ASSIGNMENT - 4

MADHAN. T
B008149116

1 a) NO PRUNING

So to calculate numbers of nodes in space tree when no pruning is

$$(2^{n+1} - 1)$$

Here $n = 5$,

$$(2^{5+1} - 1) = 63$$

b) PRUNED STATE SPACE TREE.

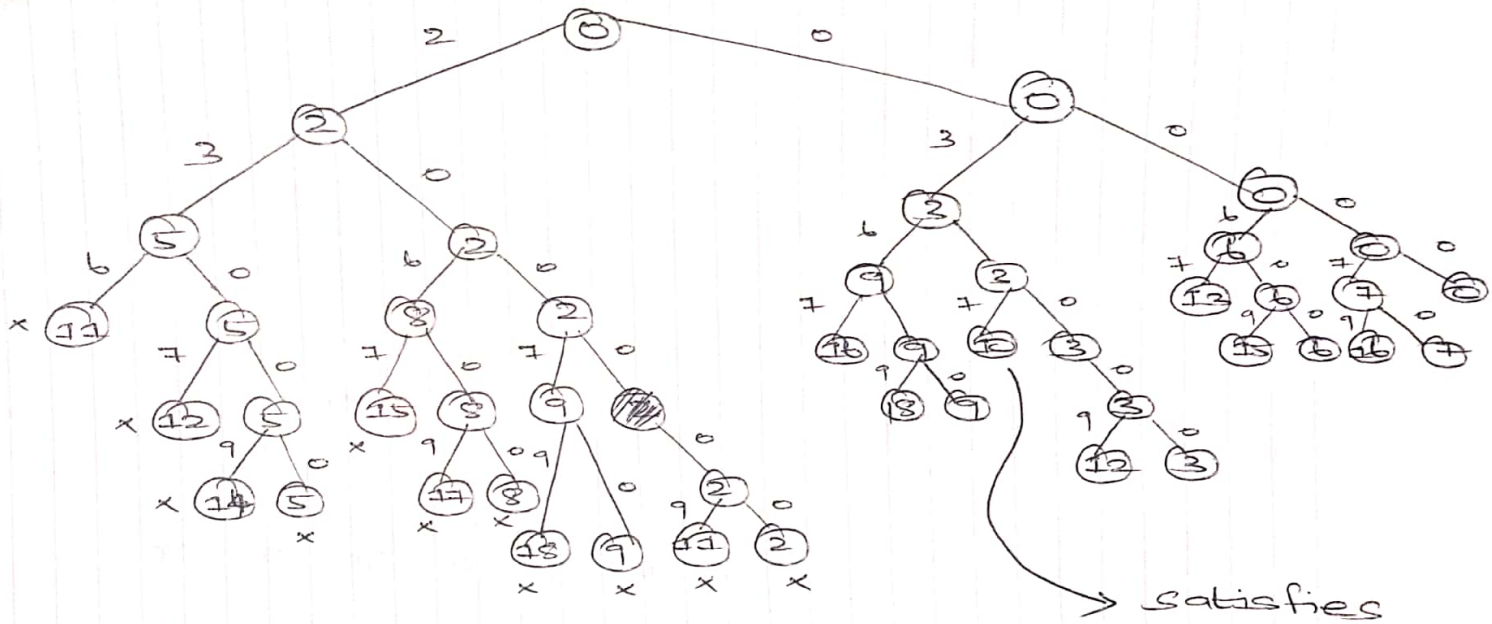
The following diag shows the pruned state space tree by tracking it using Back track algorithm

{ 3, 9, 2, 6, 7 }

sum = 10

Ascending order will be

{ 2, 3, 6, 7, 9 }



x ⇒ doesn't satisfy

↗ ⇒ satisfy

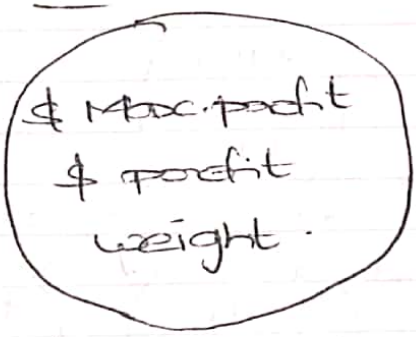
Total no of nodes = 45.

