K. J. Somaiya College of Engineering

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

Batch: A1	**Roll No.:** 16010120015
Experiment No.5	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implementation of Knapsack Problem using Greedy strategy

Objective: To learn the Greedy strategy of solving the problems for different types of problems

CO to be achieved:

CO 2	Describe various algorithm design strategies to solve different problems and analyze Complexity.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://lcm.csa.iisc.ernet.in/dsa/node184.htm**
4. **http://students.ceid.upatras.gr/~papagel/project/kruskal.htm**
5. **http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/kruskalAlgor.html**
6. **http://lcm.csa.iisc.ernet.in/dsa/node183.html**
7. **http://students.ceid.upatras.gr/~papagel/project/prim.htm**
8. **http://www.cse.ust.hk/~dekai/271/notes/L07/L07.pdf**

Pre Lab/ Prior Concepts:

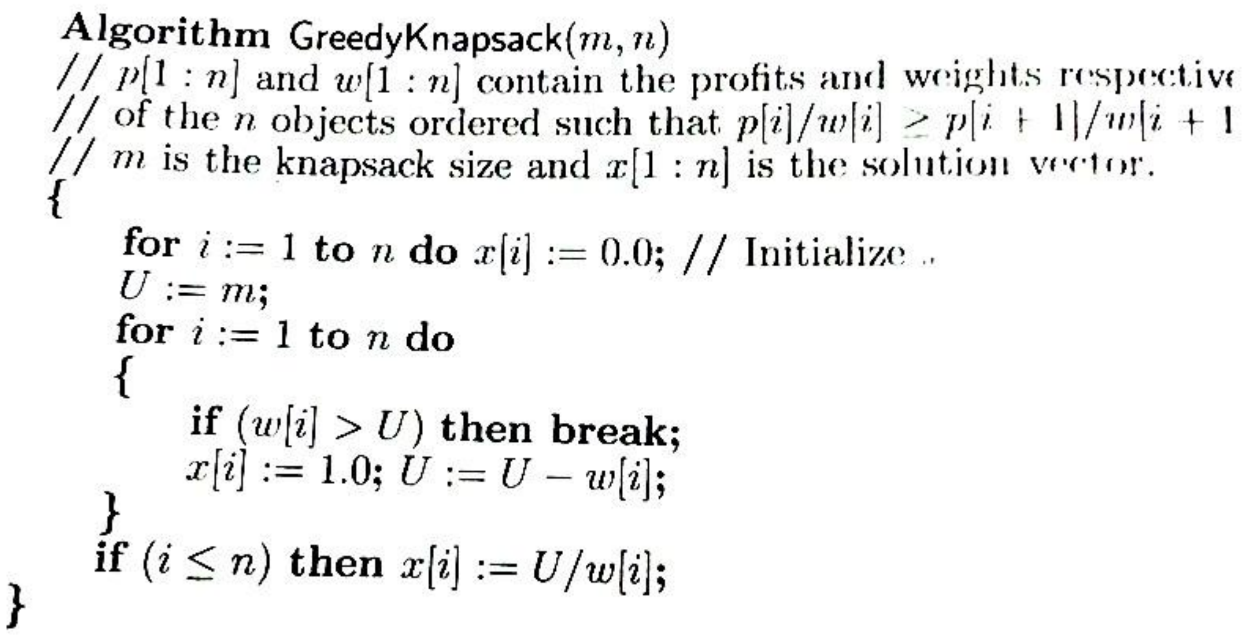
Historical Profile:

The knapsack problem represents constraint satisfaction optimization problems' family. Based on nature of constraints, the knapsack problem can be solved with various problem saolving strategies. Typically, these problems represent resource optimization solution.

Given a set of n inputs. · Find a subset, called feasible solution, of the n inputs subject to some constraints, and satisfying a given objective function. · If the objective function is maximized or minimized, the feasible solution is optimal. · It is a locally optimal method.

New Concepts to be learned:
Application of algorithmic design strategy to any problem, Greedy method of problem solving Vs other methods of problem solving, optimality of the solution, knapsack problem and their applications

