AI project: the connection game

It is played on a rectangular (chessboard-like) board consisting of a number of rows and a number of columns that structure the board into small squares or cells. Every cell has (up to) four adjacent cells to the north, south, east, and west of it. In the initial configuration, every such cell contains different types of pipes. Those pipes may cross each other, go around the corner, connect one side of the tile to the opposite side, and so on. Including the empty cell, there are 6 different types of pipes, see Figure 1.

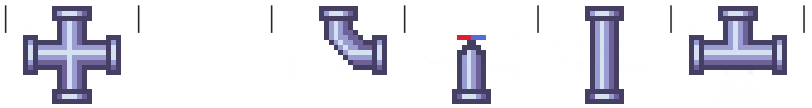

Figure 1: The six different pieces (including the empty one).

If there is a pipe running from the center of a tile to the middle of one of its sides, then this side is called **open**. Otherwise, the side is called **closed**. If the pipes are filled with water, then the system will possibly leak at an open side of some tile. The only way of preventing this is to have another tile with an open side in the adjacent cell, so that the water can flow on into the open pipe in the adjacent cell. This motivates the following definition:

Definition: Two tiles in adjacent cells form a safe pair if they either touch each other in open sides or touch each other in closed sides.

The goal of the game is to bring the pipe system into a safe state where all the pairs of tiles in adjacent cells form safe pairs. The pipe system is allowed to consist of several connected components in a safe system. An instance of the connection problem consists of a rectangular board and of rotated tiles in the cells. By default, the water entry point is the upper-left tile on its left, and the down-right tile on its right is the output. All border sides are considered as closed, except the entry point and the output point. Note that in this setting, there is no constraint on connecting the entry point to the output point. The question is to decide whether the system can be brought into a safe state, see Figure 2.

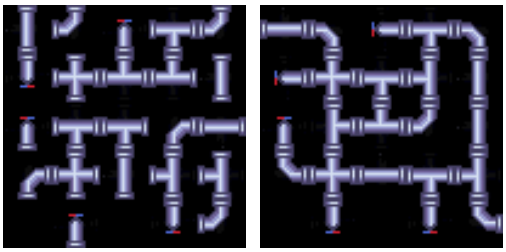

Figure 2: A starting game position (left) and its solution (right) on a 5×5 board.

During the first three hours of this project, you will only work on the preliminary study. The last six hours will be dedicated to the actual implementation. You will write an initial report containing all your responses to this preliminary study. The deadline for submission is set for **October 27, 2024**, at 8 PM. This report should reasonably be no less than 2 pages and no more than 5 pages. For compliance purposes, you will use the ACM template, available at : https://www.acm.org/publications/authors/submissions (LaTeX or docx source, pdf required).

Preliminary study

1. Modelize the problem in terms of (E, i, F, O, P, C)
2. Study the stucture of the search tree:
   - what is its size ?
   - is it symmetrical ?
   - what is its branching factor ?

3. Propose a strategy for puzzle generation.
4. Given your anwsers, select, and justify this choice, a convenient way of exploring the search tree.
5. Could you elaborate heuristics to help out the search ? (e.g. constraints, scheduling, state evaluation functions,...)
6. There exists a variant of this game where a subset of the pieces are not placed on the board, but on its side, and are to be correctly placed. What is the impact of this change on your modelization and your complexity study.

The final report will include, in addition to all elements from the preliminary study, a detailed description of the implementation and responses to the questions indicated in the *Implementation* section. Once again, at least 5 pages and at most 12 pages seems reasonable here, still using the same ACM template. You will also be required to provide a Git link to your project containing the code necessary to reproduce the results described in the report. The deadline for submission is set for **November 29, 2024**, at 8 PM.

Implementation

1. Write a function that, given a board size, generate a starting game position. Make sure that the problem has at least one solution.
2. (Optional) Write a function drawing/updating the game on the screen. You should have the necessary material to do it rather quickly.
3. Write a function that verify if a state is terminal.
4. Implement the final function searching for a solution of a given problem.
5. Incorporate the heuristic(s) defined during the preliminary study into your search.
6. Using a sufficiently large grid, perform a comparative analysis between your first solution and the different heuristic(s) that you proposed.