Greedy Algorithms

Fractional Knapsack

```
wt[]: An array containing weight against each item.
value[]: An array containing market value of each item.
items[]: An array containing the item numbers.
N: Total #of items.
M: Maximum capacity of Knapsack.

FKSP( items[], wt[], value[], N, M)
{
//create a new array "V_W_Ratio" of type double to store the ratio of (value/wt)
   V_W_Ratio[1...N]
   For(i=1 to N)
   {
      V_W_Ratio[i] = value[i]/wt[i]
   }
//Now sort the data of all the arrays in descending order on the base of (value/wt) ratio. Since we want to maximize the profit, so the ideal scenario is "maximum value and minimum weight" that is why we are going to sort on the base of (value/wt) ratio.
   Sort(items, wt, value, V_W_Ratio, N)
//Now calculate the profit and keep the track of selected items providing maximum profit.
   profit = 0
   vector<int> Selected_Items
   For(i=1 to N)
   {
      if(wt[i] < M) {
//if the available capacity "M" is greater than the current item's weight "wt[i]" then select the current item completely.
         profit += value[i]
         Selected_Items.push_back(items[i])
      }
      else
      {
//if the available capacity "M" is less than the current item's weight "wt[i]" then select the fractional part of current item.
         profit += (M/wt[i]) *value[i];
//After selecting the fractional part of this item, the remaining capacity "M" will become zero so break the loop.
         break;
      }
   }
   return (profit, Selected_Items)
}
```