**Nicolás Aguilar León - 201530741**

**Juan Sebastián Millán Lejarde - 201516818**

**Algoritmo de solución:**

El algoritmo escogido se selecciona a partir de la similitud del problema B con el problema del morral, por lo cual se puede hacer la analogía en donde el porcentaje de participación de cada accionista equivaldría al peso de cada objeto y el porcentaje de participación que se desea alcanzar al peso máximo que aguanta el morral. Dado lo anterior, se hizo uso de la técnica de programación dinámica para desarrollar este algoritmo. Además, se tuvo en cuenta que, usando otra solución más directa, se harían múltiples cálculos repetidos innecesarios y precisamente la programación dinámica evita estos contratiempos. Se usaron las etapas que se definieron en clase para desarrollar la solución.

**Definición de lenguaje:**

Primero se define sePuede(*porcentaje* ,j) como una función que retorna verdadero en caso de que se pueda construir el entero *porcentaje* a partir de una suma que incluya a p.n y/u otros elementos en tal que .

De tal manera que la respuesta a este problema se encuentra buscando el mínimo porcentaje mayor a 50 que se puede armar a partir de una sumatoria que incluya a p.n y/o el porcentaje de participación de los otros accionistas. Formalmente:

*Se utiliza el mínimo porcentaje que cumpla estas condiciones ya que de esta manera se maximiza la ganancia del n-esimo accionista.*

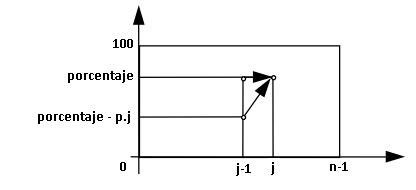
***Recurrencia:***

*Si no se cuenta con ningún otro accionista para armar mediante una suma el porcentaje deseado. Solo se podrá armar a porcentaje si este es igual al p.n.*

*Si el porcentaje de participación del j-esimo accionista es mayor que el porcentaje que intentamos armar no es posible incluirlo en la sumatoria.*

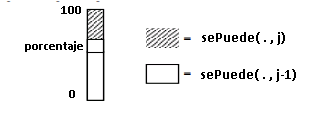
*Si el porcentaje de participación del j-esimo accionista es menor al porcentaje que intentamos armar, existen dos opciones, que este se incluya en la sumatoria o no.*

**Diagrama de necesidades:**

**

***Invariante:***

Como al final solo nos interesa la última columna, no es necesario almacenar la matriz completa, por lo tanto, podemos hacer uso como estructura de datos un arreglo de booleanos de tamaño 101 donde se conserve la siguiente invariante si se llena el diagrama de necesidades de arriba abajo y de izquierda a derecha:



Al final se toma el arreglo que da como resultado, se empieza a recorrer desde porcentaje = 51 hasta 100, se toma el primer porcentaje en este orden que indique true en esa posición del arreglo y se reemplaza en esta fórmula.

**Análisis de complejidades espacial y temporal:**

**Espacial:**

La complejidad espacial depende del número de accionistas en el problema debido a que se guarda el porcentaje que le corresponde a cada uno en un arreglo de tamaño n al inicio del algoritmo. Pero si el número de accionistas es menor a 101 la complejidad espacial será constante debido a que el tamaño del arreglo que se escogió como estructura de datos para obtener G sin importar el caso será siempre de tamaño 101.

**Temporal:**

La complejidad temporal depende del número de accionistas cuando se llena la estructura de datos y del último recorrido que se realiza sobre la mitad del arreglo final, este último tiempo siempre será constante. Por lo tanto, predomina la complejidad de llenar la estructura de datos seleccionada.

**Comentarios finales:**

En esta solución de manera similar a la solución estudiada en clase del problema del morral, se hace el uso de la programación dinámica para evitar hacer uso de llamadas recurrentes que implicarían hacer cálculos repetidos. Por medio de la programación dinámica cada sub-problema se resuelve solo una vez, y cada vez que se necesite ese valor, solo bastara con sacarlo directamente de la estructura usada, lo que nos permite ahorrar mucho tiempo. El uso de una estructura de datos auxiliar nos permite cambiar espacio por tiempo.