Backtracking

▼ **TOC**

Introduction
N-Queens Problem
    Advantages
    Disadvantages
    Example
Graph Coloring
    Advantages
    Disadvantages
    Example
Hamiltonian Cycle
    Advantages
    Disadvantages
    Algorithm
    Example

Introduction

Backtracking is a problem-solving technique used to find solutions by systematically exploring all possible paths or states, while discarding any paths that cannot possibly lead to a solution. It is useful for problems with a large search space and involves recursively making decisions and exploring all possible outcomes until a solution is found. Backtracking is often used in combination with other techniques, such as pruning and heuristics, to make the search more efficient. The N-Queens problem is a common example of a problem that can be solved using backtracking.

Time and space complexity of all backtracking algorithm is **O(n!)** i.e. **O(n^n)** and **O(n)** respectively.

N-Queens Problem

The N-Queen Problem is a classic puzzle in the field of computer science and mathematics. It involves placing N chess queens on an NxN chessboard, so that no two queens threaten each other. The problem is to determine all the possible arrangements of the N queens on the board, given the constraints.

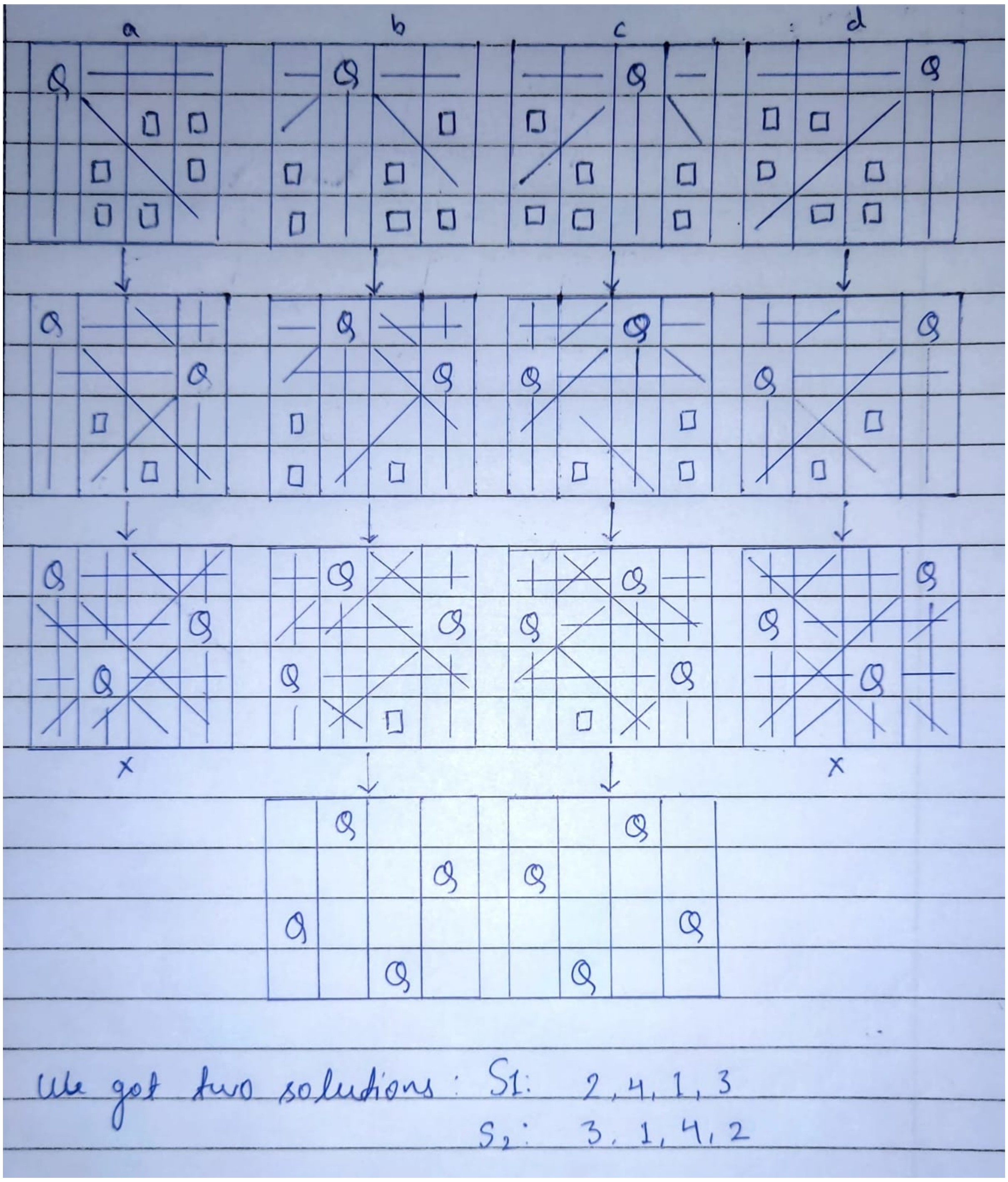
Advantages

- It has a wide range of applications in computer science, including optimization algorithms, AI, and game theory.
- The N-Queen Problem can be used to test the performance of different search algorithms and optimization techniques.

