**Batch: D2 Roll No.: 16010122323**

**Experiment No. 4**

**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/Grap hAlgor/kruskalAlgor.html**](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:**

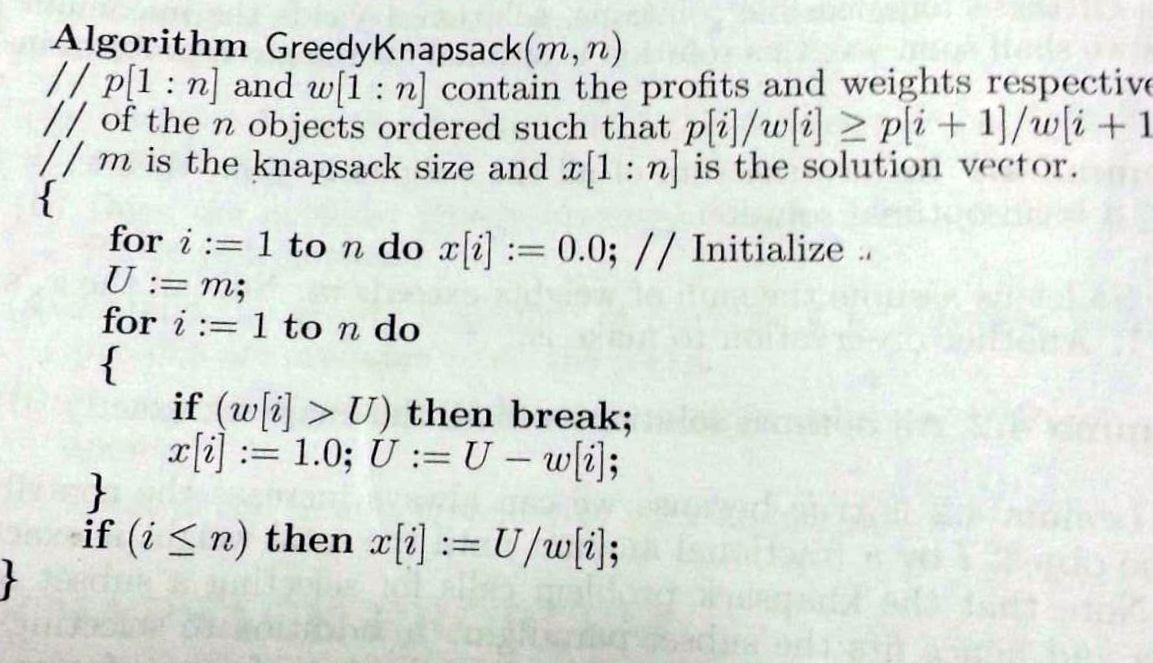
The knapsack problem represents constraint satisfaction optimization problems’ family. Based on nature of constraints, the knapsack problem can be solved with various problem saolving strategies. Typically, these problems represent resource optimization solution.

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.

