Backtracking strategy

Dr. Priyadarshan Dhabe,
Ph.D (IIT Bombay)

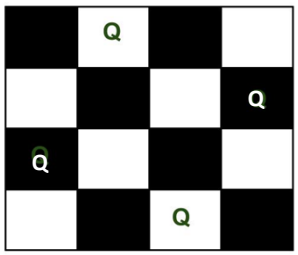
Backtracking strategy- general approach

- Backtracking is an algorithm design technique that uses brute force approach to solve the problems.
- Problems which can have multiple solutions can be solved by this approach (not optimization problems, want all solutions)
- Solutions has some constraints (called bounding functions)
- Backtracking uses **DFS** (depth first search) to generate possible solutions as opposed to **branch and bound** that works in (**BFS)** Breadth First manner.
- It generates solutions (incrementally) by series of decisions and when a wrong decision is made we can backtrack to earlier state.
- It is used to solve the problems in which a sequence of objects are chosen from a specified set so that the sequence satisfy some **constraints**.

https://www.youtube.com/watch?v=DKCbsiDBN6c

N queens problem- Backtracking strategy

- The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. For example, following is a solution for 4 Queen problem.



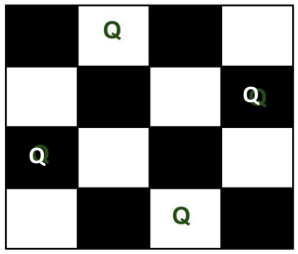
A queen **attacks** another queen, if both of them are in **same** row, column and diagonal

N queens problem- Backtracking strategy

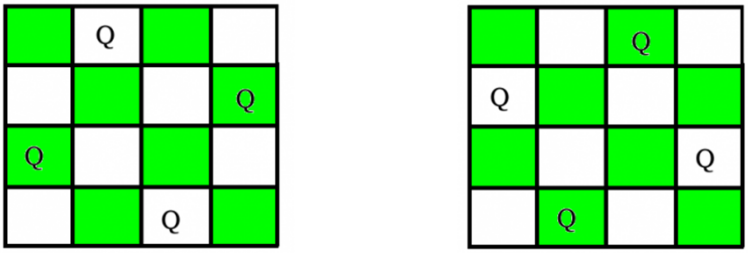
- The expected output is a **binary matrix** which has **1s** for the blocks where queens are placed. For example, following is the output matrix for above **4 queen solution**.

{ 0, 1, 0, 0}
{ 0, 0, 0, 1}
{ 1, 0, 0, 0}
{ 0, 0, 1, 0}

binary matrix solution



N queens problem- Multiple solutions



https://www.youtube.com/watch?v=xFv_Hl4B83A

N queens problem- Backtracking strategy

- 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 do not find such a row due to clashes then we backtrack and return false.

{ 0, 0, 0, 0}
{ 0, 0, 0, 0}
{ 0, 0, 0, 0}
{ 0, 0, 0, 0}

binary matrix at start

N queens problem Backtracking algorithm

1) Start in the leftmost column
2) If all queens are placed return true
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 mark this **[row, column]** as part of the solution and recursively check if placing queen here leads to a solution. (This queen should not attack other)
b) If placing the queen in **[row, column]** leads to a solution then return true.
c) If placing 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 to trigger backtracking.

https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/

Efficient method than brute force approach

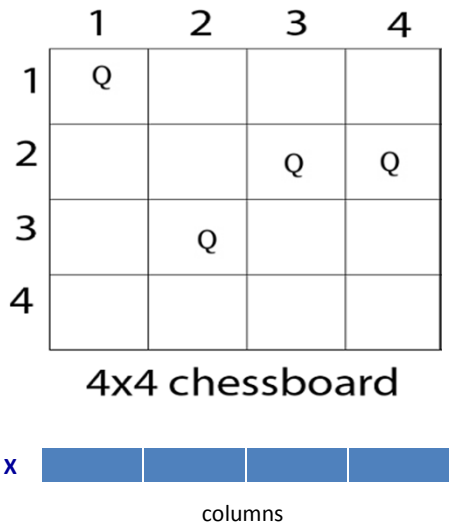
Total possibilities

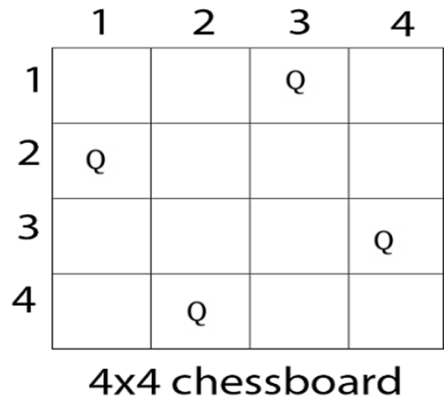
$$^{16}C_4 = \frac{16!}{4!*12!}$$

-Let each queen **Qi** –need to be placed in the **i th** row
-Now we simply need to decide the columns where each queen need to be placed

