Fidel Morales Briones A01198630
Tarea 5 manacher
29 de septiembre, 2024

Pseudocódigo manacher:

```
Unset
string longestPalindromicSubstring(string T) {
	string manacherString = añadir "#" entre cada caractér de T y añadir
"@" al incio y al final "&"

	// arreglo para guardar el tamaño del palindromo tomando el elemento i
como el centro del palíndromo
	P[manacherString.size()]

	int C, R, maxIndex, maxSize = 0

	for int i = 1 until manacherString.size() - 1 {
		// calcular valor espejo del palíndromo
		int mirror = 2 * C - 1

		// copiar el valor del mirror
		if i < R {
			P[i] = min(R - i, P[mirror])
		}

		// revisar sí es un palíndromo válido y calcular su tamaño
		while manacherString[i + (1 + P[i])] == manacherString[i - (1 +
P[i])] {
			P[i]++
		}

		// Cambiar el centro porque se encontró un palíndromo
		if i + P[i] > R {
			C = i
			R = i + P[i]
		}

		// Actualizar el palíndromo más grande
		if P[i] > maxsize {
			maxIndex = i
			maxSize = P[i]
		}
	  }


	}
```

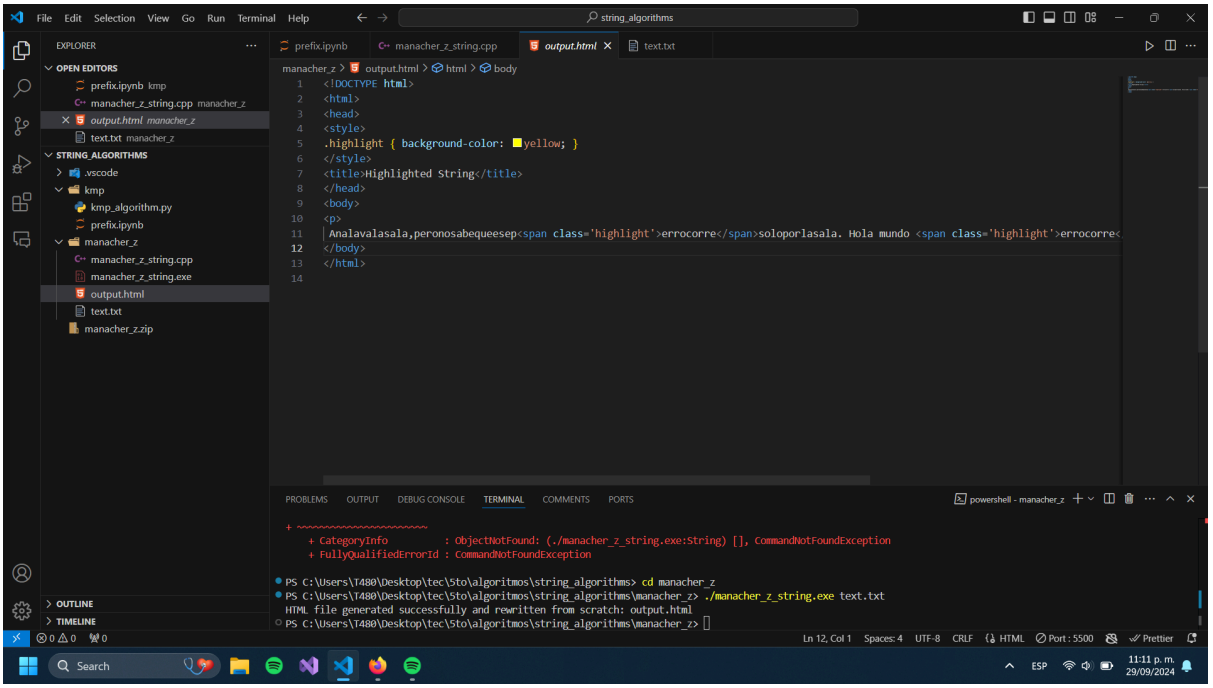
```
	// reconstruir el palíndromo en base al centro y su tamaño
	for int i = 1 until i <= manacherString.size() / 2;{
		rightIndex = maxIndex + (2 * i);
		leftIndex = maxIndex - (2 * i);

		palindrome = manacherString[leftIndex] + palindrome +
manacherString[rightIndex];
	}

	return palindrome
}
```

Análisis de complejidad:

