Backtracking

Unit-3 - II

Introduction :

Backtracking is an algorithmic Technique for solving the problems recursively. This method is used in many problems which deal with searching for set of solutions by satisfying some constraints.

In Backtracking applications the desired solution is expressed as 'n' tupples of $x_1, x_2, \ldots \ldots x_n$.

All the solutions solved by using backtracking satisfy two constraints namely.

(1) Explicit constraints and ..

(2) Implicit constraints -

(1). Explicit constraints :

These constraints are the rules which restrict each x_i to take the values from the given set only..

(2). Implicit constraints :

These constraints determines which of the tupples in solution space satisfy the criterian function.

BACKTRACKING

Applications of Backtracking :

Various applications based on backtracking are

(1). N-Queen's problem (4-Queens & 8-Queen's problem)
(2). Sum of subsets problem
(3). Graph colouring problem
(4). Hamiltonian cycle.
(5) Knapsack problem.

(1). **N-Queen's problem** : This problem is of two types namely 4-Queen's problem and 8-Queen's problem.

4-Queen's problem :

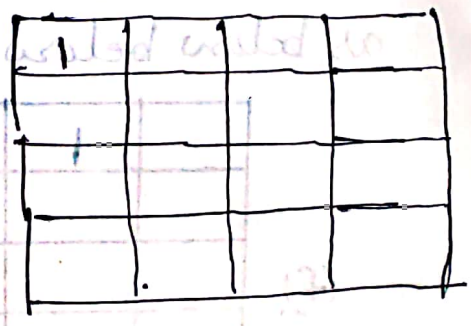
Consider an 4X4 chess board. let there are 4Queens. These 4-Queens are to be placed on the 4X4 chess board so that no two queen's are on the same row, same column & diagonal.

The explicit constraints are 4-Queens are to be placed on 4X4 chess board in 4^4 ways.

The implict constraints are no two Queens are in the same same row or column or diagonal.

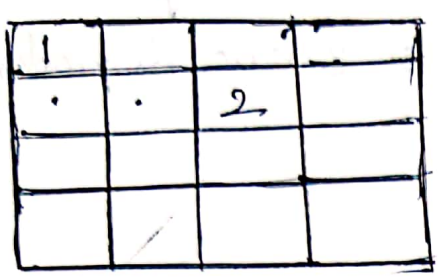
Let $\{x_1, x_2, x_3, x_4\}$ be the solution vector where x_i column number on which the Queen 'i' is placed.

First Queen '1' is placed in first row and first column.

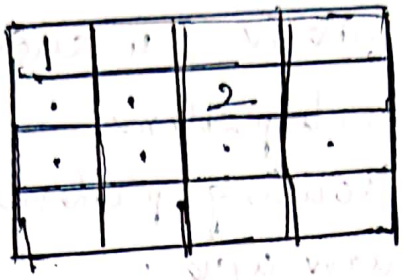


(a).

The second Queen is not placed in 1st row and 2nd column It should be placed in 2nd row and in 2nd, 3rd & 4th column. If we place in 2nd column both will be in same diagonal so place it in 3rd column.

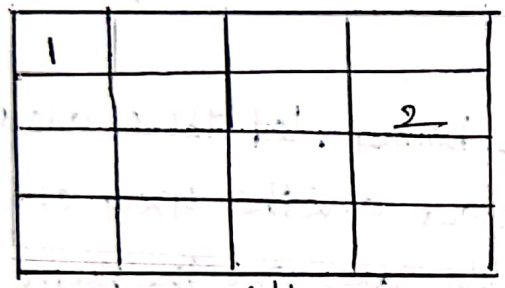


(b)

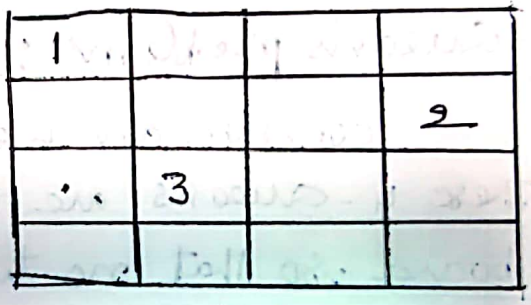


(c)

