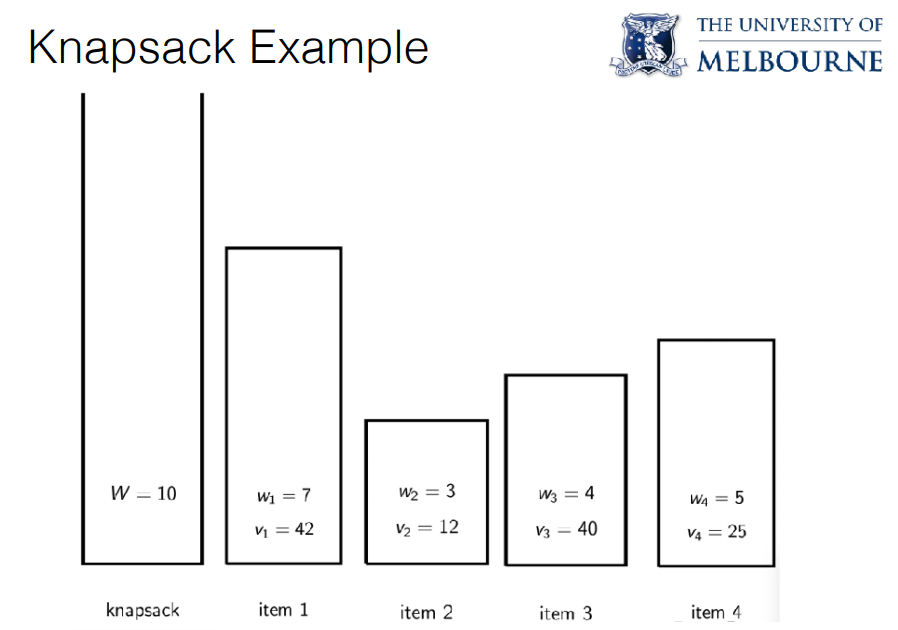
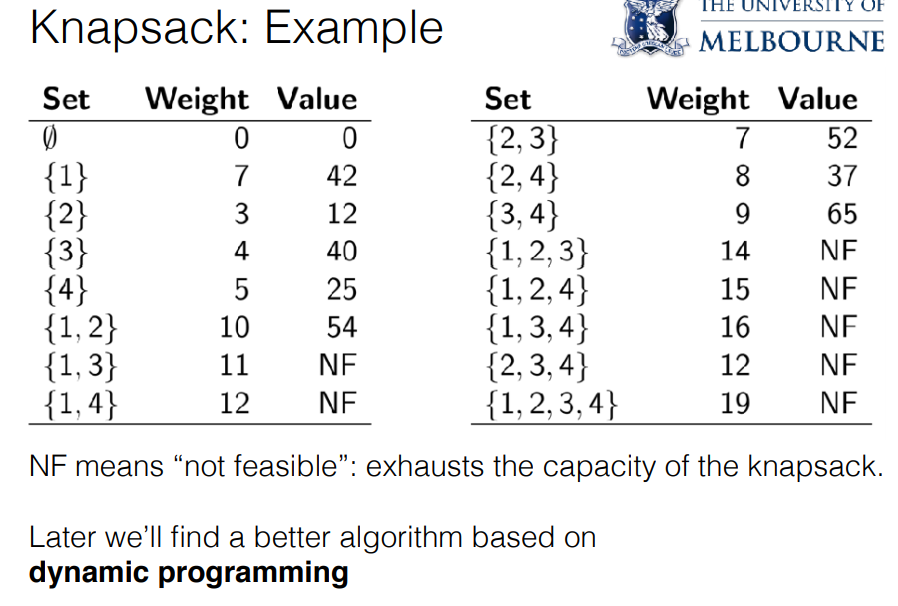
**HOW DOES DYNAMIC PROGRAMMING WORK FOR THE KNAPSACK PROBLEM**

What is a knapsack Problem?

The Knapsack or it can even be a bag, would consider that given n items with weights w1, w2, w3,…wn and the values v1,v2,v3,…,vn and a knapsack of Capacity W we would have the most valuable selection of items whose combined weight would not exceed W.





Say if the weight limit of the given knapsack is 10, then we would get a distribution like the above.

Dynamic Programming approach for the knapsack problem would be:

The Dynamic programming would give a better solution.

The critical step is to find a good answer to the sub problem what is the sub problem.

