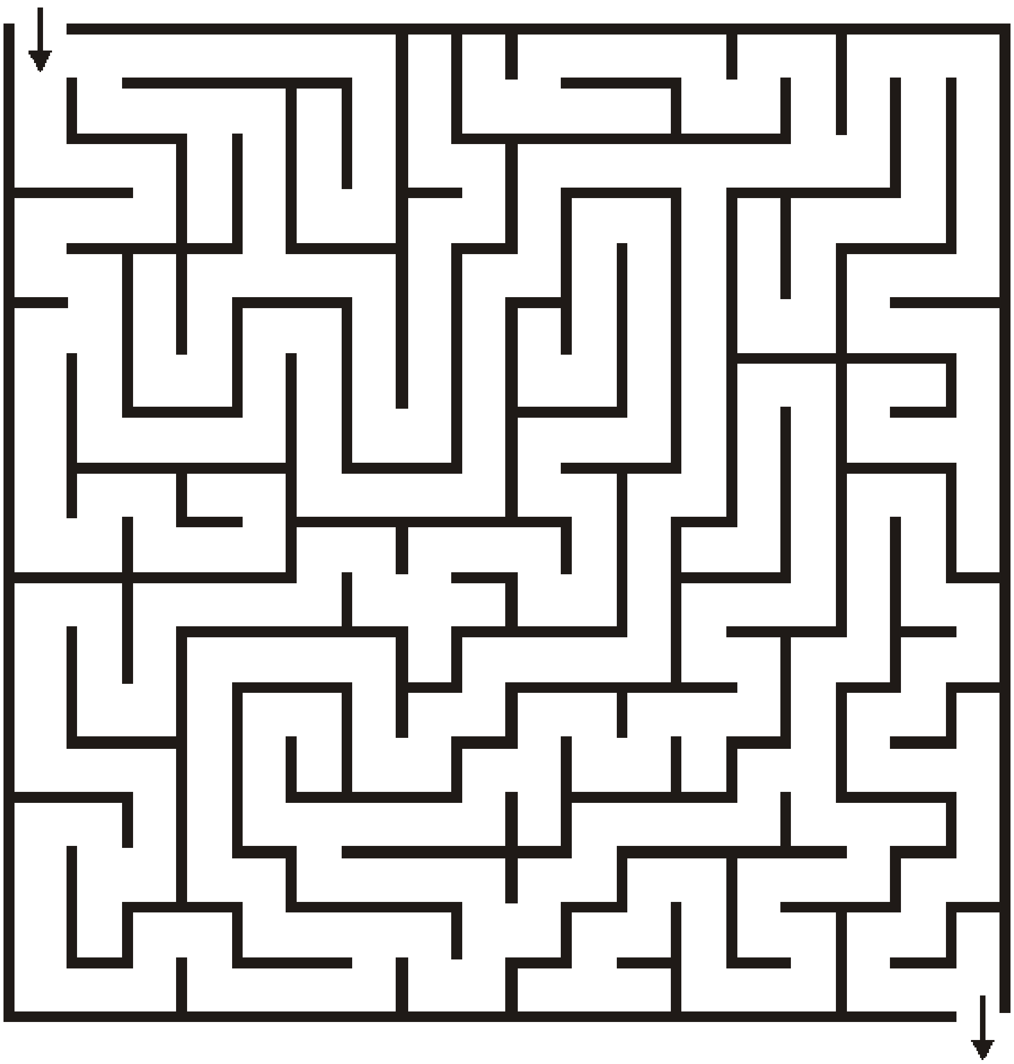
1. Backtracking & N-Queen Problem

What is Backtracking?

Backtracking is a method used to solve problems by trying out possible solutions and backing up (reversing) when a solution doesn't work.

Example:

Imagine you're in a maze: - Try a path. - If it leads to a dead end, go back and try another.



This process of trying, checking, and rewinding is backtracking.

What is the N-Queen Problem?