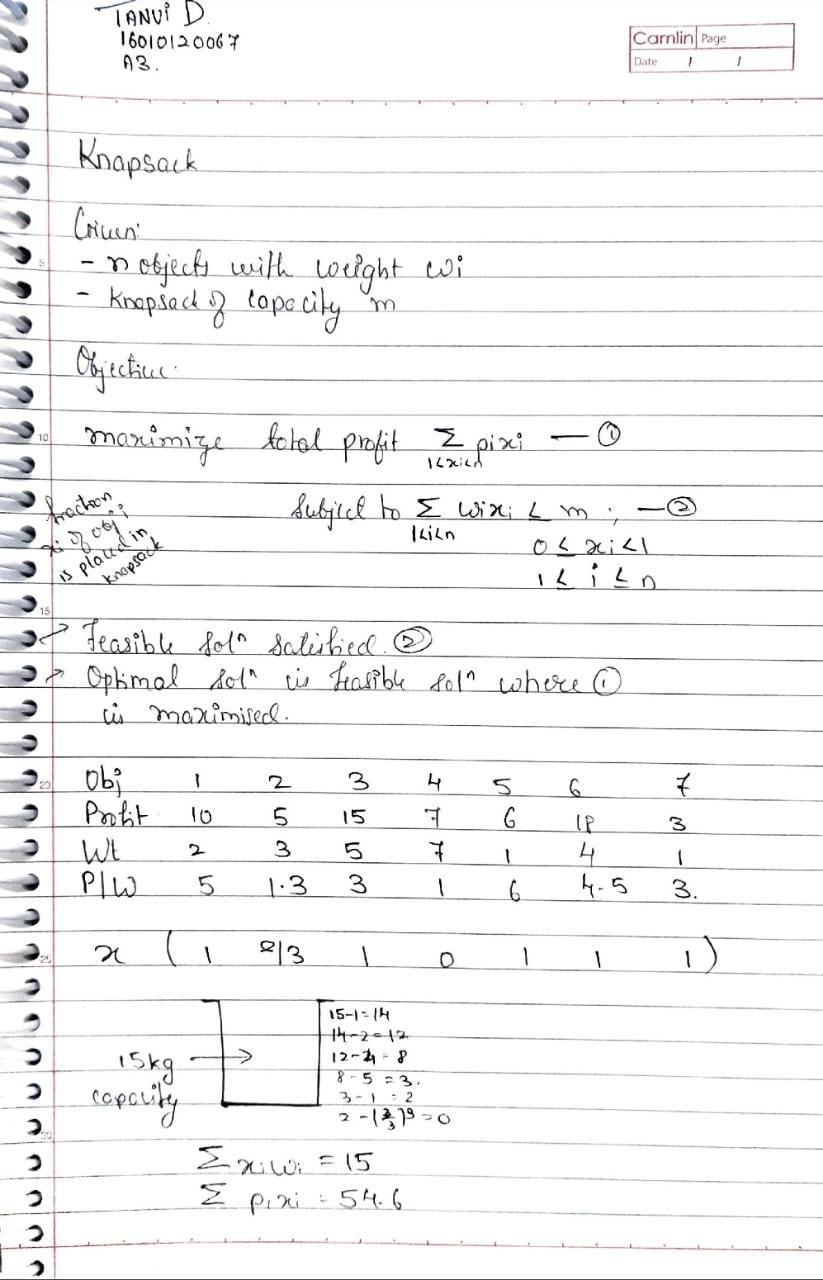
**New Concepts to be learned:**

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:**

#include <bits/stdc++.h>

using namespace std;

struct item {

int profit, weight, X;

item(int profit, int weight): profit(profit), weight(weight), X(0) {} };

bool compare(struct item a, struct item b)

{

double pwRatio1 = (double)a.profit / a.weight;

double pwRatio2 = (double)b.profit / b.weight;

return pwRatio1 > pwRatio2;

}

double knapsack(struct item items[], int capacity, int size)

{

sort(items, items + size, compare);

int currentWeight = 0;

double maxProfit = 0.0;

for (int i = 0; i < size; i++)

{

if (currentWeight + items[i].weight <= capacity)

{

currentWeight += items[i].weight;

maxProfit += items[i].profit;

items[i].X = 1;

}

else

{

int currentCapacity = capacity - currentWeight;

items[i].X = (double) currentCapacity / items[i].weight;

maxProfit += items[i].profit \* ((double) currentCapacity / items[i].weight); break;

}

}

return maxProfit;

}

int main() {

int capacity = 50;

item items[] = {{ 120, 30 },

{ 60, 20 },

{ 100, 50 }};

int size = sizeof(items) / sizeof(items[0]);

cout << "Max Profit " << knapsack(items, capacity, size) << endl;

cout << "Fraction Collected: ";

for (auto &item: items)

{

cout << item.X << " ";

}

return 0;

}

**OUTPUT:**



**Time Complexity:**

* Sorting of the array takes O(nlogn) time if we consider a merge or a quick sort
* Greedy Knapsack takes O(n) time
* Therefore, total time complexity is O(n + nlogn) **= O(nlogn) Space Complexity:**
* **O(3n + 2) = O(n)** as we only traverse through the arrays.

**Conclusion:**

Through this experiment we learnt about the fractional knapsack problem and learnt to implement it in C++.