Objective

The objective of this project is to implement a parallel backtracking algorithm using Microsoft MPI (MSMPI) to search for valid configurations of the IQ Fit puzzle (11×5). Students will design a scalable solution capable of efficiently exploring a large solution space by distributing workloads among multiple MPI processes.

Background

The IQ Fit puzzle involves placing 12 uniquely-shaped 2D pieces on an 11×5 grid, ensuring no overlap and complete coverage. Each piece can be flipped and rotated, resulting in multiple valid orientations. Figure 1 shows a solved state of the puzzle.

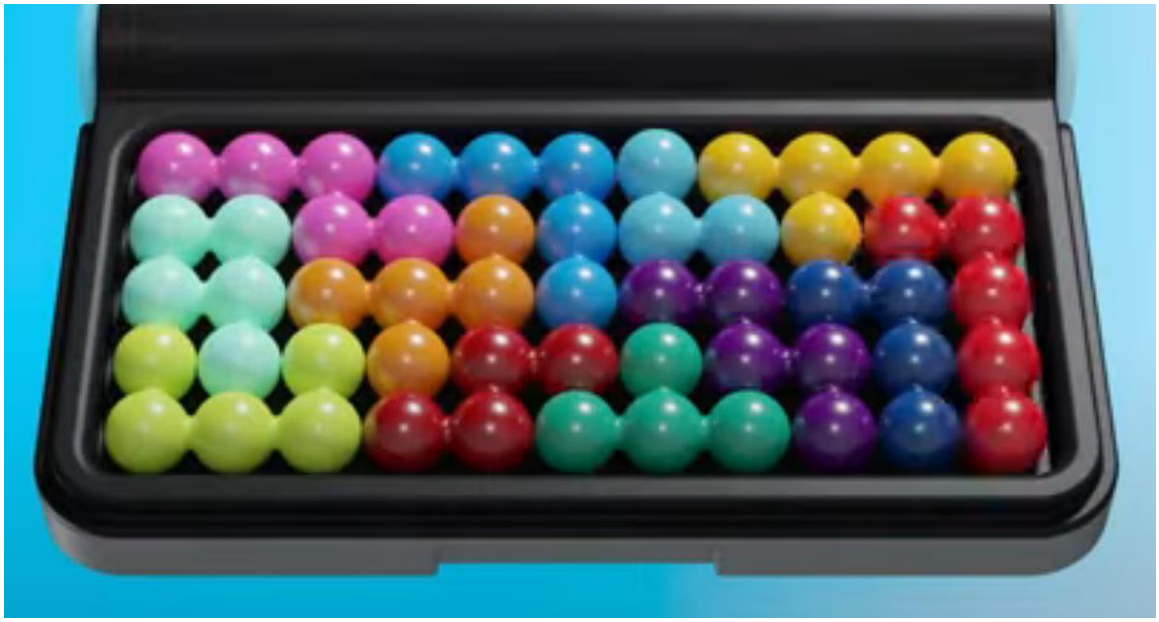

Figure 1 The IQ fit Puzzle

Given the exponential number of configurations, the problem becomes computationally intensive. Therefore, parallelization is essential to reduce runtime and allow the discovery of multiple or all valid solutions.

Project Tasks

- Modeling the Puzzle:

Define a suitable data structure for the 11×5 board and 12 pieces. He 12 pieces is presented in Figure 2.

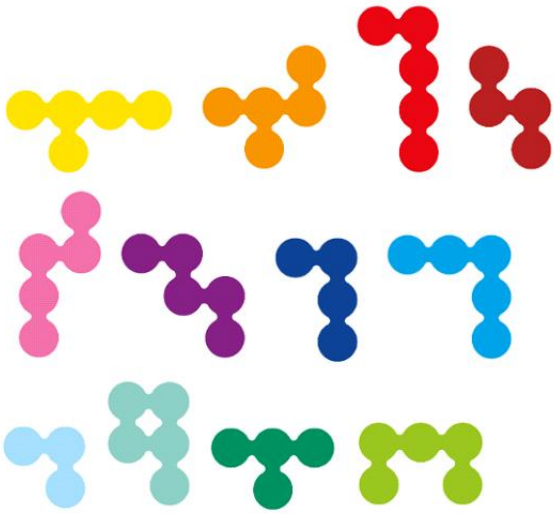

Figure 2 The 12 pieces of the puzzle

Implement functionality to rotate and flip each piece. Validate whether a piece can be placed at a given location.

- Sequential Baseline:

  Implement a recursive backtracking algorithm to find solutions sequentially.
  Ensure correctness before parallelization.

- Parallelization with MSMPI:

  Use Microsoft MPI to distribute the solution space.
  Design a task division strategy (e.g., split by fixed first N pieces).
  Each MPI process explores a distinct subset of the configuration tree.
  Use collective communication (e.g., MPI_Gather, MPI_Reduce) to combine results.

- Result Management:

  Save or print all unique valid solutions found by any process.
  Display them using letters. For example

```
AAABBBCDDDD
EEAAGBCCDJJ
EEGGGBKKLLJ
FEFGIIMKKLJ
FFFIIMMMKLJ
```

  Ensure each solution is stored only once (**handle duplicates if necessary**).

- Performance Evaluation:

  Measure speedup, efficiency, and scalability across varying process counts (e.g., 2, 4, 8, 16). Plot performance metrics and analyze bottlenecks.

- Documentation and Report:

  Clearly explain your approach, design decisions, and optimization strategies.
  Include performance graphs, code structure overview, and example outputs.
  Discuss limitations and possible extensions.

Deliverables

1. Source Code (C/C++ with MSMPI)

2. Project Report (PDF)

3. README File (build instructions, dependencies, and run instructions)

Assessment Criteria

Criterion	Weight
Correctness of Solution	25%
Parallel Algorithm Design	25%
Performance and Scalability Results	20%
Code Structure and Documentation	15%
Report	15%

Optional Extensions (Bonus Marks)

• Implement a GUI or visualization to show piece placements.

Resources

• Microsoft MPI (MSMPI) Documentation: https://learn.microsoft.com/en-us/message-passing-interface/microsoft-mpi

• Access to lab machines with MSMPI or instructions to install MSMPI locally.