Algoritmos y Estructuras de Datos III
Primer Cuatrimestre 2020
Trabajo Práctico 1

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

Optimizando Jambo-tubos

Contexto y motivación

La cadena de supermercados Jambo quiere construir una serie de robots con el fin de optimizar diversas operaciones de sus locales. Por ejemplo, debido a la reducción en el uso de bolsas de plástico, Jambo desea ofrecer un servicio de *empacado* de productos en los denominados *Jambo-tubos*. En cada tubo se apilan los distintos productos, uno arriba del otro, es decir, un producto no puede estar ubicado al lado de otro, solamente arriba o abajo.

Como es de conocimiento general, hay ciertos productos (e.g. un paquete de papas fritas) que si tienen cierto peso encima se aplastan, lo cual es una situación indeseable. Por lo tanto, cada robot debe ir considerando los productos que vienen por la cinta transportadora y decidiendo si lo agrega a su tubo o no. Como programar robots es una tarea difícil, se propone hacer una prueba de concepto donde hay un solo Jambo-tubo. El objetivo del robot entonces es, sabiendo el orden en que los elementos vienen por la cinta transportadora, su *peso* y su *resistencia*, es decir, cuánto peso pueden tener arriba sin ser aplastados, decidir cuáles productos poner el tubo de manera tal de poder guardar la mayor cantidad posible. Observar que el tubo tiene una resistencia propia, de manera tal que hay un límite de peso total que se puede cargar.

El problema

Dado un Jabotubo con resistencia R, y una secuencia ordenada de n productos S, cada uno con un *peso* asociado w_i y una resistencia asociada r_i, determinar la máxima cantidad de productos que pueden apilarse en un tubo sin que ninguno esté aplastado. En la Figura 1 se ilustra un ejemplo con $R = 50$, $n = 5$, $w = [10, 20, 30, 10, 15]$ y $r = [45, 8, 15, 2, 30]$. La solución óptima es 3, y consiste en tomar los elementos 1, 3 y 4. Notar que la solución alternativa tomando los elementos 1, 3 y 5 no es factible porque la suma de sus pesos es $55 > R$.

Nota: Los productos solamente pueden ser apilados en el orden en que son recibidos de la cinta transportadora. Para este problema, asumiremos que todos los valores mencionados son enteros no negativos.

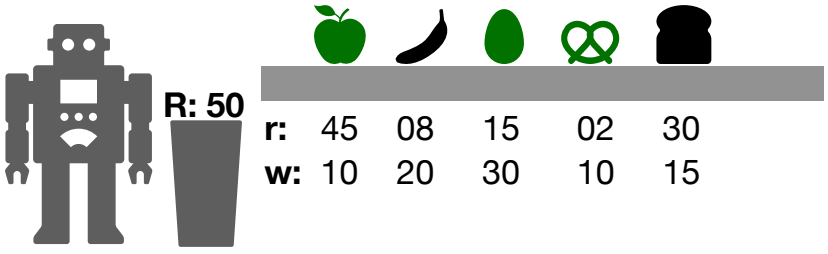

Figura 1: Ejemplo de instancia del problema de Jambo-tubos.

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 productos a considerar y R la máxima resistencia entre todos los productos):

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

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** (sin contar la carátula) que desarrolle los puntos mencionados.

Parámetros y formato de entrada/salida

La entrada consistirá de una primera línea con dos enteros n, R correspondientes a la cantidad de productos disponibles y a la resistencia del jambotubo, respectivamente. Luego le sucederán n líneas con dos enteros, w_i y r_i, correspondientes a los pesos y a las resistencias de cada uno de los productos.

La salida consistirá de un único número entero que representará la máxima cantidad de productos que pueden apilarse sin que ninguno sea aplastado.

Entrada de ejemplo	Salida esperada de ejemplo
5 50 10 45 20 8 30 15 10 2 15 30	3

Fechas de entrega

- *Formato Electrónico:* Domingo 17 de Mayo de 2020, hasta las **23:59 hs**, enviando el trabajo (informe + código) en un zip a través del campus virtual.

Fechas de reentrega

- *Formato Electrónico:* Domingo 7 de Junio de 2020, hasta las **23:59 hs**, enviando el trabajo (informe + código) en un zip a través del campus virtual.