Greedy Method

- Most straightforward design technique
  - Most problems have n inputs.
  - **Solution** contains a subset of inputs that **satisfies a given constraint**.
  - **Feasible solution**: Any subset that satisfies the constraint.
  - Need to find a feasible solution that ***maximizes or minimizes*** a given **objective function** – ***optimal solution***.
- Used to determine a feasible solution that may or may not be optimal
  - At every point, make a decision that is locally optimal; and hope that it leads to a globally optimal solution.
  - Leads to a powerful method for getting a solution that works well for a wide range of applications.
  - May not guarantee the best solution.
- Ultimate goal is to find a ***feasible solution*** that ***minimizes [or maximizes] an objective function***; which is known as an optimal solution

General Abstractions of Greedy Algorithm(for subset paradigm)

```
Algorithm Greedy(a,n)
// a[1:n] contains the n inputs.
{
        solution:= Φ ; // Initialize the solution.
        for i:=1 to n do
        {
                x:= Select(a);
                if Feasible(solution,x) then
                        solution:= Union(solution,x);
        }
        return solution;
}
```

Select , **Feasible** and **Union** functions are implemented depending up on the given problem.

Knapsack Problem(Fractional)

- Given n objects a knapsack or bag. Object *i* has a weight w_i and the knapsack has a capacity m. If a fraction x_i, $0 < x_i < 1$,of object *i* is placed into the knapsack, then a profit $p_i x_i$ is earned. The ***objective*** is to obtain a filling of the knapsack that ***maximizes the total profit earned*** . Since the knapsack capacity is m, we require the total weight of all chosen objects to be at most m. Formally, the problem can be stated as

$$\text{maximize} \sum_{1 \leq i \leq n} p_i x_i$$

$$\text{subject to} \sum_{1 \leq i \leq n} w_i x_i \leq m$$

$$\text{and } 0 \leq x_i \leq 1, \quad 1 \leq i \leq n$$

The profits and weights are positive numbers

- A feasible solution is any set $(x_1,x_2,....x_n)$ satisfying $\sum_{1\le i\le n} w_i x_i \le m$ and
$0 \le x_i \le 1, \quad 1 \le i \le n$
- An optimal solution is a feasible solution that $\text{maximize} \sum_{1\le i\le n} p_i x_i$

Example

Consider the following instance of the knapsack problem n=3, m=20 , (p1,p2,p3)=(25,24,15) and (w1,w2,w3)=(18,15,10) .

S.No	(x1,x2,x3)	$\Sigma w_i x_i$	$\Sigma p_i x_i$
1	(1/2,1/3,1/4)	=18*1/2 +15*1/3+10*1/4 =9+5+2.5 =16.5	=25*1/2+24*1/3*15*1/4 =12.5+8+3.75 =24.25
2	(1,2/15,0)	20	28.2
3	(0,2/3,1)	20	31
4	(0,1,1/2)	20	31.5

Strategy 1: Arrange the objects in the decreasing order of the profits

Object	Profit	Weight
1	25	18
2	24	15
3	15	10

Object	Profit	Weight
1	25	18
2	24	15
3	15	10

Given m = 20,
Initialize X =(0,0,0), remaining weight(rw) = m

Object Selected	Is feasible	Profit earned	Remaining Capactiy
Object 1	18 < rw ? Yes	P= 25*1=25	rw = 20-18 =2
Object 2	15< rw ? No partition Object 2	P = 25 + 2/15 *24 = 25 + 3.2 = 28.2	rw = 2 – (2/15) *15 =0

Solution Vector X is (1, 2/15, 0)

Profit earned is 28.2, which is not optimal

Strategy 2: Arrange the objects in the increasing order of the weights

Object	Profit	Weight
3	15	10
2	24	15
1	25	18

Object	Profit	Weight
1	25	18
2	24	15
3	15	10

Given m = 20,
Initialize X =(0,0,0), remaining weight(rw) = m

Object Selected	Is feasible	Profit earned	Remaining Capactiy
Object 3	10 < rw ? Yes	P= 15*1=15	rw = 20 - 10 = 10
Object 2	15< rw ? No partition Object 2	P = 15 + 10/15 *24 = 15 + 14.4 = 29.4	rw = 10 – (10/15) *15 =0

Solution Vector X is (0, 2/3, 1)

Profit earned is 29.4 which is not optimal

Strategy 3: Arrange the objects in the decreasing order of the P/W ratios

- Calculate P/W ratio and
- arrange the objects in the decreasing order of P/W ratios

Object	Profit(P_i)	Weight(W_i)	P_i/W_i
1	25	18	1.3
2	24	15	1.6
3	15	10	1.5

Object	Profit(P_i)	Weight(W_i)	P_i/W_i
2	24	15	1.6
3	15	10	1.5
1	25	18	1.3

Given m = 20,
Initialize X =(0,0,0), remaining weight(rw) = m

Object Selected	Is feasible	Profit earned	Remaining Capactiy
Object 2	15 < rw ? Yes	P= 24*1=24	rw = 20 - 15 = 5
Object 3	10< rw ? No partition Object 3	P = 24 + 5/10 *15 = 24 + 7.5 = 31.5	rw = 5 – (5/10) *10 =0

Solution Vector X is (0, 1, 1/2)

Profit earned is 31.5 which is optimal

```
Algorithm GREEDY _KNAPSACK(P, W, M, X, n)
// P[1:n] and W[1 :n] contain the profits and weights respectively of the n
//objects ordered so that P[i]/W[i] >= P[i + 1]/W[i + 1].
//M is the knapsack size and X[1:n] is the solution vector
{
        X = 0                //initialize solution to zero
        rw = M          // rw = remaining knapsack capacity
        for i = 1 to n do
        {
            if W[i] > rw then
                break;
            X[i] = 1
            rw = rw - W[i]
        }
        if i <= n then
          X[i] = rw/W[i];
}
```