Prepared By Abhiyan Dev Rimal
https://github.com/abhirimal/be-software-notes

Date
Page

Chapter-6 (1 or 2 Q)

V. Imp

1) Briefly explain the Backtracking method for problem solving? or, (Explain Backtracking algorithm)

Ans: A backtracking algorithm is a problem solving algorithm that uses a brute force approach for finding the desired output. (satisfy some criteria)

-> The Brute force approach tries ~~to~~ out all the possible solutions and chooses the desired/best solutions.

-> Backtracking systematically searches for a solution to a problem among all available options.

-> The term backtracking suggests that if the current solution is not suitable, then backtrack and try other solutions. Thus, recursion is used in this approach.

(In backtracking, we start with one possible option out of many available options and try to solve the problem. If we are able to solve the problem with the selected move then we will print the solution else we will backtrack and select some other option and try to solve it. If none of the options work out we will claim that there is no solution for the problem.)

-> This approach is used ~~for~~ to solve problems that have multiple solutions. If we want an optimal solution, we must go for dynamic programming.

Scanned with CamScanner

→ There are three types of problems in backtracking:

1) Decision Problem - In this, we search for a feasible solution.
2) Optimization Problem - In this, we search for the best solution.
3) Enumeration Problem - In this, we find all feasible solutions.

Algorithm:

Step 1: if current_position is goal, return success
Step 2: else,
Step 3: if current_position is an end point, return failed.
Step 4: else, if current_position is not end point, explore and repeat above steps.

Applications of Backtracking

1) N-Queen Problem
2) Sum of subset problem
3) Graph coloring
4) Hamilton Cycle

→ A State Space Tree is used in Backtracking
→ A state space tree is a tree representing all the possible states (solution or nonsolution) of the problem from the root as an initial state to the leaf as a terminal state

Scanned with CamScanner

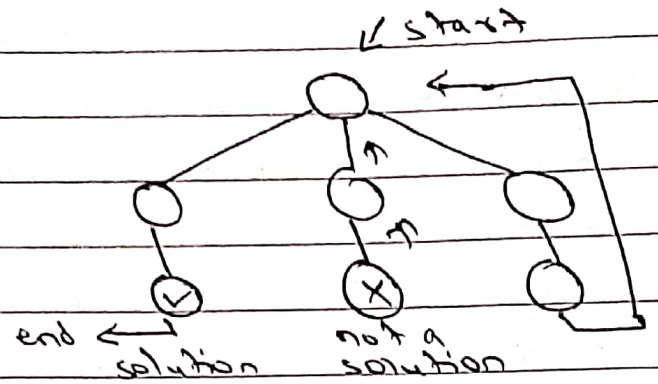


fig: State Space Tree

Example Backtracking Approach

Q) You want to find all the possible ways of arranging 2 boys and 1 girl on 3 benches. Constraint: (girl should not be on the middle bench)

soln:-

There are total of $3! = 6$ possibilities.

We will try all the possibilities and get the possible solutions. We recursively try all the possibilities.

⇒

All the possibilities are:- (B_1, B_2, G), (B_1, G, B_2), (B_2, B_1, G), (B_2, G, B_1), (G, B_1, B_2), (G, B_2, B_1).

