"Análisis de Algoritmos"
José Helo Guzmán

Contenido


## ▬▬▬ 1. Introducción.

En este caso analizaremos el problema del “knapsack”, conocido como el problema de la mochila.

Existen muchas versiones de este problema, por lo que siempre es importante analizar las condiciones iniciales para determinar cuál problema es el que se ha propuesto.

Por tratarse de un problema complejo, se construirá un algoritmo ambicioso, en inglés “Greedy Algorithm”, que sea capaz de resolver este problema.

## ▬▬▬▬ 2. The Knapsack Problem.

Supongamos que usted es un ladrón ambicioso. Usted está dentro de un supermercado donde puede encontrar varios artículos de diferente valor y peso.

Existe solamente una unidad de cada artículo, por lo que puede tomar únicamente un artículo de cada cosa. Además cuenta con una mochila que a lo sumo puede cargar 10 kilos de peso. Veamos la lista de artículos.

| Artículo | Ganancia | Peso |
|---|---|---|
| apio | $10 | 1 kilos |
| papas | $40 | 2 kilos |
| bananos | $20 | 1 kilos |
| pan | $15 | 3 kilos |
| queso | $60 | 5 kilos |

Vamos a representar los artículos de la siguiente manera:

```
objetos:
[
[apio,    [10, 1] ],
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[queso,   [60, 5] ]
];
```

## ▬▬▬▬ 3. Solución Exhaustiva.

Si resolvemos el problema de manera exhaustiva, vamos a tener que revisar la siguiente cantidad de subconjuntos:

conjunto vacío: ⌀

conjuntos de tamaño 1: c(5,1) = 5

conjuntos de tamaño 2: c(5,2) = 10

conjuntos de tamaño 3: c(5,3) = 10

conjuntos de tamaño 4: c(5,4) = 5

conjuntos de tamaño 5: c(5,5) = 1

En total debemos revisar 32 posibles combinaciones.

En general para 5 objetos se tienen 2^5=32 posibildades

Nota:

```
(%i) powerset({apio,papas,bananos,pan,queso});

{

{},

{apio   },
{papas  },
{bananos},
{pan    },
{queso  },

{apio,   papas  },
{apio,   bananos},
{apio,   pan    },
{apio,   queso  },
{papas,  bananos},
{papas,  pan    },
{papas,  queso  },
{bananos,pan    },
{bananos,queso  },
{pan,    queso  },

{apio,   papas,  bananos},
{apio,   papas,  pan    },
{apio,   papas,  queso  },
{apio,   bananos,pan    },
{apio,   bananos,queso  },
{apio,   pan,    queso  },
{papas,  bananos,pan    },
{papas,  bananos,queso  },
{papas,  pan,    queso  },
{bananos,pan,    queso  },

{apio, papas,  bananos,pan  },
{apio, papas,  bananos,queso},
{apio, papas,  pan,    queso},
{apio, bananos,pan,    queso},
{papas,bananos,pan,    queso},

{apio,bananos,pan,papas,queso},

}



(%i) cardinality(powerset({apio,papas,bananos,pan,queso}))
(%o) 32



(%i)  length(powerset({apio,papas,bananos,pan,queso}));
(%o)  32
```

