**Recursive Solution:**

**=> Here we use Memoization(Top down approach), (**start recursive calls from bottom**), so we could form iterative DP formula by this approach.**

**Approach :**

-> Recur for :

1) Include the Current nth item in subset and add its Value to value got by rest (n-1) items and **Weight decreased by weight of current item.**

2) Don’t include current item and return value got by rest (n-1) items **with remaining Weight of Knapsack as it is** bcoz we didn’t include the item.

-> Here each of ‘N’ items has 2 choices so we end up recurring for **O(2^n)** subproblems in below code, but actual possible values are only N\*W so we could store it in a DP table, if already calculated. (Code for it is below this code)

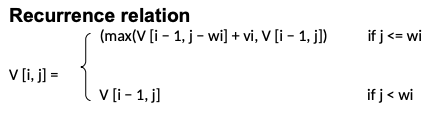
<https://ide.geeksforgeeks.org/dYyuUFZn1O>

**RECURSION USING MEMOIZATION:**

-> We can solve this problem by simply creating a 2-D array that can store a particular state (n, w) if we get it the first time. Now if we come across the same state (n, w) again instead of calculating it in exponential complexity we can directly return its result stored in the table in constant time.

<https://ide.geeksforgeeks.org/PD8a8cpGqc>

**Iterative DP**

****

Code : <https://ide.geeksforgeeks.org/MnYFoCQ1s8>