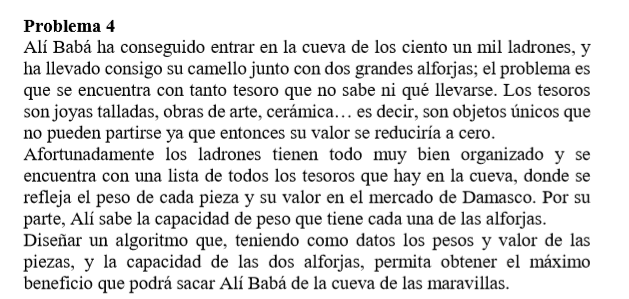
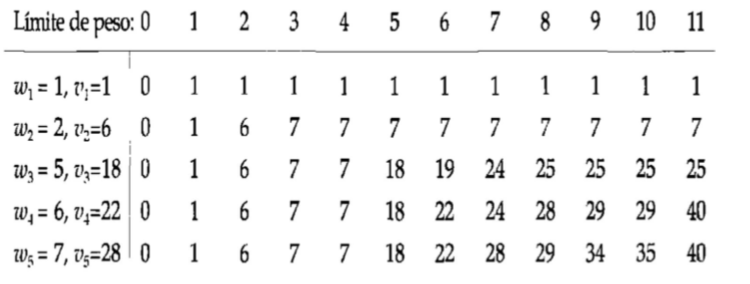
PL4

JAVIER MARTÍN GÓMEZ

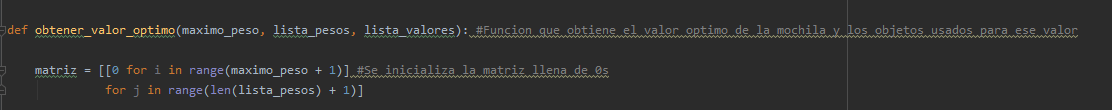
47231977M



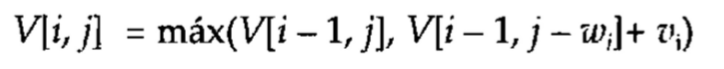
Queremos obtener el máximo beneficio utilizando dos mochilas sin excedernos del peso de ambas. Primero de todo, tendremos que crear una función que nos cree una matriz como la que hemos visto en teoría, tal que así:



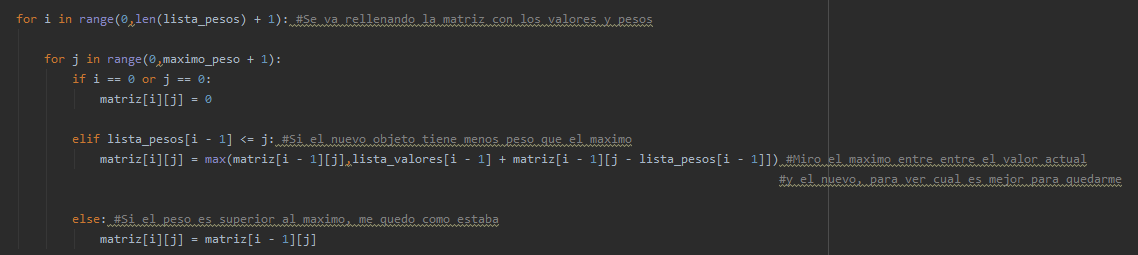
Donde w es el peso y v es el valor.

Por lo tanto, primero tendremos que inicializar la matriz llena de 0s con tantas filas como valores haya y con tantas columnas como máximo peso haya.

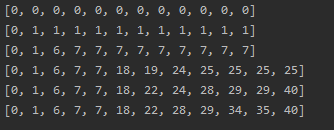
Después, iremos rellenando la tabla gracias la regla general, explicada en teoría:



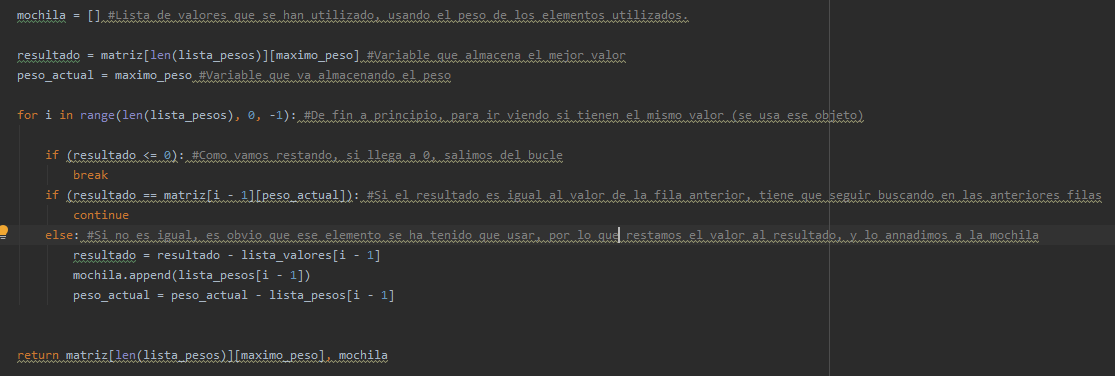
Comprobamos, que, si el peso del objeto nuevo es menor que el máximo peso, aplicamos la fórmula anterior, el máximo entre el valor del objeto actual y el del nuevo. Si el peso se excede del máximo, se queda igual. Para la primera posición se rellenará con un 0.



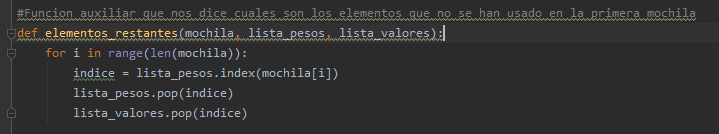
De esta manera, obtenemos la matriz con el máximo valor que podemos obtener. Para comprobar que la matriz se crea correctamente, podemos introducir los datos del ejemplo anterior. Como podemos ver, se crea la matriz del ejemplo anterior:



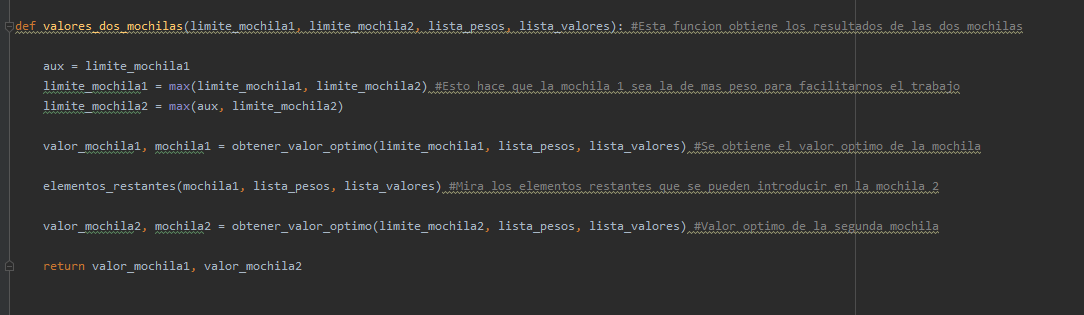
Ahora, tenemos que hallar los valores que componen ese máximo valor. Para ello, tenemos que ir mirando la matriz. Empezamos desde la última posición, ya que tendrá el valor máximo. Si el valor de la posición anterior es igual, significa que ese valor no se ha usado, por lo que pasamos al siguiente. Si no es igual, significa que ese valor se ha usado, por lo que lo añadimos a la mochila y restamos ese valor al resultado actual. Cuando el resultado llegue a 0, salimos del bucle y devolvemos, para usarlo posteriormente, la mochila con los valores usados y el valor de ella.



Ya hemos conseguido hallar el algoritmo para una mochila, pero nos faltaría saber cómo obtener el mejor resultado utilizando dos mochilas. Para ello, primero usamos una función auxiliar que comprueba los elementos que no se han usado en la primera mochila, ya que, para la segunda mochila, no se pueden utilizar.



Por último, creamos una función que obtiene el valor de la primera mochila (primero siempre vamos a usar la de más peso) y el valor de la segunda, llamando a la primera función que hemos creado que obtenía el valor óptimo a partir de una lista de valores, de pesos y de un límite de peso en la mochila.



A partir de los siguientes objetos compuestos por los siguientes valores y pesos



podemos obtener el valor de dos mochilas de 11 y 8 como máximo peso.