Si se resuelve el problema de manera exhaustiva, revisando cada una de las rutas anteriores se obtiene los siguientes valores para cada grupo.

```
[

-----------------

{}
-----------------

{[apio,[10,1]]
}
-----------------▶ Total:[10,1]


{[papas,[40,2]]
}
-----------------▶ Total: [40,2]


{[bananos,[20,1]]
}
-----------------▶ Total: [20,1]


{[pan, [15,3]]
}
-----------------▶ Total:[15,3]


{[queso,[60,5]]
}
-----------------▶ Total: [60,5]

{[apio,   [10,1]],
 [papas,  [40,2]]
}
-----------------▶ Total: [50,3]


{[apio,   [10,1]],
 [bananos,[20,1]]
}
-----------------▶ Total: [30,2]


{[apio,   [10,1]],
 [pan,    [15,3]]
}
-----------------▶ Total: [25,4]


{[apio,   [10,1]],
 [queso,  [60,5]]
}
-----------------▶ Total: [70,6]


{[papas,  [40,2]],
 [bananos,[20,1]]
}
-----------------▶ Total: [60,3]


{[papas, [40,2]],
 [pan,   [15,3]]
}
-----------------▶ Total: [55,5]
```

```
{[papas, [40,2]],
 [queso, [60,5]]
}
-----------------▶ Total: [100,7]


{[bananos,[20,1]],
 [pan,    [15,3]]
}
-----------------▶ Total: [35,4]


{[bananos,[20,1]],
 [queso,  [60,5]]
}
-----------------▶ Total: [80,6]


{[pan,    [15,3]],
 [queso,  [60,5]]
}
-----------------▶ Total: [75,8]


{[apio,   [10,1]],
 [papas,  [40,2]],
 [bananos,[20,1]]
}
-----------------▶ Total: [70,4]


{[apio,   [10,1]],
 [papas,  [40,2]],
 [pan,    [15,3]]
}
-----------------▶ Total: [65,6]


{[apio,   [10,1]],
 [papas,  [40,2]],
 [queso,  [60,5]]
}
-----------------▶ Total: [110,8]


{[apio,   [10,1]],
 [bananos,[20,1]],
 [pan,    [15,3]]
}
-----------------▶ Total: [45,5]


{[apio,   [10,1]],
 [bananos,[20,1]],
 [queso,  [60,5]]
}
-----------------▶ Total: [90,7]


{[apio,   [10,1]],
 [pan,    [15,3]],
 [queso,  [60,5]]
}
-----------------▶ Total: [85,9]


{[papas,  [40,2]],
```

```
 [bananos,[20,1]],
 [pan,    [15,3]]
}
-----------------▶ Total: [75,6]


{[papas,  [40,2]],
 [bananos,[20,1]],
 [queso,  [60,5]]
}
-----------------▶ Total: [120,8]


{[papas,  [40,2]],
 [pan,    [15,3]],
 [queso,  [60,5]]
}
-----------------▶ Total: [115,10]


{[bananos,[20,1]],
 [pan,    [15,3]],
 [queso,  [60,5]]
}
-----------------▶ Total: [95,9]

{[apio,   [10,1]],
 [papas,  [40,2]],
 [bananos,[20,1]],
 [pan,    [15,3]]
}
-----------------▶ Total: [85,7]


{[apio,   [10,1]],
 [papas,  [40,2]],
 [bananos,[20,1]],
 [queso,  [60,5]]
}
-----------------▶ Total: [130,9]


{[apio,   [10,1]],
 [papas,  [40,2]],
 [pan,    [15,3]],
 [queso,  [60,5]]
}
-----------------▶ Total: [125,11]
-----------------▶ No cabe en la mochila.


{[apio,   [10,1]],
 [bananos,[20,1]],
 [pan,    [15,3]],
 [queso,  [60,5]]
}
-----------------▶Total: [105,10]


{[papas,  [40,2]],
 [bananos,[20,1]],
 [pan,    [15,3]],
 [queso,  [60,5]
},
-----------------▶ Total: [135,11]
-----------------▶ No cabe en la mochila.
```

```
-----------------

{[apio,   [10,1]],
 [bananos,[20,1]]
 [pan,    [15,3]],
 [papas,  [40,2]],
 [queso,  [60,5]]
},
-----------------▶ Total: [145,12]
-----------------▶ No cabe en la mochila.

]
```

```
>>> Se escoge la mejor solución.
```

Y finalmente se escoge la solución, cuyo peso cabe en la mochila y con el mayor valor:

```
[ [apio,    [10,1]],
  [papas,   [40,2]],
  [bananos, [20,1]],
  [queso,   [60,4]]
]

Total: [130,9]
```

Se obtiene una ruta de $130 y 9 kilos de peso.

## ▬▬▬ 4. Análisis de Complejidad.

Tal como se ha mostrado, para una mochila con 5 artículos de tienen $2^5$ posibles grupos para colocar en la mochila.

En general:

5 artículos se tienen $2^5 = 32$ grupos.

6 artículos se tienen $2^6 = 64$ grupos.

7 artículos se tienen $2^7 = 128$ grupos.

En general para:

“n” artículos se tienen $2^n$ grupos.

