# Rivers and Stones

Assignment 5

October 30, 2025

## 1 Introduction

For Assignment 5, we will continue using the game of Stones and Rivers. As you know the game is based on using stones to score in the opponent's Score Area while strategically positioning Rivers to enable sweeping movements across the board. Stones serve as the scoring side of each piece, while Rivers act as the mobility side, guiding Stones along their flow direction. For the purpose of the assignment, there shall be a total time for all moves played by the bot in the game.

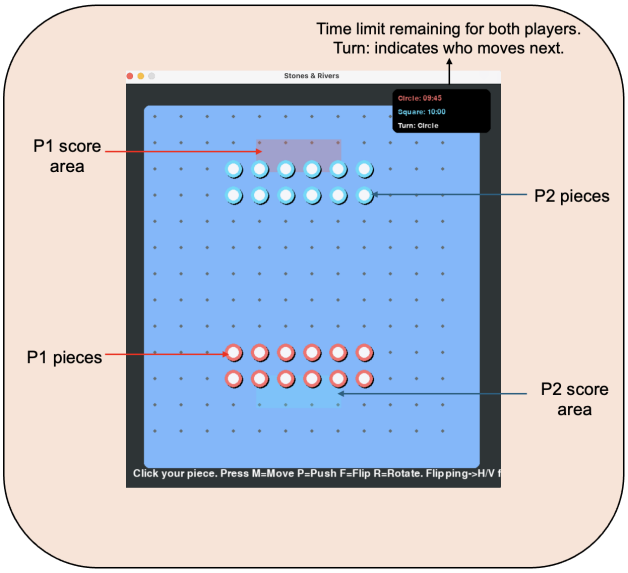



Figure 1: Example setup: A fully prepared board ready for the game.

### 1.1 Game Setup

- **The Board:** The game is played on three board configurations — small ($13 \times 12$), medium ($15 \times 14$), and large ($17 \times 16$) — with the scoring area (SA) containing 4, 5, and 6 spaces respectively.
  - **Small Board (13×12):** Includes 24 starting positions (12 for each player) and two score areas. Each score area (SA) contains 4 spaces where stones must be placed to achieve victory.
  - **Medium Board (15×14):** Includes 28 starting positions (14 for each player) and two score areas. Each score area (SA) contains 5 spaces where stones must be placed to achieve victory.

- **Large Board (17×16):** Includes 32 starting positions (16 for each player) and two score areas. Each score area (SA) contains 6 spaces where stones must be placed to achieve victory.

- **The Pieces:** Each player begins with 12, 14, or 16 identical pieces depending on the board configuration. Each piece has two sides: a **Stone side**, which is used for scoring, and a **River side**, which allows pieces to travel across the board through sweeping directional movement.
  - **The Stone Side:** A piece with no mark in between is called a *stone*.
  - **The River Side:** A piece with a thin line in between is called a *river*. The line can be horizontal or vertical, indicating the direction of the river (see Fig. 2).

## 1.2 Objective

A player is declared as winner, if

- You place the required number of your stone sides in your score area (SA)(present near the opponent's pieces at the beginning), with the stone face-up (not the river side), before your opponent achieves this goal.
- If your opponent runs out of time, and none of the players has successfully achieved the above goal, then you are the winner, and vice-versa.