Here we would design a recurrence relation using two different parameters,

1. Sequence 1,2,…,i of the items considered so far.

2. Remaining capacity  .

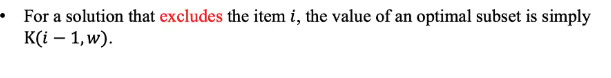


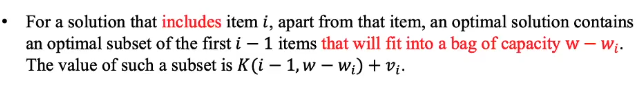


We would be then after finding the Knapsack of n items with a weight of W.

The reason why would focus on finding the Knapsack of item i with a given weight w is that we could express the solution to that recursively.

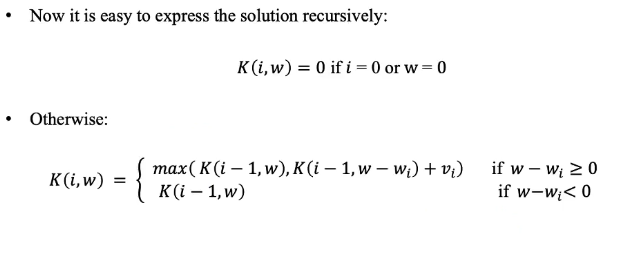
Amongst the first i items we would either pick the item i or we would not.

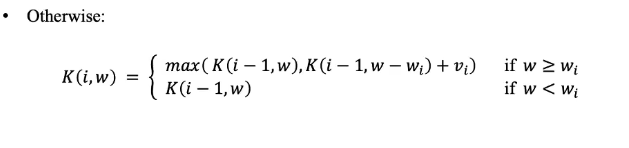


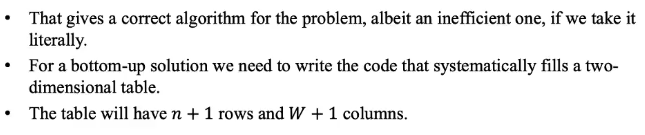


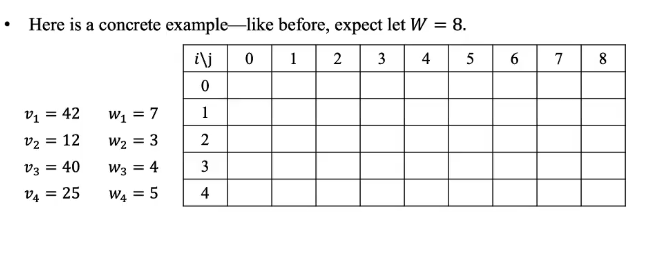
The adding of the ith item is represend by  So we have to ensure that there is enough weight available to add the ith item represented by 

