INSTITUTO POLITÉCNICO NACIONAL
UNIDAD INTERDISCIPLINARIA DE INGENIERÍA CAMPUS ZACATECAS

Profesor Roberto Oswaldo Cruz Leija

Análisis de Algoritmos

Unidad 1

Mochila Dinámica

Leslie Arellano Covarrubias

3CM1

Fecha de Entrega

07/11/2019

Introducción

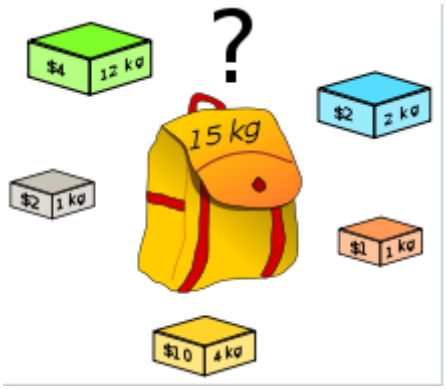
La siguiente actividad consistía en resolver el problema de la mochila basándonos en los problemas de decisión y tomando en cuenta la teoría de complejidad computacional, y las soluciones en tiempo polinómico.

Objetivo

Entender la solución de problemas NP-Completos mediante la programación dinámica.

Marco Teórico

En algoritmia, el problema de la mochila, comúnmente abreviado por KP (del inglés Knapsack problem) es un problema de optimización combinatoria, es decir, que busca la mejor solución entre un conjunto finito de posibles soluciones a un problema. Modela una situación análoga al llenar una mochila, incapaz de soportar más de un peso determinado, con todo o parte de un conjunto de objetos, cada uno con un peso y valor específicos. Los objetos colocados en la mochila deben maximizar el valor total sin exceder el peso máximo.



El problema de la mochila es uno de los 21 problemas NP-completos de Richard Karp, establecidos por el informático teórico en un famoso artículo de 1972.

Supongamos que tenemos n distintos tipos de ítems, que van del 1 al n. De cada tipo de ítem se tienen qi ítems disponibles, donde qi es un entero positivo que cumple $1 \leq qi \leq \infty$.

Cada tipo de ítem i tiene un beneficio asociado dado por v_i y un peso (o volumen) w_i. Usualmente se asume que el beneficio y el peso no son negativos. Para simplificar la

representación, se suele asumir que los ítems están listados en orden creciente según el peso (o volumen).

Por otro lado, se tiene una mochila, donde se pueden introducir los ítems, que soporta un peso máximo (o volumen máximo) W.

El problema consiste en meter en la mochila ítems de tal forma que se maximice el valor de los ítems que contiene y siempre que no se supere el peso (o volumen) máximo que puede soportar la misma. La solución al problema vendrá dada por la secuencia de variables x_1, x_2, ..., x_n donde el valor de x_i indica cuantas copias se meterán en la mochila del tipo de ítem i.

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{tal que} \quad & \sum_{i=1}^{n} w_i x_i \leq W \\ \text{y} \quad & 0 \leq x_i \leq q_i . \end{aligned}$$