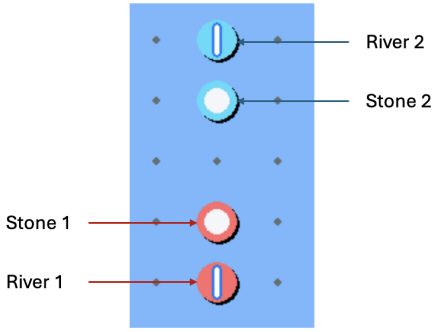


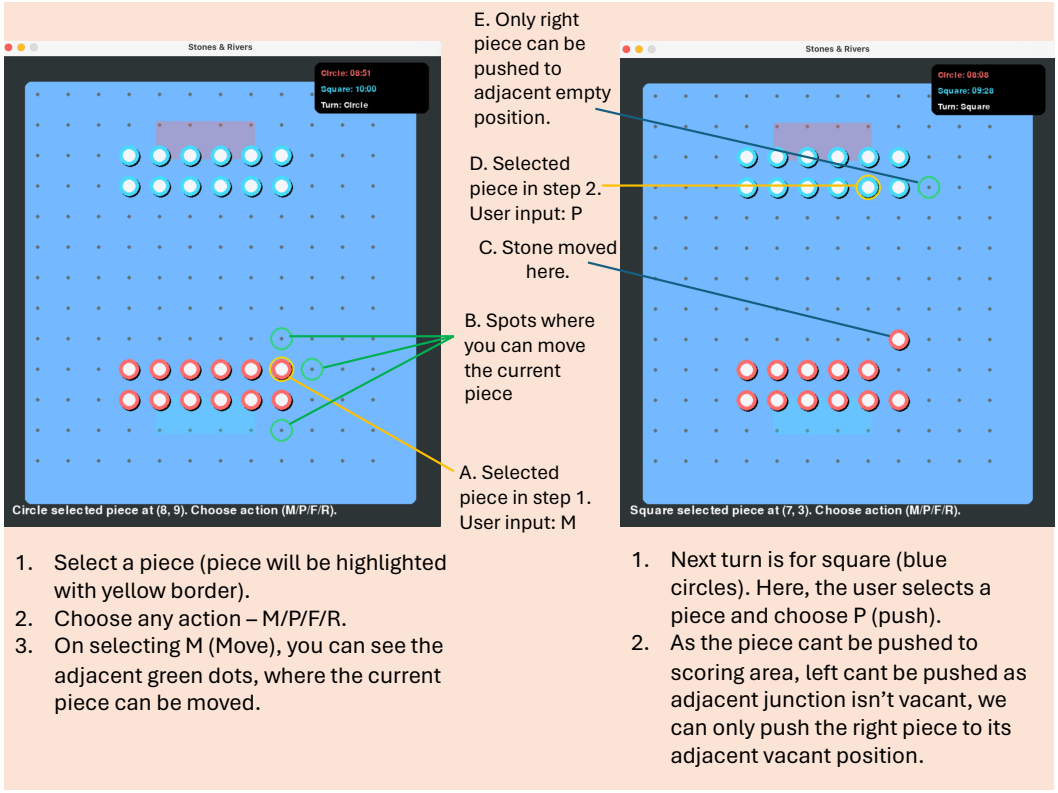

Figure 2: River and Stone pieces

## 1.3 Game Play Overview

On your turn, you may choose one action from the following four options:

- Move a Piece (Stone or River).
- Push a Piece (using either a Stone or a River).
- Flip a Piece and Rotate (switch Stone to River or reverse order, choose orientation if River).
- Rotate a River (change its flow direction).

1. **Movement Rules**
   - **General Movement**
     - Both Stones and Rivers can move.
     - Pieces move exactly one step up, down, left, or right, however, there are restrictions.
     - Movement restrictions:
       * Pieces must be placed on the intersections of the grid (all the dots visible are intersection of the grid, no half-movements in between these points are possible.
       * Cannot move into or pass through the opponent's Score Area (SA).
       * Cannot leave the board.
       * Cannot stack multiple pieces on the same intersection.
     - See Fig. 3 to understand with an illustrative example.
   - **River Movement**
     - If you step onto a River, it may travel any number of spaces along the River's flow direction.




1. Select a piece (piece will be highlighted with yellow border).
2. Choose any action – M/P/F/R.
3. On selecting M (Move), you can see the adjacent green dots, where the current piece can be moved.

1. Next turn is for square (blue circles). Here, the user selects a piece and choose P (push).
2. As the piece cant be pushed to scoring area, left cant be pushed as adjacent junction isn't vacant, we can only push the right piece to its adjacent vacant position.

Figure 3: Illustrating move and push. As the rules describe, during move, we cannot jump to a non-vaccant cell, nor inside our scoring area (SA). Note correction in the column 1 (left) figure, selected piece can push the piece backward but cannot move directly to that place.