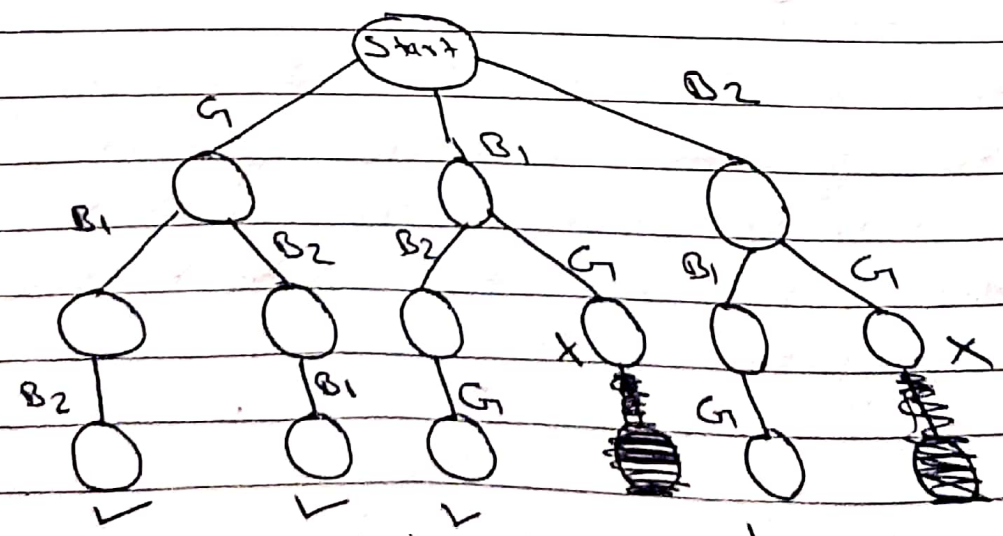


fig: State space tree of all the solutions

Scanned with CamScanner

2) Difference between the Backtracking and Recursion

→ Recursion is a technique that calls the same function again and again until you reach the best case.

→ Backtracking is algorithm that finds all the possible solutions and selects the desired solution from the given set of solutions.

3) Or,

→ In recursion, the function calls itself until it reaches a base case.

→ In backtracking, we use recursion to explore all the possibilities until we get the best result for the problem.

Scanned with CamScanner

Imp

11 3) Write down the backtracking algorithm for N-queen problem.

or

1111 How do you solve 8-queens problem using backtracking? Explain.

→ In N-Queen Problem, we are given an NXN chesboard and we have to place n queens on the board in such a way that no two queens attack each other. A queen will attack ~~each other~~ another queen if it is placed in horizontal, vertical or diagnoal points in its way.

Backtracking Algorithm: The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we donot find such a row due to clashes then we backtrack and return false.

Algorithm:

1) Start in the leftmost column
2) If all queens are placed
   return true

Bounding fn → row, column, diagonal in case of n queens problem

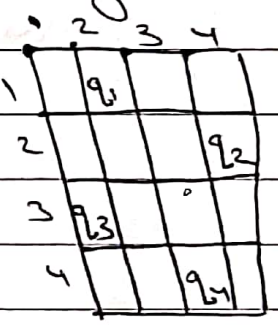
3) Try all rows in the current column. Do following for every tried row.

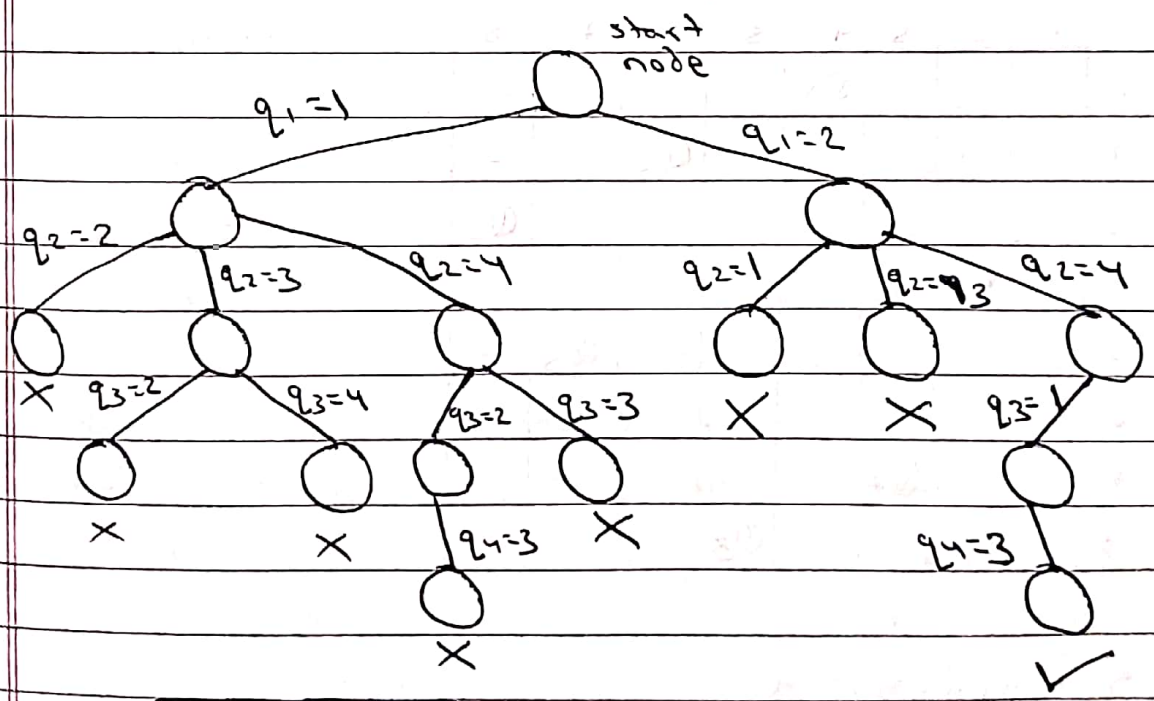
a) If the queen can be placed safely in this row then ~~most~~ mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.