So we are unable to place third Queen in 3rd row so go back to second Queen and place it in 2nd row. 4th column and then third Queen is placed in 3rd row, 2nd colu



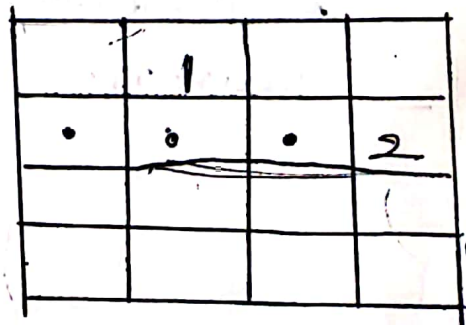
(d)



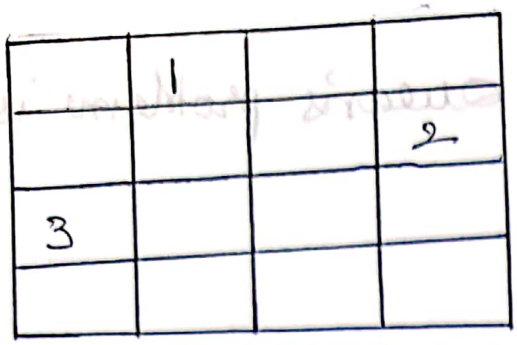
(e)

Now ~~4th~~ Fourth Queen should be placed in 4th row and 3rd column but there will be a diagonal. Then third Queen so go back and remove third Queen and place it in the next column. But it is not possible so move back to second Queen and remove it to next column. but it is not possible. so go back to third Queen and move it to next column. It can be illustrated as below below.

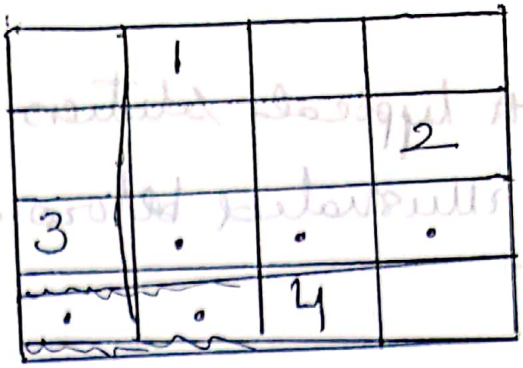
(f)



(g)



(h)



(i).

8-Queen's problem :-

In 8-Queen's problem, the 8-Queen's are to be placed on the 8×8 chess board so that no two Queen's are on the same row, same column & diagonal.

The explicit constraints are 8 Queen's are to be placed on 8×8 chess board in 8^8 ways.

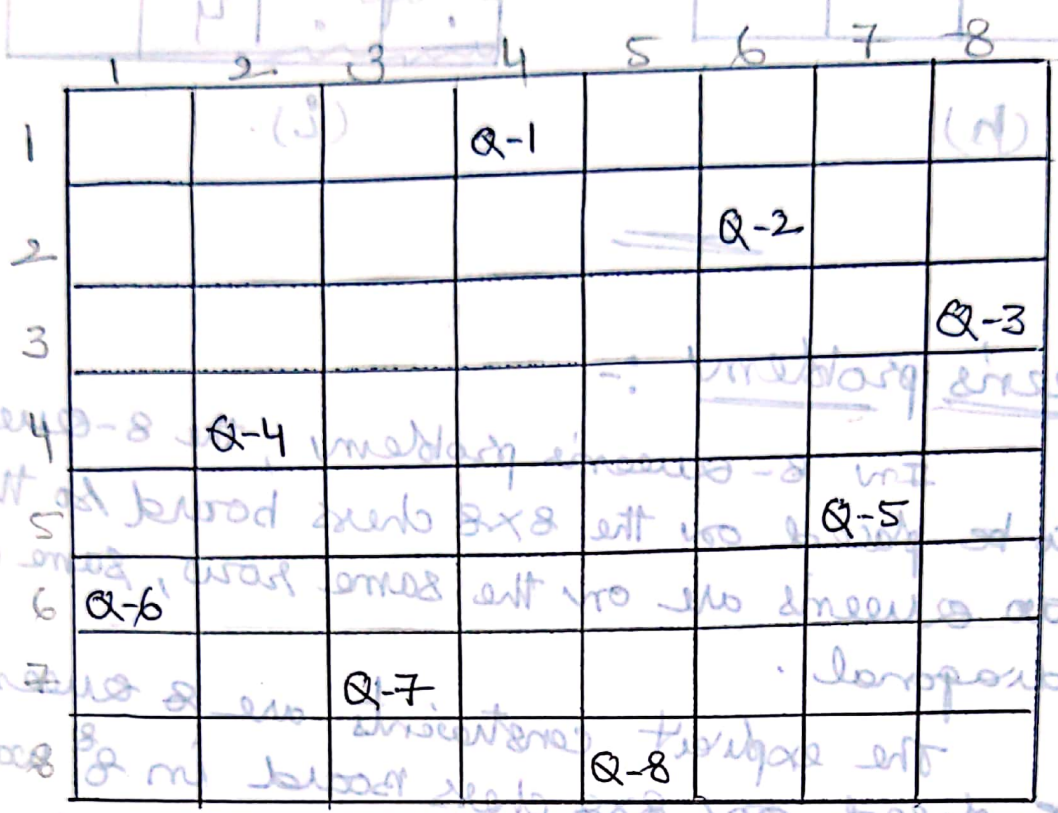
The implicit constraints are no two Queen's are in the same row, or column or diagonal.

