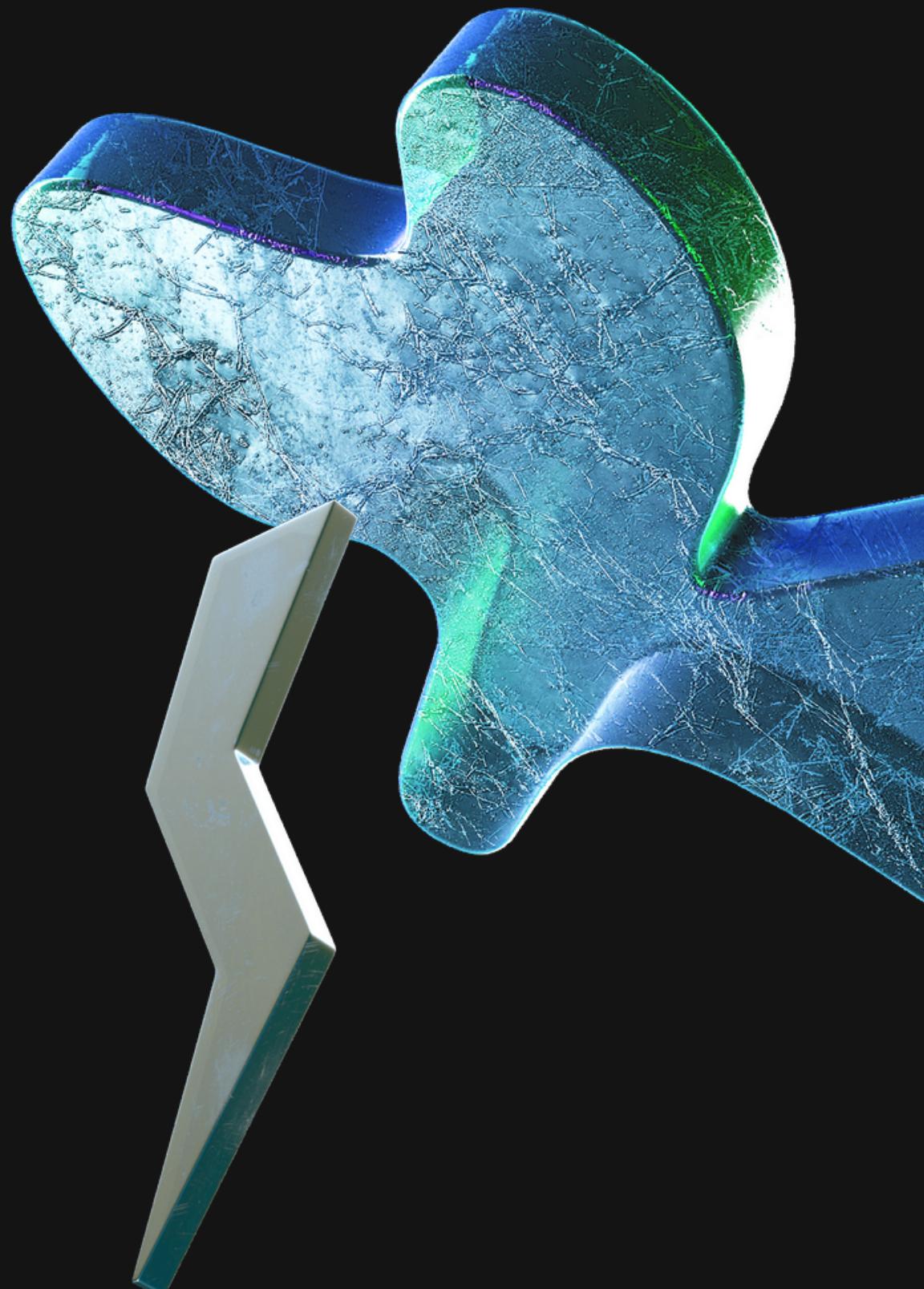
a	l	g	o
---	---	---	---

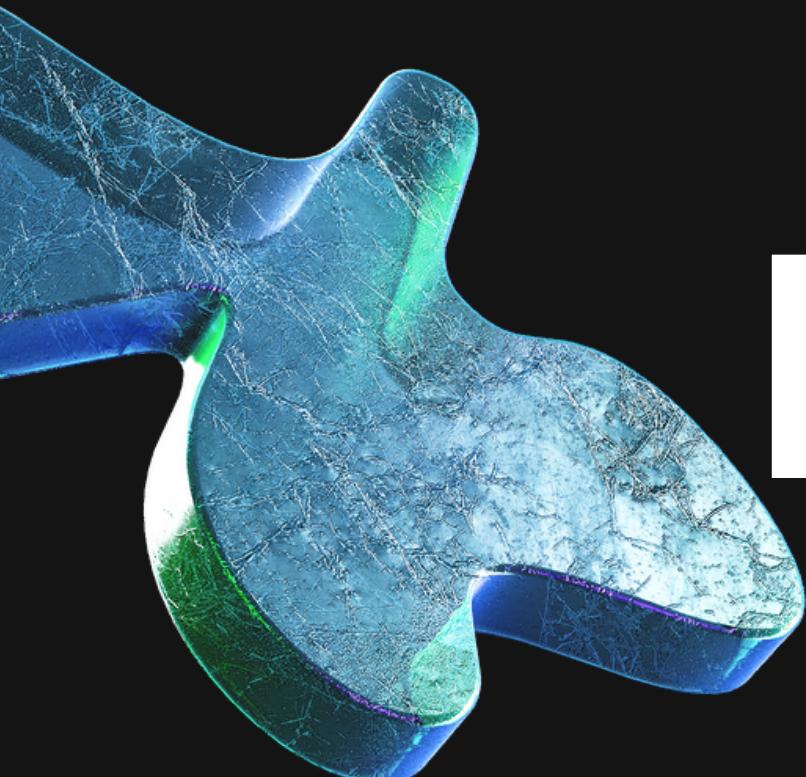
×

a	l	g	o
---	---	---	---

✓ ✓ ✓ ✓

a	l	g	o
---	---	---	---





**Manacher**  
**vs**  
**FUERZA BRUTA**

# MÁS SOBRE Manacher

## Ventajas

LA COMPLEJIDAD ES  $O(n)$  Y ESTO LO HACE MUCHO MÁS RÁPIDO QUE CUALQUIER OTRO ALGORITMO

---

ALGORITMO DISEÑADO ESPECÍFICAMENTE PARA EL PROBLEMA

---

UTILIZA UN ARREGLO P[ ] LINEAL EN VEZ DE UN VECTOR O UNA MATRIZ Y OCUPA MENOS ESPACIO

---

# MÁS SOBRE Manacher

## Desventajas

MÁS DIFÍCIL DE ENTENDER.

---

NECESITA PRE-PROCESAR LOS ELEMENTOS DEL ARREGLO

---

SOLAMENTE SE PUEDE USAR PARA UN PROBLEMA ESPECÍFICO

---



```
int encuentraPalindromoManacher(
    const std::string &contenido) { // Implementacion de Manacher O(n) para
// encontrar palindromos
std::string secuencia = "$#";
for (const auto &c : contenido) {
    secuencia += c;
    secuencia += "#";
}
secuencia += "@";
int n = secuencia.length();
std::vector<int> lps(n, 0);
int c = 0, r = 0;
for (int i = 1; i < n - 1; i++) {
    int mirr = 2 * c - i;
    if (i < r) {
        lps[i] = std::min(r - i, lps[mirr]);
    }
    while (secuencia[i + (1 + lps[i])] == secuencia[i - (1 + lps[i])]) {
        lps[i]++;
    }
    if (i + lps[i] > r) {
        c = i;
        r = i + lps[i];
    }
}
int maxLen = 0;
int centerIndex = 0;
for (int i = 1; i < n - 1; i++) {
    if (lps[i] > maxLen) {
        maxLen = lps[i];
        centerIndex = i;
    }
}
```