Para la primera se usan los valores 18 y 22 con pesos de 5 y 6 respectivamente.

Para la segunda (sin poder usar los valores anteriores) se usan los valores 28, 6 y 1, con pesos de 11, 2 y 1 respectivamente.

Otro ejemplo sería este con máximo peso 14 y 12:

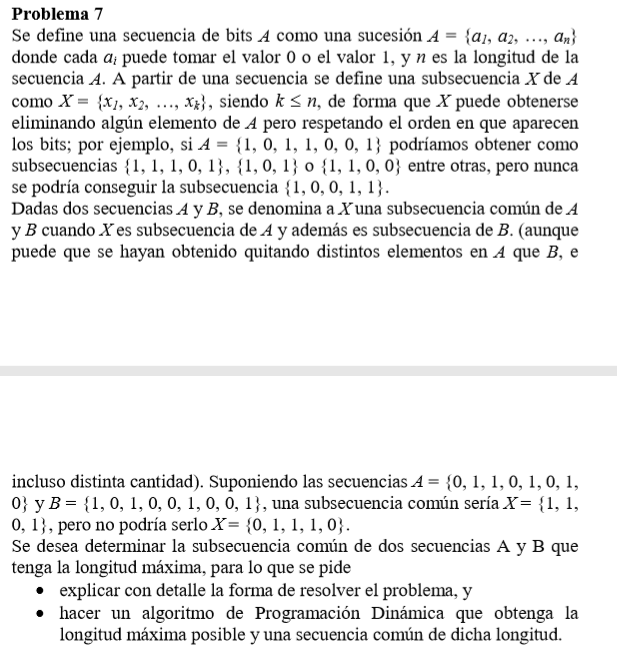


Con el resultado:

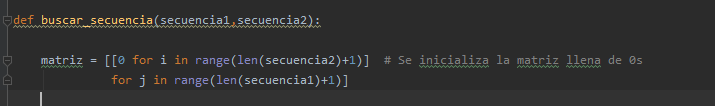


La mochila 1 tendría los valores 23,7 y 8 con pesos 9,1 y 3 respectivamente.

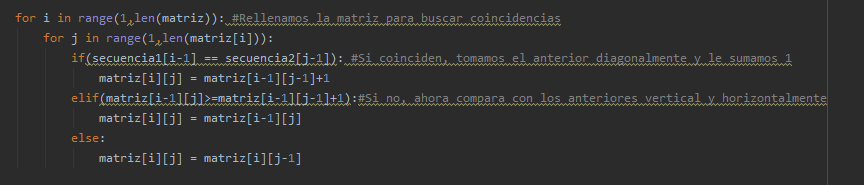
La mochila 2 tendría los valores 10 y 10 con peso 5 y 7 respectivamente



Queremos saber la subsecuencia máxima común de dos secuencias. Para ello, usaremos una matriz de dimensiones longitud\_secuencia1+1xlongitud\_secuencia2+1, que se inicializará a 0.



La matriz se irá rellenando diagonalmente, según vaya encontrando valores comunes, si encuentra coincidencia con el anterior en diagonal, le suma 1, si no, comprueba horizontal o verticalmente y arrastra ese valor a la posición.



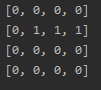
Se entenderá más claro con un ejemplo. Por ejemplo, para las secuencias A=010 y B=001

Primero se haría la matriz:



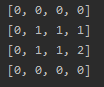
Y se empieza a comprobar la primera posición de ambas, si es igual, se suma 1 a la posición refiriéndose a que ha encontrado un bit en común. Y así, se va comprobando posición por posición, representando los números de cada fila, los bits iguales hasta esa posición.

Por ejemplo, la primera iteración del ejemplo que estamos explicando sería la siguiente:



Esos 1s se refieren a que, empezando por la primera posición de la primera secuencia, se puede obtener 1 bit en común. El 0 coincide dos veces con un 0 de la segunda secuencia, por lo que se introduce dos veces el 1 (se suma 1 a la diagonal anterior). La última posición no coincide por lo que sería 0, pero comprueba horizontal y verticalmente si hay algún número mayor y lo introduce (había un 1 a su izquierda). Por lo tanto, ahora mismo la subsecuencia mayor sería 0.