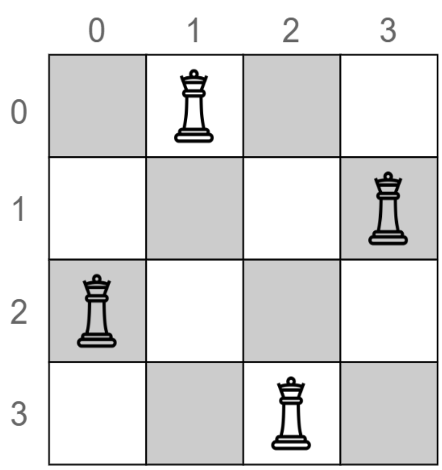
You are given a chessboard (say 4x4). You need to place 4 queens so that none attack each other. Queens attack in rows, columns, and diagonals. So, you must carefully place each one.

Why Use Backtracking?

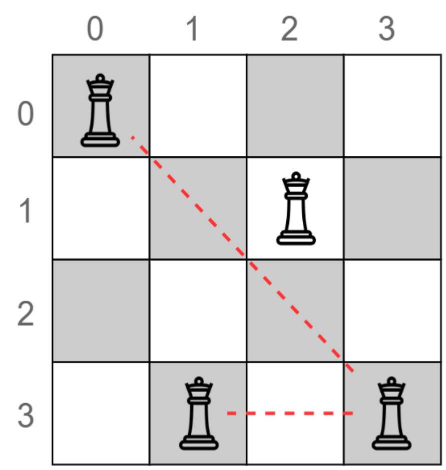
- You try to place queens one by one.
- If placing a queen causes a clash, go back and move the previous one.

Applications:

- Chess puzzles
- Sudoku solving
- Word search games
- Route finding in a maze



Valid queen positions



Invalid queen positions

---

2. Hamiltonian Circuit Problem

What Is It?

A Hamiltonian circuit is a path in a graph that visits every node exactly once and returns to the starting point.

Real Life Analogy:

You're a delivery driver who needs to visit every city once and come back home.

Why It's Hard:

There are many possible routes. You must find the one that meets the rule (no repeat, return home).

Applications:

- Travel planning
- Network routing
- Tour scheduling

---

3. Subset Sum Problem

What Is It?

You are given a list of numbers and a target value. You must find if any combination of the numbers adds up to the target.

Example:

List: [3, 34,1, 4, 12, 6, 7, 5, 2,8]
Target: 9
Answer: Yes → 4 + 5 = 9

Why It's Hard:

You may need to try many combinations.

Applications:

- Budgeting apps
- Inventory management
- Game scoring

---

4. Branch and Bound

What Is It?

A smart way to solve problems by skipping bad options and only exploring the best ones.

Real Life Analogy:

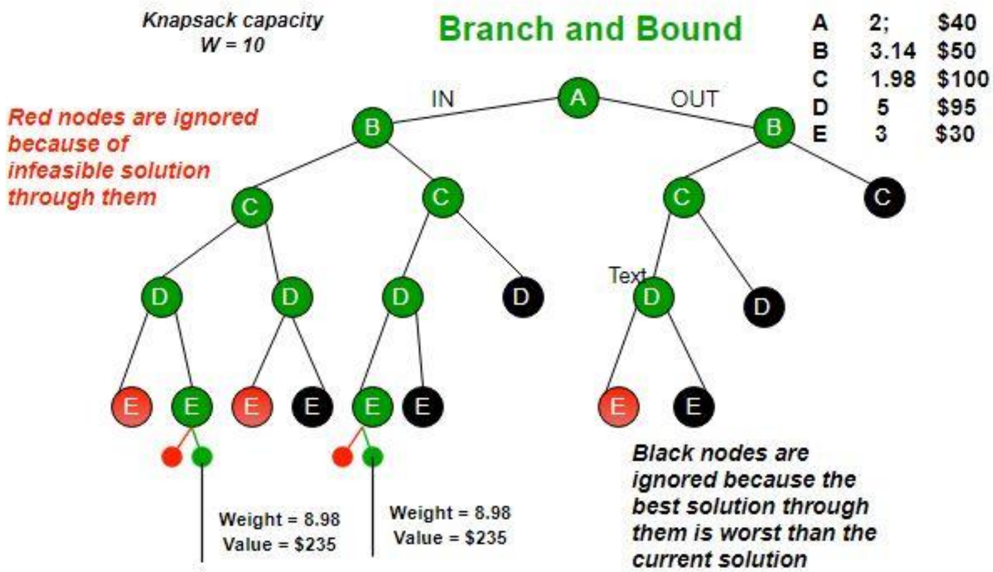
Looking for a red book in a huge library. If a section only has blue books, skip it!