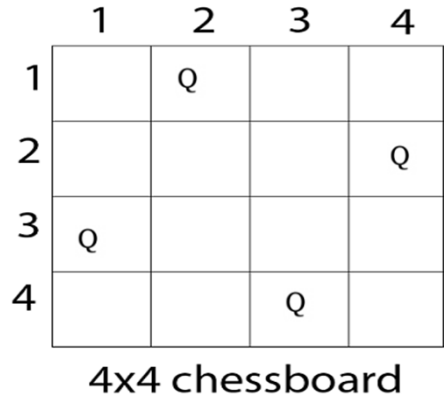
Third queen can not be placed in row3 anywhere, so back track and place 2nd queen in 4 th column

Now 4 th queen can not be placed in any place on row 4 so back track completely





X	3	1	4	2



X	2	4	1	3

-There are 2 solutions to 4 queen problem
-Time complexity by brute force approach is O(N!) for N queen problem

- We can reduce time complexity by various approaches
- Using function to find number of safe places
- Restricting Qi in i th row

N queens problem Backtracking algorithm

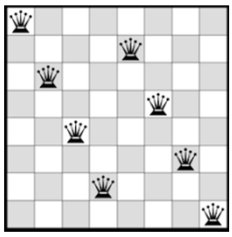
1. Worst case time complexity

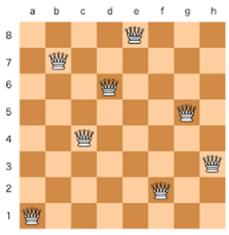
$$O(^{N^2}C_N)$$

2. If we decide to place one queen at a column then we can have $= Nx(N-1)x...x1$ possibilities thus, $O(N!)$

Further reading- Rat and maze problem using backtracking

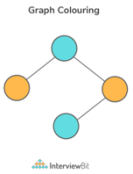
Find some solutions to 8 queen problem





Graph coloring problem-Backtracking algorithm

- **Graph/Vertex coloring** is the most commonly encountered graph coloring problem. The problem states that given m colors, determine a way of coloring the vertices of a graph such that no two adjacent vertices are assigned **same color**.
- The smallest number of colors needed to color a graph G is called its **chromatic number**. For example, the following undirected graph can be colored using minimum of 2 colors. Hence the chromatic number of the graph is 2.



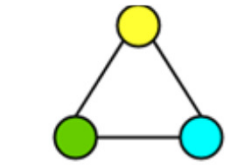
Graph coloring problem-chromatic number

1. Cycle Graph-

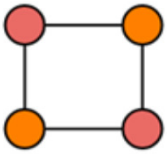
- A simple graph of 'n' vertices (n>=3) and 'n' edges forming a cycle of length 'n' is called as a **cycle graph**.
- In a cycle graph, all the vertices are of degree 2.

Chromatic number

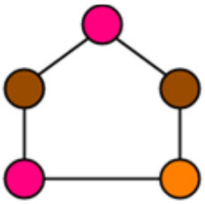
- If number of vertices in cycle graph is **even**, then its chromatic number = 2.
- If number of vertices in cycle graph is **odd**, then its chromatic number = 3.



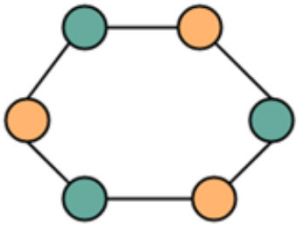
Chromatic Number = 3



Chromatic Number = 2



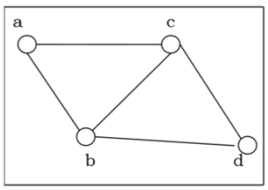
Chromatic Number = 3



Chromatic Number = 2

Graph coloring problem-chromatic number

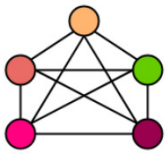
- **2. Planar Graphs-**
- A **Planar Graph** is a graph that can be drawn in a plane such that none of its edges cross each other.
- Chromatic Number of any Planar Graph = **Less than or equal to 4**
- All cycle graphs are planner graphs



Graph coloring problem-chromatic number

- **3. Complete Graphs-** A complete graph is a graph in which every two distinct vertices are joined by exactly one edge.
- In a complete graph, each vertex is **connected** with every other vertex.
- **Chromatic Number** of any Complete Graph
  = **Number of vertices in that Complete Graph**

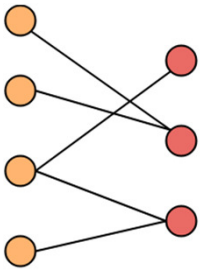
Chromatic Number = 4



Chromatic Number = 5

Graph coloring problem-chromatic number

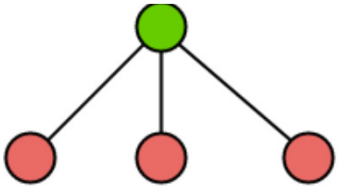
- **4. Bipartite Graph- (bipartite-composed of 2 parts)**
- A **Bipartite Graph** consists of two sets of vertices X and Y.
- The edges only join vertices in X to vertices in Y, not vertices within a set.
- Chromatic Number of any **Bipartite Graph** = 2



