Project 02: Hashiwokakero

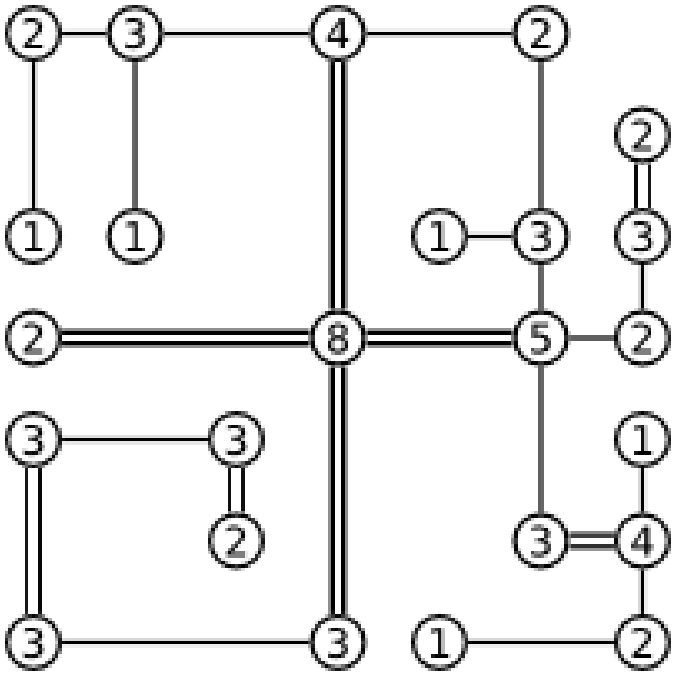
CSC14003 - Introduction to Artificial Intelligence

March 15, 2025

Contents

1 Overview

Hashiwokakero, also known as Bridges or Hashi, is a logic puzzle that challenges players to connect numbered islands with a specific number of bridges while following a set of simple rules. Published by Nikoli, this puzzle requires strategic thinking and careful planning to ensure all islands are interconnected without exceeding the allowed number of bridges per island. The game has gained popularity worldwide under different names, such as Ai-Ki-Ai in France, Denmark, the Netherlands, and Belgium. With its elegant design and logical depth, *Hashiwokakero* offers an engaging challenge for puzzle enthusiasts of all skill levels.



2 Project Description

Hashiwokakero is played on a rectangular grid with no standard size, although the grid itself is not usually drawn. Some cells start out with (usually encircled) numbers from 1 to 8 inclusive; these are the "islands". The rest of the cells are empty.

In this project, you will develop a **Hashiwokakero solver** using **Conjunctive Normal Form (CNF)** logic. to connect all of the islands by drawing a series of bridges between the islands. The bridges must follow certain criteria:

- They must begin and end at distinct islands, travelling a straight line in between.

- They must not cross any other bridges or islands.
- They may only run perpendicularly.
- At most two bridges connect a pair of islands.
- The number of bridges connected to each island must match the number on that island.
- The bridges must connect the islands into a single connected group.

An example Hashiwokakero puzzle and one of its two solutions is shown below.

In order to solve this problem, you can consider some steps:

1. **Define Logical Variables**: A logical variable is assigned to each cell of the matrix.
2. (Report) **Formulate CNF Constraints**: Write constraints for cells containing numbers to obtain a set of constraint clauses in CNF (note that you need to remove duplicate clauses)
3. (Implement) **Automate CNF Generation**: Generate CNFs automatically.
4. (Implement) **Solve Using PySAT**: Using the pysat library to find the value for each variable and infer the result.
5. (Implement) *Apply A Star Search Algorithm*: Apply A* to solve the CNF.
6. (Implement) Compare with Other Methods: Program brute-force and backtracking algorithm to compare their speed (by measuring running time which is how long it takes for a computer to perform a specific task) and their performance with A*.

3 Requirements

3.1 Inputs

Students are required to design at least 10 different input files, named according to the structure `input-01.txt,input-02.txt, ..., input-10.txt`.

For example

```
0, 2, 0, 5, 0, 0, 2
0, 0, 0, 0, 0, 0, 0
4, 0, 2, 0, 2, 0, 4
0, 0, 0, 0, 0, 0, 0
0, 1, 0, 5, 0, 2, 0
0, 0, 0, 0, 0, 0, 0
4, 0, 0, 0, 0, 0, 3
```

where zeros represent empty spaces and other numbers present islands.

3.2 Outputs

The output for above example:

```
["0", "2", "=", "5", "-", "-", "2"]
["0", "0", "0", "$", "0", "0", "|"]
["4", "=", "2", "$", "2", "=", "4"]
["$", "0", "0", "$", "0", "0", "|"]
["$", "1", "-", "5", "=", "2", "|"]
["$", "0", "0", "0", "0", "0", "|"]
["4", "=", "=", "=", "=", "=", "3"]
```

where "|" means one vertical bridge, "$" means two vertical bridges, "-" means one horizontal bridge, and "=" means two horizontal bridges.

