Complejidad temporal de la suma objetiva

- tenemos...

```java
public class SumSubArr {  2 usages  majumashe *
    /**
     * @param S La suma objetiva.
     * @return Índices del subarreglo [inicio, fin] o -1 si no se encuentra.
     */
    public static int[] findSubarrayWithSum(Integer[] arr, int S) {  1 usage  majumashe *
        HashTable<Integer, Integer> hashTable = new HashTable<>(arr.length);    //O(1)
        int sum = 0;   // O(1)

        // Guardamos la suma 0 en caso de que el subarreglo empiece en el índice 0
        hashTable.put(0, -1); // O(1)

        for (int i = 0; i < arr.length; i++) { // O(n)
            sum += arr[i]; // O(1)

            // Si (sum - S) existe en la tabla, significa que hay un subarreglo con suma S
            Integer startIdx = hashTable.get(sum - S); // O(1)
            if (startIdx != null) { // O(1)
                return new int[]{startIdx + 1, i}; // Retornamos los índices del subarreglo encontrado
            }

            // Guardamos la suma actual en la tabla hash con su índice
            hashTable.put(sum, i); // O(1)
        }

        return new int[]{-1}; // O(1),  No se encontró un subarreglo con suma S
    }                         O(n)

}
```

tenemos

que son 8n se ignoran porque

no cambian el orden de crecimiento

la complejidad final sería

O(n)