EN843407 - 2568/2 - Programming Assignment #2

Zero-Sum Games

Rockhart, an enthusiastic board game player, appreciated strategic plannings on board landscapes. He came across the classic Connect Four game among his collection. He dug deep into the game, playing endlessly until mastery became second nature, determined to dominate every component.

Connect Four is a classic two-player connection game played on a vertical grid of six rows and seven columns. Each player takes turns dropping one disc into any column, using either red or yellow discs. The goal is to connect four of your own discs vertically, horizontally, or diagonally before your opponent does. The game is simple, yet tactically sophisticated, requiring careful strategy to outperform your opponent.

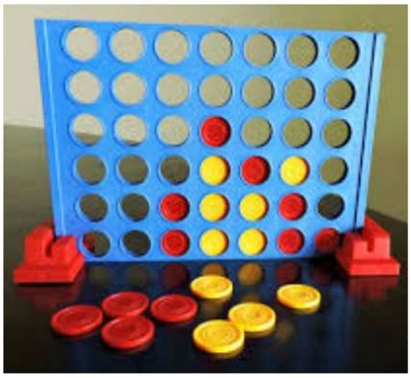


Image source: https://www.pathstoliteracy.org/adapting-connect-four-players-visual-impairments/

After mastered Connect Four, Rockhart became quite bored. To regain his interest in the game, he began creating new versions by expanding the board and changing the rules to require a greater number of connected pieces to win. **Connect-X** was the name he gave to this newly invented version. This new version requires even more sophisticated strategy planning, and it transforms his once-tamed hobby into a dynamic playground for his inventive spirit.

Game Description:

The rules of **Connect-X** are the same as the classic Connect Four, but the board's size is now ***N*** x ***M***, and the ***X*** connecting (horizontal, vertical, or diagonal) pieces are required to win the game. Just in case, you are not familiar with the rules of Connect Four, here they are:

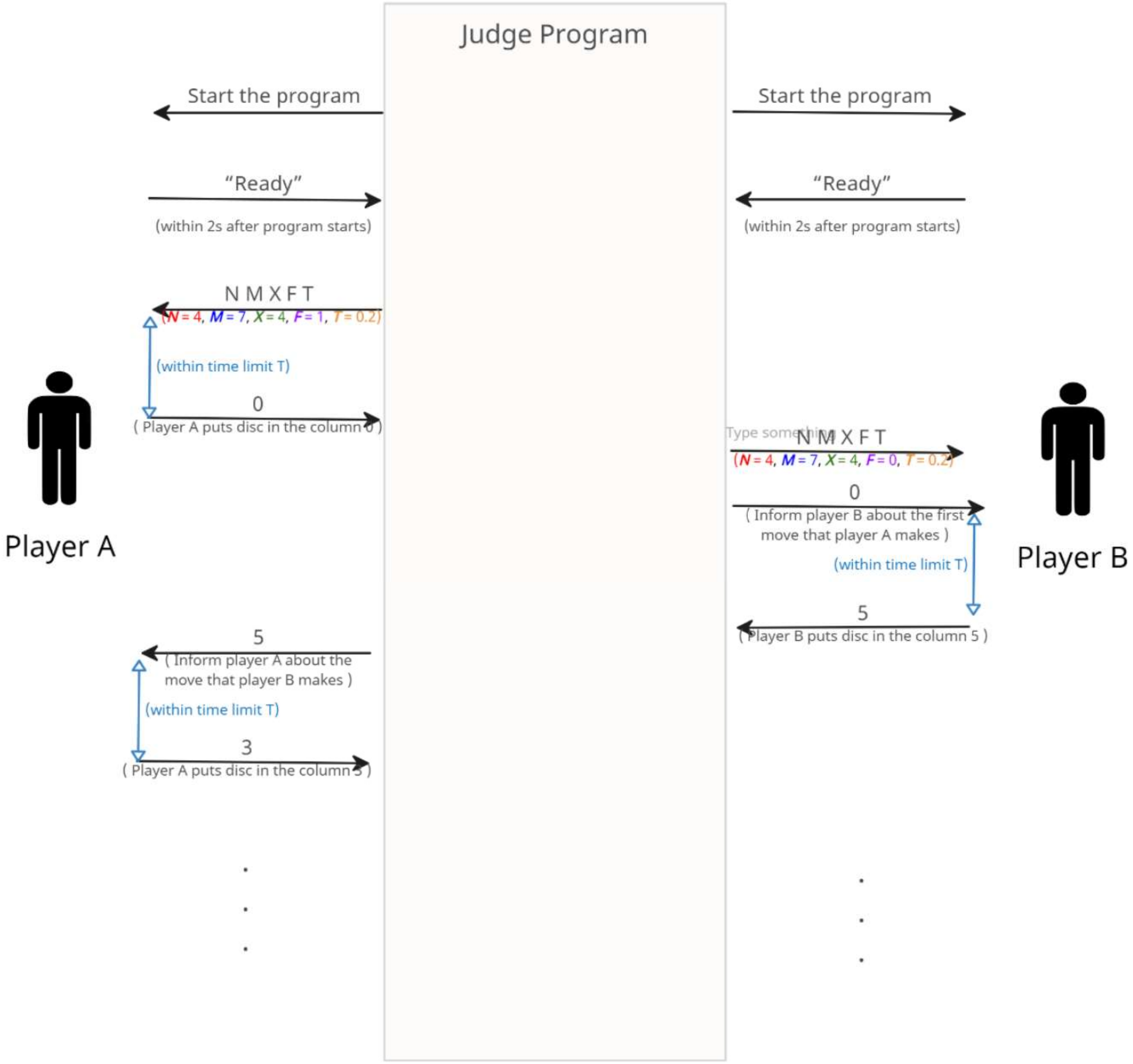
- **Board**: The standard game is a vertical ***N*** x ***M*** grid. Each turn, a disc will be dropped from the top of the grid. The disc falls down in a column either until they land on top of another disc, or until they reach the bottom level.
- **Start**: The player who gets to make the first move is chosen at random.
- **Moves**: Each player chooses a column to drop the disc into. Choosing a column that is already completely filled or an undefined column is considered an illegal move.
- **Time Limit**: There is a time limit for each player to make a move.
- **End of the Game**: The game ends in any of the following scenarios.
  - One of the players gets ***X*** discs in a row (horizontally, vertically or diagonally). This player wins and the other loses.
  - The board is full but none of the players managed to get ***X*** discs in a row. The game ends in a draw.
  - One player makes an illegal move. In this case the other player wins.
  - If one player fails to respond within the time limit, then the other player wins.

