

1.- La complejidad del algoritmo que se utilizó es lineal,  $O(n)$ . Si se compara con la versión recursiva simple, cuya complejidad es exponencial  $O(2^n)$ , se nota una mejora drástica del rendimiento; el costo computacional disminuye notablemente para valores grandes de  $n$ .

2.- La complejidad del algoritmo es  $O(n * W)$ , donde  $n$  es el número de elementos y  $W$  la capacidad de la mochila. Esto se debe a la estructura de bucles anidados: se itera por cada uno de los  $n$  objetos y, para cada uno, se actualiza el array de capacidades hasta  $W$ . La inicialización tiene un coste despreciable frente al proceso iterativo principal.

3. a.- La complejidad es  $O(n * m)$  (siendo  $n$  y  $m$  las longitudes de las cadenas), ya que es necesario llenar la totalidad de una matriz de dimensiones  $(n+1) * (m+1)$  para resolver los subproblemas superpuestos.

prematamente en cuanto se calcula un valor mayor o igual a  $k$ . Sin embargo, la complejidad asintótica del peor caso se mantiene en  $O(n * m)$ , dado que es posible que dicha longitud solo se alcance al finalizar el cómputo.

4a.- El código usa dos matrices uno para el mínimo de monedas y otro de vectores.

El coste total del programa es la suma de los dos costes de ambas matrices.

Para  $m$  su coste es de  $O(n \cdot C)$  y para  $ch$ ,  $O(n^2 \cdot C)$

4b.- Si se añadiera una restricción de número máximo de monedas por tipo eso complicaría en las decisiones, no se podría usar la formula que se basa en repetir la fila indefinidamente.

5.- La complejidad del enfoque dinámico es  $O(mn)$  tiempo y  $O(mn)$  espacio, por los bucles for de ambas matrices  $m \times n$