Pasamos a la siguiente iteración:



En esta iteración se empieza a comprobar el 1 de la primera secuencia. Al no coincidir con los dos primeros 0s de la segunda se arrastran los de arriba (comprueba horizontal y verticalmente), por eso sigue siendo 1. Al coincidir con el 1 del final, suma 1 a la diagonal anterior por lo que ahora la subsecuencia mayor es 2, ya que coincide (en negrita) la subsecuencia de A **01**0 con la de B 0**01**. Por lo tanto, ahora la subsecuencia mayor sería 01.

Pasamos a la última iteración:



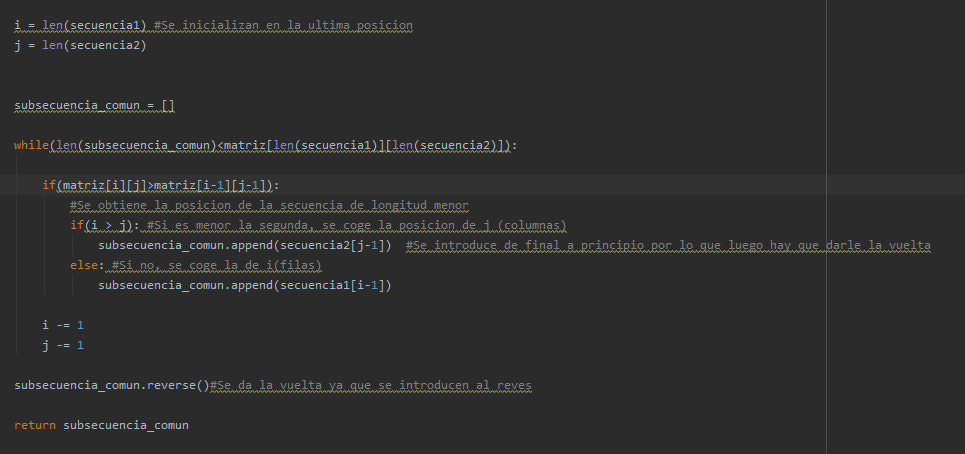
Se comprueba el último 0 de la primera secuencia. Se comprueba con el primer 0 de la segunda, como coinciden, suman 1 a la diagonal anterior. Se comprueba con el segundo 0 y al coincidir se le suma 1 al anterior diagonalmente, y al ser 2 (el anterior 1 era por la coincidencia de los 0s anteriores), habrá otra subsecuencia mayor (00). Se comprueba el 0 con el 1, y no coinciden por lo que no se le suma nada a la diagonal anterior y seguiría siendo 1, pero, al haber horizontal y verticalmente un 2, obtiene ese 2 y lo coloca en la posición (en la última posición siempre tiene que estar la longitud mayor).

Al haber dos 2 en la matriz final, significa que habrá dos subsecuencias comunes (en este caso 00 y 01).

Una vez que hemos encontrado la máxima longitud, tenemos que encontrar una subsecuencia de esa longitud común a las dos secuencias.

El procedimiento es el siguiente: vamos iterando la matriz diagonalmente hacia atrás desde la última posición. Si el anterior es menor a esa posición, significa que se introduce ese bit. Si no es menor (será igual) significa que ese bit no es coincidente, por lo tanto, no se introduce, y pasa a la siguiente posición.

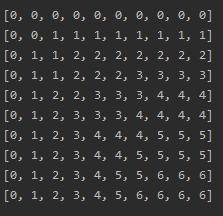
Se van insertando los bits en una lista empezando por el final, por lo que luego habrá que darle la vuelta.



La salida para el ejemplo anterior sería:



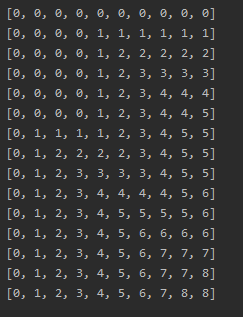
Otro ejemplo sería A=011010101 y B=101001000, cuya matriz sería:



Y la salida sería:



Otro ejemplo más podría ser A=1111000111110 B=000111101 cuya matriz sería:



Y la salida sería:



El algoritmo solo devuelve una cadena, aunque es posible que haya más de una habitualmente.