Algoritmos y Estructuras de Datos III
Segundo Cuatrimestre 2018
Trabajo Práctico 1

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Eligiendo justito

Contexto y motivación

En muchas ocasiones sucede que se tiene un conjunto de ítems con un valor asociado y se quiere elegir un subconjunto de ellos que sume exactamente un valor objetivo. Esto en general está asociado a situaciones en donde se tiene un cierto recurso limitado y se quiere encontrar la manera de elegir ciertos ítems que completen de manera exacta ese recurso. Encontrar el mínimo o el máximo de la suma de los ítems es una tarea muy sencilla. Sin embargo, saber si se puede sumar exactamente un número dado, es una tarea mucho más compleja.

El problema que se menciona se puede encontrar en la literatura como el *Problema de suma de subconjuntos* o *Subset Sum* en inglés. El problema es fundacional en Ciencias de la Computación y se encuentra en la lista de 21 problemas *difíciles* presentada por Karp en [1].

Existen muchos problemas del mundo práctico en donde se debe resolver un *Subset Sum* o una variación del mismo. Supongamos que tenemos un conjunto de procesos que tenemos que asignar a dos procesadores. Cada proceso tiene asociado un valor t_i que es la cantidad de slots de tiempo que requiere para ser ejecutado. Lo que desearíamos en este contexto es dividir los procesos de tal manera que minimicemos el tiempo final T requerido para ejecutarlos todos. Para lograr esto, lo ideal sería dividir de manera equitativa los procesos. Es decir, encontrar un subconjunto de tareas cuya duración sume exactamente $\frac{T}{2}$. Esta situación no es más que un caso particular del problema de *Subset Sum*. En particular a esta especialización del problema (en donde hay que sumar exactamente la mitad del total) se lo conoce como *Partition Problem*.

Así como la mencionada, hay otras situaciones en donde se requiere de este problema, el cual no tiene una resolución algorítmica simple, lo que motiva su estudio en este trabajo.

El problema

Dado un conjunto de n ítems S, cada uno con un *valor* asociado v_i, y un valor objetivo V, decidir si existe un subconjunto de ítems de S que sumen exacto el valor objetivo V. Si existe dicho conjunto, decir cual es la mínima cardinalidad entre todos los subconjuntos posibles. En otras palabras decidir si existe $R \subseteq S$ tal que $\sum_{i \in R} v_i = V$, y si existe, devolver la menor cardinalidad posible de R.

Para este problema, asumiremos que todos los valores mencionados son enteros no negativos.

Enunciado

El objetivo del trabajo práctico es resolver el problema propuesto de diferentes maneras, realizando posteriormente una comparación entre los diferentes algoritmos utilizados.

Se debe:

1. Describir el problema a resolver dando ejemplos del mismo y sus soluciones.

Luego, por cada método de resolución:

2. Explicar de forma clara, sencilla, estructurada y concisa, las ideas desarrolladas para la resolución del problema. Para esto se pide utilizar pseudocódigo y lenguaje coloquial combinando adecuadamente ambas herramientas (**¡sin usar código fuente!**). Se debe también justificar por qué el procedimiento desarrollado resuelve efectivamente el problema.
3. Deducir una cota de complejidad temporal del algoritmo propuesto (en función de los parámetros que se consideren correctos) y justificar por qué el algoritmo desarrollado para la resolución del problema cumple la cota dada.
4. Dar un código fuente claro que implemente la solución propuesta.

   El mismo no sólo debe ser correcto sino que además debe seguir las *buenas prácticas de la programación* (comentarios pertinentes, nombres de variables apropiados, estilo de indentación coherente, modularización adecuada, etc.).

Por último:

5. Realizar una experimentación computacional para medir la performance de los programas implementados, comparando el desempeño entre ellos. Para ello se debe preparar un conjunto de casos de test que permitan observar los tiempos de ejecución en función de los parámetros de entrada, analizando la idoneidad de cada uno de los métodos programados para diferentes tipos de instancias.

A continuación se listan los algoritmos que se deben considerar, junto con sus complejidades esperadas (siendo n la cantidad de ítems a considerar y V el valor objetivo):

- Algoritmo de fuerza bruta. Complejidad temporal perteneciente a $\mathcal{O}(n \times 2^n)$.
- Algoritmo de *Backtracking*. Complejidad temporal perteneciente a $\mathcal{O}(n \times 2^n)$. Se deben implementar dos podas para el árbol de backtracking. Una poda por factibilidad y una poda por optimalidad.
- Algoritmo de Programación Dinámica. Complejidad temporal perteneciente a $\mathcal{O}(n \times V)$.

Solo se permite utilizar $c++$ como lenguaje para resolver el problema. Se pueden utilizar otros lenguajes para presentar resultados.

La entrada y salida de los programas **deberá hacerse por medio de la entrada y salida estándar del sistema**. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Deberá entregarse un informe impreso **con a lo sumo 10 paginas** que desarrolle los puntos mencionados.

Parámetros y formato de entrada/salida

La entrada consistirá de una primera línea con dos enteros n y V, correspondientes a la cantidad de ítems disponibles y al valor objetivo, respectivamente. Luego le sucederán n líneas con un entero, v_i, correspondientes al valor de cada uno de los ítems.

La salida consistirá de un único número entero que representará el valor mínimo cardinal entre todos los subconjuntos posibles. En caso de no existir ningún subconjunto, debe devolverse un -1.

Entrada de ejemplo	**Salida esperada de ejemplo**

`5 25`
`10`
`15`
`5`
`10`
`5`
	`2`

Fechas de entrega

- *Formato Electrónico:* Domingo 9 de Septiembre de 2018, hasta las **23:59 hs**, enviando el trabajo (informe + código) a la dirección `algo3.dc@gmail.com`. El subject del email debe comenzar con el texto `[TP1]` seguido del apellido del alumno.
- *Formato físico:* Lunes 10 de Septiembre de 2018, a las 18 hs.

Importante: El horario es estricto. Los correos recibidos después de la hora indicada no serán considerados.

Referencias

[1] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.