0-1 Knapsack Problem

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.

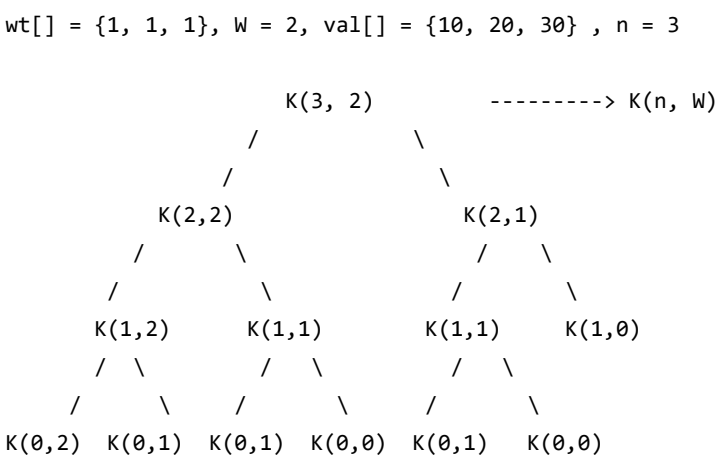
In other words,given two integer arrays val[0 .. n-1] and wt[0 .. n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the complete item, or don't pick it (0-1 property).

A simple solution is to consider all subsets of items and calculate the total weight and value of all subsets. Consider the only subsets whose total weight is smaller than W. From all such subsets, pick the maximum value subset.

1. **Optimal Substructure:** To consider all subsets of items, there can be two cases for every item: (1) the item is included in the optimal subset, (2) not included in the optimal set.
   Therefore, the maximum value that can be obtained from n items is max of following two values.
   2.1. If weight of nth item is greater than W, then the nth item cannot be included, hence maximum value obtained by n-1 items and W weight (excluding nth item).
   2.2 else Value of nth item plus maximum value obtained by n-1 items and W minus weight of the nth item (including nth item).

Note that function recursiveKnapsack() computes the same subproblems again and again:

The recursion tree for following sample inputs.



Complexity of DP solution

Time Complexity: O(n*W) where n is the number of items and W is the capacity of knapsack.