

| **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**](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.

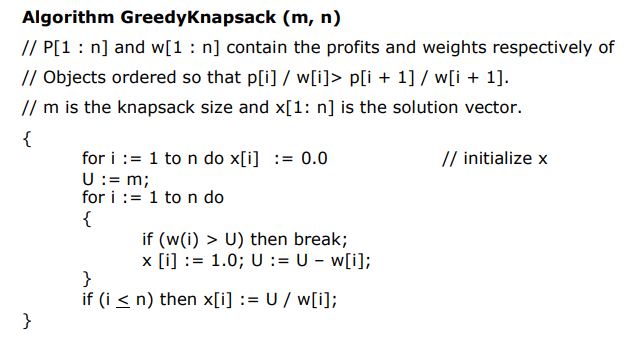


**New Concepts to be learned:**

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**

****

**#include <bits/stdc++.h>**

**using namespace std;**

**void printArray(vector<vector<int>> &v)**

**{**

**cout << "The array is\n";**

**for(int i = 0; i < v.size(); i++)**

**{**

**cout << "Item: " << v[i][0] << " Weight: " << v[i][1] << " Value: " << v[i][2] << "\n";**

**}**

**}**

**bool by\_weight(vector<int> a, vector<int> b)**

**{**

**return a[1] < b[1];**

**}**

**bool by\_value(vector<int> a, vector<int> b)**

**{**

**return a[2] > b[2];**

**}**

**bool by\_ratio(vector<int> a, vector<int> b)**

**{**

**return a[2]/a[1] > b[2]/b[1];**

**}**

**void greedy(vector<vector<int>> &v, int m)**

**{**

**vector<int> items;**

**int value = 0;**

**int a = 0;**

**for(int i = 0; i < v.size(); i++)**

**{**

**if(m >= v[i][1])**

**{**

**a++;**

**items.push\_back(v[i][0]);**

**value += v[i][2];**

**m -= v[i][1];**

**}**

**else**

**{**

**break;**

**}**

**}**

**if(m == 0 || a == v.size())**

**{**

**cout << "items taken: ";**

**for(auto &i: items)cout << i << " ";**

**cout << "\n";**

**cout << "value: " << value << "\n";**

**return;**

**}**

**float fraction = (float)m/v[a][1];**

**if(fraction != 0)**

**{**

**value += fraction \* v[a][2];**

**}**

**cout << "items taken: ";**

**for(auto &i: items)cout << i << " ";**

**cout << fraction << " of " << v[a][0];**

**cout << "\n";**

**cout << "value: " << value << "\n";**

**return;**

**}**

**int main()**

**{**

**int n;**

**cout << "Enter the number of items\n";**

**cin >> n;**

**vector<vector<int>> v(n, vector<int> (3));**

**for(int i = 0; i < n; i++)**

**{**

**cout << "Enter weight and value of item " << i + 1 << "\n";**

**v[i][0] = i + 1;**

**cin >> v[i][1];**

**cin >> v[i][2];**

**}**

**cout << "Enter the capacity of a knapsack\n";**

**int m; cin >> m;**

**cout << "Knapsack by weight\n";**

**sort(v.begin(), v.end(), by\_weight);**

**greedy(v, m);**

**cout << "Knapsack by value\n";**

**sort(v.begin(), v.end(), by\_value);**

**greedy(v, m);**

**cout << "Knapsack by ratio\n";**

**sort(v.begin(), v.end(), by\_ratio);**

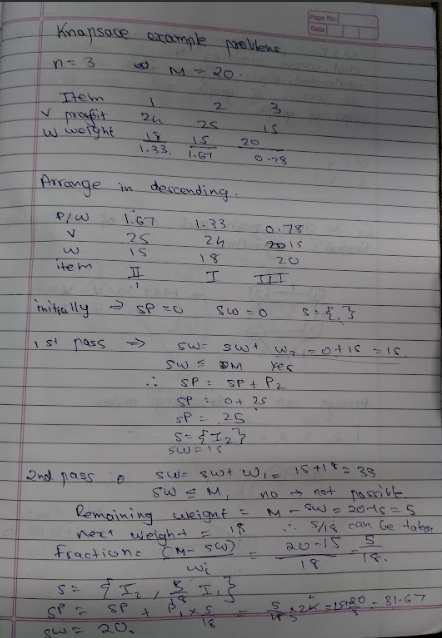
**greedy(v, m);**

**// printArray(v);**

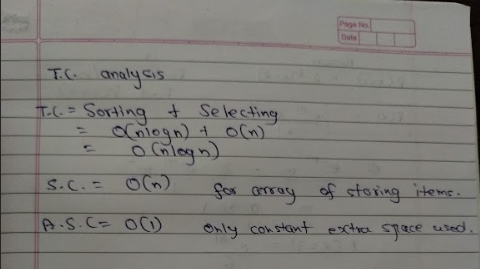
**}**



**Example: Knapsack Problem**



**Analysis of Knapsack Problem algorithm:**

****

**Conclusion:**

**We have learned about the fractional knapsack problem and know that sorting by ratio of value by weight is the most optimal.**