Disadvantages

- The N-Queen Problem is NP-complete, which means that there is no known algorithm that can solve it efficiently for large values of N.
- The problem becomes exponentially harder as N increases, and the search space quickly becomes too large to explore fully.

Example



We got two solutions : S_1: 2, 4, 1, 3

S_2: 3, 1, 4, 2

Graph Coloring

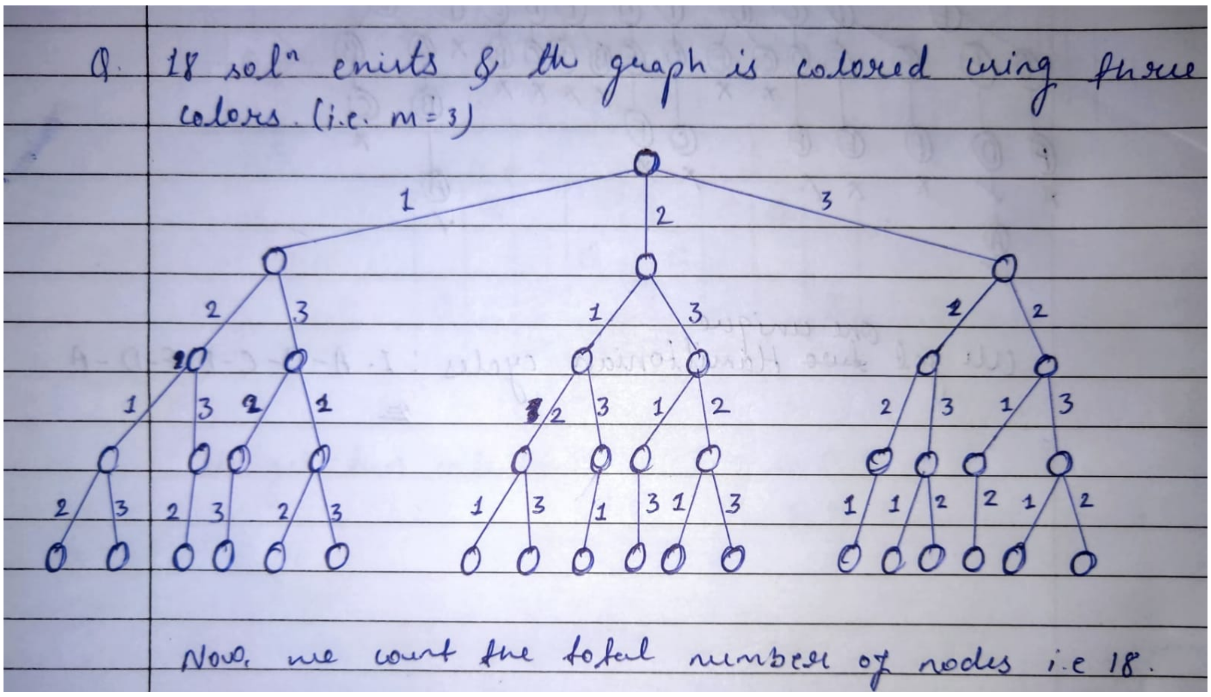
Graph coloring is the process of assigning colors to the vertices of a graph such that no two adjacent vertices have the same color. The goal is to use the smallest number of colors possible while still satisfying this constraint.

Advantages

Disadvantages

- Graph coloring has practical applications in fields such as scheduling, map coloring, and register allocation in compilers.
- Solving the graph coloring problem can be a good exercise in algorithm design and optimization, as there are many different ways to approach the problem and many different algorithms that can be used to solve it.
- The graph coloring problem is an NP-complete problem, which means that it is believed to be computationally intractable for large graphs.
- Finding a proper coloring for a graph can be difficult, as there are many possible colorings to explore and it is not always clear how to find a good coloring.

Example


Q. If sol^n exists & the graph is colored using three colors. (i.e. m=3)

Now, we count the total number of nodes i.e 18.

Hamiltonian Cycle

A Hamiltonian cycle, also known as a Hamiltonian circuit, is a path in a graph that visits every vertex exactly once and ends at the starting vertex. It is a special case of a Hamiltonian path, which is a path that visits every vertex exactly once, but does not necessarily end at the starting vertex.

Articulation point, pendant vertices and other such things make the Hamiltonian cycle not possible.

Advantages

- The Hamiltonian cycle problem is a good example of a combinatorial optimization problem, which is a common type of problem in computer science and other fields.
- Solving the Hamiltonian cycle problem can have practical applications in fields such as transportation planning and network design.