b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing a queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

4) If all rows have been tried and nothing worked, return false.





q → | 2 | 4 | 1 | 3 |
q1 q2 q3 q4

column no

, here queens can be placed in a 4×4 chessboard.

fig:- State space tree of 4 Queen Problem.

Similarly, if we continue this process we will find another solution i.e 3,1,4,2

	1	2	3	4
1			Q_1	
2	Q_2			
3				Q_3
4		Q_4		

	1	2	3	4
1		Q_1		
2				Q_2
3	Q_3			
4			Q_4	

Two solutions i.e 2,4,1,3 & 3,1,4,2.

Q) For 8 queen problem

→ One possible solution of 8 queen problem is, (4,6,8,2,7,1,3,5).

	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			

other solution {7,5,3,1,6,8,2,4}

4) Apply backtracking to sum of subsets problem.

→ (The problem is to find a subset of a given set $S = \{S_1, S_2, \ldots S_n\}$ of n +ve integers whose sum is equal to a given +ve integer "m".)

→ In this problem, there is a given set with some integer elements. And another sum value is also provided, we have to find a subset of the given set whose sum is the same as given ~~some~~ sum value.

→ We are considering

1) the set contains non-negative value.

2) the input set is unique (no duplicates are present)

→ Here backtracking approach is used for trying to select a valid subset. When an item is not valid, we will backtrack to get the previous subset and add element to get the solution.

Algorithm:

1) Start with an empty set

2) Add the next element from the list to set

3) If the subset is not feasible or if we have reached the end of the set, then backtrack through the subset until we find the most suitable value.

4)

P.T.O →

understand

{ Backtracking:

→ In Backtracking algorithm as we go down along depth of tree we add elements so far, and if the added sum is satisfying explicit constraints, we will continue to generate child nodes further. Whenever the constraints are not met, we stop further generation of sub-trees of that node, and backtrack to previous node to explore the nodes not yet explored. We need to explore the nodes along the breadth and depth of the tree. Generating nodes along breadth is controlled by loop and nodes along the depth are generated using recursion (post order traversal). }

Steps: (Algorithm)

1) Start with an empty set
2) Add the next element from the list to the set.
3) If the subset is having sum M, then stop with that subset as solution.
4) If the subset is not feasible or if we have reached the end of the set, then backtrack through the subset until we find the most suitable value.
5) If the subset is feasible (sum of subset < M) then go to step 2.
6) If we have visited all the elements without finding a suitable subset and if no backtracking is possible then stop without solution.

Scanned with CamScanner

Example:

Q) Let $w = \{1,3,4,5\}$ and $m=8$. Find the possible subsets of w that sum to m using backtracking.

