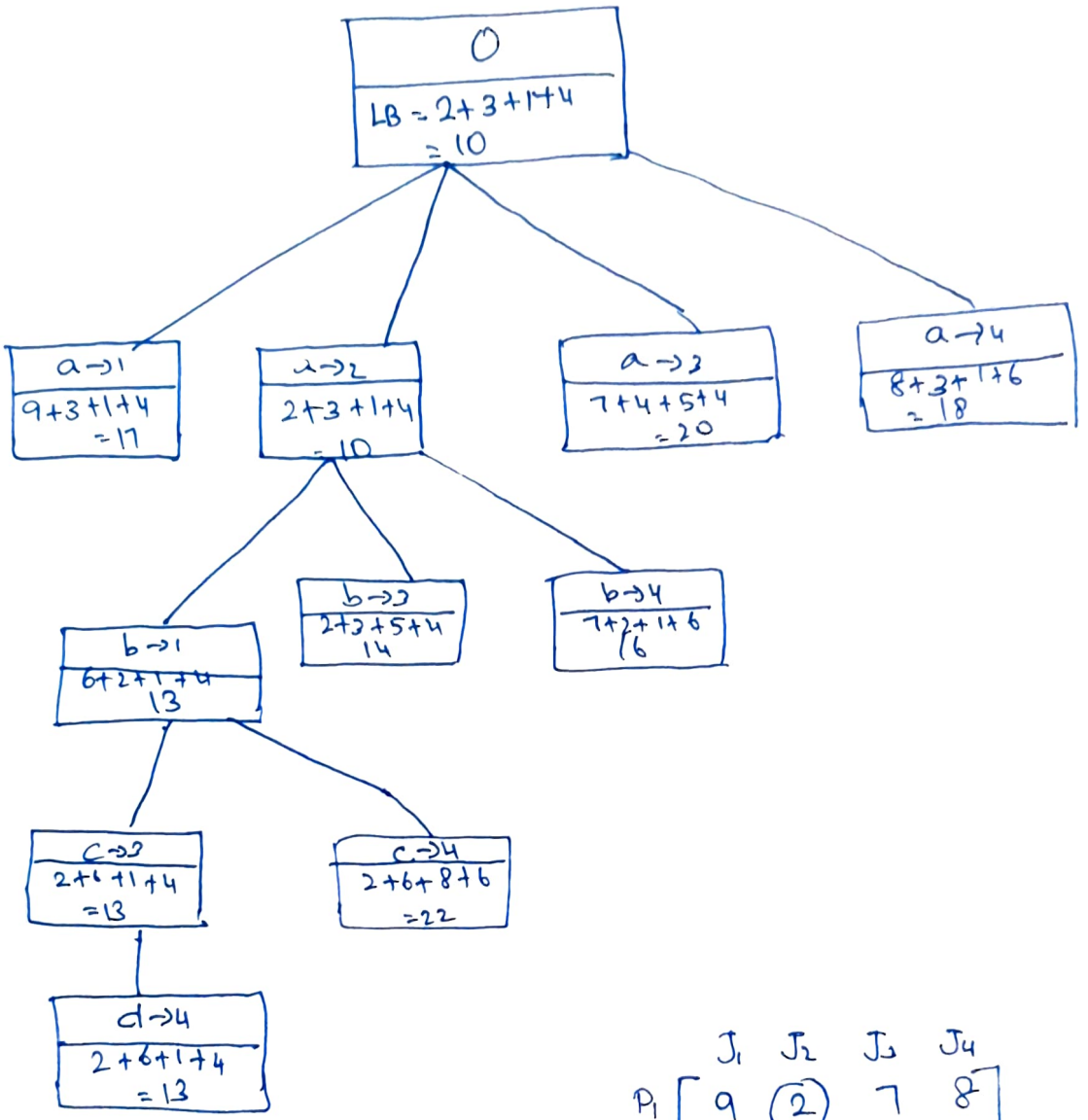
DAA Asignment 2

Senthil Sivaraman.S
CSE - 'C'

(1)

	J_1	J_2	J_3	J_4
P_1	9	2	7	8
P_2	6	4	3	7
P_3	5	8	1	8
P_4	7	6	9	4



	J_1	J_2	J_3	J_4
P_1	9	(2)	7	8
P_2	(6)	4	3	7
P_3	5	8	(1)	8
P_4	7	6	9	(4)