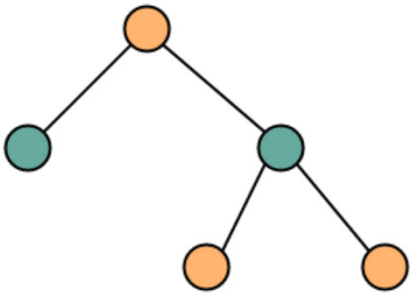
Chromatic Number = 2

Graph coloring problem-chromatic number

A **tree** is a bipartite graph, thus has chromatic number =2



Chromatic Number = 2



Chromatic Number = 2

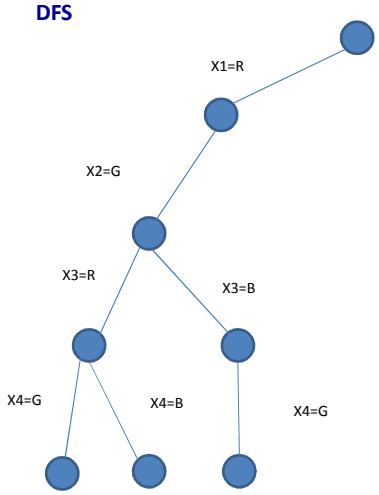
https://www.gatevidyalay.com/graph-coloring-chromatic-number/#:~:text=In%20this%20graph%2C-,No%20two%20adjacent%20vertices%20are%20colored%20with%20the%20same%20color,number%20of%20this%20graph%20%3D%203.

Graph coloring problem-Backtracking algorithm

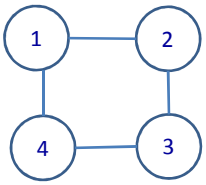
- **Input:-** a graph G with vertices V, m colors and Adjacency matrix M
- **Output:- all possible solutions** of the graph coloring
- **Algorithm- Backtracking**
- By using the backtracking method, the main idea is to assign colors one by one to different vertices right from the first vertex.
- Before color assignment, check if the adjacent vertices have same or different color by considering already assigned colors to the adjacent vertices.
  - If the color assignment does not violate any constraints, then we mark that color as part of the result. If color assignment is not possible then backtrack and return false.

Graph coloring problem-Backtracking algorithm

DFS



State space tree



m={R,G,B}

We got third solution here
3) R, G, B,G

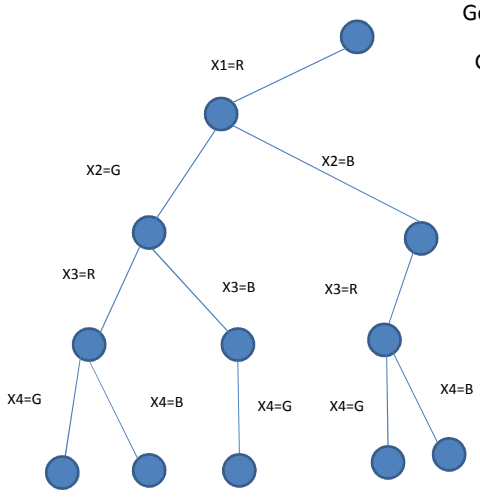
Now all colors are tried to node 4 and 3 go **back to node 2**

At this point of time we have 2 solutions

1) **R,G,R,G**
2) **R,G,R,B**

Try next color for 3 rd vertex

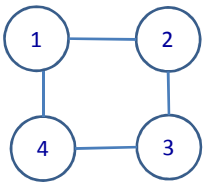
Graph coloring problem-Backtracking algorithm



Got one more solution **R,B,R,G**

Got one more solution **R,B,R,B**

In this way we can continue and find all possible solutions



m={R,G,B}

https://www.youtube.com/watch?v=052VkKhIaQ4

Graph coloring problem-Backtracking algorithm Time complexity

- Assume number of colors as m and vertices in the graph as **v**.
- If we don't consider any constraint on coloring, we can find the total number of nodes generated to infer time complexity
- We start with a single node then it can have m child's. Each of these m childs have their m childs, thus nodes at various levels can be summed as

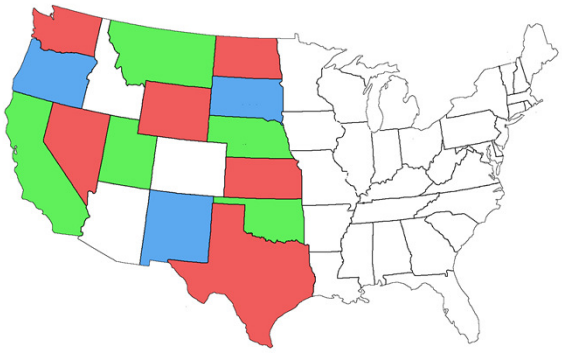
$$1 + m + m^2 + m^3 + \ldots + m^v$$

Thus, time complexity is $O(m^v)$

Graph coloring problems

- 1. can a given graph be colored using m colors?-**m coloring decision problem (Y/N)**
- 2. Minimum number of colors required to color the graph? **m coloring optimization problem**

Map coloring as application of Graph coloring



-Each region in the map can be considered as a node in a graph. If two regions they share the common boundary then they can be represented as adjacent nodes in a graph.
- Graph coloring can be applied to the region coloring in this way and thus map coloring is possible.