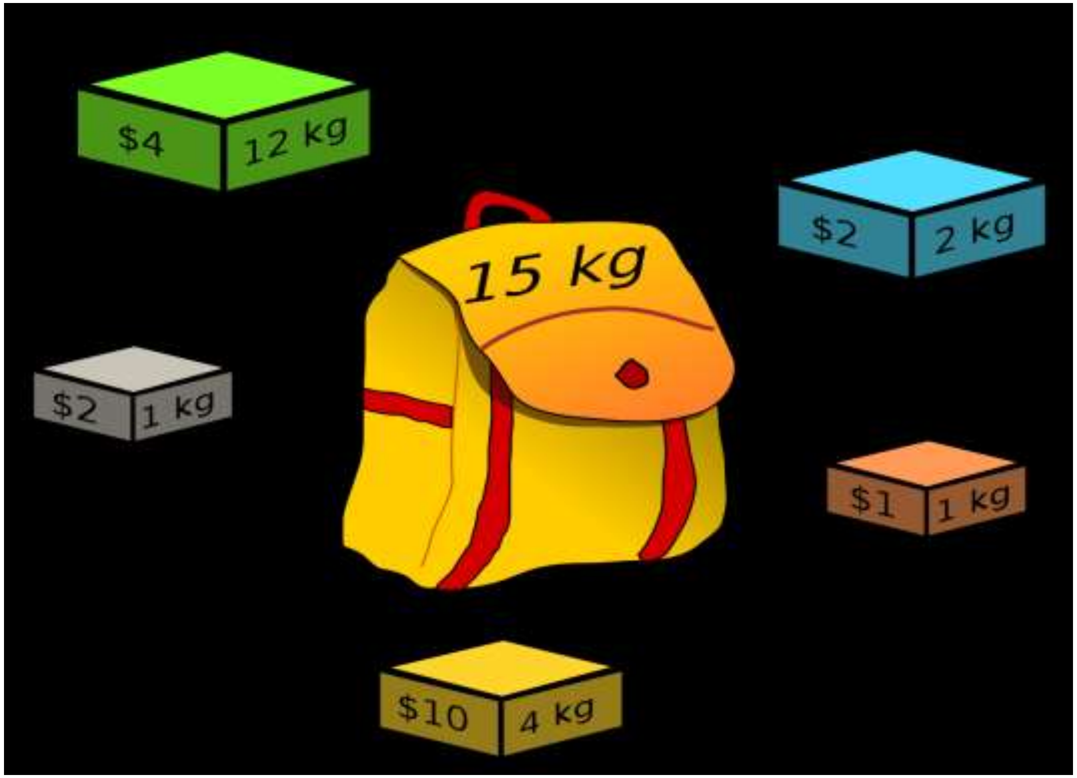
UNIT II – Brute Force and Divide and Conquer

- **Brute Force Design Technique**
  - Selection Sort
  - Bubble Sort
  - Sequential Search
  - Closest pair and Convex hull problem
  - Travelling Salesman problem
  - **Knapsack problem**
  - **Assignment problem**

Travelling Salesman problem, Knapsack problem, Assignment problem → Exhaustive Search

Knapsack problem

- Given n items of known weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack.



Weight: 100 gms
Value: 1.5 Kg Honey

1

Weight: 150 gms
Value: 3 Kg Honey

2

Weight: 500 gms
Value: ???

