## 0/1 Knapsack Problem

In a knapsack problem , we are given a set of n items where each item i is specified by as size $w_i$ and a value $V_i$ . We are also given a size bound W, the size of our knapsack.

| Item # | Size(w) | Value(V) |
|---|---|---|
| 1 | 1 | 8 |
| 2 | 3 | 6 |
| 3 | 5 | 5 |

There are two versions of the problem:

(0-1) Knapsack Problem : Items are indivisible: you either take an item or not. Solved with *dynamic programming*

Fractional Knapsack Problem: Items are divisible: you can take any fraction of an item. Solved with a *greedy algorithm*

## Recurrence relation

$$V[i, j] = \begin{cases} (\max(V[i-1, j-wi] + vi, V[i-1, j]) & \text{if } j <= wi \\ V[i-1, j] & \text{if } j < wi \end{cases}$$

Where V [i, j] to be the best value that can be achieved for the instance with only the first i items and capacity j. The final answer will be V [n,W].

**Base cases**
No capacity available: V [i, 0] = 0
No items available: V [0, j] = 0

With the 0-1 Knapsack, you need to know which parts you should do to get the best total value possible. We want maximizing our chance to get more value.

## Algorithm

```
KNAPSACK-DP(w1, . . . ,wn, v1, . . . , vn,W)
for i <- 0, . . . , n do
        for j <-  0, . . . ,W do
                if i = 0 or j = 0 then
                        V [i, j]  = 0
                else
                        if j < wi then
                                V [i, j]  = V [i − 1, j]
                        else
                                V [i, j] =  max(V [i − 1, j], V [i − 1, j − wi] + vi)
                        end if
                end if
        end for
end for
return V [n,W]
```

# Knapsack DP example

**Input:**

$W = 5$

| | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **w** | **2** | **1** | **3** | **2** |
| **v** | **12** | **10** | **20** | **15** |

```
if i = 0 or j = 0 then
        V [i, j]  = 0
```

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | | | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[1,1]**

**if** j < wi **then**

V [i, j] = V [i − 1, j]

**1<2 then**

**V[1,1]** = V [1 − 1, 1] =V[0,1] =0

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | **0** | | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[1,2]**

**if** j >=wi **then**

V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)

**2**>=**2 then**

**V[1,2]** = max{V [1 − 1, 2] , V[1-1, 2-2] +12}

= max{0 , 0 +12}=12

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | **12** | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[1,3]**

**if** j >=wi **then**

V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)

**3**>=**2 then**

**V[1,3]** = max{V [1 − 1, 3] , V[1-1, 3-2] +12}

= max{0 , 0 +12}=12

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | **12** | | |
| 2 | 0 | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[1,4]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**4**>=**2 then**

**V[1,4]** = max{V [1 − 1, 4] , V[1-1, 4-2] +12}

= max{0 , 0 +12}=12

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | **12** | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[1,5]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**5**>=**2 then**

**V[1,5]** = max{V [1 − 1, 5] , V[1-1, 5-2] +12}

= max{0 , 0 +12}=12

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | **12** | **12** |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[2,1]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**1**>=**1 then**

**V[2,1]** = max{V [2 − 1, 1] , V[2-1, 1-1] +10}

= max{0 , 0 +10}=10

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | **10** | | | | |
| 3 | 0 | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 0 | | | | | |

**For V[2,2]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**2**>=**1 then**

**V[2,2]** = max{V [2 − 1, 2] , V[2-1, 2-1] +10}

= max{12 , 0 +10}=12

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | **12** | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[2,3]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**3**>=**1 then**

**V[2,3]** = max{V [2 − 1, 3] , V[2-1, 3-1] +10}

= max{12 , 12 +10}=22

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | **22** | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[2,4]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**4**>=**1 then**

**V[2,4]** = max{V [2 − 1, 4] , V[2-1, 4-1] +10}

= max{12 , 12 +10}=22

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | **22** | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[2,5]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**5**>=**1 then**

**V[2,5]** = max{V [2 − 1, 5] , V[2-1, 5-1] +10}

= max{12 , 12 +10}=22

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | **22** |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

**For V[3,1]**

**if** j < wi **then**
V [i, j] = V [i − 1, j]
**1<3 then**

**V[3,1]** = V [3 − 1, 1] =V[2,1] =10

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | **22** |
| 3 | 0 | **10** | | | | |
| 4 | 0 | | | | | |