Let $\{x_1, x_2, \ldots x_8\}$ represents a solution in which x_i is the column number on which the Queen 'i' is placed in the same column.

Let A[1:8, 1:8] be the two dimensional array representing the squares of the chess board. The squares are numbered 1 through 8. Each Queen must be on a different row.

Suppose [i, j] and [k, l] are the two positions for two Queens. They are on the same diagonal if $|j-l| = |i-k|$.

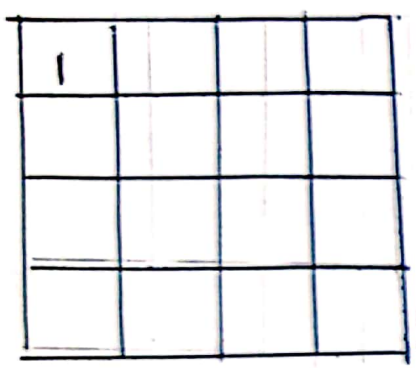
A typical solution to 8-queen's problem is as illustrated below.



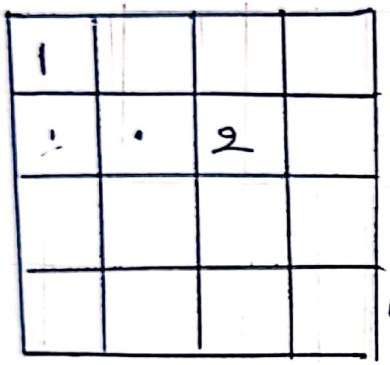
The solution of 8-queen's problem is

$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\} = \{4, 6, 8, 2, 7, 1, 3, 5\}$

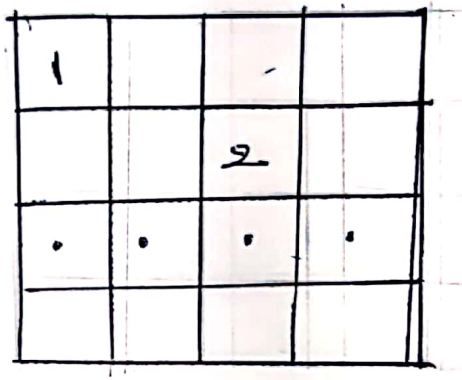
4-Queen's problem



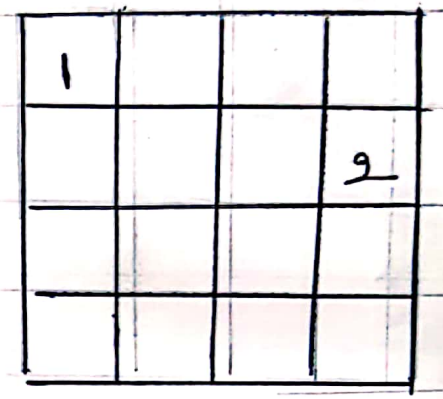
(a)