Soln:-

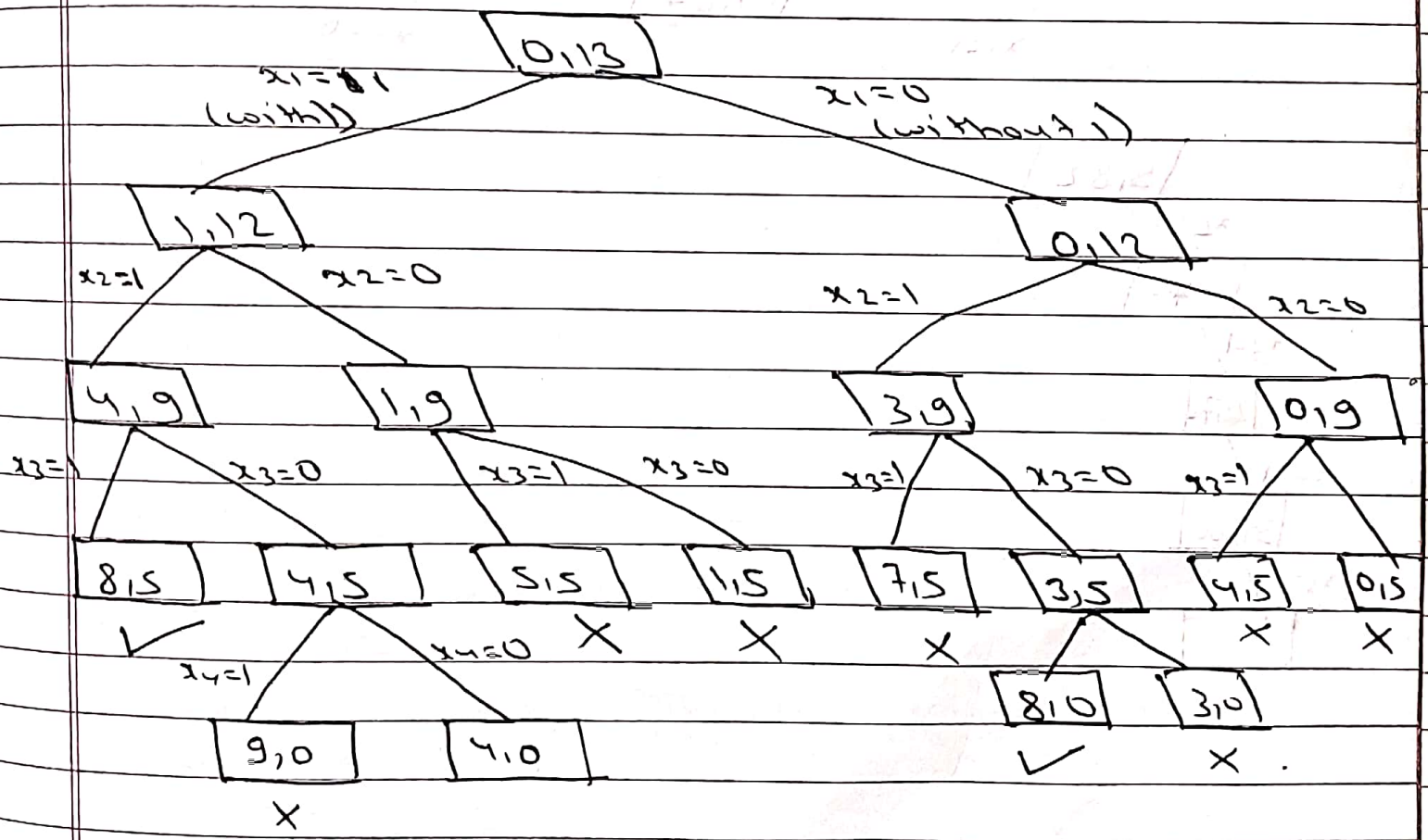
Here,

$n = 4$, $m = 8$.

$x_i = 1$ (with)

$x_i = 0$ (without)

x	1	3	4	5
	x_1	x_2	x_3	x_4



State Space Tree

~~Hence~~

Here two solutions are obtained

i.e. two subsets whose sum is 8

$\therefore \{1,3,4\}$ & $\{3,5\}$

1 a) Let $w = \{5, 7, 10, 12, 15, 18, 20\}$ and $m = 35$. Find the possible subsets of w that sum to m using backtracking. Also draw the portion of the state space tree that is generated.

Soln:- Here,

$n = 4$, $m = 35$

$x_i = 1$ (with), $x_i = 0$ (without)