# MÁS SOBRE FUERZA BRUTA

## Ventajas

ALGORITMO SIMPLE

---

FÁCIL DE IMPLEMENTAR

---

ENCUENTRA TODAS  
LAS POSIBLES  
SOLUCIONES

---

# MÁS SOBRE FUERZA BRUTA

## Desventajas

RECORRE TODOS LOS ELEMENTOS

---

DEPENDE DEL NÚMERO DE ELEMENTOS EL TIEMPO DE EJECUCIÓN

---

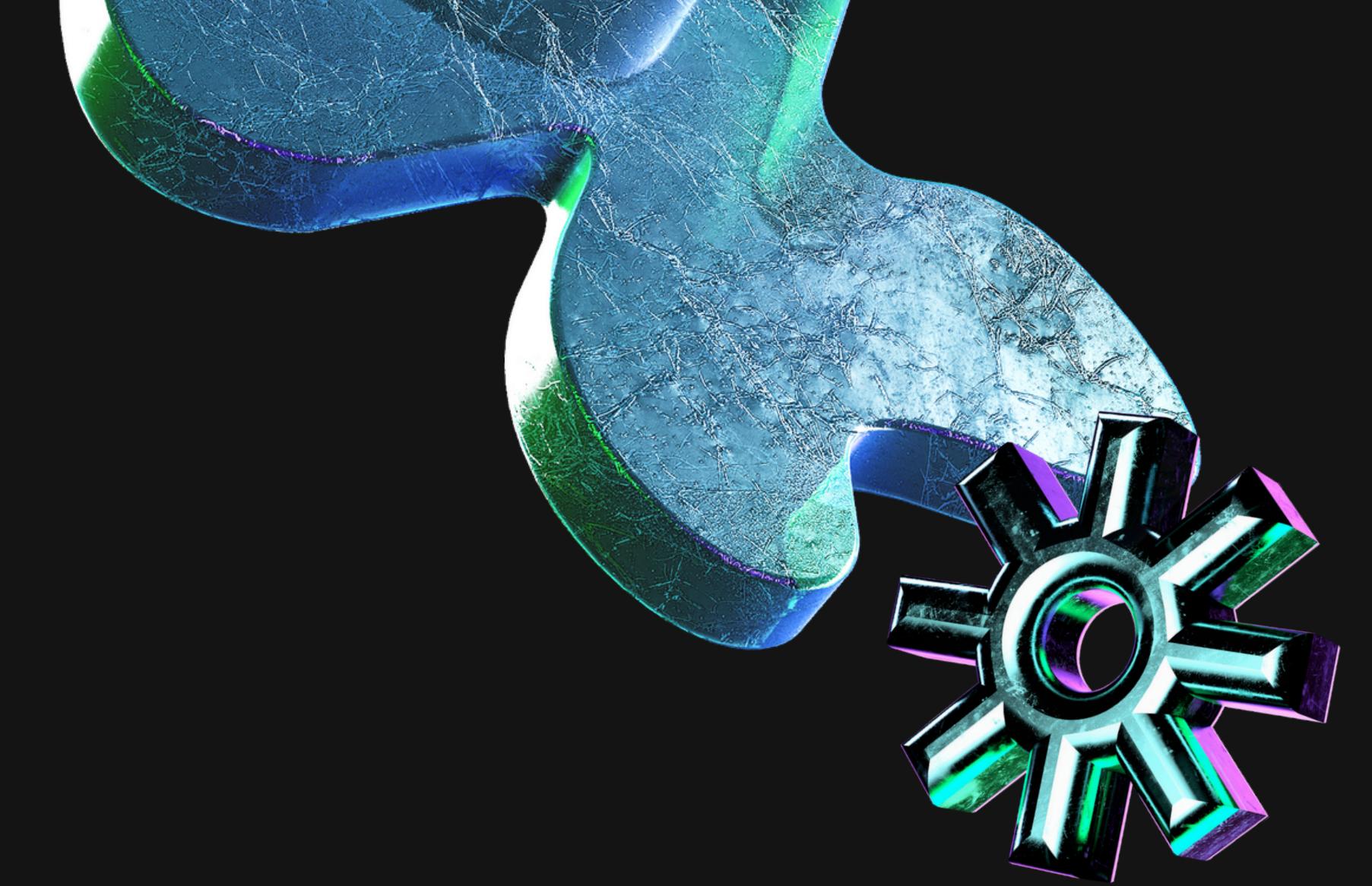
CONSUME BASTANTE MEMORIA

---



```
int encuentraPalindromoBruto(
    const std::string &contenido) { // Implementacion de busqueda bruta
 0(n^3)
    // para encontrar palindromos
    int n = contenido.length();
    int maxLen = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            bool esPalindromo = true;
            for (int k = i; k <= (i + j) / 2; k++) {
                if (contenido[k] != contenido[j - (k - i)]) {
                    esPalindromo = false;
                    break;
                }
            }
            if (esPalindromo) {
                maxLen = std::max(maxLen, j - i + 1);
            }
        }
    }
    return maxLen;
}
// basado en
https://www.tamps.cinvestav.mx/~ertello/algorithms/sesion07.pdf
```





¿CONCLUSIÓN?

```
(false) Secuencia no encontrada
Tiempo de ejecución: 0.0007ms

(true) Secuencia
<<17271>> encontrada en el archivo:
Situacion_Problema_1/transmission02.txt en la posición:
17
Tiempo de ejecución: 0.00023ms

///// Parte 2 /////

Encuentra palindromo más largo por Manacher --->

(true) Palindromo de longitud: 11 encontrado en el archivo:
Situacion_Problema_1/transmission01.txt
Tiempo de ejecución: 0.00244ms