2. Given

i) size = 28
n = 3

Items	Profit	Weight
1	$ 22	8
2	$ 110	20
3	$ 50	5

CALCULATIONS



is zero // Initialy

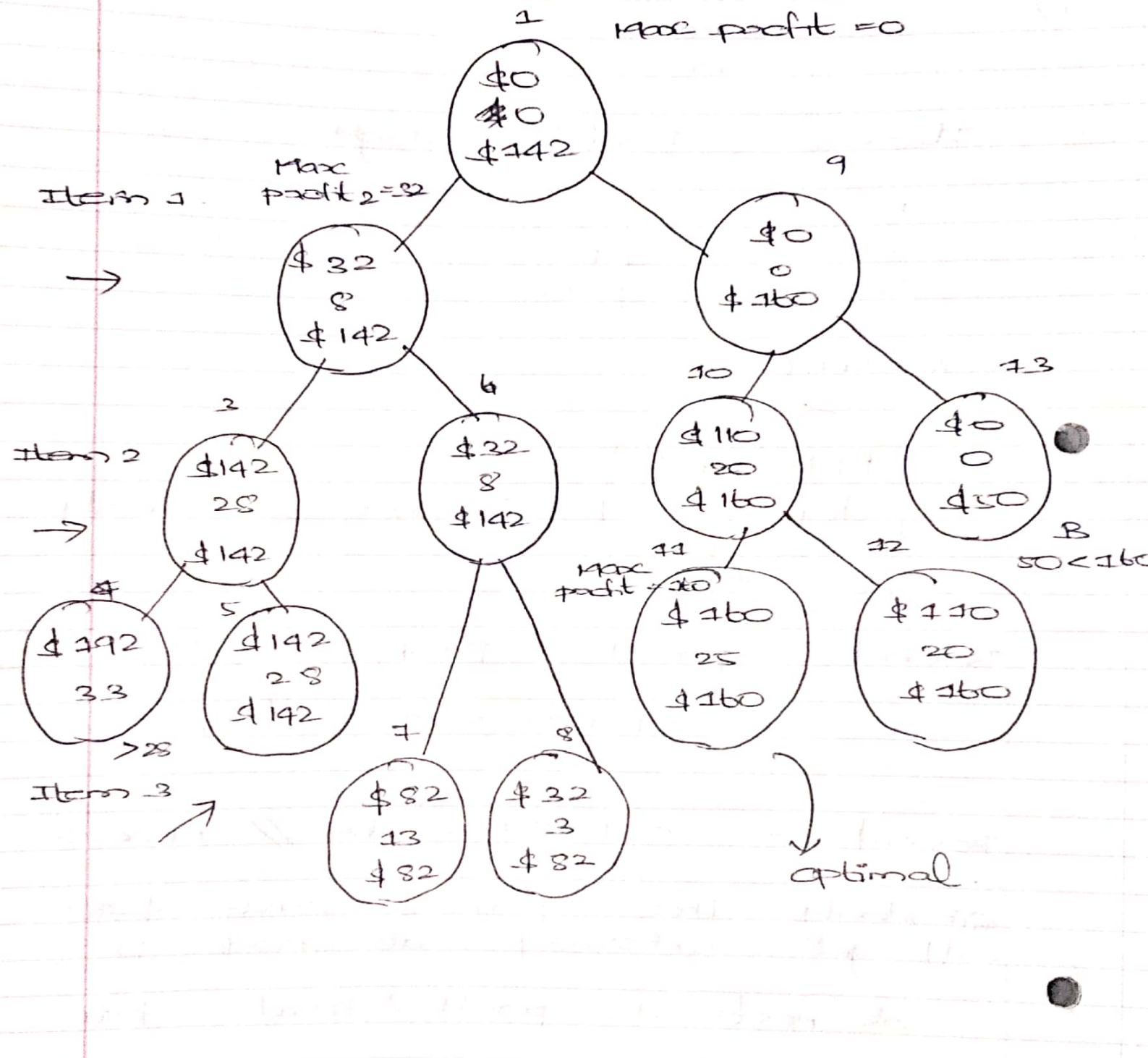
Bound = profit + P1 + P2 + (C − 28) * $\frac{P3}{W3}$

