CLA 2 Assignment: Branch and Bound on the 8-Puzzle Problem

Objective

In this assignment, you will apply the Branch and Bound (B&B) search strategy to the **8-Puzzle problem**.
Your primary goal is to explore how different **lower-bound heuristics** affect:

1. **Search time / number of expanded nodes**
2. **Ability to reach the optimal solution**
3. **Accuracy of the lower bound** (i.e., how close the heuristic estimate is to the true optimal number of moves)

Task Overview

You will:

1. **Implement four different lower-bound heuristics** (from simplest to more advanced).
2. **Use Branch and Bound** to search for the optimal solution using each heuristic.
3. **Generate ~1000 random solvable puzzle instances**.
4. For each instance and each heuristic:
    - Compute the lower bound.
    - Solve the puzzle optimally using B&B.
    - Record **time taken**, **number of nodes expanded**, and **optimal move cost**.
5. **Compute the average ratio**

$$\text{ratio} = \frac{\text{optimal moves}}{\text{lower bound estimate}}$$

for every heuristic.

6. **Compare and analyze** how heuristic quality affects performance.

Lower-Bound Heuristics to Implement

Implement the following heuristics in increasing order of effectiveness:

---

(1) H_1: Trivial Heuristic (h = 0)

- Always returns **0**.
- Represents **uninformed search**.
- Serves as a baseline.

---

(2) H_2: Misplaced Tiles Heuristic

- Number of tiles not in their correct position.
- Simple to compute.
- Admissible but not very accurate.

---

(3) H_3: Manhattan Distance Heuristic

- Sum of Manhattan distances of each tile to its goal position.
- Classic heuristic.
- More accurate lower bound.

---

(4) H_4: Manhattan Distance + Linear Conflict (or Pattern Database if you prefer advanced)

Choose **ONE** advanced heuristic:

Option A — *Linear Conflict Heuristic*

- Manhattan distance + 2×(# of linear conflicts)
- Admissible and significantly tighter.

Option B — *Pattern Database (PDB)*

- Precompute exact distance for a subset of tiles.
- Combine additive PDBs for a strong lower bound.
- Most advanced option (more work).

You may choose either one depending on your course level.

What Students Must Implement

A. Branch and Bound Search

- Use a B&B search that:
    - Expands nodes in order of **cost + lower bound**
    - Prunes branches when **current path cost + heuristic ≥ best known solution**
    - Eventually finds the **optimal** solution

B. Random Puzzle Generator

- Generate **1000+ solvable random puzzles**.
- Ensure solvability using inversion count parity check.

C. Data Collection

For each puzzle and each heuristic:

- Lower-bound estimate
- Optimal solution cost (moves)

- Execution time to find the optimal solution
- Number of expanded nodes

D. Compute Average Ratios

For each heuristic H_i:

$$\text{Average ratio}(H_i) = \frac{1}{1000} \sum_{k=1}^{1000} \frac{\text{optimal moves}(k)}{\text{LB estimate}(k)}$$

Interpretation:

- Ratio ≈ 1 → heuristic is tight (very good)
- Ratio >> 1 → heuristic is weak / loose (poor lower bound)

Expected Results / Learning Outcomes

Students will observe that:

- **H_1: h=0**
    - Bound is useless → ratio very large
    - Search time explodes
- **H_2: Misplaced Tiles**
    - Better, but still weak
    - Search still slow
- **H_3: Manhattan Distance**
    - Significantly tighter
    - Search becomes practical

- **H_4: Linear Conflict**
    - Very close to optimal → ratio close to 1
    - Drastically reduces search time

This demonstrates the importance of good heuristics in Branch and Bound search.

Deliverables

1. Code

- Well-structured implementation of B&B and all heuristics.
- Code for generating random puzzles.
- Code for computing ratios and collecting statistics.

2. Report

Include:

1. Description of each heuristic
2. Explanation of Branch & Bound implementation
3. Performance comparison table
4. Plots:
    - Time vs. heuristic
    - Average ratio vs. heuristic
    - Nodes expanded vs. heuristic
5. Discussion & conclusion about heuristic strength