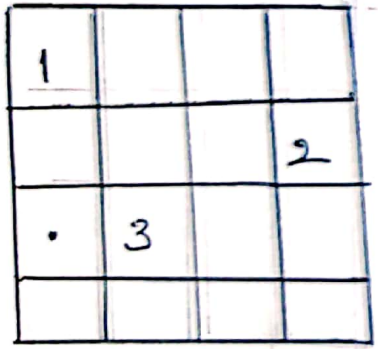
(b)



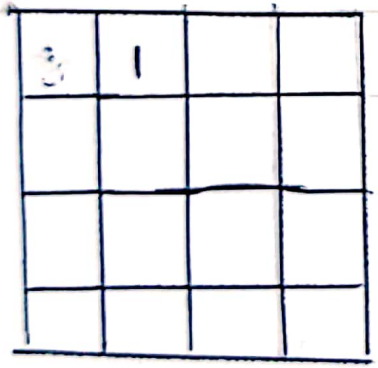
(c)



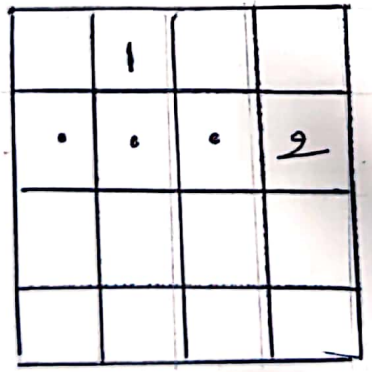
(d)



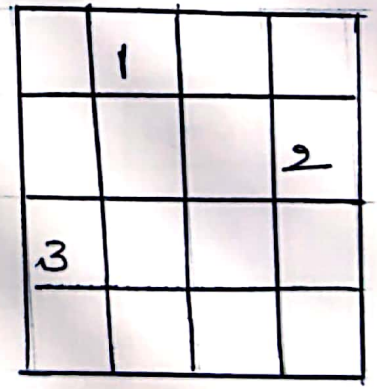
(e)



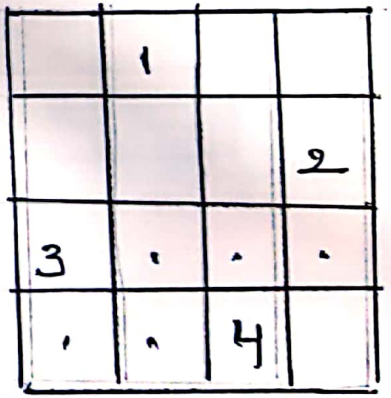
(f)



(g)



(h)



(i)

4- Queen's problem

	1	2	3	4
1		Q-1		
2				Q-2
3	Q-3			
4			Q-4	

The solution is

$\{x_1, x_2, x_3, x_4\} = \{2, 4, 1, 3\}$

8- Queen's problem

	1	2	3	4	5	6	7	8
1				Q-1				
2						Q-2		
3								Q-3
4		Q-4						
5							Q-5	
6	Q-6							
7			Q-7					
8					Q-8			

The solution is

$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$

$= \{4, 6, 8, 2, 7, 1, 3, 5\}$

Sum of subsets problem

oblem statement :

Let $S = \{s_1, \ldots, s_n\}$ be a set 'n' +ve integers then sum of ubsets 's' is equal to +ve integer ' i.e $s_1 \leq s_2 \leq \ldots \leq s_n$.

gorithm :

ep 1 : start with an empty set.

tep 2 : Add next element to the list.

tep 3 : if sum of subsets = d then this is the solution.

step 4 : if sum Exceeds with 'd' then backtracks.

step 5 : if subsets sum not Exceeds with 'd' then repeat step ②.

step 6 : if visited all the elements, backtracking is not possible, then stop with out the solution

Example :

Consider a set 'S' = {5, 10, 12, 13, 15, 18} and d = 30. Solve it for obtaining sum of subsets.

