Eg:- We want to buy the best car.
For that all the brands available are checked. Checking with the features we select the best one by sorting latest released cars.

So we have own method of selection and based on the selection method we get some optimal solution.

Knapsack Problem

→ Combinational optimization problem.

→ It appears as a subproblem in many more complex mathematical models of real world.

→ According to the problem statement

- There are n items in the store.
- Weight of i^{th} item, $w_i > 0$.
- Profit for i^{th} item, $P_i > 0$
- Capacity of knapsack, w.

eg:-

→ Suppose there are 7 objects.

→ Every object have some profit associated with it.

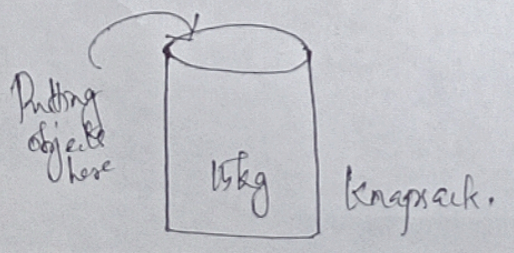
→ Also object have some weight.

→ There is a bag or knapsack here. Its capacity is 15kg.

ie, $n = 7$

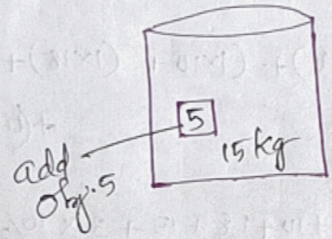
$m = 15kg$.

Object	1	2	3	4	5	6	7	
Profit (P)	10	5	15	7	6	18	3	
weight	2	3	5	7	1	4	1	



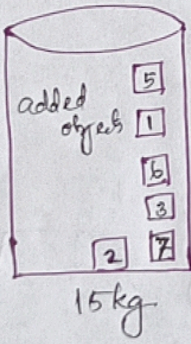
Object	1	2	3	4	5	6	7
Profit (P)	10	5	15	7	6	18	3
Weight (w)	2	3	5	7	1	4	1
P/w	5	1.3	3	1	6	4.5	3.

① So 1st we are adding 5th object
ie, we get profit 6 (obj. weight = 1)



15 − 1 = 14 kg remaining

② Next high profit = 5
for object 1 with weight 2.
Remaining weight of bag = 14 kg
So add object 1 to the bag and get the profit full as 5.



x_1 15 − 1 = 14 kg
x_2 14 − 2 = 12 kg
x_3 12 − 4 = 8 kg
x_4 8 − 5 = 3 kg.
x_5 3 − 1 = 2 kg.
$x_6 = 2/3$.

$x_1 = 1$ (1 means we are taking full kg of that particular object)

$x_2 = 1$
$x_3 = 1$
$x_4 = 1$
$x_5 = 1$
$x_6 = 2/3$
$x_7 = 0$.

Total Weight = $x_1 \times$ (Weight of obj 5) + $x_2 \times$ (Weight of obj 1) +
$x_3 \times$ (Weight of obj 6) + $x_4 \times$ (Weight of obj 3) +
$x_5 \times$ (Weight of obj 7) + $x_6 \times$ (Weight of obj 2)
+ $x_7 \times$ (Weight of obj 4)

$= (1\times1) + (1\times2) + (1\times4) + (1\times5) + (1\times1) + (2/3\times3)$
$+ (0\times7)$

$= 1+2+4+5+1+2+0 = 15\,kg.$

Total profit $= x_1 \times$ profit of object 5 +
$x_2 \times$ profit of object 1 +
⋮

$= (1\times6) + (1\times10 + (1\times18) + (1\times15) + (1\times3) + (2/3\times5)$
$+ (0\times7)$

$= 6+10+18+15+3\times10/3+0.$
$= 54.6.$

So the Constraint in the given problem

$$\sum_{i=1}^{n} x_i w_i \leq m \rightarrow$$ max^m capacity of bag.

Objective.

