BACKTRACKING (DYNAMIC PROGRAMMING)

CSE [AIML], 6TH SEMESTER

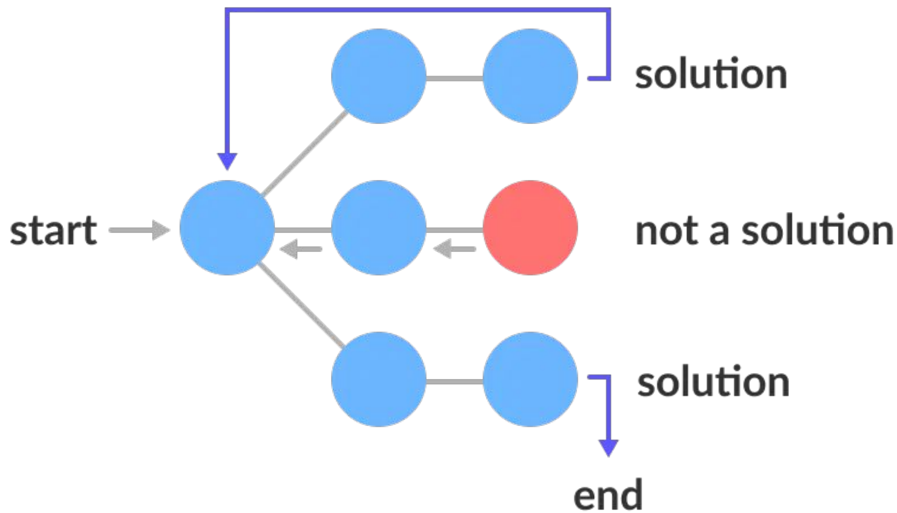
DESIGN & ANALYSIS OF ALGORITHM

Introduction

- A backtracking algorithm is a problem-solving algorithm that uses a **brute force approach** for finding the desired output.
- 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.
- This approach is used to solve problems that have multiple solutions. If you want an optimal solution, you must go for **dynamic programming**.

State Space Tree

- A space state 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.



Backtracking Algorithm

```
Backtrack(x)
    if x is not a solution
        return false
    if x is a new solution
        add to list of solutions
    backtrack(expand x)
```

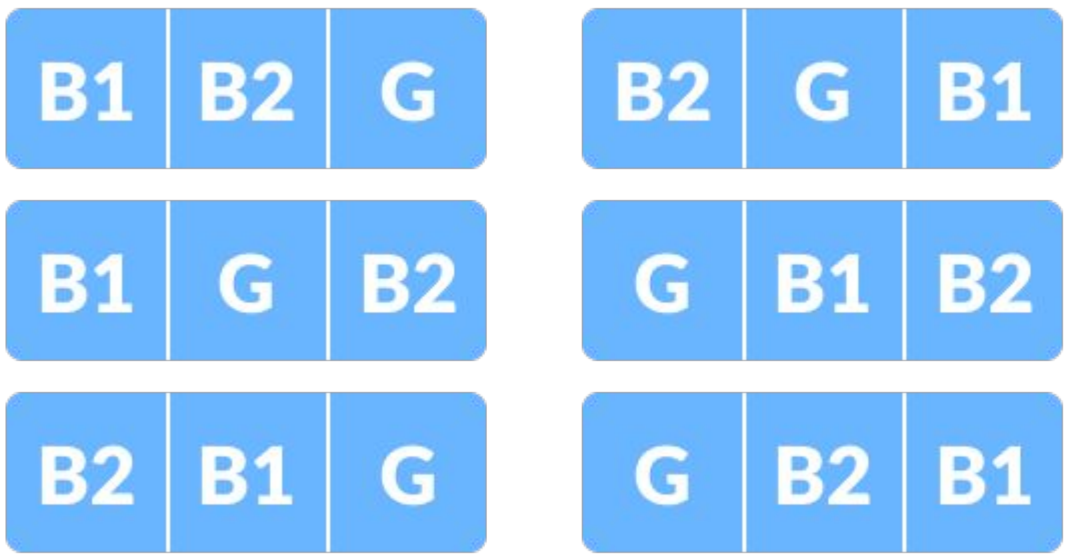
Example Backtracking Approach

Problem: 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.

Solution: There are a total of 3! = 6 possibilities. We will try all the possibilities and get the possible solutions. We recursively try all the possibilities.

