**🤔 Viva Questions & Answers:**

1. **Q:** What is A\* search?  
   **A:** A\* is a best-first search algorithm that finds the shortest path using the formula f(n) = g(n) + h(n), combining actual cost and estimated cost.
2. **Q:** What is the difference between g(n) and h(n)?  
   **A:** g(n) is the actual cost to reach node n from the start. h(n) is the heuristic estimate to reach the goal from node n.
3. **Q:** What kind of heuristics can be used for 8-puzzle?  
   **A:** Common heuristics include **Manhattan distance** and **number of misplaced tiles**.
4. **Q:** Why is A\* better than DFS or BFS?  
   **A:** A\* is more efficient because it uses heuristics to focus the search on promising paths and avoid unnecessary exploration.

**🔍 Definitions:**

* **g(n)**:  
  The **actual cost** to reach a node n from the start.  
  In the 8-puzzle, this is the **number of moves taken** so far from the starting configuration.
* **h(n)**:  
  The **heuristic estimate** of the cost to reach the goal from node n.  
  It’s an educated guess — in the 8-puzzle, this could be how far tiles are from their goal positions (like Manhattan distance).
* **f(n)**:  
  The **total estimated cost** of the cheapest solution path that goes through node n.  
  So,

f(n) = g(n) + h(n)

**🔷 What is the 8-puzzle problem?**

* A 3x3 board with 8 tiles numbered 1 to 8 and one empty space (-1 here).
* Goal: Reach a desired configuration (goal state) from a given start state by sliding tiles into the empty space

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| print\_board() | Nicely prints the 3x3 puzzle grid |
| solvable() | Checks if a given start state can be solved using **inversion count** |
| heuristic() | Manhattan distance: total of distances each tile is away from its goal |
| moveleft()/right()/up()/down() | Move the empty tile (swap with neighboring tile) |
| movetile() | Generates all possible moves and chooses the best one (with lowest f(n)) |
| solveEight() | Recursive solver that makes best move and prints board |
| main() | Takes user input and starts the process |

**✅ 📋 Suggestions (for Viva):**

* Be confident explaining g, h, f.
* Know the Manhattan distance formula.
* Understand inversion count logic.
* Be ready to explain the difference between BFS, DFS, and A\*.

**🔁 What is Inversion Count in the 8-Puzzle Problem?**

In the context of the **8-puzzle**, **inversion count** is a way to determine if a puzzle configuration is **solvable** or **unsolvable**.

**✅ Definition:**

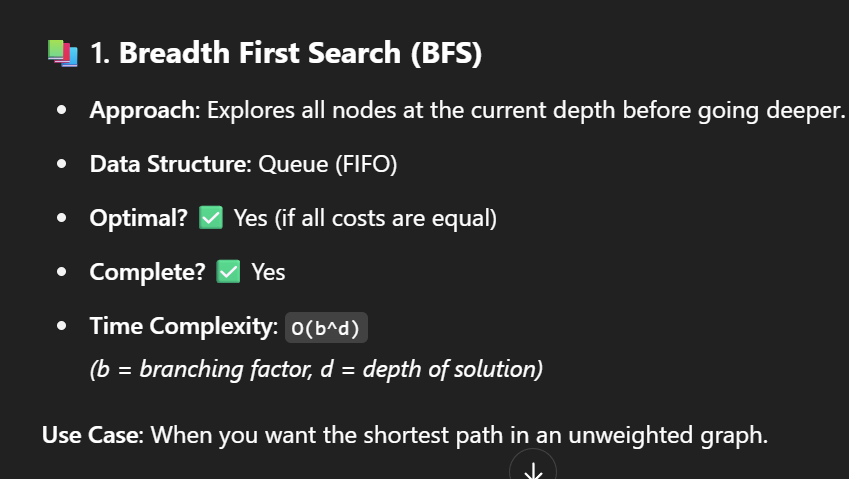
An **inversion** is a pair of tiles (a, b) such that:

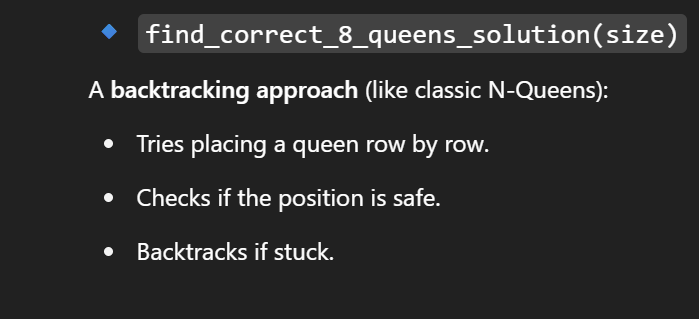
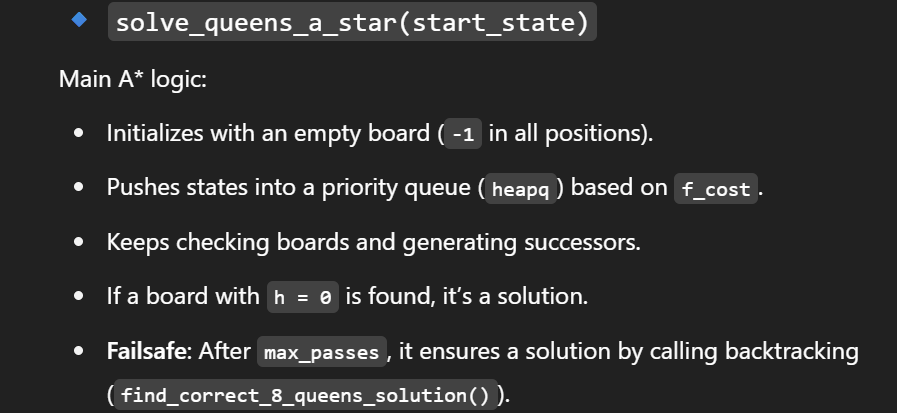
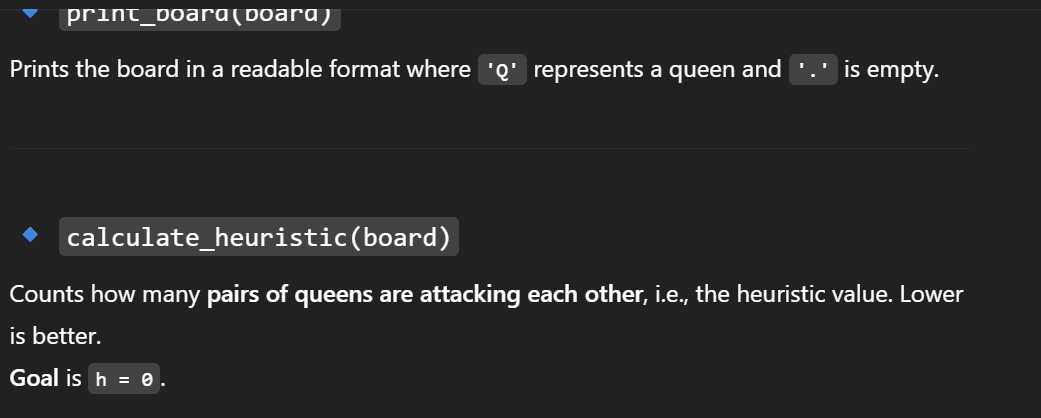
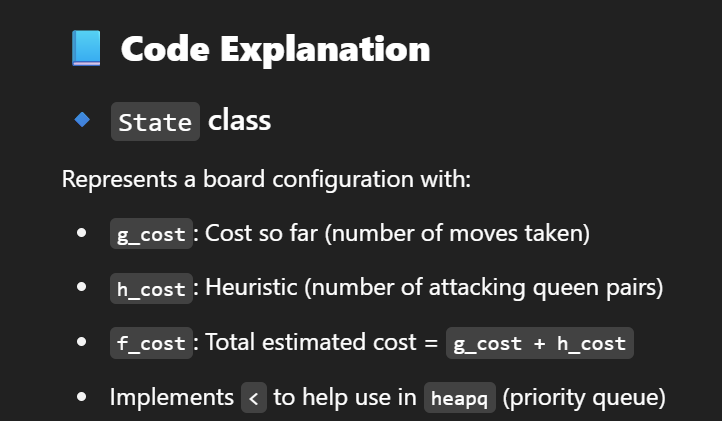
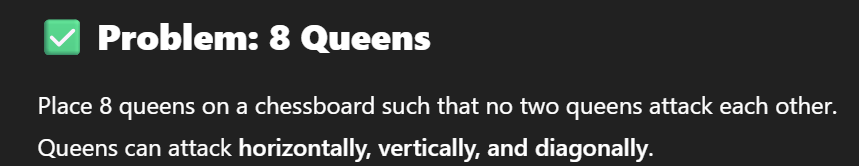
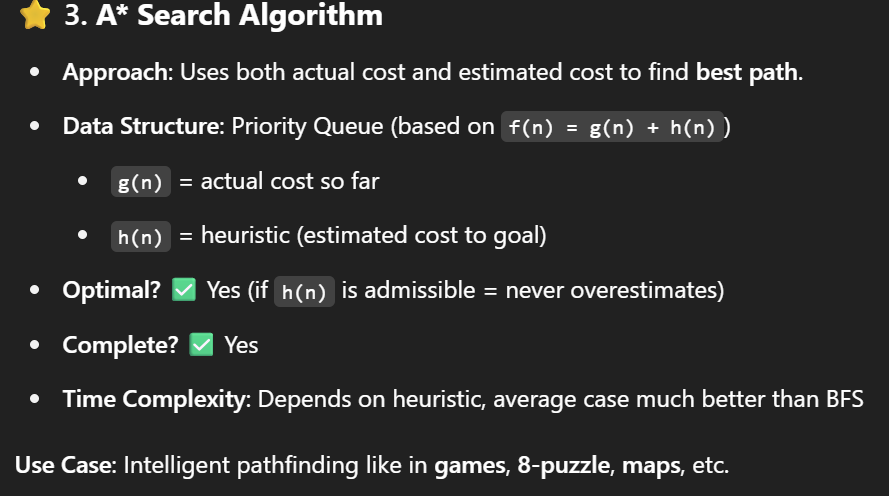
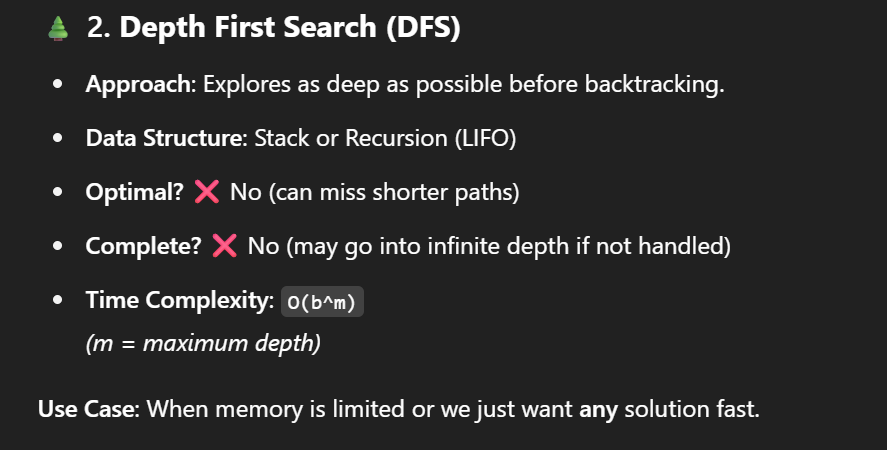
* a appears before b in the list, but
* a > b (i.e., the tiles are out of order)

We **ignore the blank tile (-1 or 0)** when counting inversions.

**🧠 Why is it important?**

In an **8-puzzle (3x3)**, the puzzle is solvable **only if the total number of inversions is even**.



V**2. What type of problem is this?**

**Ans**: It's a **Constraint Satisfaction Problem (CSP)**.

**3. Which algorithms are used here?**

**Ans**:

* **A\* Search** (with heuristic = number of attacks)
* **Backtracking** (used as a backup for guaranteed solution)

N QUEENS PROBLEM

* **3. What is the time complexity of this solution?**
* **Ans:** In the worst case, it's O(N!) — because for each row, we try N columns.