Instituto Politécnico Nacional

Programa Académico: Ingeniería en Sistemas Computacionales
Unidad de Aprendizaje: Análisis de Algoritmos
Alumno: Nava Álvarez José Andrés
Fecha: 04/11/2019

Problema de la Mochila

Tabla de contenido

Introducción

En la presente actividad realizamos el análisis del Problema de la mochila en el cual pusimos a prueba lo aprendido en cuanto al análisis de algoritmos. Posteriormente llevamos a cabo la realización de un programa para resolver dicho problema, apoyados por supuesto en la programación dinámica.

Marco Teórico

Problema de la mochila

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.

Definición Formal;

Supongamos que tenemos$n$ distintos tipos de ítems, que van del 1 al n. De cada tipo de ítem se tienenq_i ítems disponibles, dondeq_1 es un entero positivo que cumple $1 < q_1 < \infty$.

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 dado por la secuencia de variables $x_1, x_2, \ldots, x_n$ donde el valor de x_i indica cuantas copias se meterán en la mochila del tipo de ítem i.

El problema se puede expresar matemáticamente por medio del siguiente programa lineal:

$$\text{Maximizar } \sum_{i=1}^{n} v_i xi$$

$$\text{Tal que} \quad \sum_{i=1}^{n} w_i x_i \leq W$$

$$\text{y} \qquad 0 \leq x_i \leq q_i$$

Si$q_1 = 1$ para$i = 1,2, \ldots, n$ se dice que se trata del problema de la mochila 0-1. Si uno o más $q_1$es infinito entonces se dice que se trata del problema de la mochila no acotado también llamado a veces problema de la mochila entera. En otro caso se dice que se trata del problema de la mochila acotado

Programacion Dinamica

En informática, la programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas.

Una subestructura óptima significa que se pueden usar soluciones óptimas de subproblemas para encontrar la solución óptima del problema en su conjunto. Por ejemplo, el camino más corto entre dos vértices de un Grafo se puede encontrar calculando primero el camino más corto al objetivo desde todos los vértices adyacentes al de partida, y después

usando estas soluciones para elegir el mejor camino de todos ellos. En general, se pueden resolver problemas con subestructuras óptimas siguiendo estos tres pasos:

- Dividir el problema en subproblemas más pequeños.
- Resolver estos problemas de manera óptima usando este proceso de tres pasos recursivamente.
- Usar estas soluciones óptimas para construir una solución óptima al problema original.

Los subproblemas se resuelven a su vez dividiéndolos en subproblemas más pequeños hasta que se alcance el caso fácil, donde la solución al problema es trivial.

Desarrollo

Comenzaremos nuestro análisis, teniendo una lista de objetos con los que vamos a llenar la mochila. De los cuales tendremos su peso y su beneficio.

Para obtener el beneficio maximizado nos valdremos del método dinámico, de este modo utilizaremos una matriz de beneficios en la cual sacaremos en orden el beneficio máximo con cada peso de la mochila. Esto empezando en 1 y aumentando de uno en uno con cada fila de la matriz.

Comenzaremos creando una matriz con número de filas igual al número de ítems más uno y número de columnas igual al peso máximo de la mochila más 1. Rellenaremos la primera fila y columna con 0.

La matriz será entonces completada con los mejores beneficios posibles. Para esto utilizaremos el modelo dinámico. Comenzaremos llenando la matriz por filas (cada fila representa un articulo). En cada fila anotaremos el mejor beneficio posible teniendo en cuenta que en cada columna aumentamos por uno el peso hasta llegar al total de la mochila.

En el análisis tomaremos en cuenta la fila anterior, ya que restándole el peso del artículo que se está analizando al peso de la mochila, podremos ver si es posible sumar resultados anteriores con peso restante para así obtener el mejor beneficio.

Ejemplo con 4 artículos y una mochila de peso 4;

Artículos \ Peso	0	1	2	3	4
0	0	0	0	0	0
1	0	13	13	13	13
2	0	13	13	13	17
3	0	24	37	37	37
4	0	24	37	44	57

Matriz de beneficios

Id	Valor	Peso
0	13	1
1	4	3
2	24	1
3	20	2

Artículos Utilizados

Una vez calculada la matriz, escribiremos la solución a manera de una lista de los artículos escogidos con su peso y su valor. En seguido pondremos también el peso utilizado y el beneficio total.

Resultados

Prueba de una mochila con capacidad de 4 y 4 artículos con pesos de 1 a 3 y valores de 1 a 25;

```
Matriz de Beneficios
0, 0, 0, 0, 0,
0, 0, 12, 12, 12,
0, 17, 17, 29, 29,
0, 17, 18, 29, 30,
0, 17, 25, 42, 43,
Solucion:
2-25.0
1-1.0
1-17.0
Peso total: 4 de 4
Beneficio total: 43
```

Prueba de una mochila con capacidad de 8 y 10 artículos con pesos de 1 a 4 y valores de 1 a 25;

```
Matriz de Beneficios
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 21, 21, 21, 21, 21, 21, 21, 21,
0, 21, 21, 21, 21, 29, 29, 29, 29,
0, 23, 44, 44, 44, 44, 52, 52, 52,
0, 23, 44, 56, 56, 56, 56, 64, 64,
0, 23, 44, 56, 56, 56, 56, 67, 67,
0, 23, 44, 57, 69, 69, 69, 69, 80,
0, 23, 44, 57, 69, 69, 78, 78, 80,
0, 23, 44, 57, 69, 69, 78, 78, 80,
0, 23, 44, 57, 69, 69, 80, 80, 89,
0, 23, 44, 57, 69, 69, 80, 86, 89,
```

```
Solucion:
2-11.0
2-9.0
1-13.0
1-12.0
1-23.0
1-21.0
Peso total: 8 de 8
Beneficio total: 89
```

Prueba de una mochila con capacidad de 20 y 5 artículos con pesos de 1 a 3 y valores de 1 a 10;

```
Matriz de Beneficios
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
0, 0, 0, 9, 9, 9, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
0, 7, 7, 9, 16, 16, 16, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
0, 10, 17, 17, 19, 26, 26, 26, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31,
Solucion:
1-10.0
1-7.0
3-9.0
3-5.0
Peso total: 8 de 20
Beneficio total: 31
```

Prueba de una mochila con capacidad de 10 y 10 artículos con pesos de 1 a 200 y valores de 1 a 5;

```
Matriz de Beneficios
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
Solucion:
Peso total: 0 de 10
Beneficio total: 0
```

*En este caso ningún elemento entro a la mochila ya que no tenían el peso necesario.

Prueba de una mochila con capacidad de 5 y 5 artículos con pesos de 1 y valores de 1;

```
Matriz de Beneficios
0, 0, 0, 0, 0, 0,
0, 1, 1, 1, 1, 1,
0, 1, 2, 2, 2, 2,
0, 1, 2, 3, 3, 3,
0, 1, 2, 3, 4, 4,
0, 1, 2, 3, 4, 5,
Solucion:
1-1.0
1-1.0
1-1.0
1-1.0
1-1.0
Peso total: 5 de 5
Beneficio total: 5
```

Conclusiones

En esta actividad es fácil percibir la utilidad y el beneficio de la programación dinámica. Ya que sin ella este problema exigiría demasiado poder de computación para resolver la situaciones en los que la mochila tenga demasiada capacidad y/o sean demasiados artículos.

Con esta metodología podemos pasar de un problema de permutaciones a simplemente completar una matriz de m x n.