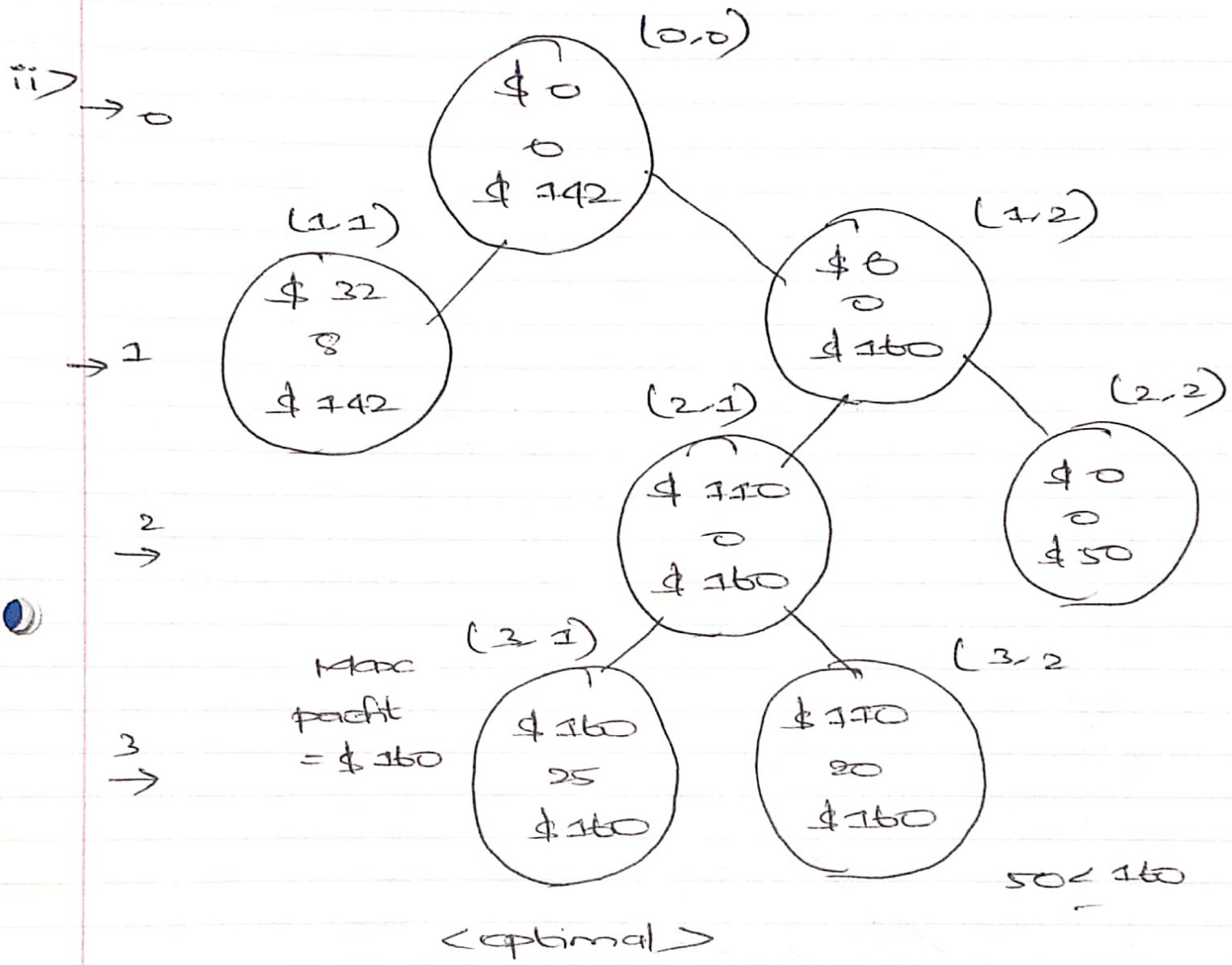
= 22 + 110 = 142. // Node 1

Bound = 50 + 110 = 160 // Node 2

Similarly the optimal value $ 160 will get obtained at node 11.

at node 11 profit & Bound = $ 160.





3.a) The given problem belongs to class P, which means it can be solved in polynomial time.

→ The problem consists of two parts, first sort the array, next is to compose the 5000th element with 100000.

→ The sorting can be done in polynomial time using common sorting algorithm [O(nlogn)].

→ comparision can be done in constant time.

→ so the decision can be done by average polynomial time of O(nlogn).

ALGORITHM

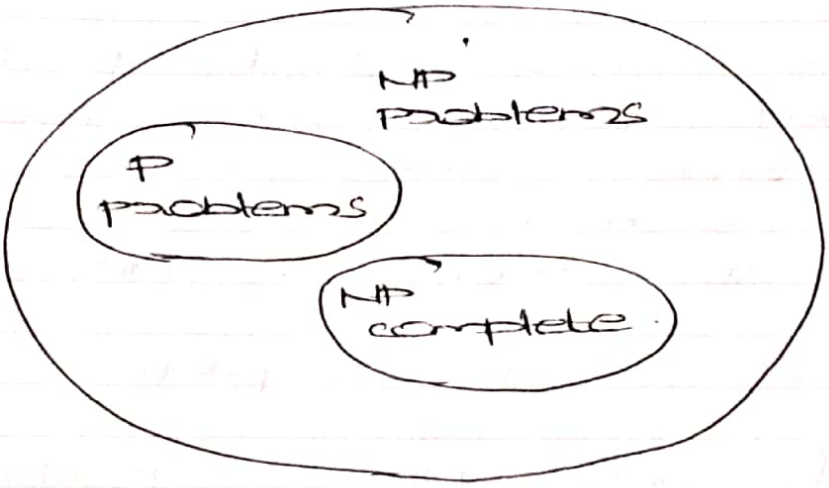
```
sort A [ ]
if A[5000] > 100000
   return "Greater"
else if A[5000] < 100000
   return "lesser"
else
   return "equal".
```

b) Yes

→ P complexity class is a subset of NP class.

→ So, the problem belongs to P class as also in NP class.

c) PROBABLY IN BOTH P & NP.



Above diag, represents the class inclusion in different complexity classes.

→ Both P & NPC belongs to NP class. but the Mutual inclusion is not clear.

4) a) P → Prob can be solved in poly time.

NPC → The hardest problem

NP → Prob can be solved in non deterministic poly. time

NP Hard → prob is as hard as NPC.

cc <p NPC [x <p y],

which means that prob x can be reduced to another prob y in poly. time means x is as hard as prob y. Therefore cc is atleast as hard as prob in NP complete.

∴ cc is NP-complete.

Also if any NP-C prob in P, then P = NP

If prob x in NP-C, then x is solvable in polynomial time if P=NP.

Proof

If P = NP, the x can be solved in polytime. suppose x is solvable in polytime, let y be prob

in NP. We can solve Y in polytime. reduce it to X. Therefore every problem in NP has a polytime algorithm and P = NP.

A problem R is NP-complete if

1) R is NP
2) Every NP prob P reduces to R.