Key Ideas:

- **Branching**: Trying different paths.
- **Bounding**: Skipping the ones that can't work.

Applications:

- Puzzle solving (e.g., 8-puzzle)
- Job assignments
- Optimization problems



---

5. Assignment Problem

What Is It?

Assign people to tasks in the most efficient way (least cost, least time, etc.).

Real Life Analogy:

You have 3 workers and 3 tasks. You know how much time each worker takes for each task. Assign jobs to minimize total time.

How To Solve:

Using a method called the Hungarian Algorithm (a smart technique).

Applications:

- Job scheduling
- Matching students to projects
- Allocating resources

---

6. Approximation Algorithms for NP-Hard Problems

What Is It?

Sometimes problems are so hard (NP-Hard) that finding the exact answer quickly is almost impossible. Approximation algorithms give us a **good enough answer** quickly.

NP-hard problems are a class of problems in computer science that are at least as hard as the hardest problems in the complexity class NP (Nondeterministic Polynomial time). Finding an efficient (polynomial-time) algorithm for even one NP-hard problem would mean that all problems in NP could be solved efficiently.

Real Life Analogy:

You want to pack your suitcase in the best way possible, but you're in a rush. You pick the **almost best** items quickly, instead of spending hours testing every combination.

Why Use It?

- Exact solutions take too long.
- Approximation gives results fast.

Applications:

- Scheduling large tasks
- Big data clustering
- Internet routing

---

7. Knapsack Problem & Travelling Salesman Problem (TSP)

Knapsack Problem

What Is It?

You have a bag (knapsack) with a weight limit. You are given items with values and weights. Choose items to put in the bag to get the **maximum value without exceeding the weight limit**.

Real Life Analogy:

Packing your backpack with snacks, books, and gadgets — you want the **most useful mix** that fits.

Applications:

- Cargo loading
- Budget planning
- Resource allocation

Travelling Salesman Problem (TSP)

What Is It?

A salesman must visit a number of cities once and return home, using the **shortest route** possible.

Real Life Analogy:

You're planning a trip — visit each tourist spot once and return home using the least amount of fuel or time.

Why It's Hard:

- Many possible routes.
- Need to check which one is shortest.

Applications:

- Delivery services
- Route optimization
- GPS pathfinding

Summary Table

Algorithm	Simple Idea	Used In
Backtracking	Try, fail, go back, and try again	Puzzles, mazes, Sudoku
N-Queen Problem	Place queens so none attack each other	Chess, constraint solving
Hamiltonian Circuit	Visit all places once and return	Route planning, delivery
Subset Sum	Pick numbers that add to a target	Finance apps, games
Branch and Bound	Skip paths that can't work, try smarter ones	Optimization, puzzles
Assignment Problem	Match people to jobs in the best way	Job/task scheduling

Algorithm	Simple Idea	Used In
Approximation Algorithms	Fast, near-best solutions for hard problems	Scheduling, big data
Knapsack Problem	Pick items for max value under weight limit	Packing, budgeting
Travelling Salesman Problem	Visit cities once with the shortest round trip	Routing, logistics, GPS