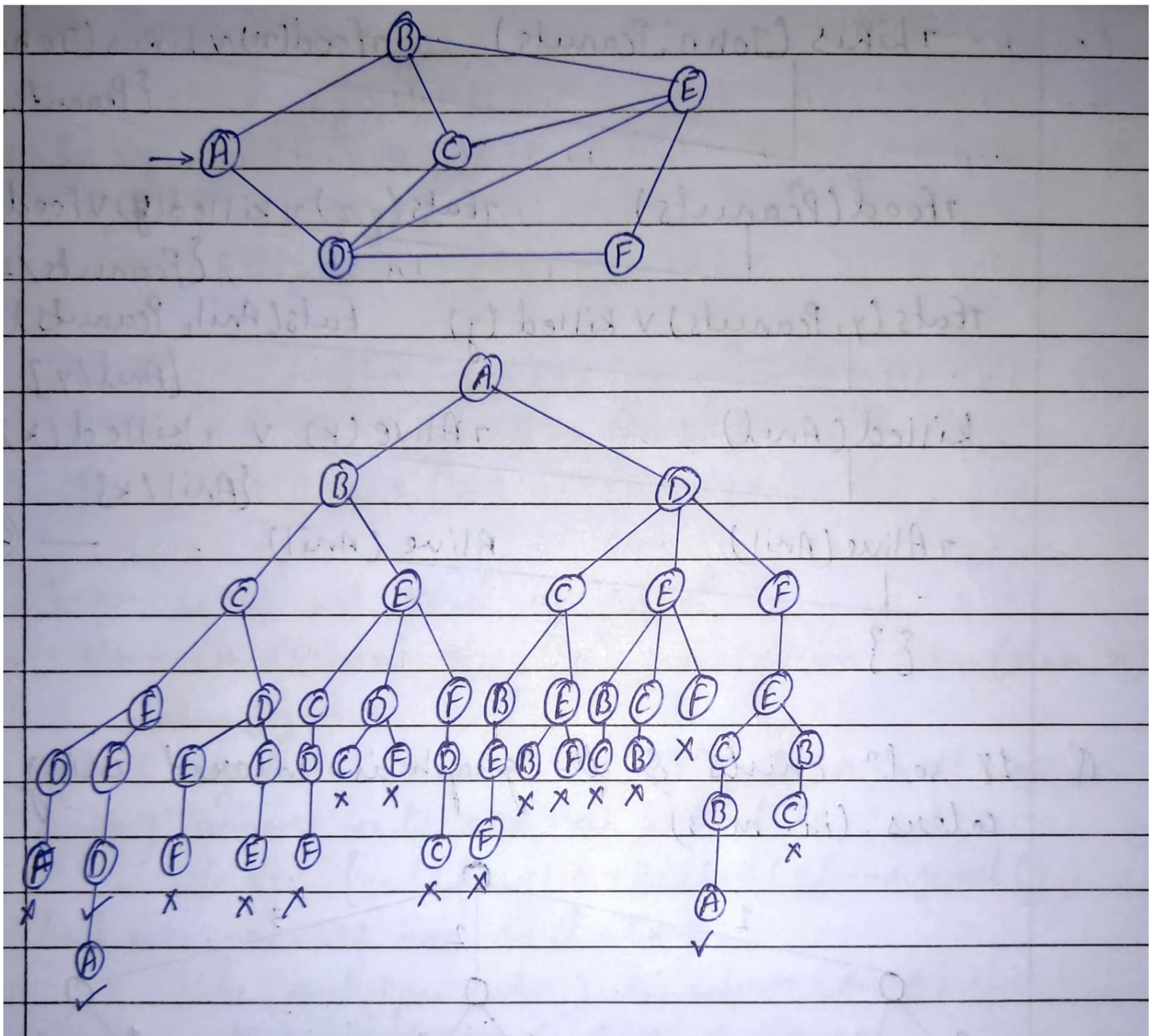
Disadvantages

- The Hamiltonian cycle problem is an NP-complete problem, which means that it is believed to be computationally intractable for large graphs.
- Finding a Hamiltonian cycle in a graph can be difficult, as there are many possible paths to explore and it is not always clear how to find a good path.

Algorithm

```
1. Initialize a list "path" to be empty.
2. Choose a starting vertex "v" in the graph.
3. Add "v" to the "path".
4. For each adjacent vertex "w" of "v":
   a. If "w" is already in "path", continue to the next adjacent vertex.
   b. Add "w" to "path".
   c. If "path" contains all the vertices in the graph, return "path".
   d. Otherwise, recursively call the algorithm with "w" as the new starting vertex.
   e. If the recursive call returns a path, return that path.
   f. Remove "w" from "path".
5. If no Hamiltonian cycle is found, return "None".
```

Example



We got ~~two~~ one unique Hamiltonian cycles : 1. A-B-C-E-F-D-A