Thus from above instances. We can say that the problem CC is in NP.

b) No, professor Green shouldn't get prize for showing P = NP.

5. a) Given

professor showed a reduction of satisfiablity to the prob YY in poly time algorithm.

→ Reduction of any prob from A to B means all instances can be converted to B in poly. time not vice versa, which means if B is reduced from A, then it atleast as hard to solve as A.

→ Since satisfiability is an NP hard problem, hence we can say that YY prob is also NP hard.

→ As the prof showed a polytime verification algo. hence the prob is also a NP prob.

thus YY is a NP and NP hard.

ie: YY is NP-c problem.

b) As YY is NP- complete, hence it is a harder problem. so if any algorithm solves YY in poly. time, then the algo can also solve less toughes problems of NP class.

→ Now if the prof finds a poly time solution to YY problem then same transformation to this same algorithm can solve other difficult problems.

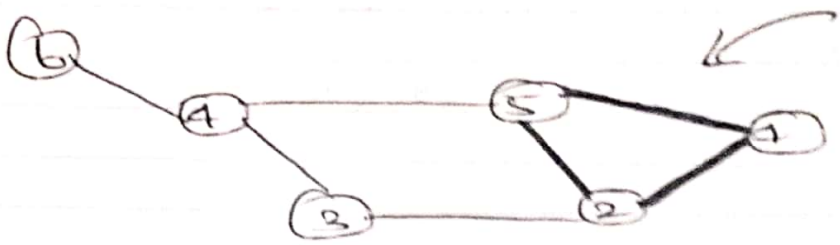
→ thus we can argue that all problems in NP class is thus solvable in polytime and we can say P = NP!

Hence proved, the prof should get the each prize.

b) Given

The problem is the decision clique problem to determine whether a graph G contains a clique of atleast a given size k.

This is a NP- complete problem.



professor Basiletti's algorithm takes $O(n^k)$ for this problem, which is polynomial time algorithm

We know $P < NP$.

In this case professor has showed $P = NP$.

7) Sum of subset is a decision prob.

→ It takes exponential time to find particular sum of subset. so it is a NP complete problem.

It can be divided into two subset problems.

* include the last element, recur for n = n-1, sum = set[n-1]

* Exclude the last element, recur for n = n-1.

If any of above returns true, then return true.

Therefore it is a NP-complete problem.

8) KNAPSACK ALGO

Consider knapsack algorithm with two i/p with one as an array and other as integer.

The array consists of n items with each items having a weight and index value. Let's say the maximum weight is W.

Let's consider there are 5 items and max weight is 3. (cout).

$$T(n) = O(n * W) = O(3 \times 5) = O(15)$$

If the array size is doubled then,

$$T(n) = O(10 \times 3) = O(30)$$

But when weight W is doubled, then its binary bit's gets doubled. (8 bits long)

Now, $T(n) = O(n * W) = O(5 \times 40) = O(200)$

Thus due to exponential increase, it is a NP-complete problem.

9.

a) It is a NP, this problem cannot be solved in a poly. time because for the factorials of very large number the problem would require exponential amount of time to solve.

b) It is a P, this can be solved in poly time. Although it is a complex problem but the careful scheduling might lead to the solution of the problem in poly time.

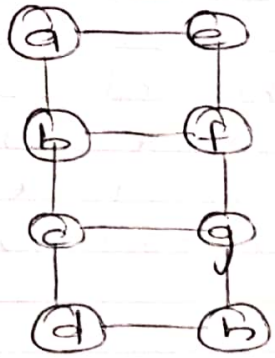
c) It is NP, since the wall is infinity long and it is midnight, either the agent would be able to find the door right away or he/she can keep looking for all their life.

