**Batch: C1 Roll No.: 16010122221**

**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 analyse 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:**

Data structures, Concepts of algorithm analysis

**Historical Profile:**

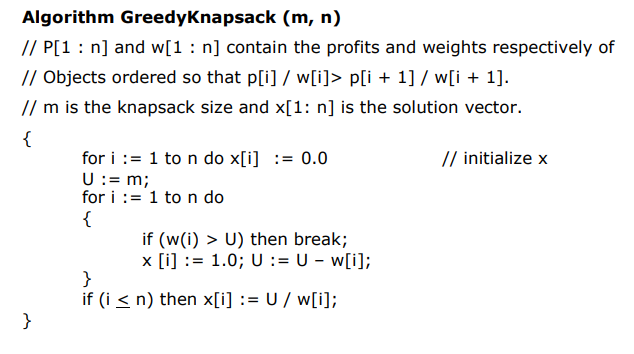
The knapsack problem represents constraint satisfaction optimization problems’ family. Based on nature of constraints, the knapsack problem can be solved with various problem saolving strategies. Typically, these problems represent resource optimization solution.

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.

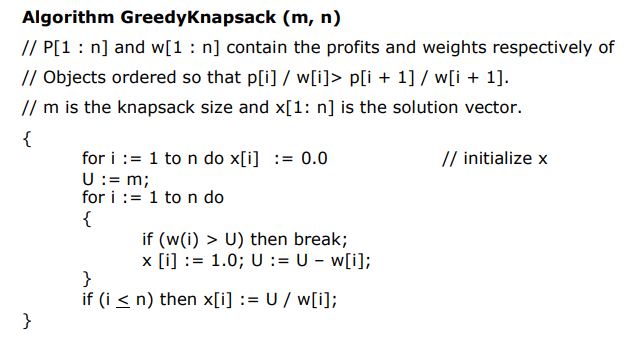
**New Concepts to be learned:**

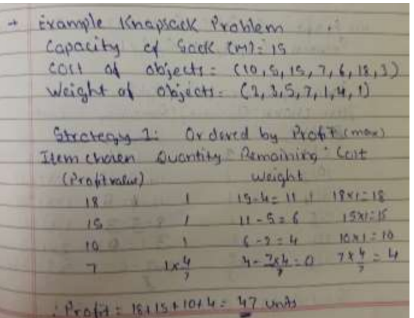
Application of algorithmic design strategy to any problem, Greedy method of problem solving Vs other methods of problem solving, optimality of the solution, knapsack problem and their applications

**Knapsack Problem Algorithm**

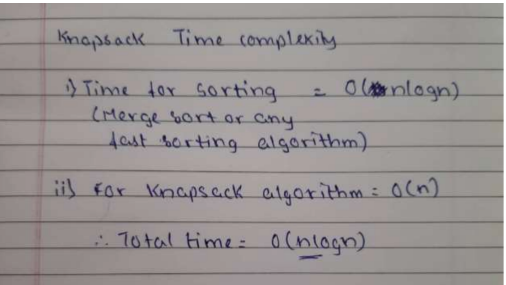
****

**Example: Knapsack Problem**

****

****

**Analysis of Knapsack Problem algorithm:**

****

**CODE AND OUTPUT:-**

class Item:

def \_\_init\_\_(self, profit, weight):

self.profit = profit

self.weight = weight

def fractionalKnapsack(W, arr):

arr.sort(key=lambda x: (x.profit/x.weight), reverse=True)

finalvalue = 0.0

for item in arr:

if item.weight <= W:

W -= item.weight

finalvalue += item.profit

else:

finalvalue += item.profit \* W / item.weight

break

return finalvalue

if \_\_name\_\_ == "\_\_main\_\_":

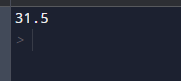
W = 20

arr = [Item(25, 18), Item(24, 15), Item(15, 10)]

max\_val = fractionalKnapsack(W, arr)

print(max\_val)

**OUTPUT:-**

****

**Conclusion:** HENCE WE ARE ABLE TO IMPLEMENT KNAPSACK ALGORITHM USING GREEDY APPROACH