Elements	Sum 'S'	compare 'S' with 'd'	comment
Empty set = {}	sum = 0	-	
5	5	-	Then add next element
5, 10	15	15 < 30	Add next element
5, 10, 12	27	27 < 30	Add next element
5, 10, 12, 13	40	40 > 30	sum exceeds then backtrack
5, 10, 12, 15	42	42 > 30	" " "
5, 10, 12, 18	45	45 > 30	" " "
5, 10			
5, 10, 13	28	28 < 30	Add next element
5, 10, 13, 15	33	33 > 30	sum Exceeds then backtracks
5, 10			
5, 10, 15	30	30 = 30	solution obtained

∴ Solution obtained is sum = 30 = d.

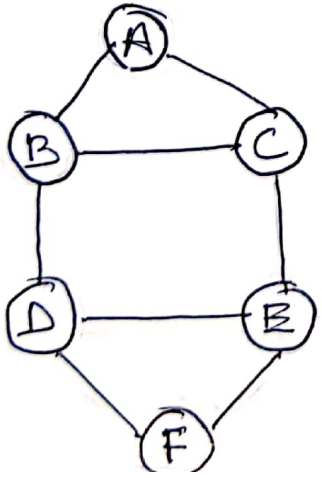
graph coloring problem

problem statement :

graph coloring is a problem of coloring each vertex in a graph in such a way that no two adjacent vertices have the same color and yet m-colors are used. This problem is called as m-coloring problem.

The least no. of colors needed to color the graph is called its chromatic number.

For example consider the following graph, it require three colors to color the graph. The following steps are used to solve the problem



step (1) : The graph 'G' has the vertices A to F. The colors used here are Red, Green and Blue.

step (2) :

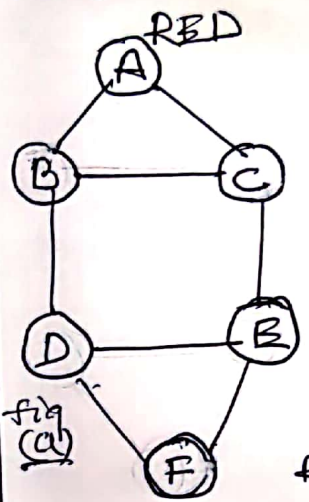


fig (a)

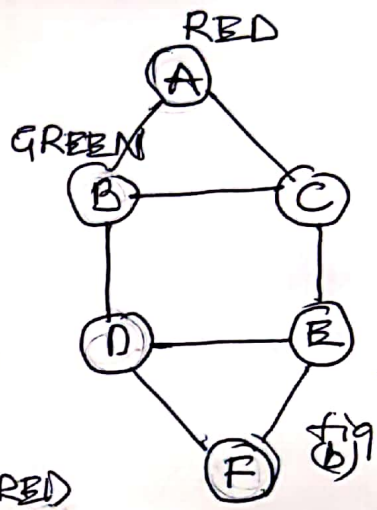


fig (b)

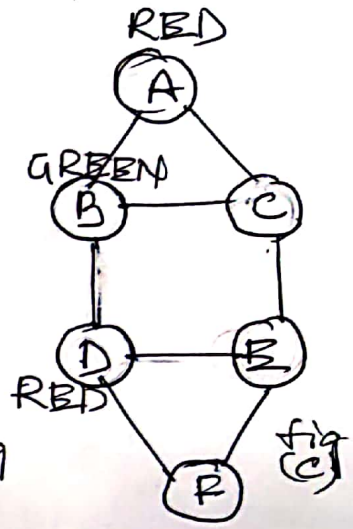


fig (c)

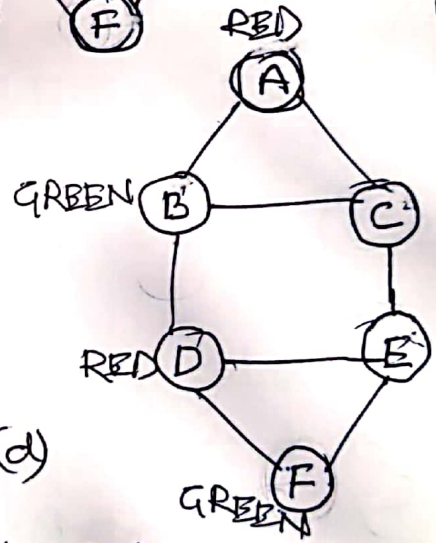


fig (d)

