* **Variations and strategy**

1. **Fractional Knapsack Problem**

* **Problem Statement:-**
* Let’s say thief is doing robbery at mall, he has knapsack with capacity(C), in mall there are various weighted item and various profit. But thief is trying to take only those item which has high profit, you can take fraction of item also. Now your job is to find the maximum profit earn by thief in given capacity.

**Solution:**

* One approach to solve fractional knapsack problem is using brute-force method, in which we would be try to generate all possible subset with all different fraction but that will be too much time consuming.
* An efficient solution for this problem is use greedy approach. Here we take the profit by weight ratio and then we sorted this ratio in descending order, then we take the item which has highest ratio and subtract its weight from the capacity at last we can take the full item or fraction of item to use full capacity. This approach always gives optimal solution.
* If profit/weight ratio already in sorted manner then fractional knapsack using greedy algorithm gives more better performance.
* If ratio was not already sorted then we have to spend time for sorting array for ratio and then we can complete our problem.

**Variation:**

* If thief want that he has to take only full item not fraction of item allowed, then this problem is know as 0-1 knapsack problem and if allow to take fraction of item also then it is fractional knapsack problem.

**Limitation:**

* We can’t solve 0-1 knapsack problem using greedy approach sometimes because sometimes it will not give the optimal solution even if we have some more capacity.

1. **Making Change Problem**

* **Problem Statement:**
* The problem is that suppose you have N money and someone says that give me the minimum number of coins whose value is equal to N, and coins are provided by that person and imagine you have n numbers of coins for that.

**Variation:**

* It is most common variation of the coin change problem, a general case of partition in which, given the denominations of an infinite set of coins, and you have to find out how many ways to make a change for a specific amount of money. For this purpose you have to use Dynamic Programming(DP) by using it you can solve problem in polynomial time.

**Solution:**

* Here the problem is simple that you have find minimum number of coin to reach specific amount of money, so that you have to do first sorting in decreasing order of the all available coins and the subtract the coin’s value which highest among all and the small than amount of money and you have find the minimum number of coins required, this is the greedy approach to solve this problem but some times it not give optimal solution.

**Example:**

Coins: 1 5 15 20 Amount = 30

* By algorithm you will get answer 3 coins(20,5,5) but it is obvious that we can reach to this amount by only two coins(15,15). This is the drawback of the this approach.

**Limitation:-**

* Here it might possible that we can’t make change of given N in two case : :

1. If we have limited number of coins.
2. If in values of coins we have not sufficient value of coins, it might be possible that we never found that set of coins whose value is equal to N.