- Travel stops when encountering another piece with stone side up (yours or your opponent's).
- Both players' River pieces can be used for movement.
- If you land on another River, continue moving in its new direction.

- **Clarifications**
  - Stone and river both pieces can use River movement.
  - You may continue the movement until blocked by another piece or you reach the end of the board; there is no hard-and-fast rule upto which you want to continue or where you want to stop.
  - River movement cannot enter or cross the opponent's Score Area (SA). (see Fig. 4)

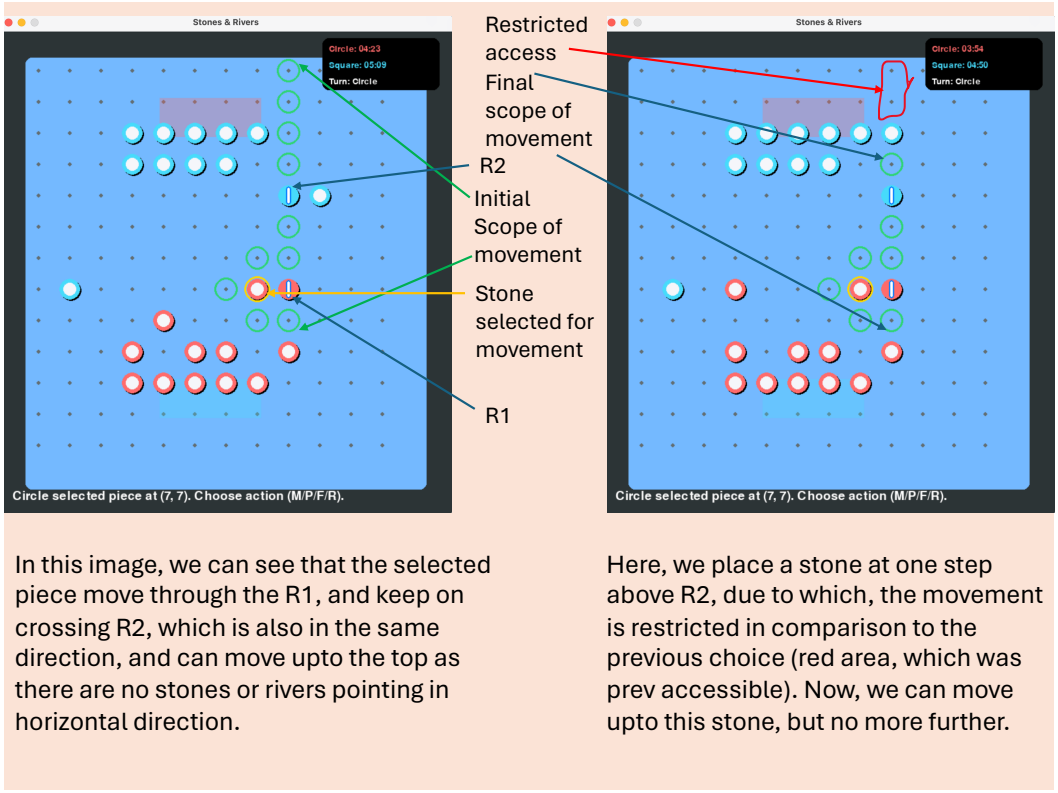
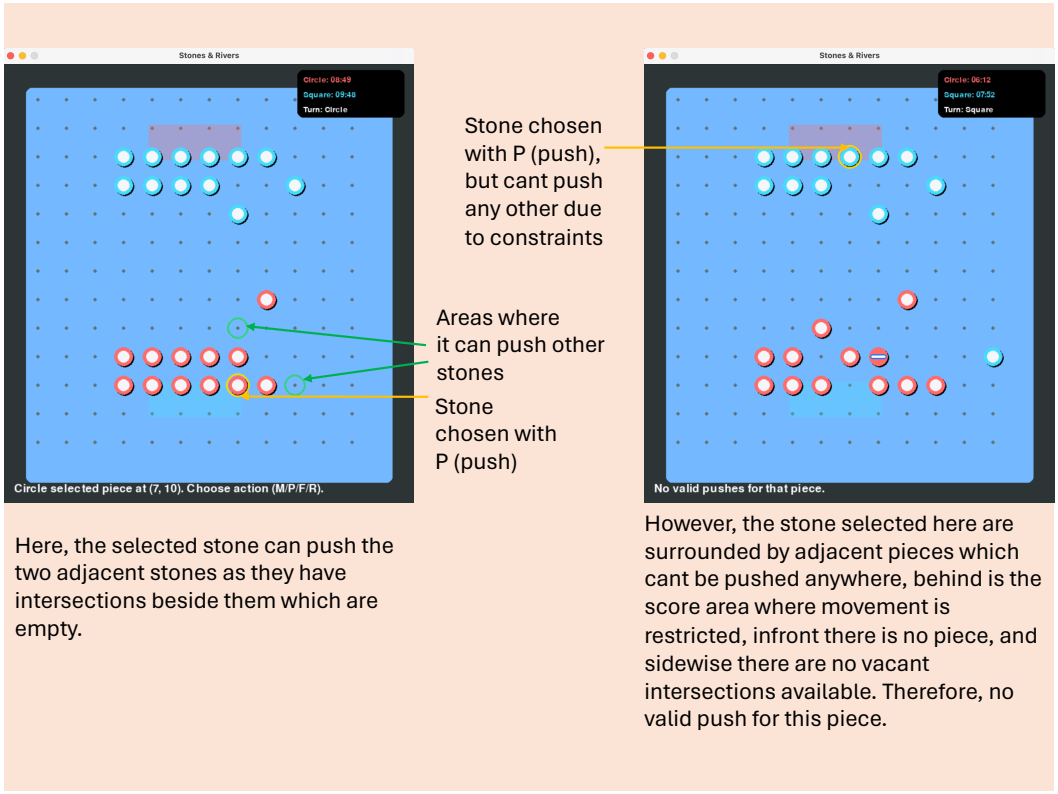