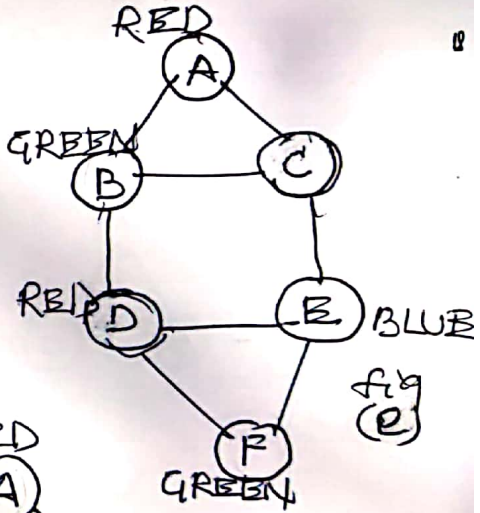
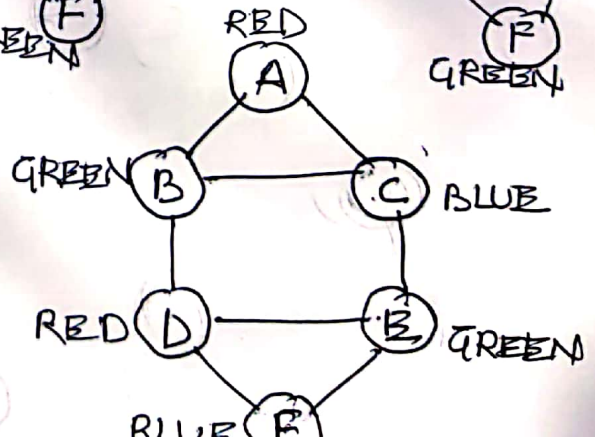


fig (e)

step (3) :

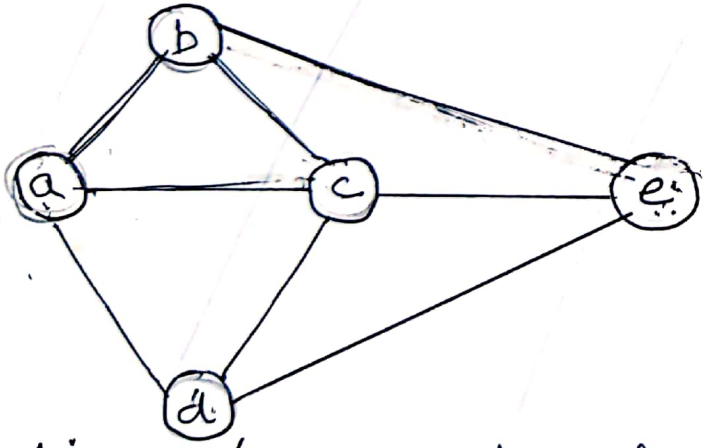


Hamiltanian cycle :

Definition :

Given an undirected graph of nodes x and y Then find a path from x to y visiting each node in the graph exactly once.

Example : consider a graph 'G' Then the Hamiltanian cycle is a-b-c-e-d-a.



This problem can be solved by using Backtracking approach. The state space tree is generated to find all the Hamiltanian cycles in the graph.

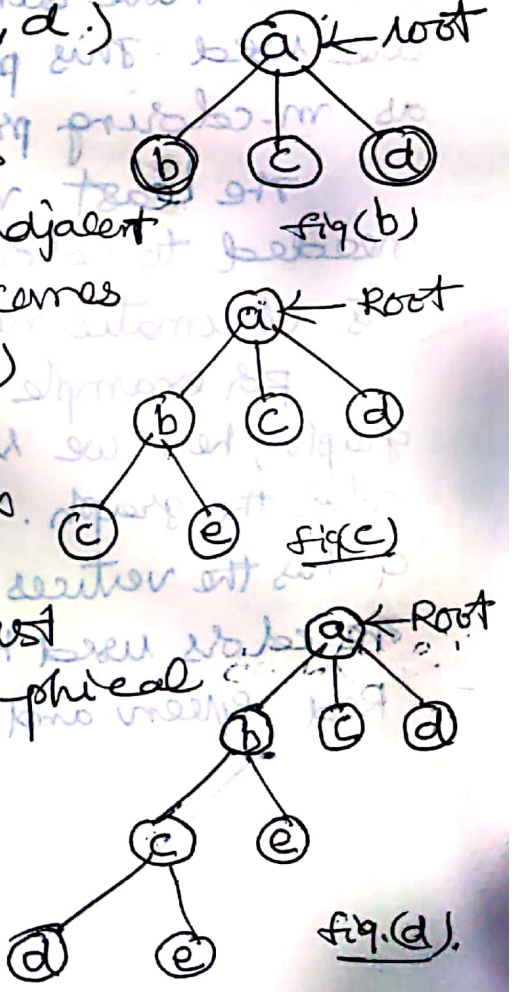
Initially start the search with vertex 'a'. the vertex 'a' becomes root of the tree