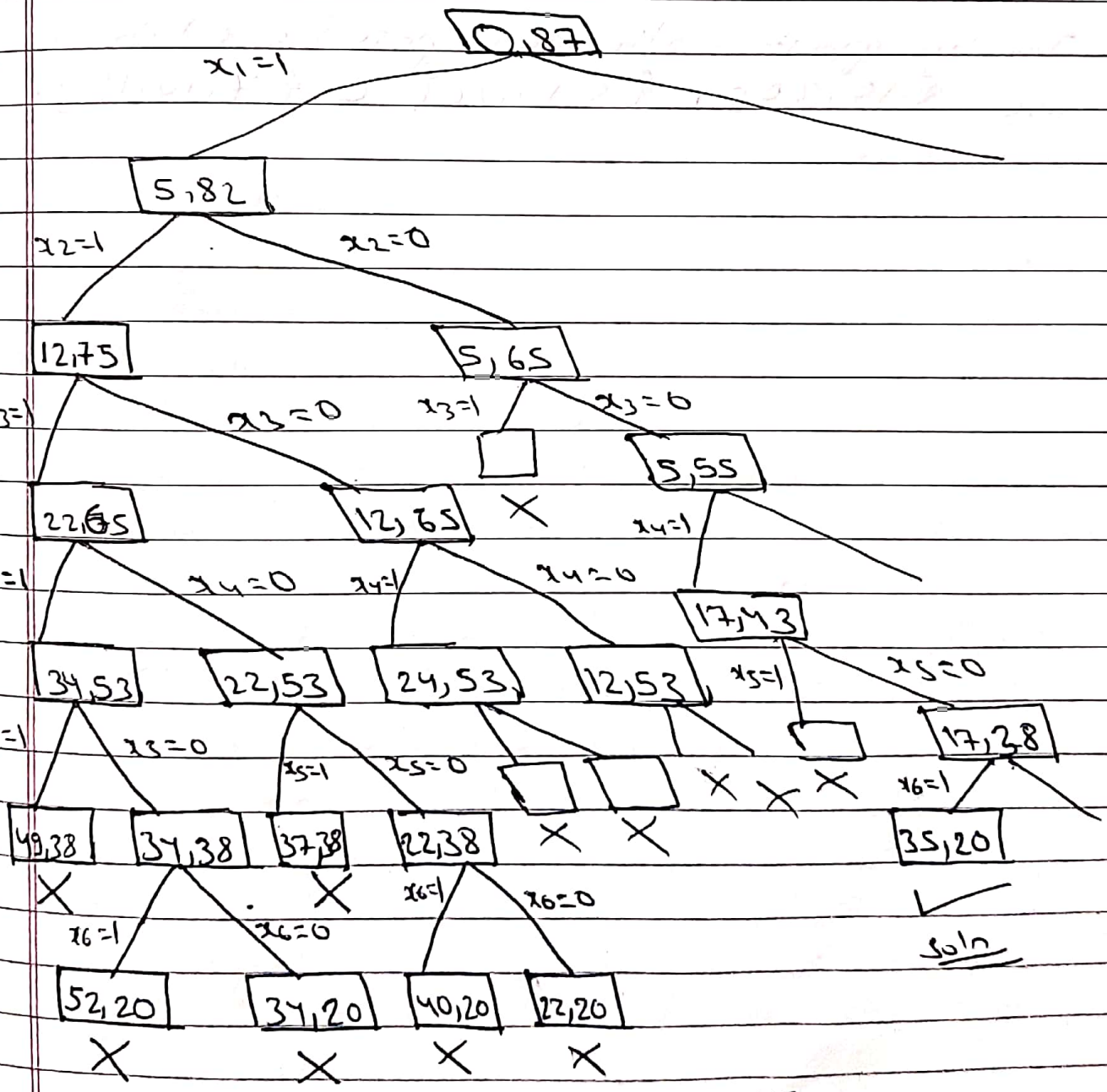


fig:- State Space Tree

Scanned with CamScanner

Here we found a solution on the path

$x_1=1, x_2=0, x_3=0, x_4=1, x_5=0, x_6=1$
i.e $\{5, 12, 18\}$ whose sum is 35.

Similarly, if we continue this process we will find many desired solutions.

In this question other two possible solutions are, $\{15, 20\}$, $\{5, 10, 20\}$ & $\{7, 10, 18\}$.

Scanned with CamScanner

classmate
Date
Page

10) Graph colouring Problem. Explain in brief!

or

What do you mean by graph coloring problem. How this problem would be solved with the help of Backtracking algorithm. explain with algorithm.

-> In this problem, an undirected graph is given. There is also provided m colors.
The problem is to find if it is possible to assing m different colors to each vertex of a graph G, such that no two adjacent vertices of the graph are of the same colors.

-> If the solution exists, then display which color is assigned on which vertex.

-> The objective is to minimize the number of colors while coloring a graph.

-> The smallest number of colors required to color a graph G is called its chromatic number of that graph.

-> Graph coloring problem is a NP complete problem.

Application of graph coloring

1) Map coloring
2) Data mining
3) Networking

Scanned with CamScanner

Using Dynamic/Naive Approach

-> In this ~~method~~ approach using the brute force method, we find all permutations of color combinations that can color the graph.