**For V[3,2]**

**if** j < wi **then**
V [i, j] = V [i − 1, j]
**2<3 then**

**V[3,2]** = V [3 − 1, 2] =V[2,2] =12

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | **22** |
| 3 | 0 | 10 | **12** | | | |
| 4 | 0 | | | | | |

**For V[3,3]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**3**>=**3 then**

**V[3,3]** = max{V [3 − 1, 3] , V[3-1, 3-3] +20}

= max{22 , 0 +20}=22

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | **22** |
| 3 | 0 | 10 | 12 | **22** | | |
| 4 | 0 | | | | | |

**For V[3,4]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**4**>=**3 then**

**V[3,4]** = max{V [3 − 1, 4] , V[3-1, 4-3] +20}

= max{22 , 10 +20}=30

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | **30** | |
| 4 | 0 | | | | | |

**For V[3,5]**

**if** j >=wi **then**
V [i, j] = max(V [i − 1, j], V [i − 1, j − wi] + vi)
**5**>=**3 then**

**V[3,5]** = max{V [3 − 1, 5] ,  V[3-1, 5-3] +20}

= max{22 ,  12 +20}=32

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | **32** |
| 4 | 0 | | | | | |

**For V[4,1]**

**if** j < wi **then**
V [i, j]  = V [i − 1, j]
**1<2 then**

**V[4,1]** = V [4 − 1, 1] =V[3,1] =10

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | **10** | | | | |

**For V[4,2]**

**if** j >=wi **then**
V [i, j] =  max(V [i − 1, j], V [i − 1, j − wi] + vi)
**2**>=**2 then**

**V[4,2]** = max{V [4 − 1, 2] ,  V[4-1, 2-2] +15}

= max{12 ,  0 +15}=15

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | **15** | | | |

**For V[4,3]**

**if** j >=wi **then**
V [i, j] =  max(V [i − 1, j], V [i − 1, j − wi] + vi)

**3**>=**2 then**

**V[4,3]** = max{V [4 − 1, 3] ,  V[4-1, 3-2] +15}

= max{22 ,  10 +15}=25

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | 15 | **25** | | |

**For V[4,4]**

**if** j >=wi **then**
V [i, j] =  max(V [i − 1, j], V [i − 1, j − wi] + vi)
**4**>=**2 then**

**V[4,4]** = max{V [4 − 1, 4] ,  V[4-1, 4-2] +15}

= max{30 ,  12 +15}=30

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | 15 | **25** | **30** | |

**For V[4,5]**

**if** j >=wi **then**
V [i, j] =  max(V [i − 1, j], V [i − 1, j − wi] + vi)
**5**>=**2 then**

**V[4,5]** = max{V [4 − 1, 5] ,  V[4-1, 5-2] +15}

= max{32 ,  22 +15}=37

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | 15 | 25 | 30 | **37** |

**To identify which items to include in the knapsack**

All of the information we need is in the table V[n,W] is the maximal value of items that can be placed in the Knapsack

Let i=n,   k = W

While i, k > 0 {

If(V[i,k]≠V[i-1,k])

Then do   i=i-1, k=k-wi

Mark the item i

Else

i=i-1

}

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | 15 | 25 | 30 | **37** |

Item 4 is been include in knapsack{4} because V[i,k]≠V[i-1,k]

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | 15 | 25 | 30 | **37** |

Item 3 is not been included in knapsack{4} because V[i,k] =V[i-1,k]

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | 15 | 25 | 30 | **37** |

Item 2 is been included in knapsack{4,2} because V[i,k]≠V[i-1,k])

| i/j | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| 4 | 0 | 10 | 15 | 25 | 30 | **37** |

Item 1 is been included in knapsack{4,2,1} because V[i,k]V[i-1,k]

The optimal knapsack should contain {1, 2,4} with value{12+10+15=37}

**Analysis**: run time: Clearly the run time of this algorithm is O(nW), based on the nested loop structure and the simple operation inside of both loops. When comparing this with the previous $O(2^n)$, we find that depending on W, either the dynamic programming algorithm is more efficient or the brute force algorithm could be more efficient. (For example, for n=5, W=100000, brute force is preferable, but for n=30 and W=1000, the dynamic programming solution is preferable.)