Fundamentos de Algoritmia

Grados en Ingeniería Informática

Convocatoria ordinaria, 20 de enero de 2023. Grupos B, D y G

1. (3.5 puntos) Dado un vector de enteros, y un entero $c > 0$, se dice que un tramo (secuencia de valores consecutivos) de dicho vector es un *c-tramo* cuando la suma de sus elementos es igual a c.

Desarrolla un algoritmo **iterativo eficiente** que, dado un vector de enteros, **todos ellos positivos**, y un entero $c > 0$, devuelva `true` si el vector contiene algún *c-tramo*, y `false` en caso contrario. Debes, asimismo, justificar la corrección (precondición, postcondición, invariante, cota) y el orden de complejidad del algoritmo.

Tu algoritmo se probará mediante casos de prueba que constarán de tres líneas:

- En la primera línea aparecerá el número n de elementos del vector ($0 \leq n \leq 1000000$).
- En la segunda línea aparecerán, en orden, los elementos del vector.
- En la tercera línea aparecerá el valor c que define los *c-tramos*.

La lista de casos de prueba finalizará con una línea con -1. Para cada caso de prueba, el programa imprimirá `SI` si el vector contiene algún *c-tramo*, y `NO` si no contiene *c-tramos*.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	Salida
6 1 10 1 2 3 1 7 5 10 10 1 2 3 17 5 1 2 3 4 10 9 5 10 1 2 3 4 9 5 10 1 2 3 4 21 -1	SI NO SI SI NO

2. (2.5 puntos) En una secuencia de enteros siempre destacan el elemento (o elementos) más pequeños y el elemento (o elementos) más grandes. Para todos los demás elementos se suele utilizar la locución "ni fu ni fa": números que resultan indiferentes y no suscitan ningún tipo de interés.

Desarrolla un algoritmo **divide y vencerás eficiente** que reciba un vector de enteros **ordenado crecientemente** y devuelva un valor booleano que indique si el vector tiene algún elemento *ni fu ni fa*. Debes, así mismo, determinar justificadamente el coste del algoritmo.

Tu algoritmo se probará mediante casos de prueba que constarán de dos líneas:

- En la primera línea aparecerá el número n de elementos del vector ($0 < n \leq 1000000$).
- En la segunda línea aparecerán los elementos del vector ordenado.

La lista de casos de prueba finalizará con una línea con -1. Para cada caso de prueba, el programa imprimirá `SI` si el vector contiene algún elemento *ni fu ni fa* y `NO` en caso contrario.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	Salida
3 1 1 2 6 10 10 10 10 20 30 -1	NO SI

3. (4 puntos) Dos piratas deben repartirse un cofre lleno de valiosas monedas de oro. Para ello han estimado el valor de cada moneda y han decidido que el reparto debe ser lo más justo posible. Como las monedas no se pueden partir (perderían gran parte de su valor), es posible que el botín no se pueda dividir en 2 partes exactamente iguales, pero tratarán de que la diferencia de valor entre las dos partes sea la **mínima** posible. Además, cada uno de los piratas debe obtener un número mínimo de monedas y no pueden quedar monedas sin repartir.

Diseña e implementa un algoritmo **de vuelta atrás** que ayude a los piratas a realizar el reparto. El algoritmo debe calcular: (1) la mínima diferencia de valor alcanzable con repartos válidos, y (2) el número de formas distintas de repartir el botín que permiten alcanzar esa diferencia mínima. Se valorará el uso de estrategias efectivas para reducir el espacio de búsqueda.

Tu algoritmo se probará mediante casos de prueba que constarán de dos líneas:

- La primera línea contendrá 2 enteros que representan, respectivamente, el número de monedas del cofre ($0 < n \leq 100$) y el número mínimo de monedas que debe obtener cada pirata ($0 \leq m \leq n/2$).
- La segunda línea contendrá n números enteros positivos que representan el valor de las monedas.

La lista de casos de prueba finalizará con una línea con -1. Para cada caso de prueba, el programa imprimirá dos números que representan la mínima diferencia de valor alcanzable con repartos válidos y el número de repartos que permiten alcanzar dicha diferencia mínima.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	**Salida**
2 1 7 3 3 1 5 9 3 4 2 3 3 3 3 6 2 1 2 1 1 2 9 8 4 1 2 3 4 6 7 8 9 10 3 4 5 9 3 7 9 9 8 5 2 -1	4 2 1 2 0 6 4 6 0 8 1 76