Por lo tanto el problema será:

$O(2^n)$

## ▬▬▬ 5. Primera Implementación “Greedy”.

La primera aproximación ambiciosa para resolver este problema sería la siguiente:

• Tome el artículo de mayor valor que cabe en la mochila.

• Repita este proceso hasta que no quepan más artículos en la mochila.

Recordemos los artículos que tenemos:

```
objetos:
[
[apio,    [10, 1] ],
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[queso,   [60, 5] ]
];
```

Veamos cómo funciona este procedimiento para una mochila de tamaño 10:

>>> Iniciamos con la mochila vacía:

```
objetos:
[
[apio,    [10, 1] ],
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[queso,   [60, 5] ]
];


[ [],
  [0,0]
]
```

>>> Primera Iteración:

```
objetos:
[
[apio,    [10, 1] ],
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[queso,   [60, 5] ]  ◄
];


[
  [queso,[60,5]],
  [60,5]
]
```

```
>>> Segunda Iteración:

objetos:
[
[apio,    [10, 1] ],
[papas,   [40, 2] ], ◄
[bananos, [20, 1] ],
[pan,     [15, 3] ],
];


[
  [queso,[60,5]],
  [papas,[40,2]],
  [100,7]
]




>>> Tercera Iteración:

objetos:
[
[apio,    [10, 1] ],
[bananos, [20, 1] ], ◄
[pan,     [15, 3] ],
];


[
  [queso,  [60,5]],
  [papas,  [40,2]],
  [bananos,[20,1]],
  [120,8]
]




>>> Cuarta Iteración:

objetos:
[
[apio,    [10, 1] ],
[pan,     [15, 3] ], ◄ No podemos agregar el pan!
];


[ [queso,  [60,5]],
  [papas,  [40,2]],
  [bananos,[20,1]],
  [120,8]
]



>>> Quinta Iteración:

objetos:
[
[apio,    [10, 1] ], ◄
];
```

```
[ [queso,  [60,5]],
  [papas,  [40,2]],
  [bananos,[20,1]],
  [apio,   [10,1]]
  [130,9]
]
```

Podemos ver como con 4 iteraciones fue posible construir la mochila con un peso de 9 kilos y una ganancia de $130.

## ▬▬▬ 6. Análisis de Complejidad.

En la implementación anterior, al no estar ordenados los objetos, tenemos que recorrer la lista en busca del máximo. El peor caso se presenta cuando todos los artículos caben en la mochila. Por lo que el tiempo de ejecución estará dado por:

$$n + (n-1) + (n-2) + \ldots + 2 + 1 = \left( \frac{n*(n+1)}{2} \right)$$

$$1 + 2 + \ldots + (n-2) + (n-1) + n = \left( \frac{n*(n+1)}{2} \right)$$

Que sabemos es equivalente a:

$O(n^2)$

## ▬▬▬ 7. Una Segunda Implementación.

Podemos hacer otra implementación. Pero en este caso vamos a ordenar la lista de acuerdo a la ganancia de cada artículo.

Dada la lista inicial:

```
objetos:
[
[apio,    [10, 1] ],
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[queso,   [60, 5] ]
];
```

La ordenamos de mayor a menor, de acuerdo a su ganancia.

```
objetos:
[
[queso,   [60, 5] ],
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[apio,    [10, 1] ]
];
```

Y hacemos un recorrido completo de la lista para ver cuáles objetos caben en la mochila.

>>> Iniciamos con el primer elemento:

```
objetos:
[
[queso,   [60, 5] ], ◄
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[apio,    [10, 1] ]
];

mochila:
[
  [queso,[60,5]],
  [60,5]
]
```

>>> Continuamos con el siguiente elemento:

```
objetos:
[
[papas,   [40, 2] ], ◄
[bananos, [20, 1] ],
[pan,     [15, 3] ],
[apio,    [10, 1] ]
];

mochila:
[
  [queso,[60,5]],
  [papas,[40,2]],
  [100,7]
]
```