Figure 4: River movement with restriction example.

2. **Pushing Rules**

- **Pushing with a Stone Piece**
  When a Stone moves into a space already occupied by another piece, it may push that piece(stone side) forward by exactly one step in the same direction. This push is only valid if the next intersection is empty and lies within the boundaries of the board. A push is considered invalid if the displaced piece would be forced off the board, moved into the opponent's Score Area (SA), or has no valid move in the chosen direction. In addition, Stones cannot push a series of pieces lined up together—only a single piece may be pushed at a time (see Fig. 5).

- **Pushing with River Piece**
  Rivers allow a more flexible form of pushing. When a River moves into an occupied space, it




Here, the selected stone can push the two adjacent stones as they have intersections beside them which are empty.

However, the stone selected here are surrounded by adjacent pieces which cant be pushed anywhere, behind is the score area where movement is restricted, infront there is no piece, and sidewise there are no vacant intersections available. Therefore, no valid push for this piece.

Figure 5: Illustrating pushing with a stone piece.

can push the other piece any number of spaces along one of its flow directions, with the distance chosen by the active player. The pushed piece must follow the rules of River movement during this displacement and cannot push other pieces further along its path. After the push is resolved, the River that initiated the action is flipped over to its Stone side. Both your own pieces and your opponent's may be pushed in this way, and the displaced piece must be a Stone only. However, the push is invalid if no valid landing space exists or if the resulting move recreates the exact same board position as the previous turn (see Fig. 6).