2. $W = 11$

item No	Weight (kgs)	Value	Value/weight
1	10	60	6
2	7	28	4
3	4	20	5
4	2	24	12

$$u_0 = V + (W - w)\,[(v_{i+1})/(w_{i+1})]$$

No : $I = 0$, $w = 0$, $V = 0$

$$u_0 = 0 + (11)(6) = 66$$

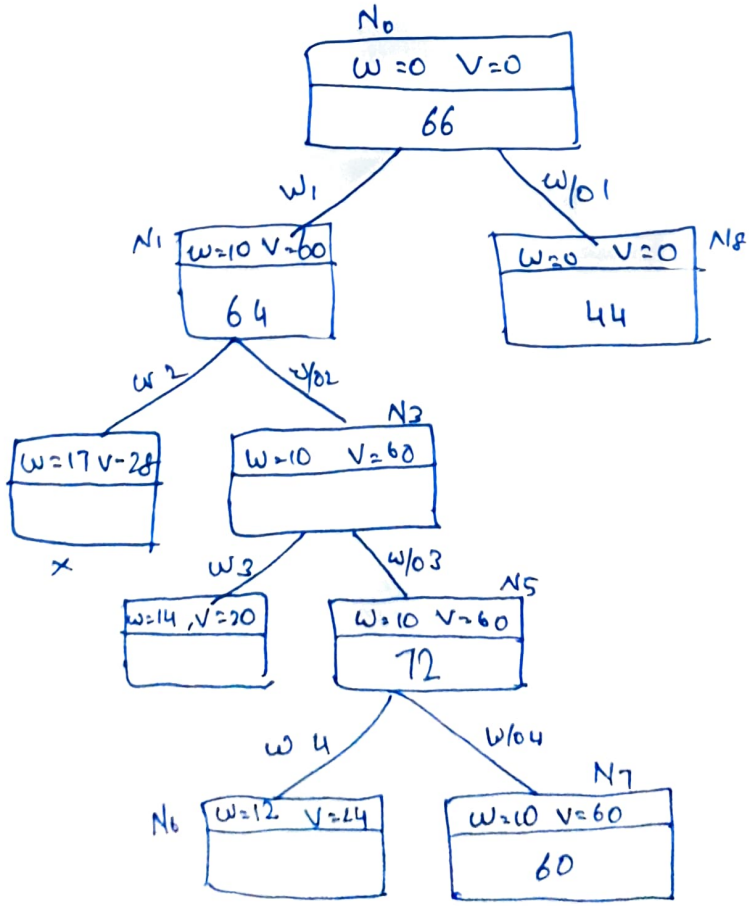
N1 : $I = 1$, $w = 10$, $V = 60$ $\Rightarrow$ $60 + (11-10)(4) = 64$

N3 : $I = 2$, $w = 10$, $V = 60$ $\Rightarrow$ $60 + 5 = 65$

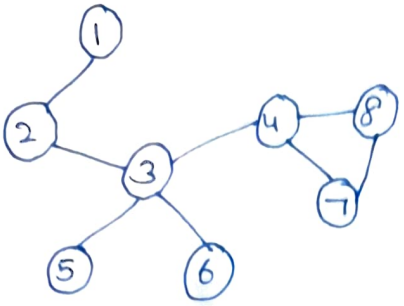
N5 : $I = 3$, $w = 10$, $V = 60$ $= 60 + 12 = 72$

N7 : $I = 4$, $w = 10$, $V = 6$ $\Rightarrow$ $60 + 0 = 60$

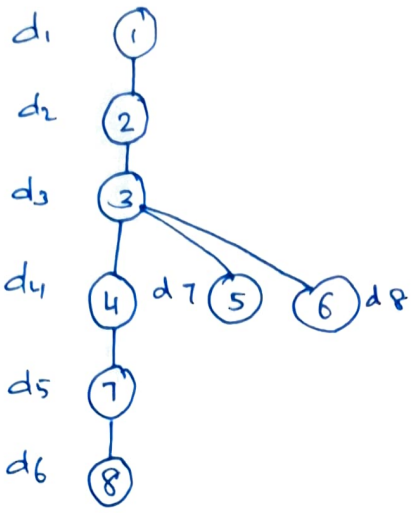
N8 : $I = 1$, $w = 0$ $V = 0$ $ub = 44$.



3)



DFS



V	1	2	3	4	5	6	7	8
d	1	2	3	4	7	8	5	6
L	1	1	1	3	3	3	4	4

To find articulation point:

root - U, V - Parent

$L[v] \geq d[u]$

$L[3] \geq d[1] \rightarrow 1 \geq 2$ ×

$L[5] \geq d[3] \rightarrow 3 \geq 3$ ✓

$L[6] \geq d[3] \rightarrow 3 \geq 3$ ✓

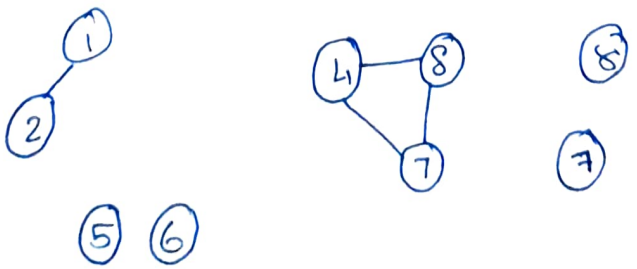
$L[3] \geq d[4] \rightarrow 1 \geq 4$ ✓

$L[7] \geq d[4] \rightarrow 4 \geq 4$ ✓

$L[8] \geq d[4] \rightarrow 4 \geq 4$ ✓

∴ Articulation pts are 3, 4.