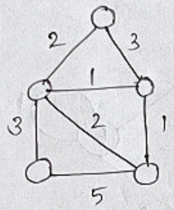
$$max \sum_{i=1}^{n} x_i P_i \text{ (max profit)}$$

Algorithm

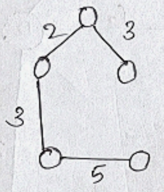
```
Greedy-fractional-knapsack (w, m, x[])
{ for i=1 to n
      do x[i] = 0
   weight = 0
   while (weight < m)
         do i = best remaining item
            if weight + wi ≤ m
               the x[i] = 1
               weight = weight + wi
         else
            x[i] = (m-weight) / wi
               weight = m.
}           return x
```

Minimal Spanning Tree.

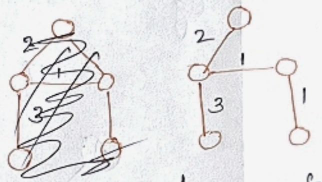
A minimum spanning tree or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with minimum possible total edge weight. i.e, It is a spanning tree whose sum of edge weights is as small as possible.



Graph



Spanning tree
Cost = 13



Minimum cost spanning tree
Cost = 7

$G = (V, E)$

In MST $G' = (V', E')$,

$V' = |V|$

& $E' = |V| - 1$

{ No: of spanning trees can be made out of a graph $= {}^{|E|}C_{|V|-1}$ − no: of cycles }

There are algorithms (Greedy method) finding out MST without finding all possible spanning trees. They are

1) Prim's algorithm
2) Kruskal's algorithm

Prim's algorithm.

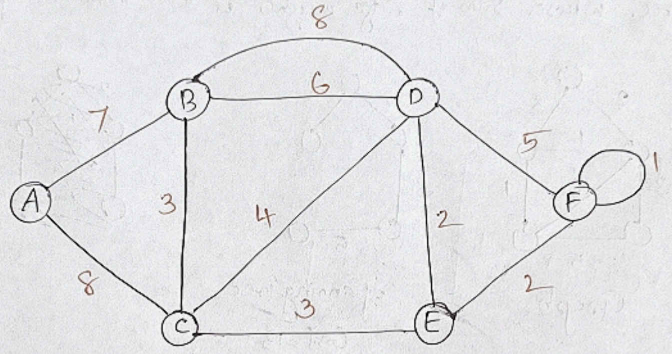
→ It is a greedy algorithm that finds MST for a connected weighted undirected graph.

→ It finds a subset of edges that form a tree that includes every vertex, where the total weight of all the edges in a tree is minimized.

Steps.

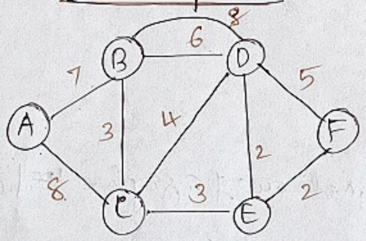
1) Initialize the MST with a vertex chosen at random (minimum).
2) Find all the edges that connect the tree to new vertices.
Find minimum & add it to the tree.
3) Keep repeating step (2) until we get a MST.

eg:-

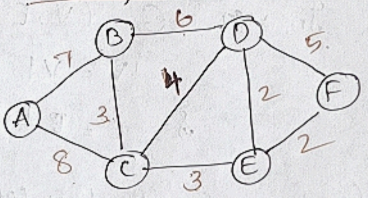


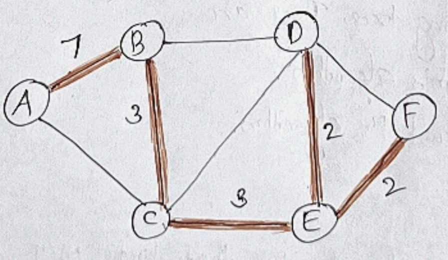
First
Eliminate
self loops &
parallel
edges.

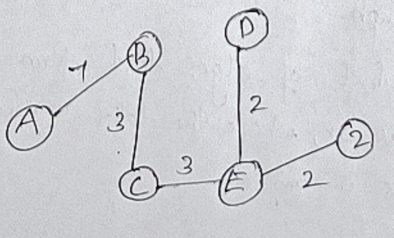
★ Remove loop.



★ Remove parallel edges.







Cost = 17