Greedy Method:

The greedy method is perhaps (maybe or possible) the most straight forward design technique, used to determine a feasible solution that may or may not be optimal.
Feasible solution:- Most problems have n inputs and its solution contains a subset of inputs that satisfies a given constraint(condition). Any subset that satisfies the constraint is called feasible solution.
Optimal solution: To find a feasible solution that either maximizes or minimizes a given objective function. A feasible solution that does this is called optimal solution.
The greedy method suggests that an algorithm works in stages, considering one input at a time. At each stage, a decision is made regarding whether a particular input is in an optimal solution.
Greedy algorithms neither postpone nor revise the decisions (ie., no back tracking).

Example: Kruskal's minimal spanning tree. Select an edge from a sorted list, check, decide, and never visit it again.
Application of Greedy Method:

1. Job sequencing with deadline
2. 0/1 knapsack problem
3. Minimum cost spanning trees
4. source shortest path problem

Algorithm for Greedy method

```
Algorithm Greedy(a,n)
//a[1:n] contains the n inputs.
{
Solution :=0;
For i=1 to n do
{
X:=select(a);
If Feasible(solution, x) then
Solution :=Union(solution,x);
}
Return solution;
}
```

Selection Function, that selects an input from a[] and removes it. The selected input's value is assigned to x.
Feasible Boolean-valued function that determines whether x can be included into the solution vector.
Union function that combines x with solution and updates the objective function.

Knapsack problem:

The knapsack problem or rucksack (bag) problem is a problem in combinatorial optimization: Given a set of items, each with a mass and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

There are two versions of the problems
1. 0/1 knapsack problem
2. Fractional Knapsack problem
a. Bounded Knapsack problem.
b. Unbounded Knapsack problem

Solutions to knapsack problems
Brute-force approach:-Solve the problem with a straight farward algorithm
Greedy Algorithm:- Keep taking most valuable items until maximum weight is reached or taking the largest value of eac item by calculating vi=valuei/Sizei
Dynamic Programming:- Solve each sub problem once and store their solutions in an array.

Greedy algorithm for knapsack.

```
Algorithm GreedyKnapsack(m,n)
// p[i:n] and [1:n] contain the profits and weights respectively
// if the n-objects ordered such that p[i]/w[i]>=p[i+1]/w[i+1], m size of knapsack and
x[1:n] 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];
}
If(i<=n) then x[i]:=U/w[i];
}
```