fig(a) (a) ← root

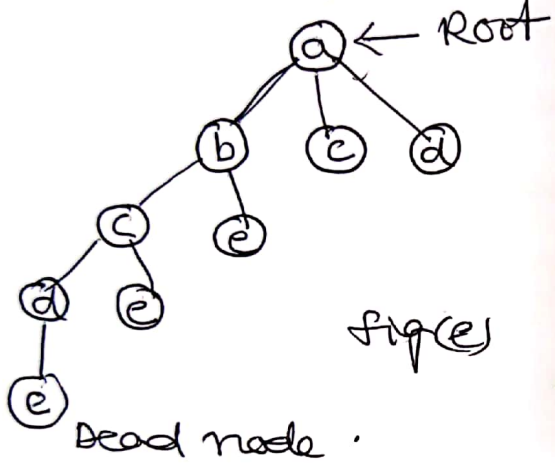
Next choose the vertex 'b' adjacent to 'a'. as it comes first in lexicographical order (b, c, d.)

Next vertex 'c' is selected which is adjacent to 'b' and which comes in the order (c, e)

Next select vertex 'd' adjacent to 'c' which comes first in the lexicographical order (d, e)

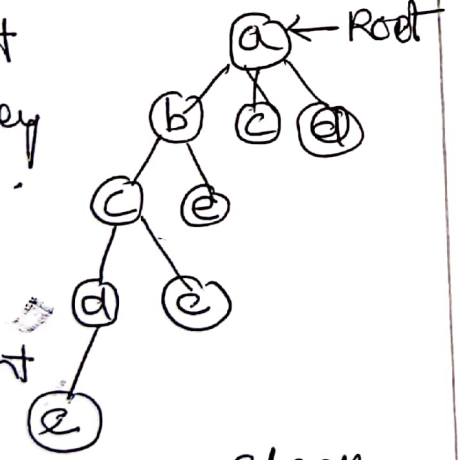


Next vertex 'e' is selected adjacent to 'd'. If we choose vertex 'a' then we do not get Hamiltanian cycle.



fig(e)

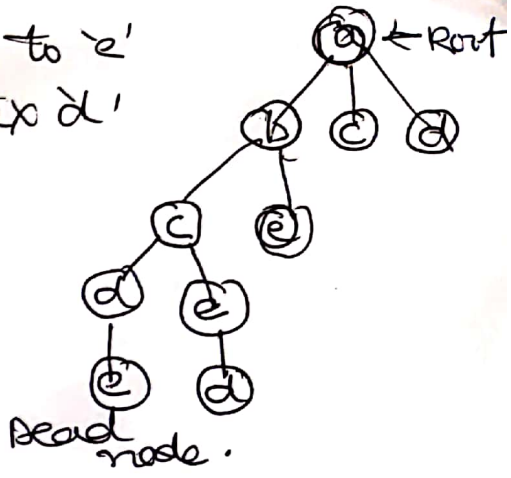
The vertex adjacent to 'e' are b,c,d but they are already visited. So backtrack one step and remove vertex 'e'



The vertex adjacent to 'd' are e,c,a-from which vertex 'e' has already visited. checked.

and by choosing 'a' we do not get Hamiltanian cycle so backtracks one step. Hence select vertex 'e' adjacent to 'c'.