3.3 Programming language

The source code must be written in **Python** (3.7 or later). You are allowed to use any supporting libraries; however, the main algorithms directly related to the search process must be implemented by you.

3.4 Report

- Member information (Student ID, full name, etc.)
- Work assignment table, which includes information on each task assigned to team members, along with the completion rate of each member compared to the assigned tasks.
- Self-evaluation of the project requirements.
- Detailed explanation of each algorithm (implementation process, heuristic function, etc.). Illustrative images and diagram are encouraged. **Please DO NOT PUT TOO MUCH SOURCE CODE INTO THE REPORT**.
- Description of the test cases and experiment results (memory usage, time complexity, etc.) Highlight challenges and compare overall behavior of your algorithms.
- The report needs to be well-formatted and exported to PDF. If there are figures cut off by the page break, etc., points will be deducted.
- References (if any).

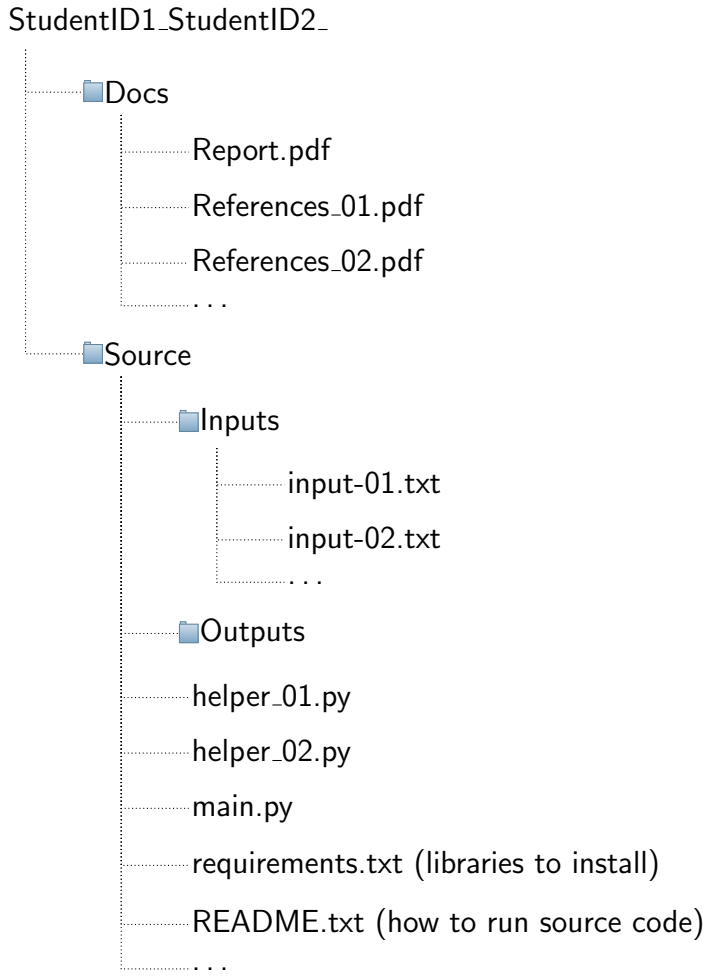
3.5 Demonstration videos

- Demo videos (recording the running process of your program for some test case) should be uploaded to YouTube or Google Drive, and the public URLs are included in the report.
- In the video, students should start from compiling or running their code, then walk through the key steps of the program's execution to make it easy to follow.

3.6 Submission

- Your report, source code and test cases must be contributed in the form of a compressed file (.zip, .rar, .7z) and named according to the format **StudentID1_StudentID2_...**
- If the compressed file is larger than 25MB, prioritize compressing the report and source code. Test cases may be uploaded to the Google Drive and shared via a link.

Please follow this he directory organization:


StudentID1_StudentID2_
Docs
Report.pdf
References_01.pdf
References_02.pdf
...
Source
Inputs
input-01.txt
input-02.txt
...
Outputs
helper_01.py
helper_02.py
main.py
requirements.txt (libraries to install)
README.txt (how to run source code)
...

4 Assessment

No	Criteria	Scores
1	Solution description: Describe the correct logical principles for generating CNFs.	30%
2	Generate CNFs automatically.	10%
3	Use the PySAT library to solve CNFs correctly.	10%
4	Implement A* to solve CNFs without using a library.	10%
5	Implement additional algorithms for comparison: 1) Brute-force algorithm to compare with A* (speed); 2) Backtracking algorithm to compare with A* (speed).	10%
6	Documentation and analysis: 1) Write a detailed report (30%); 2) Thoroughness in analysis and experimentation; 3) Provide at least 10 test cases with different sizes (7×7, 9×9, 11×11, 13×13, 17×17, 20×20) to verify your solution; 4) Compare results and performance.	30%

5 Notices

Please pay attention to the following notices:

1. This is a GROUP assignment. Each group has 4 members.
2. Duration: about 3 weeks.
3. Any plagiarism, any tricks, or any lie will have a 0 point for the course grade.
4. AI-Generated Content must lower than 30% in report.