(true) Palindromo de longitud: 11 encontrado en el archivo:
Situacion_Problema_1/transmission02.txt
Tiempo de ejecución: 0.00724ms

Encuentra palindromo más largo por busqueda bruta --->

(true) Palindromo de longitud: 11 encontrado en el archivo:
Situacion_Problema_1/transmission01.txt
Tiempo de ejecución: 0.00554ms

(true) Palindromo de longitud: 11 encontrado en el archivo:
Situacion_Problema_1/transmission02.txt
Tiempo de ejecución: 0.046319ms

///// Parte 3 /////

(true) Substring de longitud: 19 encontrado entre los archivos:
Situacion_Problema_1/transmission01.txt y Situacion_Problema_1/transmission02.txt
Tiempo de ejecución: 0.00429ms

(true) Substring de longitud: 145 encontrado entre los archivos:
Situacion_Problema_1/transmission01.txt y Situacion_Problema_1/transmission02.txt
Tiempo de ejecución: 0.01682ms
```

# REFERENCIAS

Bultrón, J. (2017). Algoritmo KMP [Presentación]. Recuperado de <https://prezi.com/p/9fxptyllxpzz/algoritmo-kmp/>

Bustos, B. (s.f.). Búsqueda de texto. Recuperado de <https://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/BusqTexto/>

Universidad Mayor de San Andrés. (s.f.). Algoritmo de fuerza bruta. Recuperado de <https://www.studocu.com/bo/document/universidad-mayor-de-san-andres/taller-de-programacion-lab-inf/algoritmo-de-fuerza-bruta/4599639>

GfG. (2021, February 11). Manacher s Algorithm Linear Time Longest Palindromic Substring Part 1. GeeksforGeeks. <https://www.geeksforgeeks.org/manachers-algorithm-linear-time-longest-palindromic-substring-part-1/>