**Knapsack Optimization Problem**

**Problem 1.** Formulate your own procedure for solving this problem.

**Solution**: My solution will be similar to the explanation given under section (3) of problem 2. I will first create a new set S which has as its elements relative value or **vi/wi**ordered decreasingly, where **vi** and **wi** are value and the weight of **si** respectively coming from the set **S**. Then, I will iterate through this set starting from the beginning element, and pick only those items such that the sum of weight of these items is less than or equal to the maximum allowable weight, i.e., **W**.

**Problem 2**. Greedy Strategies:

Given: S = {0, 1, 2}, v[] = {1, 3, 4}, w[] = {1, 2, 4}, W = 4.

1. Try arranging S in increasing order of weight. Then load the knapsack with items from S until it becomes impossible to add any more because of the weight restriction.

**Solution**: Arranging in decreasing order of weight gives us: **S = {0, 1, 2}, v[] = {1, 3, 4}, w[] = {1, 2, 4}**. In this strategy, we can only pick the first and second items as their overall weight is below the maximum, i.e., 4. We cannot add the third element as it would mean that the sum of weights of items picked becomes greater than 4, violating the restriction. So, now we then add the values of the first two items, which gives us 1+3 = 4. Obviously, this strategy gave us the optimum result. However, it should be noted that knowing that the answer is optimal wouldn’t have been easy had the set been big.

1. Try arranging S in decreasing order of value. Then load the knapsack with items from until it becomes impossible to add any more because of the weight restriction.

**Solution:** Arranging in the decreasing order of value gives us: S = {2, 1, 0}, v[] = {4, 3, 1}, w[] = {4, 2, 1}, W = 4. Now, we pick items by staring from the beginning until the sum of the items doesn’t exceed 4. So, here only the first item which has a weight of 4 and value of 4 at the same time can be picked. Obviously, the value we got in this strategy is the maximum possible under given restriction and the set provided. However, compared to the strategy we used in (1) before, the weight is greater by 1 (it was 3 in the case above). So, overall it’s not the best solution, but still gives the correct answer as the value is the maximum possible under the restriction (weight of items in the bag not exceeding 4).

1. Try arranging S in decreasing order of value per weight. This strategy is in line with my solution for problem 1. For each i, let **bi = vi/w**i. Then we compute: **b0 = 1, b1 = 1.5, b2 = 1**. If we arrange by decreasing order of the bi, we get **S = {1, 0, 2}, v[ ] = {3, 1, 4},w[ ] = {2, 1, 4}.**

Here also, we can pick the first two items as their total weight, which is **2 + 1 = 3**, is below 4. However, we cannot add the fourth items because of the restriction (Max weight allowed = 4). Now we add the values of the two items selected, which is **3 + 1 = 4**. This strategy also gives us the maximum possible value given the restriction. Moreover, we total weight of the items in the bag is 3, which is the lowest possible to get the value of 4.

**Problem 3**. Try answering the following:

1. You are trying to solve a knapsack problem; as always, you are given **S = {s0, s1, . . ., sn−1}, w[ ] = {w0, w1, . . ., wn−1}, v[ ] = {v0, v1, . . ., vn−1}** and a maximum weight W. Is it possible that an optimal solution will not make use of item sn−1 (in other words, that a particular optimal solution S0 does not contain the item sn−1)? Give an example to illustrate your idea.

Solution: The optimal solution may or may not include the item sn−1. Take for example:

**S = {1, 0, 2}, v[ ] = {3, 4, 3}, w[ ] = {2, 1, 4}.** In this case, the optimal value is **3 + 4 = 7**, which corresponds to the first and second elements of the set **S**. The last element is not included in the optimal solution. The best strategy to that I consider to solve problems such as this is the one described under section 3 of problem 2.

1. You are trying to solve a knapsack problem; as always, you are given **S = {s0, s1, . . ., sn−1}, w[ ] = {w0, w1, . . ., wn−1}, v[ ] = {v0, v1, . . ., vn−1}** and a maximum weight **W**. Suppose that an optimal solution S0 S makes use of the item sn−1 (in other words, **sn−1 S0)**. Suppose now that we remove **sn−1** from **S**, giving us the smaller set **S0 = {s0, s1, . . ., sn−2}**, and remove **sn−1** from **S0**, giving us **S’0**, and also change **W** to **W’** = **W−wn−1.** Is it true that **S’0** is now a solution for the knapsack problem with items **S0**, weights **{w0, w1, . . ., wn−2}**, values **{v0, v1, . . ., vn−2}**, and maximum weight **W’**? Explain.

**Solution**: My answer is **YES**. Let’s assume that we use the strategy given under (3) of problem 2 (arranging S in decreasing order of value per weight). Given W’ = W – wn-1, the sum of weights of elements in **S’0** respects the restriction: Suppose, W0 = . Then, **W’0 = W0 – wn-1 =** .

**W0 W => W’0 + wn-1  W** …subtracting wn-1 both sides of the equation gives

**W’0 W - wn-1**…… which is equivalent to

**W’0 W’**  ….... Note that **W - wn-1  = W’**

So, the weight of items left S’0 doesn’t exceed **W’**.

Now, the next question is can we add more element into **S’0** and still the total weight is below W’? The answer is **NO**. We assumed that S0 had the optimal value and it was not possible to add any more elements to **S0**because then the weight restriction is violated. If that is the case, then adding any more element to **S’0** will have the same consequence, because W’ is less only by wn-1 than W, which is the weight of element removed from S0 to give S’0. It is easy to give mathematical proof of this, but I think it’s straight forward to deduce that based on my explanations above.

Another question is, can we replace any element in **S’0** with another element from the set **S** which is not already included in **S’0** and still get a better optimal solution?

The answer is again is **NO**. If there were such element in the set **S**, it would have already been included in the original **S0** in favor of the element that we now consider to replace.