# Reporte: Algoritmo Mochila (Dinámico)

Osvaldo de Luna

5 de noviembre de 2019

# Problema Mochila

Hay $x$ número de artículos que se pueden llevar en una mochila, cada uno de ellos provee un peso (o costo) y un beneficio. Lo que se desea es llevar aquella combinación de artículos que nos provean el mayor beneficio posible estando debajo, o igual, del peso máximo que puede soportar la mochila.

## Solución propuesta

Para obtener una solución dinámica se ideó la creación de una 'Matriz de Beneficios', la cual contendrá el beneficio máximo obtenible de una menor cantidad de peso máximo de la mochila y menor cantidad de artículos de la misma lista de artículos. En otras palabras, el beneficio máximo posible con el primer artículo, con los dos primeros, con los tres primeros, y así sucesivamente hasta llegar a los $x$ número de artículos; tomando un peso máximo de mochila cero, uno, dos y así hasta llegar al solicitado en el problema. De esta manera se pretende conocer el beneficio máximo posible de $n+1$ artículos cuando ya conocemos ese valor para $n$ artículos, y de manera análoga con un peso $p+1$, cuando sabemos el comportamiento con el peso $p$. para de esta manera ya no recalcular mejores combinaciones y sólo recalcular acorde al nuevo ítem o artículo hasta poder llegar a los $x$ número de artículos.

Una vez teniendo la mencionada 'Matriz de Beneficios', se creó un método capaz de recuperar los ítems o artículos utilizados para la casilla que nos importa (la de la última fila y última columna), ya que la matriz solo guarda un valor entero, el cual representa el beneficio máximo posible para los $a$ artículos seleccionados de la lista y un valor $p$ de peso máximo de la mochila. Ejemplo:

Se tiene la siguiente lista de artículos.

| Ítem | Peso | Valor |
|---|---|---|
| 0 | 3 | 34 |
| 1 | 6 | 28 |
| 2 | 6 | 90 |
| 3 | 1 | 23 |
| 4 | 9 | 11 |
| 5 | 1 | 19 |
| 6 | 11 | 700 |

Cuadro 1: Lista de artículos

A partir de esta lista, se crea la matriz de beneficios. Se construye fila por fila, empezando de arriba hacia abajo y de izquierda a derecha, para llenar con un artículo los beneficios que pueden haber respecto a distintos pesos en la mochila y respecto al beneficio anterior (sin ese artículo).

| Ítem / Peso | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 34 | 34 | 34 | 34 | 34 | 34 |
| 2 | 0 | 0 | 0 | 34 | 34 | 34 | 34 | 34 | 34 |
| 3 | 0 | 0 | 0 | 34 | 34 | 34 | 90 | 90 | 90 |
| 4 | 0 | 23 | 23 | 34 | 57 | 57 | 90 | 113 | 113 |
| 5 | 0 | 23 | 23 | 34 | 57 | 57 | 90 | 113 | 113 |
| 6 | 0 | 23 | 42 | 42 | 57 | 76 | 90 | 113 | 132 |
| 7 | 0 | 23 | 42 | 42 | 57 | 76 | 90 | 113 | 132 |

Cuadro 2: Matriz de Beneficios

Así ya conocemos el beneficio máximo posible, el cual es 132. Finalmente obtenemos los ítems solución, estos se obtienen comparando a partir de la casilla resultado en la matriz de beneficios (última fila y última columna), comparamos si el valor en esa posición es diferente al que tiene justo encima (en donde se agregó ese artículo), si no es así se pasa a la fila de arriba con misma posición de columna, pero si si difieren entonces se agrega ese artículo que fue agregado en la matriz de beneficios y ahora compararemos una fila arriba (porque pasamos a donde ese artículo no está) y la posición de columna será en la que nos encontrábamos − el peso del artículo agregado a los

ítem solución.

| Ítem | Peso | Valor |
|---|---|---|
| 5 | 1 | 19 |
| 3 | 1 | 23 |
| 2 | 6 | 90 |

