**0 - 1 Knapsack Problem**

**Medium**Accuracy: 47.21% Submissions: 89928 Points: 4

You are given weights and values of **N** items, put these items in a knapsack of capacity **W** to get the maximum total value in the knapsack. Note that we have only **one quantity of each item**.  
In other words, given two integer arrays **val[0..N-1]** and **wt[0..N-1]** which represent values and weights associated with **N** items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of **val[]** such that sum of the weights of this subset is smaller than or equal to **W.** You cannot break an item, **either pick the complete item or don’t pick it (0-1 property)**.

**Example 1:**

**Input:**

N = 3

W = 4

values[] = {1,2,3}

weight[] = {4,5,1}

**Output:** 3

**Example 2:**

**Input:**

N = 3

W = 3

values[] = {1,2,3}

weight[] = {4,5,6}

**Output:** 0

**Your Task:**  
Complete the function **knapSack()** which takes maximum capacity W, weight array wt[], value array val[], and the number of items n as a parameter and returns the **maximum possible** value you can get.

**Expected Time Complexity:** O(N\*W).  
**Expected Auxiliary Space:** O(N\*W)

**Constraints:**  
1 ≤ N ≤ 1000  
1 ≤ W ≤ 1000  
1 ≤ wt[i] ≤ 1000  
1 ≤ v[i] ≤ 1000

class Solution

{

    public:

    //Function to return max value that can be put in knapsack of capacity W.

    /\*int dp[1001][1001];

    int solve(int w, int wt[], int val[], int i) {

        if (i<0 or w<=0) {

            return 0;

        }

        if (dp[i][w]!=-1) return dp[i][w];

        if (wt[i]<=w) {

            int a=val[i]+solve(w-wt[i], wt, val, i-1);

            int b=solve(w, wt, val, i-1);

            return dp[i][w]=max(a, b);

        }

        else if (wt[i]>w) {

            return dp[i][w]=solve(w, wt, val, i-1);

        }

    }\*/

    int knapSack(int W, int wt[], int val[], int n) {

       // Your code here

       /\*memset(dp, -1, sizeof(dp));

       int ans=solve(W, wt, val, n-1);

       /\*for (int i=0; i<1001; i++) {

           for (int j=0; j<1001; j++) ans=max(ans, dp[i][j]);

       }

       return ans;\*/

       vector<vector<int>> dp(n+1, vector<int> (W+1, 0));

       for (int j=1; j<W+1; j++) {

           for (int i=1; i<n+1; i++) {

               if (wt[i-1]<=j) dp[i][j]=max(val[i-1]+dp[i-1][j-wt[i-1]], dp[i-1][j]);

               else dp[i][j]=dp[i-1][j];

           }

       }

       return dp[n][W];

    }

};