Técnica de Fuerza Bruta: Esta forma de resolver el problema consiste en definir un dominio de posibles soluciones del problema que consideremos y un algoritmo que permita alcanzarlas a todas.

En nuestro problema, vamos a definir nuestro espectro de posibles soluciones al conjunto de partes del conjunto inicial, S. Este conjunto estará formado por todos los posibles subconjuntos que se puedan tomar con elementos de S. Dado que cada elemento tiene dos opciones, estar o no en un determinado subconjunto, las opciones son 2 . 2 . 2 …… 2 = 2^n.

Por la propia definición, es trivial que si llegará a existir un R contenido en S y que la sumatoria de sus elementos sea V lo vamos a poder encontrar. Además, en particular la que tenga menos elementos.

Desde el punto de vista de complejidad, la definimos en función del tamaño de entrada, ydefiniendo n como el tamaño de S, nuestro algoritmo será theta(n \* 2n), porque se recorrerá cada subconjunto del conjunto de partes de S y para cada uno revisaríamos cuáles de los n elementos de S están en ese subconjunto.

Backtracking: En esta técnica también es necesario definir el espectro de posibles soluciones del problema. Vamos a considerar también que éstas son el conjunto de partes de S.

Lo que caracteriza a esta forma de resolver el problema es que vamos a buscar definir situaciones en las que seguir computando la solución no tenga sentido. A esto se lo llama podas y existen de dos tipos: por optimalidad y factibilidad. Dado que en nuestro problema vamos a operar con los elementos de todos los posibles subconjuntos de S, es razonable imaginar que muchas operaciones se realizarán múltiples veces.

“PONER IMAGEN DE RAMA DE SOLUCIONES”

En las podas de factibilidad, buscamos caracterizar cómo es un subconjunto que **no** puede ser solución y dejar de computar todos los demás ‘subconjuntos padre’, es decir, los que contengan a éste. Por otro lado, para una cota optimalidad se identificará las situaciones en las que seguir computando un subconjunto no tenga sentido debido a que en el mejor caso, es decir, si llegara a ser solución, no sería mejor que la que ya encontramos.

En nuestro problema:

Para formar el subconjunto solución se recorrerán los n elementos de S. Sea un subconjunto vacío R, se fijará en la primera posición cada uno de los valores de S y para los siguientes posiciones de R, n, se aplicará recursivamente nuestro algoritmo con S’ = S – {e\_i}. Nuestras podas estarán determinadas por lo siguiente:

1. Por optimalidad: Si en el proceso de computo de una posible solución, se llegará al caso en que el subconjunto de elementos actual contiene más elementos que la mejor solución encontrada hasta ahora, entonces no se buscará y computará ninguno de los subconjuntos que surjan de tener como los primeros k elementos del subconjunto actual y siendo k el tamaño de éste. A nivel algorítmico significará que esa a partir de esa posición no se buscará las posibles combinaciones con los elementos restantes de S’.
2. Por factibilidad: Si en el proceso de cómputo de una posible solución, se llegara al caso en que la sumatoria de los elementos del conjunto solución actual, entonces de igual forma no se buscará conseguir los conjuntos que surjan de combinar los elementos restantes de S’ con el conjunto solución actual.

Respecto a la complejidad algorítmica y dado que la establecemos en función de tamaño de la entrada, |S| y V, y para el peor caso, también tendrá complejidad asintótica perteneciente a O(2^n \* n). El peor caso sería un valor objetivo mayor que la sumatoria de todos los elementos de S. En ese caso nunca se utilizarán ninguna de las dos podas y se llegarán a formar las 2^n combinaciones de conjuntos, con sus respectivos cómputos de sumas interiores. Hay dos diferencias claves con respecto a fuerza bruta: 1\_en bf SIEMPRE será un peor caso, la complejidad es theta(2^n \* n) porque no va a existir un caso en el que el algoritmo calcule menos, dado que va a pasar por todas las soluciones del dominio definido. En backtracking también vimos que el peor caso es O(2^n \* n), pero su cota inferior no es la misma que su cota superior, como en bf, si no que en los casos en que, por ejemplo, todos los elementos de S superen a V, entonces la cantidad de pasos realizada sería O(|S|) = O(n), dado que al poner el primer elemento entraría la poda de factibilidad y se descartarían todos los subconjuntos con el S[0] dentro, se eliminaría el elemento y se agregaría el siguientes, S[1], y el resultado sería el mismo que para S[0]... Se repite el mismo para todos los restantes elementos de S y se llega a una solución vacía.