Fundamentos de Algoritmia

Grados en Ingeniería Informática

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

1. (X 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 `int a[n]`, **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 de prueba 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. (X puntos) En un conjunto de enteros siempre destacan los extremos: 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 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 ($0 \leq n \leq 100000$).
- 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 de prueba 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. (X puntos) Tres piratas deben repartirse un suculento botín consistente en valiosos objetos de 3 tipos: objetos de oro, objetos de plata y piedras preciosas. Para ello han estimado el valor de cada objeto y han decidido que el reparto debe ser lo más justo posible e incluir todos los objetos. El problema es que los objetos no se pueden romper (perderían gran parte de su valor), así que quizás no sea posible dividir el botín en tres partes iguales. Tras mucho discutir, han decido dividir el botín en 3 partes de forma que la diferencia de valor entre la parte más valiosa y la menos valiosa sea la mínima posible. Además, todos quieren conseguir objetos de distintos tipos así que sólo considerarán soluciones en las que cada pirata consigue m objetos de oro, m objetos de plata y m piedras preciosas.

Diseña e implementa un algoritmo 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.

La entrada estará compuesta por varios casos de prueba, donde cada caso de prueba se compone de 3 líneas. La primera línea contendrá 2 enteros que representan el número de objetos del botín ($n > 0$) y el número mínimo de objetos de cada tipo que debe conseguir cada pirata ($m \geq 0$). La segunda línea contiene n números enteros positivos que representan en valor de cada objeto. La tercera línea contiene n enteros que indican el tipo de cada objeto (0 para oro, 1 para plata y 2 para piedras preciosas). La entrada termina con un -1 que no se debe procesar.

Para cada caso de prueba se escribirán dos números que representan la mínima diferencia de valor alcanzable con repartos válidos y el número de repartos que permiten alcanzar dicha diferencia mínima. En caso de que no haya ningún reparto válido se imprimirá la cadena `NO`.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	**Salida**
3 0 4 3 10 0 1 2 3 1 4 3 10 0 1 2 9 1 6 6 6 3 3 3 8 5 5 0 0 0 1 1 1 2 2 2 12 1 1 2 3 4 5 6 7 8 9 10 11 12 1 0 1 0 2 2 0 0 2 1 2 2 -1	7 5 NO 3 215 0 71

3. (X puntos) Alan y Jena son dos famosos exploradores que, tras superar mil y una desventuras, han conseguido llegar a la sala del tesoro de un antiguo templo perdido en medio de la selva. Desafortunadamente, la estructura del templo no parece demasiado estable, así que deben salir de allí lo antes posible y es posible que no puedan volver a entrar. En la sala hay muchos objetos distintos, cada uno de ellos con un determinado peso y valor, y cada uno de los exploradores cuenta con una mochila con cierta capacidad donde guardar algunos objetos.

Diseña e implementa un algoritmo que ayude a los exploradores a elegir qué objetos debe coger cada uno teniendo en cuenta que los objetos no se pueden partir (puesto que perderían su valor). El algoritmo debe calcular el mayor valor que es posible sacar del templo entre los dos de una sola vez, y el número de formas distintas en las que pueden hacerlo. Se valorará el uso de estrategias efectivas para reducir el espacio de búsqueda.

La entrada estará compuesta por varios casos de prueba, donde cada caso de prueba se compone de 3 líneas. La primera línea contendrá 3 enteros que representan el número de objetos en la sala ($n > 0$) y el peso que pueden meter en cada mochila ($m_1 \geq 0$, $m_2 \geq 0$). La segunda y tercera líneas contienen, respectivamente, los valores y pesos de cada uno de los n objetos. La entrada termina con un -1 que no se debe procesar.

Para cada caso de prueba se escribirán dos números que representan el máximo valor que los exploradores pueden sacar de una sola vez y el número de formas en las que pueden hacerlo. Para contar el número de formas debes suponer que todos los objetos son distinguibles entre sí y que importa la mochila en la que se meta cada uno.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	Salida
3 1 1 5 4 3 1 1 2 3 1 1 1 1 1 2 2 2 5 3 3 3 3 3 2 2 4 2 2 2 2 8 5 3 1 5 4 3 7 7 5 4 2 1 4 6 3 1 3 4 -1	9 2 0 1 6 2 24 2