Weight: 1000 gms
Value: 3 Kg Honey

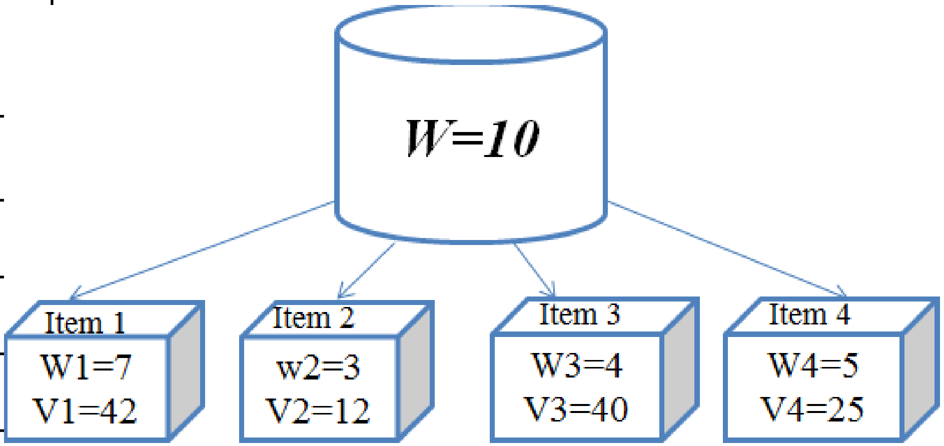
4

Weight: 350 gms
Value: 3.5 Kg Honey

3

Knapsack Problem - Example

SUBSET	Total Weight	Total Value
{null}	0	0
{1}	7	42
{2}	3	12
{3}	4	40
{4}	5	25
{1,2}	10	54
{1,3}	**11**	Not feasible
{1,4}	**12**	Not feasible
{2,3}	7	52
{2,4}	8	37
{3,4}	**9**	**65**
{1,2,3}	14	Not feasible
{2,3,4}	12	Not feasible
{1,2,4}	15	Not feasible
{1,3,4}	16	Not feasible
{1,2,3,4}	19	Not feasible



Analysis:

1. Problem size – Total no of item
2. Basic Operation
3. Count for basic Operation

No of subsets = 2^n

EX: n=4 , 16 possible subsets are available

What is the feasible solution?

Items	1	2	3	4
Weights	5	4	6	3
Values	10	40	30	50

Capacity : 10

Assignment Problem

- *n* people need to be assigned to *n* jobs, one person per job. Each person is assigned exactly one job and each job is assigned to exactly one person

P \ Jobs	Job 1	Job 2	Job 3	Job 4
Person 1	9hrs	2hrs	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

- The possibilities for allocating *n* jobs for *n* person is ***n!***
- Here n=4, 4!=24 possibilities.
- From these possibilities have to take a feasible solution.
- Small instance. When no of instances grow it is not practical.

Assignment Problem

P \ Jobs	Job 1	Job 2	Job 3	Job 4
Person 1	9hrs	2hrs	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

Possibilities of

job assignment to persons

{1,2,3,4}=9+4+1+4=18	**{2,1,3,4}=13**	{3,1,2,4}=25	{4,1,2,3}=31
{1,2,4,3}=9+4+8+9=30	{2,1,4,3}=25	{3,1,4,2}=27	{4,1,3,2}=21
{1,3,2,4}=9+3+8+4=24	{2,3,1,4}=14	{3,2,1,4}=20	{4,2,1,3}=26
{1,3,4,2}=9+3+8+6=26	{2,3,4,1}=20	{3,2,4,1}=26	{4,2,3,1}=20
{1,4,2,3}=9+7+8+9=33	{2,4,1,3}=23	{3,4,1,2}=25	{4,3,1,2}=22
{1,4,3,2}=9+7+1+6=23	{2,4,3,1}=17	{3,4,2,1}=29	{4,3,2,1}=26

Assignment Problem using
Hungarian Method

- Row Detection
- Column Detection
- Optimality Test
- Redesigning Matrix

P Jobs	Job 1	Job 2	Job 3	Job 4
Person 1	9hrs	2hrs	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

1.Row Detection

7	0	5	6
3	1	0	4
4	7	0	7
3	2	5	0

2.Column Detection

4	0	5	6
0	1	0	4
1	7	0	7
0	2	5	0

3.Optimality Test

4	**0**	5	6
0	1	0	4
1	7	**0**	7
0	2	5	**0**

J1→P2 ,J2→P1, J3→P3, J4→P4

P1,P2,P3,P4 = J2,J1,J3,P4 = **{2,1,3,4}**

Assignment Problem using

Hungarian Method - Example

Jobs ↘ Machines

5hrs	11	10	12	4
2	4	6	3	5
3	12	5	14	6
6	14	4	11	7
7	9	8	12	5