Knapsack Problem Algorithm



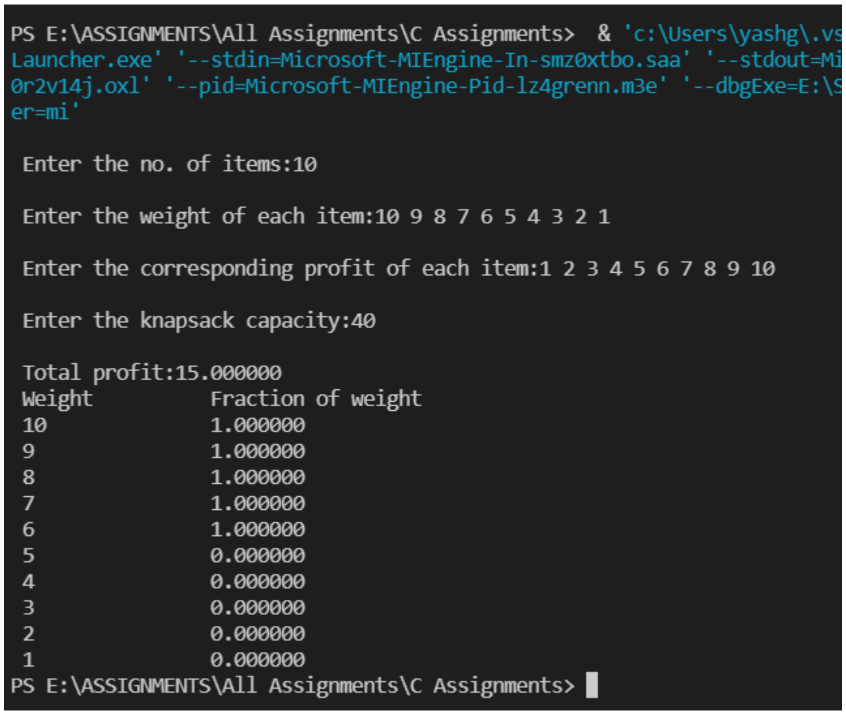
```
Algorithm GreedyKnapsack($m, n$)
// $p[1:n]$ and $w[1:n]$ contain the profits and weights respectiv
// of the $n$ objects ordered such that $p[i]/w[i] \geq p[i+1]/w[i+1$
// $m$ is the knapsack size and $x[1:n]$ is the solution vector.
{
    for $i := 1$ to $n$ do $x[i] := 0.0$; // Initialize .
    $U := m$;
    for $i := 1$ to $n$ do
    {
        if $(w[i] > U)$ then break;
        $x[i] := 1.0$; $U := U - w[i]$;
    }
    if $(i \leq n)$ then $x[i] := U/w[i]$;
}
```

Example: Knapsack Problem

Analysis of Knapsack Problem algorithm:

```
#include <stdio.h>
#include<stdlib.h>
#define MAX 50
void sort(float profit[MAX], float weight[MAX], float ratio[MAX], int n)
{
  int i, j, temp;
  for (i = 0; i < n - 1; i++)
    for (j = 0; j < n - 1 - i; j++)
    {
      if (ratio[j] > ratio[j + 1])
      {
        temp = ratio[j];
        ratio[j] = ratio[j + 1];
        ratio[j + 1] = temp;

        temp = weight[j];
        weight[j] = weight[j + 1];
        weight[j + 1] = temp;

        temp = profit[j];
        profit[j] = profit[j + 1];
        profit[j + 1] = temp;
      }
    }
}
void main()
{
  float p = 0.0, m, profit[MAX], weight[MAX], ratio[MAX], x, ans[MAX];
  int n, i;

  printf("\n Enter the no. of items:");
  scanf("%d", &n);
  printf("\n Enter the weight of each item:");
  for (i = 0; i < n; i++)
    scanf("%f", &weight[i]);
  printf("\n Enter the corresponding profit of each item:");
  for (i = 0; i < n; i++)
    scanf("%f", &profit[i]);
  printf("\n Enter the knapsack capacity:");
  scanf("%f", &m);
  x = m;
  for (i = 0; i < n; i++)
    ratio[i] = profit[i] / weight[i];
  sort(profit, weight, ratio, n);
  for (i = 0; i < n; i++)
  {
    if (weight[i] > x)
      break;
    else
    {
      p += profit[i];
      x -= weight[i];
      ans[i] = 1;
    }
  }
  if (i < n && x != 0)
  {
    ans[i] = x / weight[i];
    p += profit[i] * ans[i];
    i++;
  }
  while (i < n)
  {
    ans[i] = 0;
    i++;
  }
  printf("\n Total profit:%f", p);
  printf("\n Weight \t Fraction of weight");
  for (i = 0; i < n; i++)
    printf("\n %d\t\t %f", (int)weight[i], ans[i]);


}
```

Output:



```
PS E:\ASSIGNMENTS\All Assignments\C Assignments>  & 'c:\Users\yashg\.vs
Launcher.exe' '--stdin=Microsoft-MIEngine-In-smz0xtbo.saa' '--stdout=Mi
0r2v14j.oxl' '--pid=Microsoft-MIEngine-Pid-lz4grenn.m3e' '--dbgExe=E:\S
er=mi'

 Enter the no. of items:10

 Enter the weight of each item:10 9 8 7 6 5 4 3 2 1

 Enter the corresponding profit of each item:1 2 3 4 5 6 7 8 9 10

 Enter the knapsack capacity:40

 Total profit:15.000000
 Weight          Fraction of weight
 10              1.000000
 9               1.000000
 8               1.000000
 7               1.000000
 6               1.000000
 5               0.000000
 4               0.000000
 3               0.000000
 2               0.000000
 1               0.000000
PS E:\ASSIGNMENTS\All Assignments\C Assignments> 
```

Time complexity for knapsack problem:
O(N) + O(NlogN) + O(N) = O(NlogN)
Therefore, the time complexity of knapsack problem using greedy method is O(NlogN)

Conclusion:

By performing this experiment we have understood the greedy knapsack problem and we have learned about the knapsack problem and the methods to solve this problem. The knapsack problem is a way to solve a problem in such a way so that the capacity constraint of the knapsack doesn't break and we receive maximum profit.