Cuadro 3: Lista de artículos solución

# Análisis

Para el ejemplo anterior, tenemos

```
//PROBLEMA DE LA MOCHILA
    Mochila m = new Mochila(8);
    ArrayList<Item> items = new ArrayList<>();
    //items = Herramientas.generarArticulos(10, 25, 100);
    items.add(new Item(3,34));
    items.add(new Item(6,28));
    items.add(new Item(6,90));
    items.add(new Item(1,23));
    items.add(new Item(9,11));
    items.add(new Item(1,19));
    items.add(new Item(11,700));
    m.buscarSolucion(items);
    System.out.println(m);
```

Figura 1: Código del ejemplo

```
compile-single:
run-single:
----------Mochila----------
Capacidad: 8
Items: [Costo: 3        Beneficio: 34
, Costo: 6      Beneficio: 28
, Costo: 6      Beneficio: 90
, Costo: 1      Beneficio: 23
, Costo: 9      Beneficio: 11
, Costo: 1      Beneficio: 19
, Costo: 11     Beneficio: 700
]
Items en mochila: [Costo: 1     Beneficio: 19
, Costo: 1      Beneficio: 23
, Costo: 6      Beneficio: 90
]
Beneficio maximo obtenido: 132
Capacidad de la mochila utilizada: 8
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figura 2: Compilación del ejemplo

Para una prueba generando 1000 ítems aleatorios con un costo máximo de 10 y beneficio máximo de 1000, y una mochila con costo máximo de 10