10

a) chromatic Number:

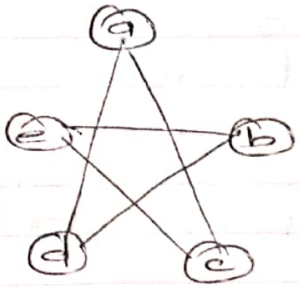
→ the smallest number of colours that needed to be assigned to the graph's vertices so that no two adjacent vertices are assigned same colour.

i]



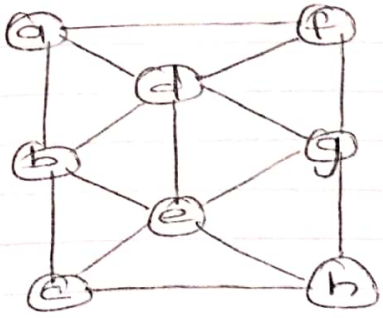
chromatic Number : 2

ii)



chromatic Number : 3

iii)



chromatic Number : 4.

b) The graph which is 2-colourable is a bipartite graph. So in this problem we have to check whether a graph is Bipartite or not, using a poly time algorithm.

ALGORITHM:

To find a graph is Bipartite using BFS.

→ set RED colour to source vertex (putting into set U)

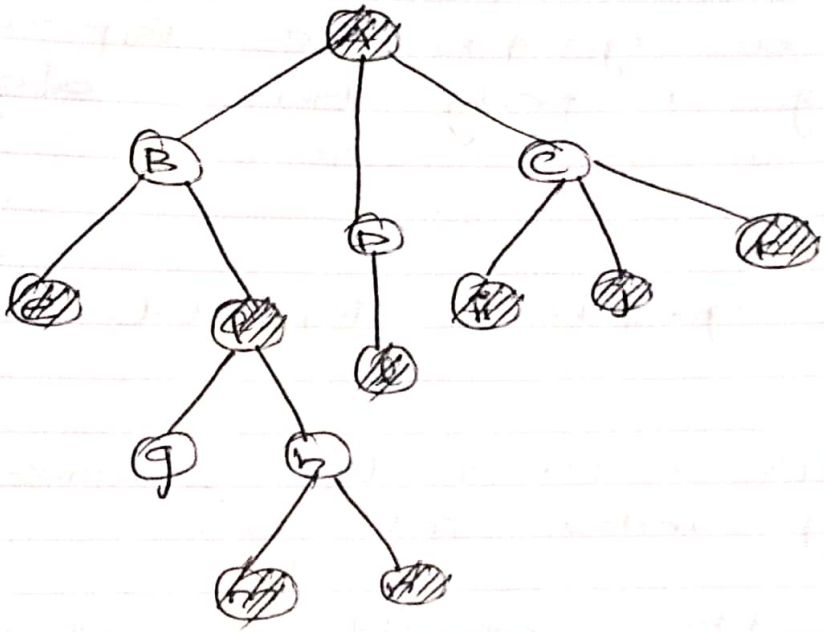
→ colour the neighbour vertex with BLUE colour. (putting into set V)

→ Then colour all the neighbour with RED (into set U)

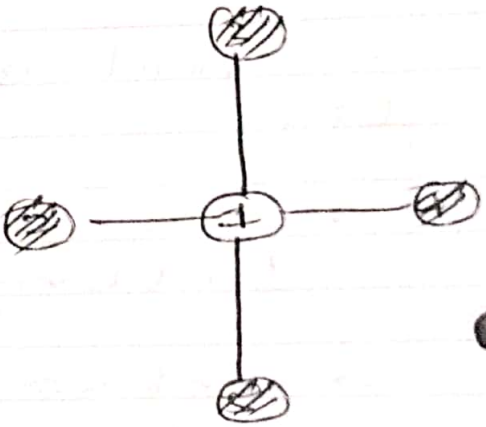
→ using the same constraints colour the remaining vertex.

→ While assigning colours, if we find a neighbour vertex with same colour then it cannot be coloured with 2 vertices (not Bipartite).

Example



(i)



(ii)