Algoritmos y Estructuras de Datos III
Segundo Cuatrimestre 2019
Trabajo Práctico 1

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

Optimizando los Portfolios

Contexto y motivación

Cotidianamente existen muchas situaciones en las que de un conjunto de ítems, se quiere elegir un subconjunto de ellos maximizando el beneficio obtenido por los mismos. Dependiendo el contexto, el beneficio asociado a un ítem dado puede tener diversos significados. En general, lo que sucede al tener estas situaciones es que no podemos simplemente elegir todos los ítems que aporten un beneficio positivo, debido a que existen restricciones de capacidad que limitan la cantidad de ítems posibles a elegir dentro del conjunto.

El problema mencionado se encuentra en la literatura como el *Problema de la Mochila* o *Knapsack problem* en inglés. Dada su importancia tanto de un punto de vista teórico como práctico, existen muchos trabajos abocados a este problema y a sus variaciones. En [2] y [3] se pueden encontrar recopilaciones completas sobre esta categoría de problemas.

Existen muchos problemas del mundo práctico en donde se debe resolver un *Knapsack problem* o una variación del mismo. Supongamos que tenemos un conjunto de acciones bursátiles en las que podemos invertir. Cada acción i tiene un precio asociado p_i y un retorno esperado r_i. Lo que nos gustaría es saber en cuales acciones invertir de manera tal que el retorno esperado total sea el mayor posible, mientras que los precios sumados de las acciones elegidas no excedan un valor fijo P, que es nuestro capital de inversión original. Si las acciones se pudieran partir, es decir si se pudiese invertir en una fracción de acción, este problema tendría una resolución golosa tal como fue demostrado en [1]. Basta con ordenar de forma decreciente las acciones según la relación $\frac{r_i}{p_i}$ e ir tomando golosamente las mejores acciones asociadas a esta relación hasta alcanzar los P pesos de capital inicial. Sin embargo, al no ser las acciones divisibles (como en muchas otras situaciones análogas), este problema no tiene una resolución tan directa y motiva justamente todo el estudio que tiene dedicado.

El problema

Dado un conjunto de n ítems S, cada uno con un *tamaño* asociado w_i y un *beneficio* asociado p_i, y una mochila con una *capacidad* asociada W, encontrar el subcojunto de ítems de S que maximice el beneficio total sin exceder la capacidad de la mochila. Es decir, encontrar $R \subseteq S$ tal que $\sum_{i \in R} p_i$ sea máxima y se cumpla $\sum_{i \in R} w_i \leq W$.

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 W el tamaño de la mochila):

- Algoritmo de fuerza bruta. Complejidad temporal perteneciente a $\mathcal{O}(n \times 2^n)$.
- Algoritmo 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 de Programación Dinámica. Complejidad temporal perteneciente a $\mathcal{O}(n \times W)$.

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 W, correspondientes a la cantidad de ítems disponibles y a la capacidad de la mochila, respectivamente. Luego le sucederán n líneas con dos enteros, w_i y p_i, correspondientes a los tamaños y a los beneficios de cada uno de los ítems.

La salida consistirá de un único número entero que representará el valor máximo de beneficio total alcanzable con una elección óptima de los ítems.

Entrada de ejemplo	**Salida esperada de ejemplo**

`5 25`
`10 5`
`15 4`
`5 13`
`10 8`
`5 8`
	`29`

Fechas de entrega

- *Formato Electrónico:* Domingo 8 de Septiembre de 2019, 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 9 de Septiembre de 2019, a las 18 hs.

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

Referencias

[1] Discrete-variable extremum problems. *Oper. Res.*, 5(2):266–288, April 1957.

[2] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems.* Springer, 2004.

[3] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Inc., New York, NY, USA, 1990.