Biconnected Components



4. Algorithm for Branch & Bound:

Branch & bound is an algorithm design Solving Combinational Optimization Problems. These problems are typically exponential in terms of time complexity & may require explore all possible Permutation in worst case. The Branch & Bound Algorithm technique solves these problems relatively quickly.

```
Connected-components (G)
{
  for each vertex V∈N
      flag [V] = -1
      Count = 0
  for V←0 to N
  {   if (flag [V] = -1)
      {   DFS (V, flag)
          Count++;
      }
  }
  Print count;
}
DFS (V, flag)
{
  flg [V] = 1;
    Print V;
    for each adj node U of V
      if (flag [U] == -1)
          DFS (U, flag);
}
```

5) NP-hard and NP Complete

Polynomial time ← Exponential Time

Linear Search - n
Binary Search - log n

0/1 Knapsack - 2^n
Traveling S.P - 2^n.

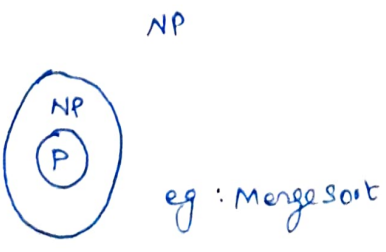
Non-deterministic Algorithm:

```
Algorithm NSearch (A, n, key)
{
    J ← Choice();
    if key = A[J]  →1
    {
        write(J)
        Success()  →1
    }
    write(0)
    Failure();  →1
}                    O(1)
```

P → Deterministic Algorithm for Polynomial time solving Prob of linear search & binary search etc.

Non-deterministic Algorithm for exponential time

Eg: 0/1 Knapsack, Hamilton etc..



eg: Merge sort

P is Subset of NP

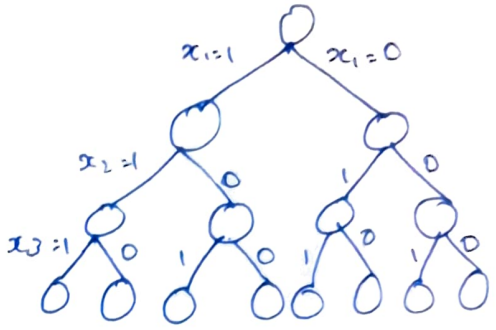
5) To check similarity b/w Exponent time solving prob we need base problem like satisfaction 2^n CNF - satisfaction

$$x_i = \{x_1, x_2, x_3\}$$

$$CNF = \underbrace{(x_1 \vee \bar{x_2} \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x_1} \vee x_2 \vee \bar{x_3})}_{C_2}$$

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$2^3 \rightarrow 2^n$



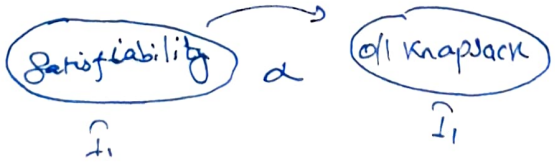
0/1 Knapsack

$P = \{10, 8, 12\}$ $n=3$ $m=8$

$w = \{5, 4, 3\}$ $x_i = \{0|1, 0|1, 0|1\}$ //Therefore knapsack also get 2^n

NP - hard:

Reduction



Satisfiability is NP hard. & if it solves any other problem then it also becomes NP- hard.

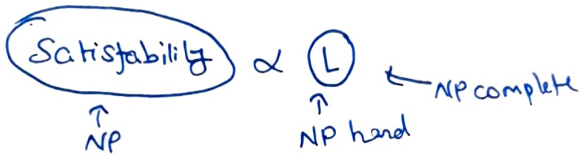
Both problems has different formulae I_1 & I_2 if I_1 is reducable to I_2. The I_2 problem becomes NP-hard

if ~~I_1~~ $I_2 \propto I_3$ (it means I_2 reduces I_3) I_3 also becomes NP-hard

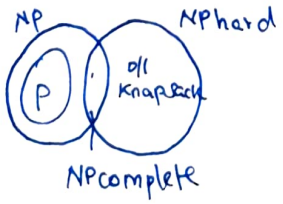
To convert formula I_1 to I_2 it takes polynomial time

NP - complete

Satisfability → NP hard
↘ NP complete



If the problem has NP-hard as well as non- Deterministic Alg. then it is known to be NP complete.



P → Deterministic Alg NP → Non Deterministic Alg

if we are sure that NP- non deterministic Algorithm Converted or prove to be P- Deterministic Alg. in future then the research will be successful

Eg: cook's

$P \subseteq NP$

$P = NP$