

DEV.F.:

Clase 4

Los dos punteros

Comenzamos en 10 min

6:40pm (hora CDMX)



Los dos punteros

El algoritmo de los dos punteros es una técnica eficiente y versátil que se usa para resolver problemas relacionados con arreglos o cadenas ordenadas. Aunque el concepto puede sonar complejo, es fácil de entender y aplicar con algo de práctica.



| ¿Para qué sirve?

- **Buscar pares de elementos que cumplan una condición:** Por ejemplo, encontrar dos números en un arreglo que sumen un valor dado.
- **Invertir un arreglo:** Puedes usar dos punteros, uno al inicio y otro al final, para intercambiar los elementos y así invertir el arreglo.
- **Eliminar duplicados en un arreglo ordenado:** Dos punteros te permiten recorrer el arreglo y eliminar los elementos repetidos de forma eficiente.
- **Comparar o combinar arreglos:** Puedes usar dos punteros para recorrer dos arreglos simultáneamente y realizar comparaciones o fusiones.



Ejemplo: Encontrar un par de números que sumen un valor dado

Supongamos que tienes un arreglo ordenado de números enteros y quieres encontrar dos números que sumen un valor objetivo.

1. Inicializa dos punteros: Uno al inicio del arreglo (**izquierda**) y otro al final (**derecha**).
2. Mientras **izquierda** sea menor que **derecha**:
 - Calcula la suma de los elementos en las posiciones izquierda y derecha.
 - Si la suma es igual al valor objetivo, ¡encontraste el par!
 - Si la suma es menor que el valor objetivo, mueve el puntero **izquierda** una posición a la derecha.
 - Si la suma es mayor que el valor objetivo, mueve el puntero derecha una posición a la **izquierda**.



| Código

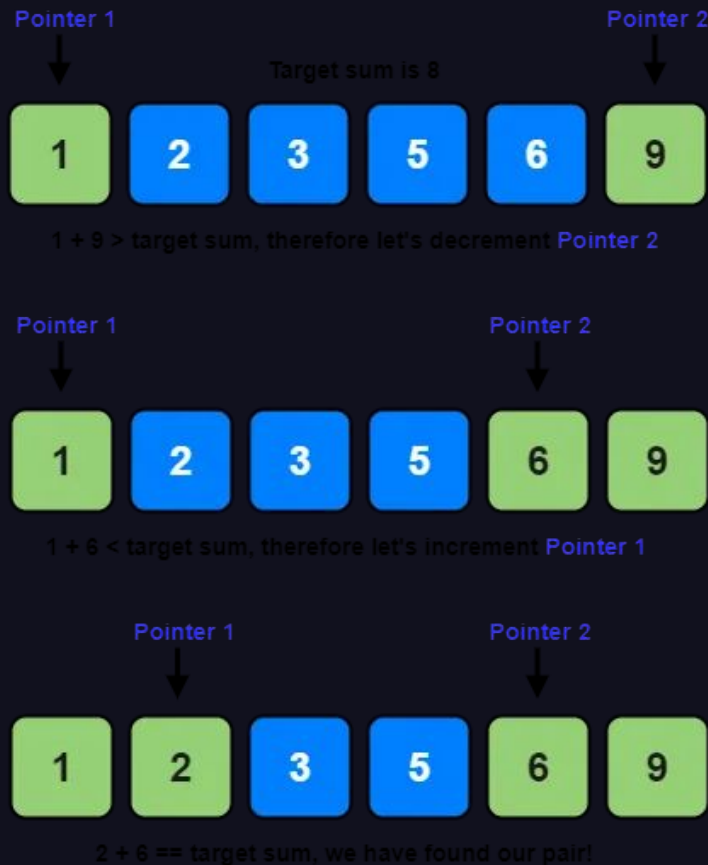
```
function encontrarPar(arr, objetivo) {  
  let izquierda = 0;  
  let derecha = arr.length - 1;  
  
  while (izquierda < derecha) {  
    let suma = arr[izquierda] + arr[derecha];  
    if (suma === objetivo) {  
      return [arr[izquierda], arr[derecha]];  
    } else if (suma < objetivo) {  
      izquierda++;  
    } else {  
      derecha--;  
    }  
  }  
  return null; // No se encontró el par  
}  
  
let arreglo = [2, 7, 11, 15];  
let objetivo = 9;  
let resultado = encontrarPar(arreglo, objetivo);  
console.log(resultado); // [2, 7]
```

DEV.F:



| Ventajas

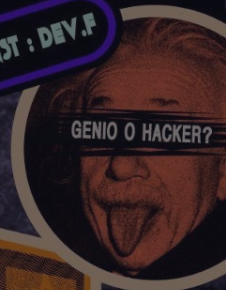
- **Eficiencia:** Reduce la complejidad temporal en muchos casos, evitando bucles anidados.
- **Simplicidad:** Es una técnica relativamente fácil de entender e implementar.



Hora de codear



CONST : DEV.F



DEV.F FUNCTION ()



No olviden



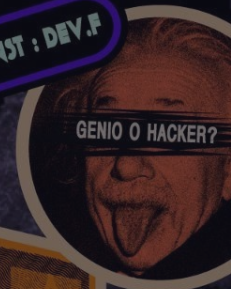
Las lecturas de **edu.devf.la**



DEVF FUNCTION ()



CONST : DEV.F



DEV.F