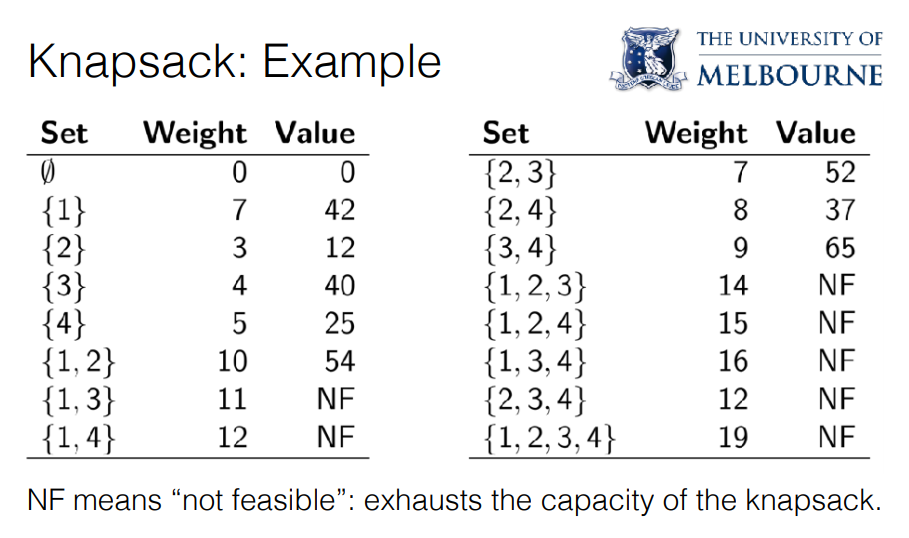


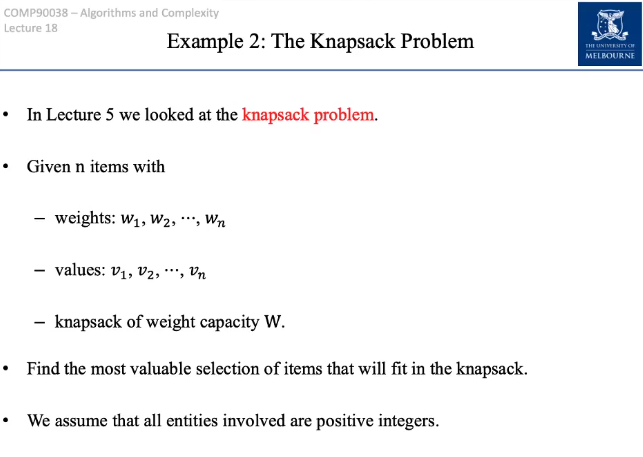


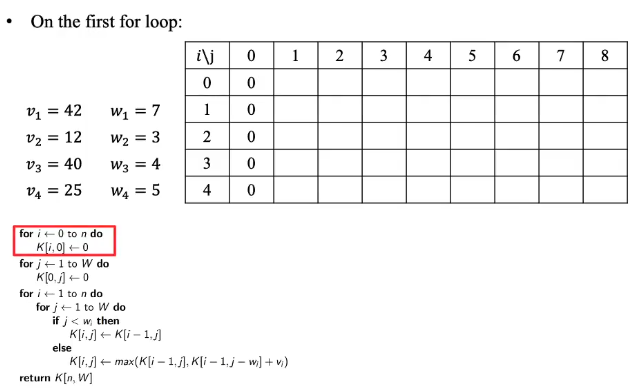












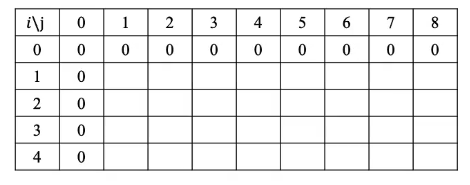
Initially we execute the operation:

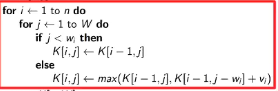


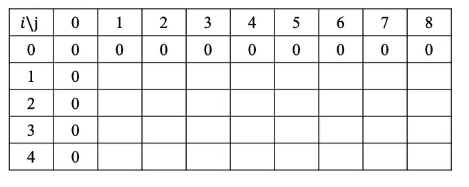
This would fill all zeros throughout for i from 1 to 4.



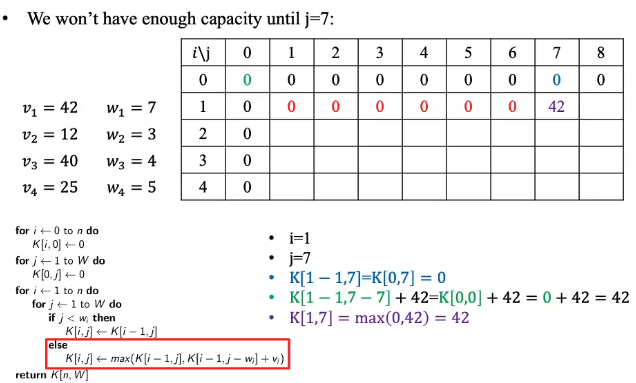
This would fill all zeros throughout for j from 0 to 8.





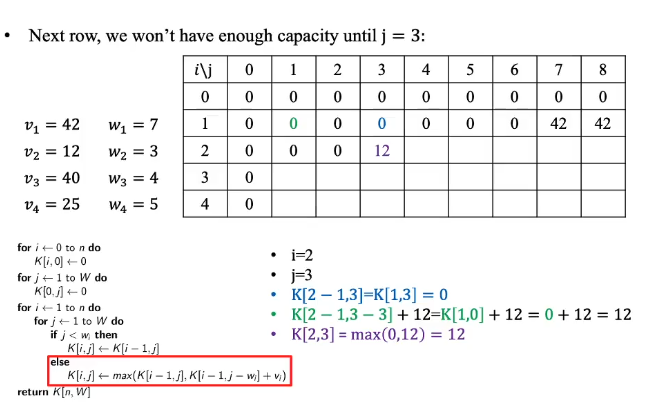


This would contain the max value.