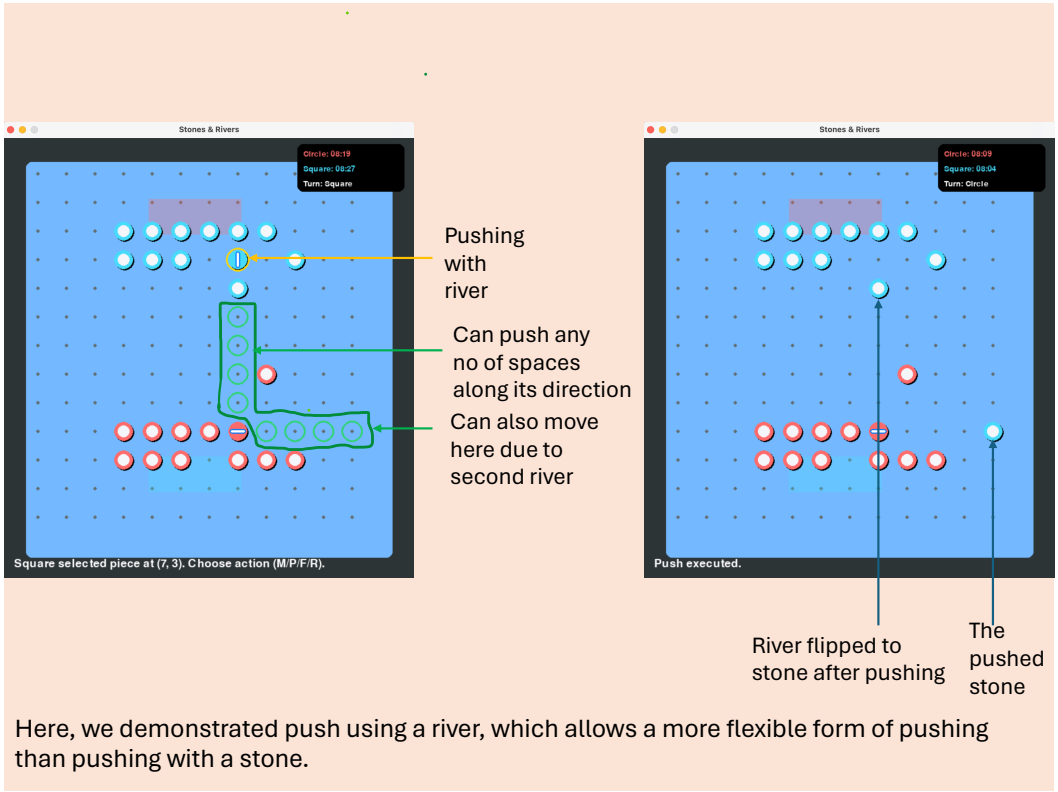



Figure 6: Pushing with a river piece, illustrating the difference wrt pushing with a stone piece.

3. **Rotating a River** Instead of moving, a player may rotate one of their River pieces by 90 degrees, switching it from vertical to horizontal or vice versa. Rotation cannot be skipped by keeping the same orientation, nor can it be rotated by 180 degrees. This action consumes the entire turn.

4. **Flipping and Rotating a Piece** Players may flip one of their pieces at any intersection. A Stone flipped into a River further requires choosing its orientation (vertical or horizontal), while a River flipped into a Stone becomes a scoring piece. This action also takes the whole turn (see Fig. 7).

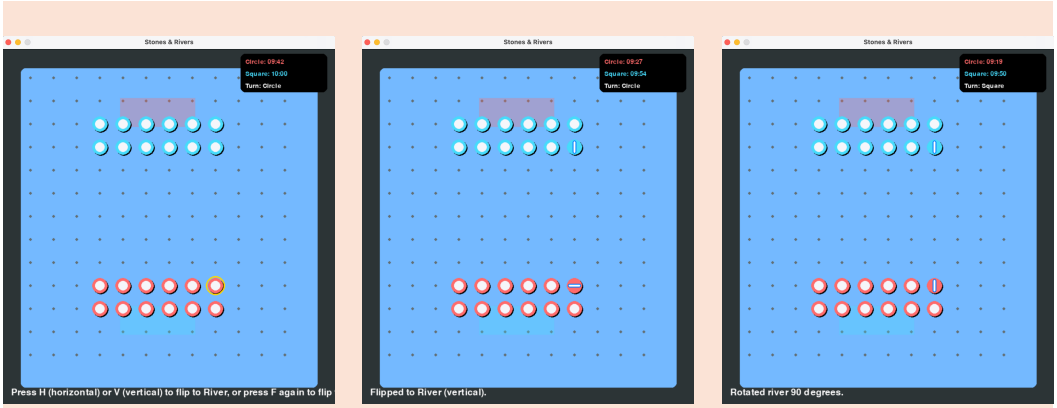



Figure 7: Flipping a piece and rotating a river.

