**Problema A**

Ricardo Enrique González

Camilo Zambrano

201425733

201515438

## 1. *Algoritmo de solución*

Para este problema solo probamos esta única alternativa.

Hay que aclarar:

ExtendedEuclid Retorna [GCD,X,Y] donde:

GCD = X* Z[j] + Y* Z[i]

Después de recuperar los datos y colocarlos en un arreglo, transformarlos en valores numéricos, se pasa a realizar un doble recorrido en donde se utiliza el algoritmo de ExtendedEuclides para hallar el MCD entre cada número, ExtendedEuclid(Z[j],Z[i]) . Si se consigue que el GCD entre alguno de los números en el Arreglo de entrada Z es 1, el algoritmo de ExtendedEuclides nos devuelve el X y el Y que forman parte del vector P respuesta.

Se imprime P.

Si no se consigue que el GCD entre alguno de los números de Z es 1, significa que no hay combinación lineal posible y se imprime un *.

**CTX**
Tamaño del problema: n
$Z[z_0..z_{n-1}]$ con $0<n<100$
Cada caso de prueba se define con una línea que contiene n+1 números naturales:
$n\ z_0\ , z_1 \ldots z_{n-1}$
con $0 \leq Z[z_i] < 10^5$

**PRE:**
Se inicializa un arreglo de Strings del tamaño de Z[0..n-1] que contiene los números a procesar.
Se inicializa otro arreglo que va contener los enteros que representan el vector respuesta P[0..n-1].
Se inicializa un ultimo arreglo de tamaño 3, que va utilizar el algoritmo ExtendedEuclid para albergar las respuestas que se van a colocar en P.
3 variables simples que van a manejar los recorridos del arreglo: k, j e i.

**POS:**
Si se consigue una combinación lineal.
$z = \langle z0, \ldots , zn-1 \rangle$ Lista entrada de enteros que representa vector z
$p = \langle p0, \ldots , pn-1 \rangle$ Lista respuesta de enteros que representa vector p donde
$CLz(P) = p\ .\ z = 1$
Se imprime p

De lo contrario, si no consigue una combinación lineal.
Se imprime *

## 2. *Análisis de complejidades espacial y temporal*

Operaciones a realizar:

Se asume que la lectura y el almacenamiento del string cuesta **O(1)**
Procesar el string obtenido por consola a Vals(arreglo con valores del problema) **O(1)**
Transformar los strings de vals en enteros para realizar los cálculos. **O(1)**

Realizar el recorrido doble del arreglo($n^2$**)** probando con ExtendedEuclid ref.[2] log(max ({a,b}).
**O($n^2$*log(max ({a,b}) )**

Como podemos observar nos hace falta la complejidad temporal de encontrar el GCD. Para la complejidad temporal de este algoritmo existen diversos criterios, en general la mayoría de se concuerdan en que la complejidad es logarítmica, lo cual tiene sentido debido a que cada vez que intentamos simplificar estamos dividiendo, es decir, estamos disminuyendo logarítmicamente la cantidad de a y b

Cabe aclarar que $0 \leq a \leq b \leq 10^5 \ \wedge \ 0 < n < 100$

Por lo tanto, en la operación mencionada anteriormente se elimina el término **log(max ({a,b})**], ya que su valor máximo es 16(log(100.000) = 16) y termina siendo constante, lo importante seria el valor de N. **O($n^2$)**

Por ultimo, se realiza un recorrido n sobre la respuesta en donde se va imprimiendo cada valor que forma parte del vector p. **O(1)**

T(n) = O(1+1+1+ $n^2$+1) = O($n^2$).

Values: arreglo que contiene los strings que serán transformados a números. **O(n)**

Vals: arreglo que contiene todos los números del vector Z. **O(n)**

Nums: arreglo que almacena la respuesta obtenida por el algoritmo de Extended Euclid. **O(3)**

Resp: arreglo que contiene todos los números del vector P. **O(n)**

Conseguimos: variable local booleana que indica si se consiguió un GCD = 1 para salirse del ciclo y continuar al siguiente problema. **O(1)**

Respuesta: variable que se utiliza para escribir la respuesta e imprimirla. **O(1)**

S(n) = θ (n+n+3+n+1+1) = θ (3n+5) = O(n)

## 3. *Comentarios finales*

El algoritmo se considera bastante eficiente, sin embargo consideramos que debe existir alguna manera mas eficiente de hallar la solución al problema. Teniendo en cuenta el tamaño de las entradas el algoritmo no se tarda mucho en conseguir la solución, pero si se quiere conseguir vectores de dimensiones mas grandes, se volverá mucho mas ineficiente. El tamaño de los valores dentro del vector nunca van afectar lo suficiente para tomar prioridad sobre la cantidad de dimensiones del vector por su naturaleza logarítmica en el algoritmo ExtendedEuclid.

Nota: Al correr el archivo si no imprime el último valor de inmediato, darle 2 veces a enter.

Referencias:

[1] https://en.wikipedia.org/wiki/Euclidean_algorithm

[2] https://cgi.csc.liv.ac.uk/~martin/teaching/comp202/Java/GCD.html