**Example: Find an optimal solution for following 0/1 Knapsack problem using dynamic programming: Number of objects n = 4, Knapsack Capacity M = 5, Weights (W1, W2, W3, W4) = (2, 3, 4, 5) and profits (P1, P2, P3, P4) = (3, 4, 5, 6).**

**Solution:**

Solution of the knapsack problem is defined as,

We have the following stats about the problem,

| **Item** | **Weight (wi)** | **Value (vi)** |
| --- | --- | --- |
| I1 | 2 | 3 |
| I2 | 3 | 4 |
| I3 | 4 | 5 |
| I4 | 5 | 6 |

* Boundary conditions would be V [0, i] = V[i, 0] = 0. Initial configuration of table looks like.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | j → |  |  |  |  |  |  |  |
|  | **Item Detail** |  | **0** | **1** | **2** | **3** | **4** | **5** |
| i=0 |  |  | 0 | 0 | 0 | 0 | 0 | 0 |
| i=1 | w1=2 | v1=3 | 0 |  |  |  |  |  |
| i=2 | w2=3 | v2=4 | 0 |  |  |  |  |  |
| i=3 | w3=4 | v3=5 | 0 |  |  |  |  |  |
| i=4 | w4=5 | v4=6 | 0 |  |  |  |  |  |

**Filling first column, j = 1**

**V [1, 1]** ⇒ i = 1, j = 1, wi = w1 = 2

As, j < wi, V [i, j] = V [i – 1, j]

V [1, 1] = V [0, 1] = 0

**V [2, 1]** ⇒ i = 2, j = 1, wi = w2=  3

As, j < wi, V [i, j] = V [i – 1, j]

V [2, 1] = V [1, 1] = 0

**V [3, 1]** ⇒ i = 3, j = 1, wi = w3=  4

As, j < wi,  V [i, j] = V [i – 1, j]

V [3, 1] = V [2, 1] = 0

**V [4, 1]** ⇒ i = 4, j = 1,  wi = w4=  5

As, j < wi, V[i, j] = V[i – 1, j]

V [4, 1] = V [3, 1] = 0

**Filling first column, j = 2**

**V[1, 2]** ⇒ i = 1, j = 2, wi = w1=  2, vi = 3

As, j ≥ wi, V [i, j]=max {V [i – 1, j], vi + V[i – 1, j – wi] }

= max {V [0, 2], 3 + V [0, 0]}

V[1, 2] = max (0, 3) = 3

**V[2, 2]** ⇒ i = 2, j = 2, wi = w2=  3, vi = 4

As, j < wi, V [i, j] = V[i – 1, j]

V[2, 2] = V[1, 2] = 3

**V[3, 2]** ⇒ i = 3, j = 2, wi = w3=  4, vi = 5

As, j < wi, V[i, j] = V[i – 1, j]

V[3, 2] = V [2, 2] = 3

**V[4, 2]** ⇒ i = 4, j = 2, wi = w4 = 5, vi = 6

As, j < wi, V[i, j] = V[i – 1, j]

V[4, 2] = V[3, 2] = 3

**Filling first column, j = 3**

**V[1, 3]** ⇒ i = 1, j = 3, wi = w1 = 2, vi = 3

As, j ≥ wi, V [i, j]=max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [0, 3], 3 + V [0, 1]}

V[1, 3] = max (0, 3) = 3

**V[2, 3]** ⇒ i = 2, j = 3, wi = w2 = 3, vi = 4

As, j ≥ wi, V [i, j] = max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [1, 3], 4 + V [1, 0]}

V[2, 3] = max (3, 4) = 4

**V[3, 3]** ⇒ i = 3, j = 3, wi = w3 = 4, vi = 5

As, j < wi, V [i, j] = V [i – 1, j]

V[3, 3] = V [2, 3] = 4

**V[4, 3]** ⇒ i = 4, j = 3, wi = w4 = 5, vi = 6

As, j < wi, V[i, j] = V[i – 1, j]

V[4, 3] = V [3, 3] = 4

**Filling first column, j = 4**

**V[1, 4]** ⇒ i = 1, j = 4,  wi = w1 = 2, vi = 3

As, j ≥ wi,  V [i, j]=max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [0, 4], 3 + V [0, 2]}

V[1, 4] = max (0, 3) = 3

**V[2, 4]** ⇒ i = 2, j = 4,  wi = w2 = 3 , vi = 4

As, j ≥ wi,  V [i, j] =max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [1, 4], 4 + V [1, 1]}

V[2, 4] = max (3, 4 + 0)  = 4

**V[3, 4]**  ⇒  i = 3, j = 4,  wi = w3 = 4, vi = 5

As, j ≥ wi,  V [i, j]=max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [2, 4], 5 + V [2, 0]}

V[3, 4] = max (4, 5 + 0) = 5

**V[4, 4]** ⇒ i = 4, j = 4, wi = w4 = 5, vi = 6

As, j < wi, V [i, j] = V [i – 1, j]

