Greedy Method

Greedy Method

- Greedy algorithms work in stages where in each stage one of the inputs is examined.
- The input is *selected* based on an optimization criterion.
- If the input forms part of an optimal solution, it is included in the solution. It is discarded otherwise.

Greedy Method

- The steps are repeated to find the complete solution.
- This model of Greedy method is called as *subset paradigm*.
- The other model called as *ordering paradigm* is based on considering the input elements in some predefined order.

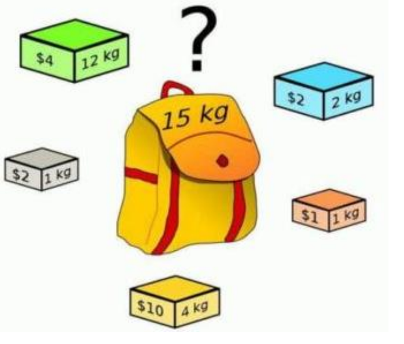
Greedy Method

Control Abstraction

```
Algorithm Greedy(a,n)
//a contains inputs
{
    solution:={};
    for i:= 1 to n do
    {
        x=Select(a);
        if feasible(solution,x) then
            solution:=solution ∪ {x};
    }
    return solution;
}
```

Knapsack Problem

- Given n objects and a knapsack. Object i has weight w_i and profit p_i and the knapsack has a capacity m.
- The objective is to maximize the total profit by selecting objects (fraction of an object) without exceeding the total capacity of the knapsack.



Knapsack Problem

- $maximize\ \sum_{1 \le i \le n} p_i x_i$
- $Subject\ to \sum_{1 \le i \le n} w_i x_i \le m$
- $0 \le x_i \le 1\ and\ 1 \le i \le n$
- x_i is 0, the object i is not selected.
- x_i is 1, the complete object i is selected.
- x_i is $0.4, 40\%$ of the object is selected and hence contributes 40% of the profit.

Knapsack Problem

- $maximize \ \sum_{1 \le i \le n} p_i x_i$ (1)
- $Subject\ to \sum_{1 \le i \le n} w_i x_i \le m$ (2)
- $0 \le x_i \le 1\ and\ 1 \le i \le n$ (3)
- A feasible solution is any set $\{x_k, \dots, x_r\}$ which satisfy (2) and (3)
- An optimum solution is a feasible solution that satisfies (1)

Knapsack Problem

- Consider the instance of the Knapsack problem: $n = 3, m = 20, (p_1, p_2, p_3) = (25,24,15), (w_1, w_2, w_3) = (18,15,10)$

S.No	(x_1, x_2, x_3)	$\sum w_i x_i$	$\sum p_i x_i$
1	$(1, \frac{2}{15}. 0)$	20	28.2
2	$(0, \frac{2}{3}, 1)$	20	31
3	$(0,1. \frac{1}{2})$	20	31.5

Item (I_i)	Weight (w_i)	Profit (p_i)	Profit/Weight (p_i/w_i)
1	40	280	7
2	10	100	10
3	20	120	6
4	24	120	5

No. of Items n = 4
Knapsack Capacity m = 60

According to profit/weight new order (descending) is **I_2, I_1, I_3, I_4**

Item Order	Before Knapsack Capacity	Condition	Solution vector (x_i) [Item Capacity]	After Knapsack Capacity
I_2	60	10<60	1	50
I_1	50	40<50	1	10
I_3	10	20<10	10/20 = 1/2	0
I_4	10	24<0	0/20 = 0	0

Solution vector according to given problem is **[1, 1, 1/2, 0]**

Total Profit = $p_1x_1 + p_2x_2 + p_3x_3 + p_4x_4$

= 280*1 + 100*1 + 120*1/2 +120*0 = **440**

Item (I_i)	Weight (w_i)	Profit (p_i)	Profit/Weight (p_i/w_i)
1	2	10	5
2	3	5	1.66
3	5	15	3
4	7	7	1
5	1	6	6
6	4	18	4.5
7	1	3	3

No. of Items n = 7
Knapsack Capacity m = 15

According to profit/weight new order (descending) is I_5, I_1, I_6, I_3, I_7, I_2, I_4

Item Order	Before Knapsack Capacity	Condition	Solution vector (x_i)	After Knapsack Capacity
I_5	15	1<15	1	14
I_1	14	2<14	1	12
I_6	12	4<12	1	8
I_3	8	5<8	1	3
I_7	3	1<3	1	2
I_2	2	3<2	2/3	0
I_4	0	0<7	0	0

Solution vector according to given problem is **[1, 2/3, 1, 0, 1, 1, 1]**

Total Profit = $p_1x_1 + p_2x_2 + p_3x_3 + p_4x_4 + p_5x_5 + p_6x_6 + p_7x_7$
= 1*10 + 2/3*5 + 1*15 + 0*7 + 1*6 + 1*18 + 1*3 = **55.33**

```
Algorithm GreedyKnapSack(m,n)
//p[1..n] contains the profits and w[1..n] contains the weights
//of n objects ordered such that
//p[i]/w[i]>=p[i+1]/w[i+1]. job[1..n] contains the IDs
//of jobs in p and w. m is the capacity of knapsack. x[1..n] is the
solution vector.
{
       for i = 1 to n do x[i] = 0.0;
       U = m;
       for i = 1 to n do
       {
              if (w[i] > U) then break;
              x[i] = 1.0;
              U = U - w[i];
              profit = profit + p[i];
       }
       if (i <= n) then
       {
              x[i] = U/w[i];
              profit = profit + p[i]* U/w[i];
       }
       return profit;
}
```

Time Complexity

- Excluding the complexity of sorting, the algorithm is $O(n)$