### 1.4 Scoring Conditions

- Only Stone sides can score.
- Pieces in the Score Area (SA) remain in play. They may still be flipped, rotated, or pushed.

## 2 Scoring

The scoring pattern for each game is described below.

### 2.1 Game Ending conditions

The game is considered to be ended in either of the following scenarios:-

- A player immediately wins once their scoring area (SA), ranging from 4 to 6 spaces depending on the board configuration, is completely filled.
- The timer runs out for both players, the game will be treated as a draw.
- If both the players have played maximum number of allowed moves (500 here), the game will be treated as a draw.
- In case of timer running out for either player, other player will be treated as winner.

### 2.2 Total Score for a game ending in Victory

The winning player gets a victory score of 100 minus the score of the losing player. Let the number of pieces of the losing player with the stone side up in its scoring arena be $n_{Lose}$ and the number of pieces (again with the stone side up) of the losing player that can reach their own scoring arena in one move be $m_{Lose}$. Let $n_{\text{total}}$ be the total number of pieces per player i.e 12, 14 and 16, and let $W$ be the reward of placing a stone in the scoring area (SA). $W = 10$ for $13 \times 12$, $W = 8$ for $15 \times 14$, and $W = 6.5$ for $17 \times 16$. Then, the total score for either player is computed as follows. ($\mathbf{1}_{[X]}$ is the indicator function that is 1 when outcome $X$ holds for the player and 0 otherwise.)

$$\text{Total Score} \;=\; \mathbf{1}_{[\text{Win}]}\,100 \;+\; W * \left(\mathbf{1}_{[\text{Lose}]} - \mathbf{1}_{[\text{Win}]}\right)\left(n_{Lose} + \frac{m_{Lose}}{n_{total}}\right) \tag{1}$$

### 2.3 Total Score for a game ending in a draw

$$\text{Total Score} \;=\; \text{Draw Score} \;+\; \frac{\text{Margin Score}}{4} \tag{2}$$

If the game draws, each player gets a **draw score of 30** (this is to discourage drawing of games) in combination with a margin score. Consider the following notation for calculating the margin score for the players:-

- $n_{self}$: Denotes the number of player's pieces, with the stone side up, that are in the player's scoring area.
- $n_{opp}$: Denotes the number of opponent's pieces, with the stone side up, that are in the opponent's scoring area.
- $m_{self}$: Denotes the number of player's pieces, with the stone side up, that can reach the player's scoring area in one move.
- $m_{opp}$: Denotes the number of opponent's pieces, with the stone side up, that can reach the opponent's scoring area in one move.

With the above notation, the margin score for either player is computed as follows.

$$\text{Margin Score} \;=\; 39 \;+\; \left(\left(n_{self} + \frac{m_{self}}{n_{total}}\right) - \left(n_{opp} + \frac{m_{opp}}{m_{total}}\right)\right) \tag{3}$$

## 3 Tournament Structure

We will have three tournaments, one for each board type. The marks will be equally distributed for the three tournaments. Each tournament comprises of the following two stages.