>>> Continuamos con el siguiente elemento:

```
objetos:
[
[bananos, [20, 1] ], ◄
[pan,     [15, 3] ],
[apio,    [10, 1] ]
```

```
];


mochila:
[
  [queso,  [60,5]],
  [papas,  [40,2]],
  [bananos,[20,1]],
  [120,8]
];



>>> Continuamos con el siguiente elemento:


objetos:
[
[pan, [15, 3]], ◄ El pan no cabe, lo eliminamos!!
[apio,[10, 1]]
];


mochila:
[
  [queso,  [60,5]],
  [papas,  [40,2]],
  [bananos,[20,1]],
  [120,8]
];



>>> Al revisar el último elemento vemos que no cabe en la mochila

objetos:
[
[apio,[10, 1]] ◄
];


mochila:
[
  [queso,  [60,5]],
  [papas,  [40,2]],
  [bananos,[20,1]],
  [apio,   [10,1]],
  [130,9]
]



>>> Por lo tanto la solución final está dada por:

objetos:
[];


mochila:
[
  [queso,  [60,5]],
  [papas,  [40,2]],
  [bananos,[20,1]],
  [apio,   [10,1]],
  [130,9]
]
```

## ▬▬▬ 8. Análisis de Complejidad.

En esta segunda implementación, se deben ordenar los artículos y luego iniciar el proceso de construcción de la mochila.

Una vez ordenada la lista, en el peor caso, se tendrá que hacer un recorrido de toda la lista. Por lo tanto el la complejidad del algoritmo estará dada por:

$O(ordernar) + O(n)$

$O(n * \log_2(n)) + O(n)$

$= O(n * \log_2(n))$

## ▬▬▬ 9. Los Algoritmos Greedy son Aproximados.

Los algoritmos “greedy” dan soluciones aproximadas, esto significa que no siempre dan la solución correcta.

Por ejemplo, si cambiamos los valores de los artículos, la mochila, obtenemos el siguiente problema.

```
Mochila de 5 kilos.
objetos:
[
[apio,    [10, 1] ],
[papas,   [40, 2] ],
[bananos, [20, 1] ],
[pan,     [30, 3] ],
[queso,   [60, 5] ]
];
```

La ordenamos:

```
Mochila de 5 kilos.
objetos:
[
[queso,   [60, 5] ],
[papas,   [40, 2] ],
[pan,     [30, 3] ],
[bananos, [20, 1] ],
[apio,    [10, 1] ]
];
```

Y el algoritmo greedy nos da la solución de:

```
[ [queso,[60,5]],
  [60,5]
]
```

Con un total de $60 y 5 kilos.

El valor correcto debe ser de 70, y hay varias soluciones:

```
[ [papas, [40,2],
  [pan,   [30,3],
  [70,5]
]
```

Con un total de $70 y 5 kilos.

## ▬▬▬ 10. Resumen.

• El “knapsack” programado de forma exhaustiva tiene un tiempo de ejecución de $O(2^n)$.

• El “knapsack” programado de forma “greedy”, sin ordenar la lista tiene un tiempo de ejecución de $O(n^2)$.

• El “knapsack” programado de forma “greedy”, ordenando la lista de artículos tiene un tiempo de ejecución de $O(n * \log_2(n))$.

• Los algoritmos “greedy” no siempre dan la respuesta exacta, en muchas ocasiones dan una aproximación.

• Pueden utilizarse otros criterios para construir el algoritmo greedy. Un mejor algoritmo “greedy” es el que saca la ganancia promedio por kilo. Es decir si las papas dan una ganancia de $40 y tienen un peso de 2 kilos. Cada kilo de papa da una ganancia de $40/2 = $20 por kilo.

## ▬▬▬ 11. Ejercicios.

### ●● Ejercicio 1.

Supongamos que usted trabaja para una empresa que envía camiones de carga por todo el país. Se necesita cargar los camiones con cajas. Todas la cajas tienen tamaños diferentes. Por supuesto se trata de usar el menor número de camiones para el envío.

Cómo escogería las cajas para minimizar el número de camiones utilizados?

Utilizaría una estrategia greedy o exhaustiva?

●● Ejercicio 2.

Investigue sobre el problema del fractional knapsack.
En que se diferencia del knapsack que presentamos aquí.

●● Ejercicio 3.

Investigue sobre el problema del bounded knapsack.
En que se diferencia del knapsack que presentamos aquí.
Se puede modificar el algoritmo presentado para solucionar este problema.

●● Ejercicio 4.

Investigue sobre el problema del unbounded knapsack.
En que se diferencia del knapsack que presentamos aquí.