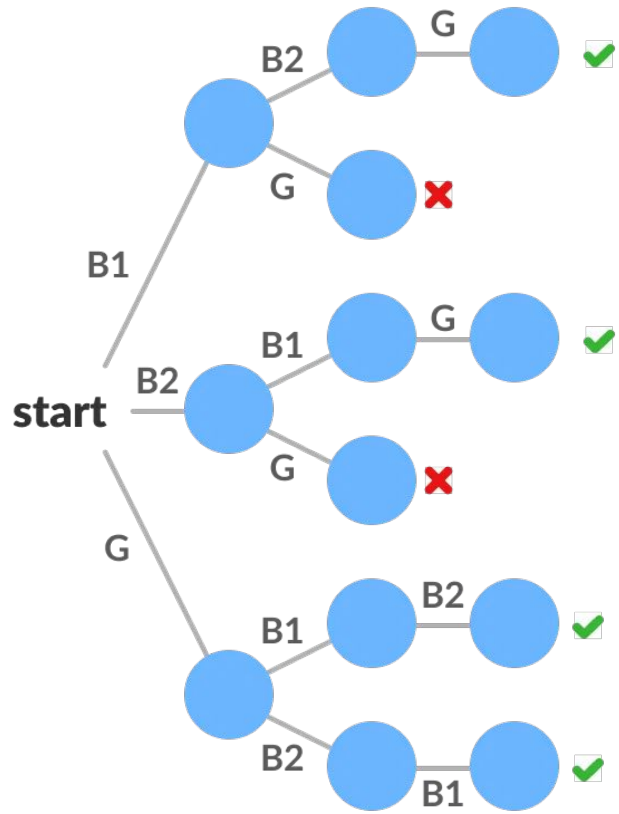
Example Backtracking Approach continues…

- All the possibilities are:



Example Backtracking Approach continues…

- The following state space tree shows the possible solutions.



Backtracking Algorithm Applications

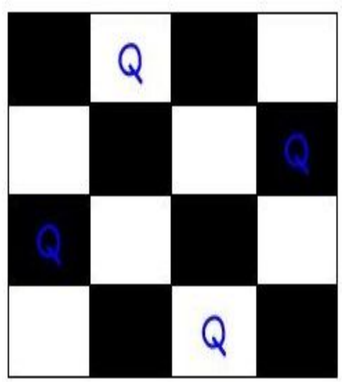
- To solve the N Queen problem.
- To solve Graph Coloring problem
- To find all Hamiltonian Paths present in a graph.

What is N Queen Problem?

In N-Queen problem, we are given an NxN chessboard and we have to place N number of queens on the board in such a way that no two queens attack each other. A queen will attack another queen if it is placed in horizontal, vertical or diagonal points in its way. The most popular approach for solving the N Queen puzzle is Backtracking.

4 Queen's Problem

- Suppose the given chessboard is of size 4x4 and we have to arrange exactly 4 queens in it. The solution arrangement is shown in the figure below –



Graph Coloring Problem Algorithm

- The Graph Coloring Problem Algorithm is designed to determine the least number of colors needed so that vertices of the graph are colored such that none of them share the same color with its neighbor. This is an NP-complete problem, and this means that it is computationally difficult to involve efficient heuristics or approximations in big graphs.

How It Works

The basic graph coloring problem algorithmic approach is relatively straightforward:

- Choose a Vertex: Start by choosing one vertex in the graph.
- Assign a Color: Designate the minimum possible color to that vertex.
- Move to the Next Vertex: Go to the next node of the graph.
- Check for Conflicts: Before painting any vertex, bear in mind that you cannot paint it with the same color you paint any of its neighboring vertices. If it does then move to the next available color on the list.
- Repeat Until All Vertices Are Colored: This has to be done until all vertices within the graph have been colored.

Hamilton Cycles and Paths

A cycle that uses every vertex in a graph exactly once is called a Hamilton cycle, and a path that uses every vertex in a graph exactly once is called a Hamilton path.

Note that if a graph has a Hamilton cycle then it also has a Hamilton path.

Advantages of Backtracking:

- Exact Solution: Retracing ensures that the solution is the best and perfect one.
- Adaptable: It works on small as well as large graphs; however, for very large graphs, it may prove to be uneconomical because of the search algorithm applied.

Disadvantages:

- Inefficiency for Large Graphs: Backtracking tends to be significantly time-consuming especially when it is dealing with large graphs because it tests out all the possible combinations of the colors.
- High Time Complexity: This of course is a function of the number of vertices and edges in the given digraph and as such, this is less feasible for backtracking when encountering very large digraphs.