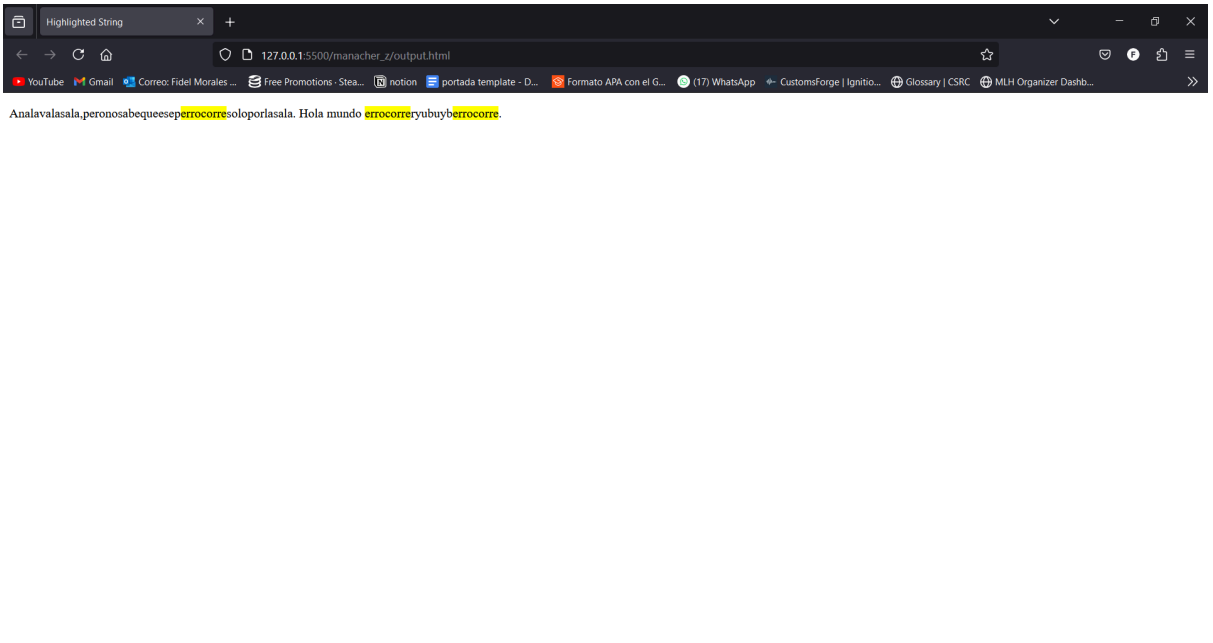
```
C/C++
string computeManacherString(string T) {
	string manacherString = ""; O(1)

	for (int i = 0; i < T.size(); i++) { O(T.size())
	manacherString = manacherString + "#" + T[i]; O(1)
	}

	manacherString =  "@" + manacherString + "#&"; O(1)

	return manacherString; O(1)
}


string longestPalindromicSubstring(string T) {
	string manacherString = computeManacherString(T); O(T.size())

	vector<int> P(manacherString.size()); O(1)

	int C, R = 0; O(1)
	int maxIndex, maxSize = 0; O(1)

	// encontrar el centro de la subcadena palíndromo más grande
	for (int i = 1; i < manacherString.size() - 1; i++) { O(T.size())
	int mirror = 2 * C - i; O(1)

	if (i < R) { O(1)
		P[i] = min(R - i, P[mirror]); O(1)
	}
```

```
	while (manacherString[i + (1 + P[i])] == manacherString[i - (1 +
P[i])]) { O(T.size())
		P[i]++; O(1)
	}

	if (i + P[i] > R) { O(1)
		C = i; O(1)
		R = i + P[i]; O(1)
	}

	// revisar si el centro actual es el más grande
	if (P[i] > maxSize) { O(1)
		maxIndex = i; O(1)
		maxSize = P[i]; O(1)
	}
	}

	// reconstruir string en base al centro
	int iter = maxSize / 2; O(1)
	string palindrome(1, manacherString[maxIndex]); O(1)
	int rightIndex, leftIndex; O(1)

	for (int i = 1; i <= iter; i++) { O(T.size()/2)
	rightIndex = maxIndex + (2 * i); O(1)
	leftIndex = maxIndex - (2 * i); O(1)

	palindrome = manacherString[leftIndex] + palindrome +
manacherString[rightIndex];    O(1)
	}

	return palindrome; O(1)
}
```

Ejecución de código que genera HTML a partir de un .txt:



HTML con palíndromos subrayados:



Bibliografía:

Code_with_Engineer. (2023a, mayo 19). *Z algorithm illustration* [Vídeo]. YouTube.

https://www.youtube.com/watch?v=Tg1w-0a1xew

Code_with_Engineer. (2023b, mayo 19). *Z algorithm implementation* [Vídeo]. YouTube.

https://www.youtube.com/watch?v=DCDPWR4k7Iw

IDeserve. (2015, 4 septiembre). *Longest palindromic substring O(N) Manacher's algorithm* [Vídeo]. YouTube. https://www.youtube.com/watch?v=nbTSfrEfo6M