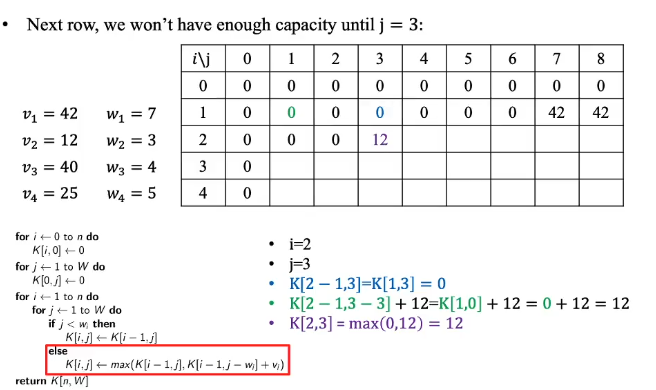


* The first item has a weight of 7 and with a value of 42. So this item can only be added to the knapsack when the value of j would be equal to 7.
* For the first item (i=1) with a value of 42, we would add it to the knapsack when the value of j is 7.
* Next we would compare 0 with 42 and put that element into the knapsack as we copy the value of 42 from the position 7 to position 8 as well.

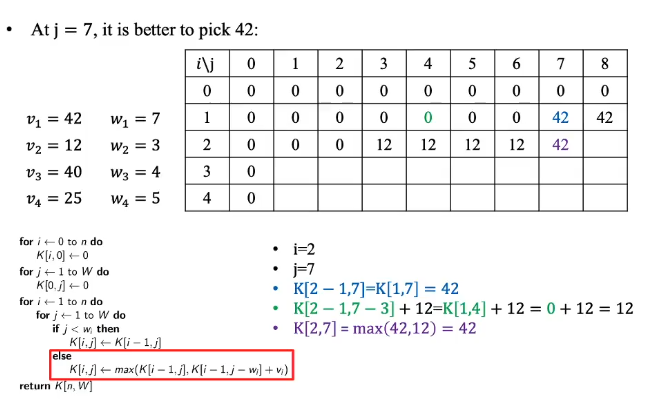
Weight is represented by the value of j, the item number is represented with the value of i .



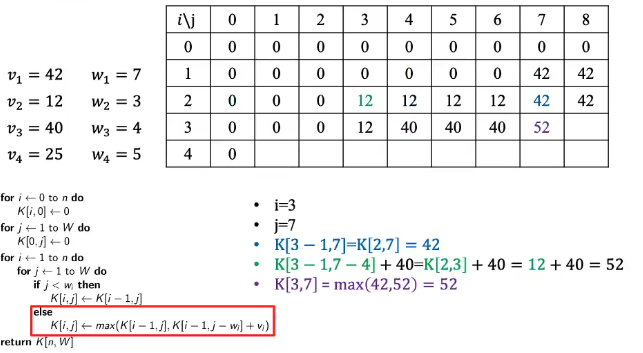
The second item with a value of 12 would have a weight of 3 and then would place in the knapsack when the value of j is 3.



At j = 7 we would jump to pick 42 as the value as we would compare 12 and 42 and it would be a better value.



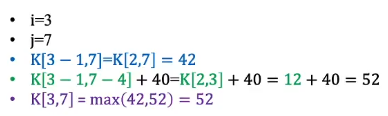
And then we would copy the better value to the adjacent cells of the knapsack.



When the value of i is 3, we have the third item with a value of 40 and a weight of 4. So when we reach a value of j equal to 3, we would do initially a comparison between 0 and 12 and the better value 12 would be placed onto the table. But then when the value of j was equal to 4, we had 40 and it would be placed into the table until when it’s 7 as we do a comparison between 40 and 52 and would keep the better value of 42 onto the table.

When the value of j is equal to 7, we can consider the 40 alone or , 40 and 12 in combination(40+12=52).

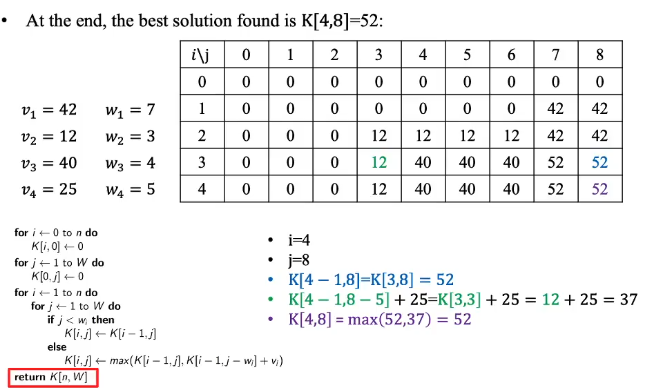
How did we get 52?





**40 is compared with the second and the third item (sum together) which would give 12+40 that’s 52.**

**And then there is comparison between 42 and 52 which would give us the max value of 52.**



**We progress in this way, until we hit the end.**