Desarrollo de la Práctica

Prueba para mochila con peso 500, 1000 artículos, 100 de peso y 100 de beneficio.

```
MochilaDinamica mochila1 = new MochilaDinamica(Articulos.crearArticulos(1000, 100, 100) ,500);
mochila1.buscarSolucion();
```

```
Tamaño: 8 ********* Beneficio: 53.0
Tamaño: 13 ********* Beneficio: 90.0
Tamaño: 4 ********* Beneficio: 66.0
Tamaño: 6 ********* Beneficio: 52.0
Tamaño: 13 ********* Beneficio: 89.0
Tamaño: 11 ********* Beneficio: 83.0
Tamaño: 9 ********* Beneficio: 68.0
Tamaño: 4 ********* Beneficio: 47.0
Tamaño: 10 ********* Beneficio: 93.0
Tamaño: 3 ********* Beneficio: 36.0
Tamaño: 7 ********* Beneficio: 48.0
Tamaño: 2 ********* Beneficio: 63.0
Tamaño: 4 ********* Beneficio: 79.0
Tamaño: 4 ********* Beneficio: 76.0
Tamaño: 14 ********* Beneficio: 90.0
Tamaño: 2 ********* Beneficio: 87.0
Tamaño: 3 ********* Beneficio: 59.0
Tamaño: 12 ********* Beneficio: 86.0
Tamaño: 12 ********* Beneficio: 89.0
Tamaño: 1 ********* Beneficio: 88.0
Tamaño: 15 ********* Beneficio: 96.0
Tamaño: 8 ********* Beneficio: 95.0
Tamaño: 7 ********* Beneficio: 72.0
Tamaño: 12 ********* Beneficio: 88.0
Tamaño: 7 ********* Beneficio: 61.0
Tamaño: 11 ********* Beneficio: 94.0
Tamaño: 6 ********* Beneficio: 89.0
Tamaño: 2 ********* Beneficio: 18.0
Tamaño: 12 ********* Beneficio: 98.0
Tamaño: 1 ********* Beneficio: 53.0
Tamaño: 3 ********* Beneficio: 73.0
Tamaño: 12 ********* Beneficio: 100.0
Tamaño: 2 ********* Beneficio: 65.0
Tamaño: 2 ********* Beneficio: 56.0
```

```
Tamaño: 5 ********* Beneficio: 46.0
Tamaño: 8 ********* Beneficio: 74.0
Tamaño: 6 ********* Beneficio: 97.0
Tamaño: 7 ********* Beneficio: 74.0
Tamaño: 7 ********* Beneficio: 89.0
Tamaño: 4 ********* Beneficio: 36.0
Tamaño: 2 ********* Beneficio: 91.0
Tamaño: 9 ********* Beneficio: 76.0
Tamaño: 6 ********* Beneficio: 70.0
Tamaño: 5 ********* Beneficio: 75.0
Tamaño: 4 ********* Beneficio: 74.0
Tamaño: 7 ********* Beneficio: 66.0
Tamaño: 5 ********* Beneficio: 63.0
Tamaño: 9 ********* Beneficio: 70.0
Tamaño: 1 ********* Beneficio: 65.0
Tamaño: 4 ********* Beneficio: 87.0
Tamaño: 5 ********* Beneficio: 44.0
Tamaño: 7 ********* Beneficio: 78.0
Tamaño: 5 ********* Beneficio: 70.0
Tamaño: 13 ********* Beneficio: 85.0
Tamaño: 8 ********* Beneficio: 94.0
Tamaño: 10 ********* Beneficio: 70.0
Tamaño: 7 ********* Beneficio: 53.0
Tamaño: 12 ********* Beneficio: 100.0
Tamaño: 1 ********* Beneficio: 85.0
Tamaño: 12 ********* Beneficio: 96.0
Tamaño: 14 ********* Beneficio: 100.0
Tamaño: 5 ********* Beneficio: 52.0
Tamaño: 5 ********* Beneficio: 91.0
Tamaño: 6 ********* Beneficio: 99.0
Tamaño: 1 ********* Beneficio: 32.0
Tamaño: 6 ********* Beneficio: 62.0
Tamaño: 5 ********* Beneficio: 62.0
Tamaño: 7 ********* Beneficio: 66.0
Tamaño: 5 ********* Beneficio: 70.0
Tamaño: 12 ********* Beneficio: 86.0
Tamaño: 9 ********* Beneficio: 90.0
```

```
Tamaño: 6 ********* Beneficio: 59.0
Tamaño: 1 ********* Beneficio: 97.0
Tamaño: 2 ********* Beneficio: 55.0
Tamaño: 5 ********* Beneficio: 46.0
BUILD SUCCESSFUL (total time: 56 seconds)
```

Conclusión

La importancia de esta clase de problemas de decisión es que contiene muchos problemas de búsqueda y de optimización para los que se desea saber si existe una cierta solución o si existe una mejor solución que las conocidas.

Algunos algoritmos existentes pueden resolver el problema de la mochila en la práctica para casos de un gran tamaño. Sin embargo, la estructura única del problema, y el hecho de que se presente como un subproblema de otros problemas más generales, lo convierten en un problema frecuente en la investigación de operaciones.