-> If any of the permutations is valid for the given graph and colors, we output the result otherwise not.

-> This method is not efficient in terms of time complexity because it finds all colors combinations rather than a single solution.

Using Backtracking algorithm

-> The backtracking algorithm makes the process efficient by avoiding many bad decision made in naive approaches.

-> In this approach we color a single vertex and then move to its adjacent (connected) vertex to color it with different color.

-> After coloring, we again move to another adjacent vertex that is uncolored and repeat the process until all vertices of the graph are colored.

-> In case, we find a vertex ~~that~~ that has all adjacent vertices colored and no color is left to make it color different, we backtrack and change the color of the last colored vertices and again proceed further.

-> If by backtracking, we come to the same vertex from where we started and all colors were tried on it, then it means the given number of

colors (i.e 'm') is insufficient to color the given graph and we require more colors (i.e a bigger chromatic number)

Algorithm

1) Different colors:

1.1) Confirm whether it is valid to color the current vertex with the current color (by checking whether any of its adjacent vertices are colored with the same color).

1.2) If yes then color it. Otherwise, try a different color.

1.3) Check if all vertices are colored or not.

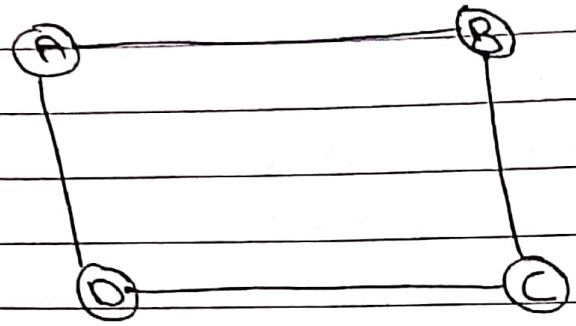
1.4) If not then move to the next adjacent uncolored vertex.

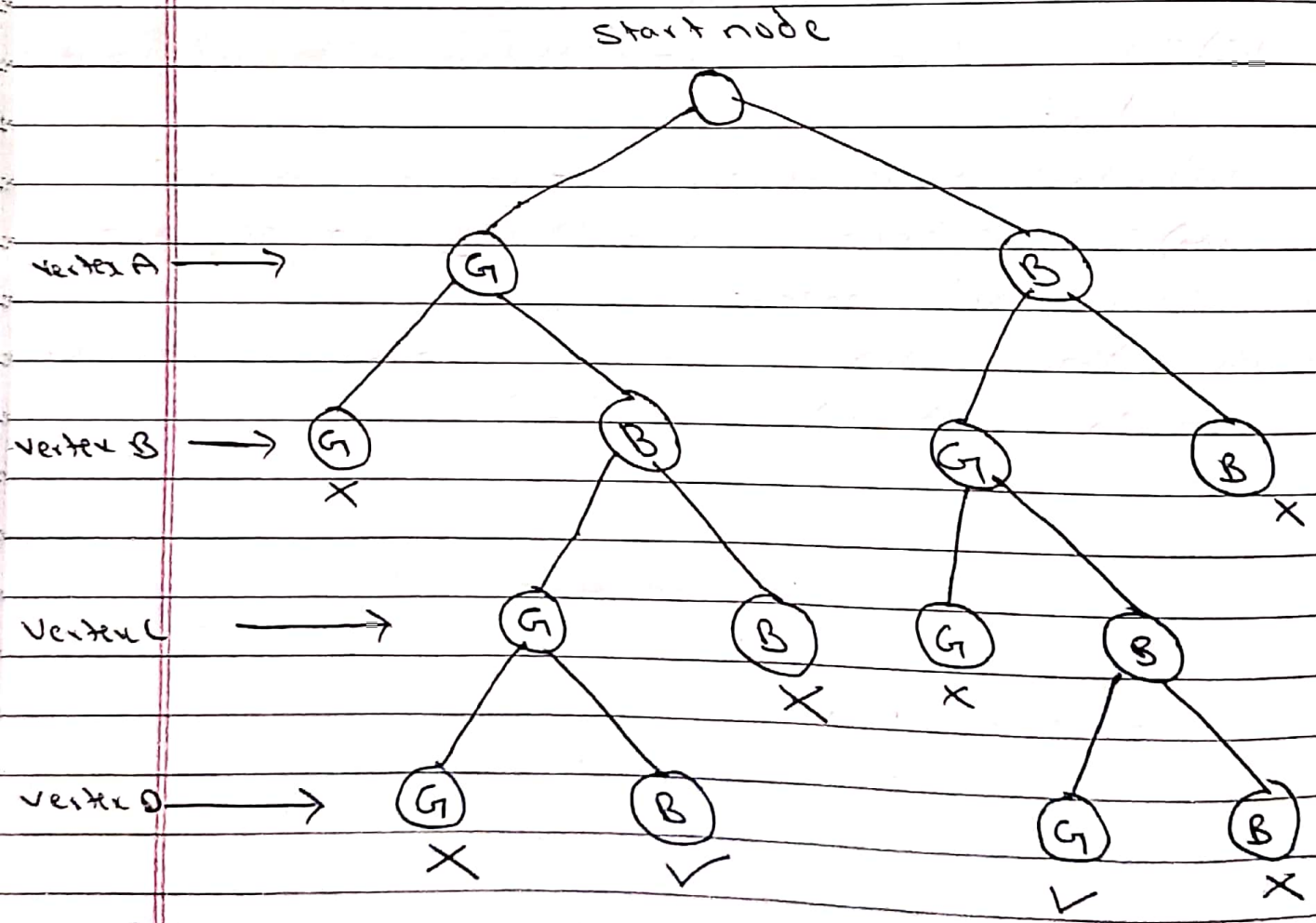
2) If no other color is available then backtrack. (i.e uncolor last colored vertex).