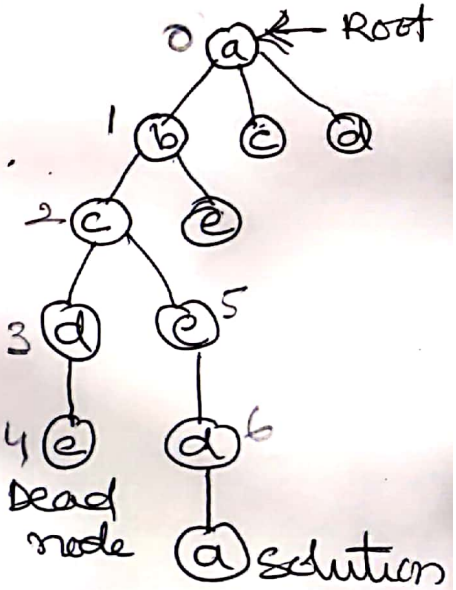
The vertex adjacent to 'e' are (b,c,d) so vertex 'd' is deleted.



~~The vertex adjacent~~

The vertex adjacent to 'd' are (a,c,e) so vertex 'a' is selected. Hence we get the Hamiltanian cycle so all vertices other than vertex 'a' is visited only once. i.e

a-b-c-e-d-a.



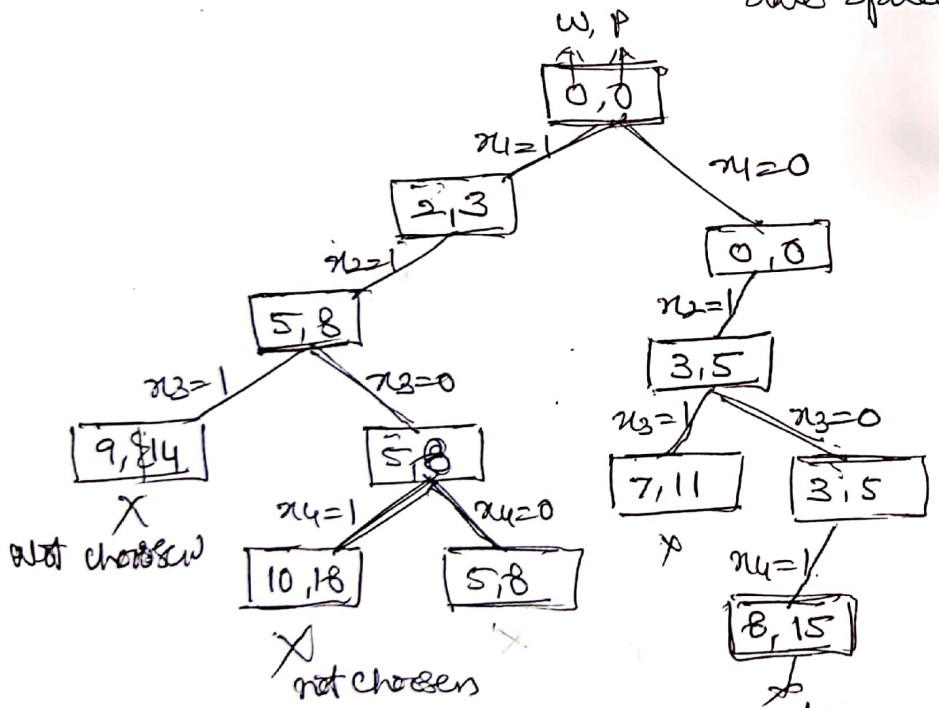
0/1 Knapsack problem using Backtracking

Find the optimal solution of the 0/1 Knapsack problem for instance $n=4$, $m=8$, $(P_1,P_2,P_3,P_4)=(3,5,6,10)$ and $(w_1,w_2,w_3,w_4)=(2,3,4,5)$ using backtracking method.

$n=4$ $m=8$

Obj	1	2	3	4
w	2	3	4	5
P	3	5	6	10

This Knapsack problem can be solved by using state space tree.



1st solution

$\{x_1,x_2,x_3,x_4\}=\{0,1,0,1\}$.

$(x_1\ x_2\ x_3\ x_4)$

Obj	1	2	3	4
sol	0	1	0	1

Final wt $3+5=8$ Kg

Final profit $5+10=15$

problem constraint

$\sum w_i x_i \le m$

$8 \le 8$

objective

max $\sum p_i x_i$

$= 15$

similarly to solve all possible solution by means of space tree using backtracking.