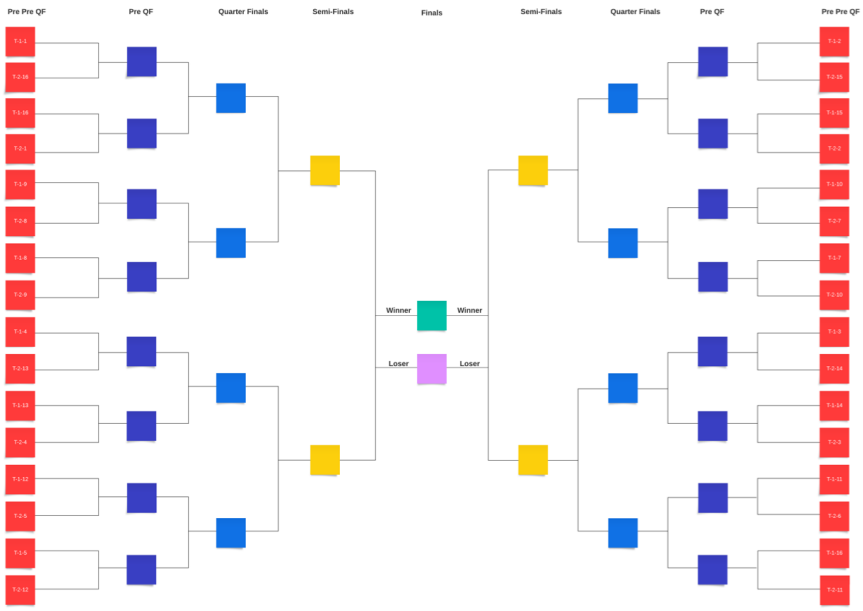



Figure 8: Game Tree

### 3.1 Group Stage

All teams will be distributed among 16 groups, $G_1, G_2, \ldots, G_{16}$, based on their assignment 2 $A_2$ scores in a modulo-16 manner, ensuring that each group reflects the overall performance distribution observed in $A_2$ . Each team plays a battle with every other team within its group. The top two teams in each group advance to the knockout stage. Each battle comprises two matches, and the scoring of each match is as described in the scoring section. Each player is allocated a total time of 2, 3, or 4 minutes per match for tournaments played on boards of size $13 \times 12$ with 12 pieces, $15 \times 14$ with 14 pieces, and $17 \times 16$ with 16 pieces per player, respectively. Average score is computed and teams are ordered based on their scores. In case of draw, margin score will be used to break the tie. If a tie still remains, a coin toss determines the result. The top two performing teams from each group advance to the knockout stage.

### 3.2 Knockouts

Let $T_{j,i}$ denote the team that ranked $j$-th in group $G_i$. The knockout stage begins with the corresponding game tree (see Fig. 8), with each winning team advancing up the ladder. Victories are determined in the same manner as in the group stage. In the event of a tie, an additional battle is played, with each player allocated 1 minute per match. If this additional battle also results in a tie, another battle is played with a time allocation of 30 seconds per player. If the tie persists, the performance in the group stage is considered. For any further ties, the result is decided by a coin toss.

## 4 Tournament Evaluation

Each tournament is worth 4 points and is graded independently. For each tournament, the following score allocation is applied:

- Winner: 12 points (will not play other tournaments)
- Runner-up: 4 points

- Second Runner-up: 3.6 points
- Third Runner-up: 3.5 points
- Losing Quarter Finals: 3.2 points
- Losing Pre-Quarter Finals: 2.8 points
- Losing Pre-Pre Quarter Finals: 2.4 points
- 3rd in Group: 2 points
- 4th in Group: 1.66 points
- 5th in Group: 1.33 points
- 6th or 7th in Group: 1 point

# 5 Code Review

This assignment package provides the complete setup for running the game. The game server, implemented in Python, manages the flow of the match, while the client server includes starter Python code for your player. On the client side, you are expected to implement your AI bot, which will communicate with the game server to send moves.

The full codebase is available at (A5 - Code). Please refer to the README in the repository for detailed instructions on how to set up, run, and interact with the system. Note that the provided game server will also be used for evaluation purposes.

# 6 Interaction with Engine

The complete codebase for the game can be accessed through the repository link provided, which also contains a detailed README file with further instructions. The game has been developed using Python (version 3.x).

## Installing Dependencies

A requirements file is included in the repository to facilitate installation. The dependencies can be installed by executing the following command:

```
pip install -r requirements.txt
```

## Running the Engine

The engine is designed to operate in two modes: either through a Graphical User Interface (GUI) or via the Command Line Interface (CLI). The execution commands for the two modes are given below.