Scanned with CamScanner

Example of graph coloring problem.

1) A graph G is given with vertices A, B, C & D. Two colors Blue and Green is given.





→ Hence, we can color ~~graph~~ graph G with vertices A, B, C, D in two ways i.e G, B, G, B & B, G, B, G respectively.

Scanned with CamScanner

11) Hamilton Cycle

or

State the Hamilton graph problem and show that backtracking can be used to solve it.

→ A Hamiltonian cycle also called a Hamiltonian circuit, is a graph cycle through a graph that visits each node exactly once.

→ Hamiltonian path in an undirected graph is a path that visits each vertex exactly once.

→ Simply, to find a hamiltonian path we have to start from some starting vertex and visit all the vertices exactly once and return back to a starting vertex.

→ The problem to check whether a graph contains a Hamiltonian path is NP-complete.

* Finding the Hamiltonian Cycle Using Backtracking

→ Using the backtracking method, we can easily find all the Hamiltonian cycle present in the given graph.

→ The idea is to use the Depth-First Search algorithm to traverse the graph until all the vertices have been visited.

→ We traverse the graph starting from a vertex (arbitrary vertex chosen as starting vertex) and

→ if at any point during traversal we get stuck (i.e, all the neighbour vertices have been visited),

Scanned with CamScanner

we backtrack to find other paths (i.e., to visit another unvisited vertex)

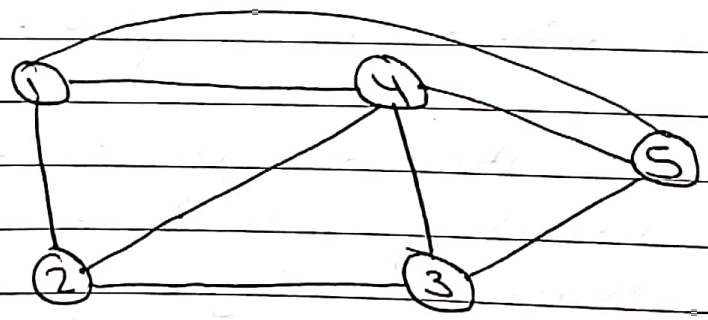
→ If we successfully reach back to the starting vertex after visiting all the nodes, it means the graph has a Hamiltonian cycle otherwise not.

Note

→ If there is a ~~Hami~~ Articulation point in a graph then Hamiltonian cycle doesnot exist in that particular graph.

Example.

1) Find a Hamiton cycle in a given graph.



The possible Hamitonian cycle in given graph G are

(1,2,3,4,5,1), (1,2,4,3,5,1), (4,1,2,3,5,4), (5,3,4,2,1,5), and many more..

→ It's application is Travelling Salesman Problem..

Scanned with CamScanner