```
Items en mochila: [Costo: 1     Beneficio: 9990
, Costo: 1      Beneficio: 9967
, Costo: 1      Beneficio: 9929
, Costo: 1      Beneficio: 9949
, Costo: 1      Beneficio: 9945
, Costo: 1      Beneficio: 9993
, Costo: 1      Beneficio: 9986
, Costo: 1      Beneficio: 9964
, Costo: 1      Beneficio: 9950
, Costo: 1      Beneficio: 9910
]
Beneficio maximo obtenido: 99583
Capacidad de la mochila utilizada: 10
BUILD SUCCESSFUL (total time: 3 seconds)
```

Figura 3: Prueba 1, información completa

Para ésta prueba se puede observar que se toma en cuenta aquellos ítems con mayor beneficio y que tengan un costo mínimo de 1, ya que la mochila solo soporta un costo de 10 y no puede permitirse el aceptar un artículo de costo mayor, sobre todo si hay tantas posibilidades que genera artículos con buen beneficio y costo de 1. Para este caso se utilizó la máxima capacidad de la mochila.

Para una prueba generando 10000 ítems aleatorios con un costo máximo de 20 y beneficio máximo de 1000, y una mochila con costo máximo de 1000.

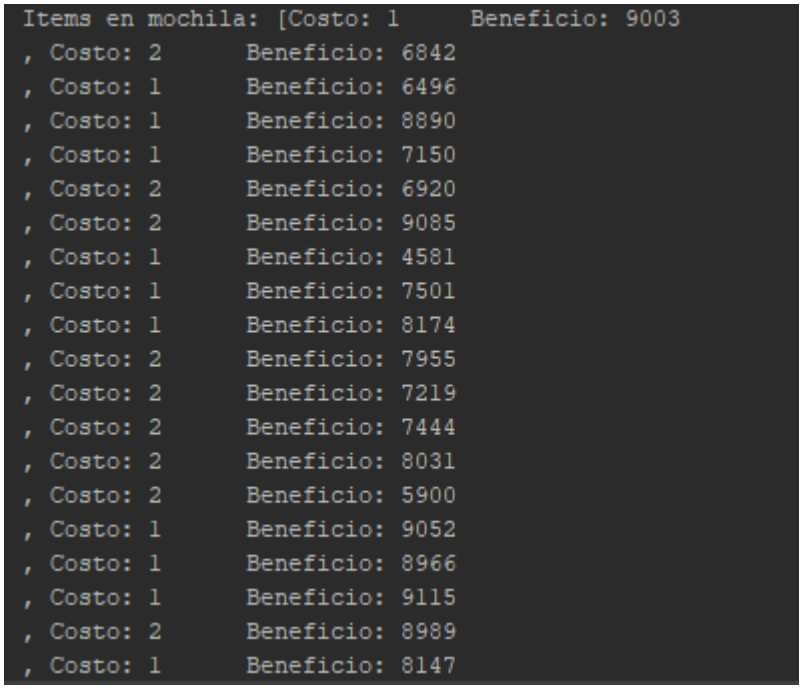


Figura 4: Prueba 2, artículos solución

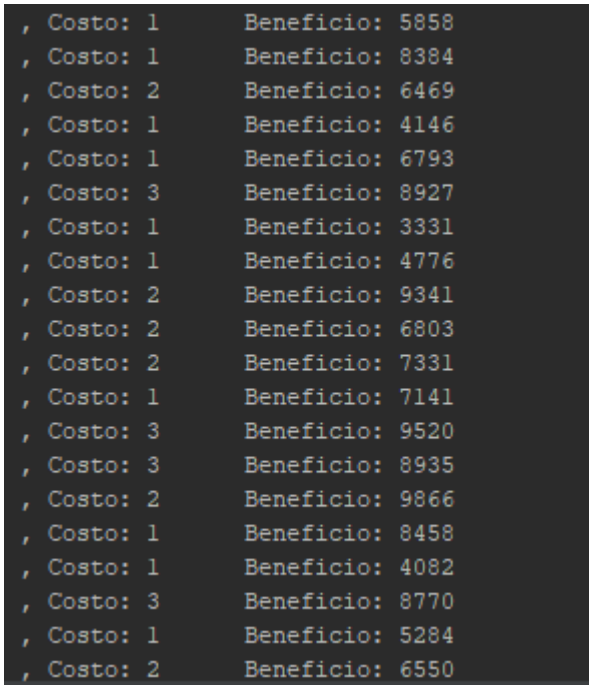


Figura 5: Prueba 2, artículos solución

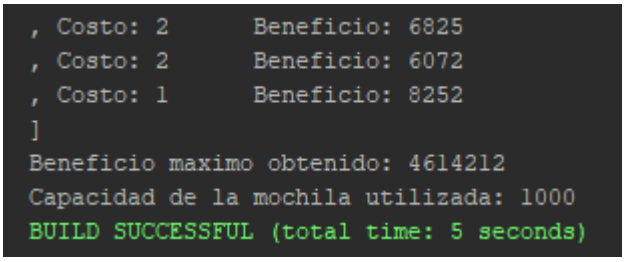


Figura 6: Prueba 2, información del beneficio máximo y peso utilizado

De manera similar a la prueba anterior, se buscan aquellos artículos con el menor costo posible, como ahora hay un costo máximo de mochila de 1000, el algoritmo puede darse el lujo de agregar ítems con costo igual o inferior a 3 (costo mayor identificado en los artículos dentro de la mochila), pero de igual manera hay tantas posibilidades que se generan artículos con costo de 1 y beneficio alto. Para este caso se utilizó la máxima capacidad de la mochila.

Para una prueba generando 10000 ítems aleatorios con un costo máximo

de 1000 y beneficio máximo de 10, y una mochila con costo máximo de 2.

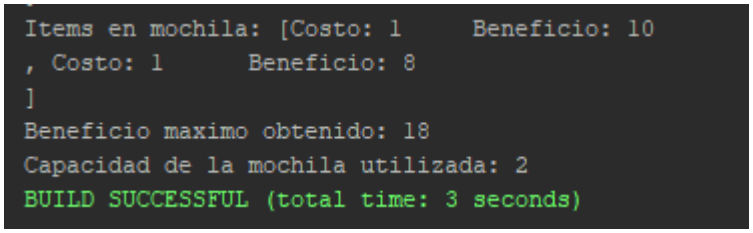


Figura 7: Prueba 3, información completa

Para el último caso se puede observar que hay demasiadas posibilidades de ítems para el peso máximo que la mochila tiene, y para mayor desventaja los ítems pueden llegar a tener un costo máximo de 1000, el beneficio es bajo para poder observar si se generaron los mejores ítems posibles, sin embargo no fue así, ya que habrían dos ítems con costo 1 y beneficio 10 (es decir, el costo mínimo y beneficio máximo del ítem). Lo que si se puede destacar es que se generó un ítem 'ideal' (con las características descritas). Para este caso se utilizó la máxima capacidad de la mochila.

Podemos ver que el algoritmo de verdad es eficaz y eficiente por lo dinámico y porque nunca se convierte en un problema de complejidad muy elevado, debido a que hace simples comparaciones y que cada que avanza en la construcción de la matriz de beneficios ya conlleva resultados anteriores óptimos.