V[4, 4] = V [3, 4] = 5

**Filling first column, j = 5**

**V [1, 5]** ⇒ i = 1, j = 5, wi = w1 = 2, vi = 3

As, j ≥ wi,  V [i, j] = max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [0, 5], 3 + V [0, 3]}

V[1, 5] = max (0, 3) = 3

**V[2, 5]** ⇒ i = 2, j = 5, wi = w2 = 3, vi = 4

As, j ≥ wi,  V [i, j] =max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [1, 5], 4 + V [1, 2]}

V[2, 5] = max (3, 4 + 3) = 7

**V[3, 5]** ⇒ i = 3, j = 5,  wi = w3 = 4, vi = 5

As, j ≥ wi, V [i, j] = max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [2, 5], 5 + V [2, 1]}

V[3, 5] = max (7, 5 + 0) = 7

**V [4, 5]**  ⇒  i = 4,   j = 5,  wi = w4=5, vi = 6

As, j ≥ wi,  V [i, j] = max {V [i – 1, j], vi + V [i – 1, j – wi] }

= max {V [3, 5], 6 + V [3, 0]}

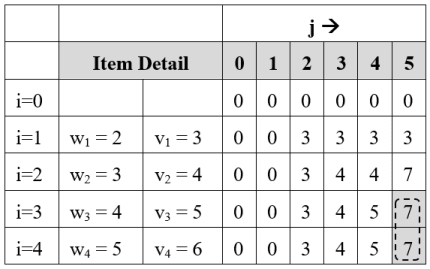
V[4, 5] = max (7, 6 + 0) = 7

Final table would be,

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | j→ |  |  |  |  |  |  |  |
|  | **Item Detail** |  | **0** | **1** | **2** | **3** | **4** | **5** |
| i=0 |  |  | 0 | 0 | 0 | 0 | 0 | 0 |
| i=1 | w1=2 | v1=3 | 0 | 0 | 3 | 3 | 3 | 3 |
| i=2 | w2=3 | v2=4 | 0 | 0 | 3 | 4 | 4 | 7 |
| i=3 | w3=4 | v3=5 | 0 | 0 | 3 | 4 | 5 | 7 |
| i=4 | w4=5 | v4=6 | 0 | 0 | 3 | 4 | 5 | 7 |

**Find selected items for M = 5**

**Step 1 :**Initially, i = n = 4, j = M = 5



V[i, j] = V[4, 5] = 7

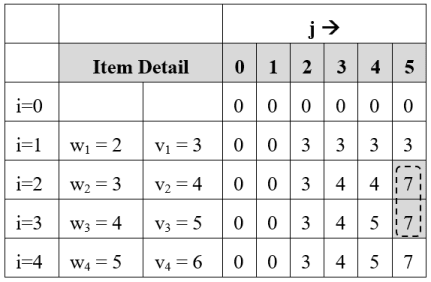
V[i – 1, j] = V[3, 5] = 7

V[i, j] = V[i – 1, j], so don’t select ith item and check for the previous item.

so i = i – 1 = 4 – 1 = 3

Solution Set S = {  }

**Step 2 :**i = 3, j = 5



V[i, j] = V[3, 5] = 7

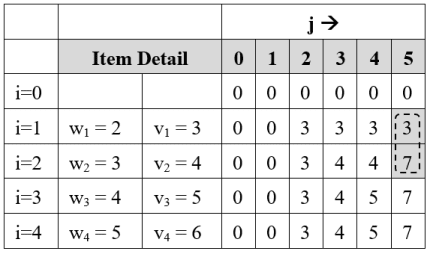
V[i – 1, j] = V[2, 5] = 7

V[i, j] = V[i – 1, j], so don’t select ith item and check for the previous item.

so i = i – 1 = 3 – 1 = 2

Solution Set S = {  }

**Step 3 :**i = 2, j = 5



V[i, j] = V[2, 5] = 7

V[i – 1, j] = V[1, 5] = 3

V[i, j] ≠ V[i – 1, j], so add item Ii = I2 in solution set.

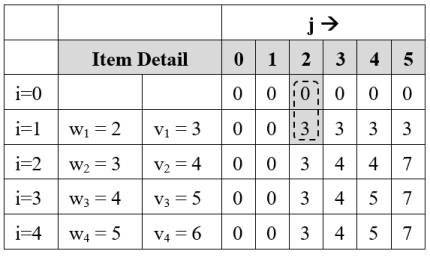
Reduce problem size j by wi

j = j – wi = j – w2 = 5 – 3 = 2

i = i – 1 = 2 – 1 = 1

Solution Set S = {I2}

**Step 4 :**i = 1, j = 2



V[1, j] = V[1, 2] = 3

V[i – 1, j] = V[0, 2] = 0

V[i, j] ≠ V[i – 1, j], so add item Ii = I1 in solution set.

Reduce problem size j by wi

j = j – wi = j – w1 = 2 – 2 = 0

Solution Set S = {I1, I2}

Problem size has reached to 0, so final solution is  
S = {I1, I2} Earned profit = P1 + P2 = 7