For running with GUI:

```
python gameEngine.py --mode hvai
```

For running with GUI with board size. There are 3 board size mapped to small, medium and large argument. Where small is default $13 \times 12$, medium is $15 \times 14$ and large is $17 \times 16$.

```
python gameEngine.py --mode hvai --board-size small
```

For running with CLI:

```
python gameEngine.py --mode hvai --nogui
```

Write your code in student_agent.py and for running it you can either use

```
python gameEngine.py --mode aivai --circle random --square student
```

or you can use:

```
python gameEngine.py --mode aivai --circle random --square student --nogui
```

# 7 Other Details

1. You are not allowed to use any libraries other than those that come with the language. This means that libraries such as boost are not allowed.
2. Your implemented algorithm should be single-threaded. You are not allowed to make use of multiple threads, asyncio or any other form of parallel execution.
3. You can use torch (*pytorch* > 1.13.1). Ensure that it compatible with cuda 11.7.
4. You may use any techniques of your choice, provided they comply with the stated constraints. These may include, but are not limited to, value functions, tablebases, neural networks, Monte Carlo Tree Search (MCTS), knowledge distillation, or self-play–based learning. Note that this is a competitive assignment, and the TA bots should not necessarily be considered the optimal benchmark for performance.

# 8 Submissions Instructions

- Submit your code for the Player in a zip file named in the format `<EntryNo1_EntryNo2>.zip`.
- Ensure that your entry numbers are correct (Kerberos IDs are not required).
- On unzipping, the following files must be present:
  - `student_agent.py`
  - `report.txt`: It must contain entry numbers of all the members on the First Line itself, before anything else.
- For C++ users, the submission must include the following files:
  1. `MakeFile` or `CMakeLists.txt`
  2. `compile.sh`
  3. `student_agent_cpp.py`
  4. `report.txt`
  5. `student_agent.cpp`
- In case you have used a machine learning or deep learning model, you must submit the training code along with the model weights. You are required to upload the trained model on Google Drive, set the access permissions to *"Anyone with the link"*, and include the link in the last line of your submission `report.txt`. Additionally, you must submit a screenshot showing the timestamp of the uploaded model along with your submission.
- Submissions not following these requirements will be penalized.
- Your code must compile and run on the provided VMs.

### Report Requirements

First Line should contain entry numbers of all the members seperated by comma. Second line of `report.txt` should start with this honor code. "Even though I/we have taken help from the following students and LLMs in terms of discussing ideas and coding practices, all my/our code is written by me/us." Third line should mention names of all students and LLMs you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you did not discuss the assignment with anyone, simply write `None`. You may include additional content after this line, but it will not be considered for grading.

### Important

Your submission will be auto-graded. It is essential that your code strictly follows the input/output specifications. Any failure to comply will result in a significant penalty.

## 9 General Guidelines

- You can work individually or in pairs.
  - If working in a pair, include your team details in the write-up.
  - It is recommended to choose a partner you communicate well with, as this will help in the more open-ended final assignment (Phase 2).
  - You must continue with the same partner for the final phase, except if your partner drops the course. In case you have changed partner, seeding will be decided based on the average score of both the partners.
- Your submission must be entirely your own work.
- You cannot use built-in libraries/implementations for search or scheduling or optimization algorithms or gameplay. To request use of a library, use Piazza, and TAs will respond in 48 hours.
- Your implemented algorithm should be single-threaded. You are not allowed to make use of multiple threads, asyncio or any other form of parallel execution.
- You must not discuss this assignment with anyone outside the class. You cannot ask LLMs to write code for you. However, you are allowed to give the LLM your code in case you are stuck in debugging, so that you can learn about your mistakes fast, and do not waste time in debugging. Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team. Please read academic integrity guidelines on the course home page and follow them carefully.
- Please do not search the Web for solutions to the problem.
- Please do not use ChatGPT or other large language models for creating direct solutions (code) to the problem. Our TAs will ask language models for solutions to this problem and add its generated code in plagiarism software. If plagiarism detection software can match with TA code, you will be caught.
- Your code will be automatically evaluated. You get a minimum of 20% penalty if your output is not automatically parsable.
- We will run plagiarism detection software. Any team found guilty of either (1) sharing code with another team, (2) copying code from another team, (3) using code found on the Web, will be awarded a suitable strict penalty as per IIT and course rules.