Your Task:

You will implement an adversarial search algorithm to play the ***Connect-X*** game. To be more specific, this variant of the game consists of five main variables:

- ***N*** - The number of rows
- ***M*** - The number of columns
- ***X*** - The number of connecting pieces required to win
- ***F*** - This indicates which player moves first. The value ***F* = 1** will be sent to the player who moves first, and ***F* = 0** will be sent to the player who moves seconds.
- ***T*** - Time limit per move, in seconds.

For example, ***N*** = 4, ***M*** = 7, ***X*** = 4, ***F*** = 0, ***T*** = 0.2 means the board is 4 rows x 7 columns, the player who aligns 4 connecting pieces first wins the game, your opponent moves first, and you have 0.2 seconds to decide each move. The figure below shows the example game, where the player A moves first.


Judge Program
Start the program
Start the program
“Ready”
(within 2s after program starts)
“Ready”
(within 2s after program starts)
N M X F T
(N = 4, M = 7, X = 4, F = 1, T = 0.2)
(within time limit T)
0
(Player A puts disc in the column 0)
Player A
Type something
N M X F T
(N = 4, M = 7, X = 4, F = 0, T = 0.2)
0
(Inform player B about the first move that player A makes)
(within time limit T)
5
(Player B puts disc in the column 5)
Player B
5
(Inform player A about the move that player B makes)
(within time limit T)
3
(Player A puts disc in the column 3)
.
.
.
.
.
.

Workflow of the Program:

The game will be played through the standard input and output, where each player takes turns deciding each move. Here is the workflow of your program:

- Program starts
- Your program makes any initialization as needed, and must print “Ready” **within two seconds** after the program starts.
- Your program reads one line from the standard input, which provides information about the game ***N M X F T .*** Each value is separated with a space character.
- Then, the game starts.Your program must take turns reading inputs for the opponent’s moves, and writing outputs to indicate the next move you want to make. Your program must print out your move **within the time limit *T*, starting at the moment that the opponent’s move is sent to you**.
    - If you move first, your program must print the first move within the time limit ***T***, after the line ***N M X F T*** is sent to you.
    - If you move second, the line ***N M X F T*** is sent to you, then **immediately** followed by the opponent’s first move.

Starting Code:

You are provided with some simple programs:

- *rand.py* program basically randomly selects one of the available columns.
- *left.py* program always selects the left-most available column.
- *right.py* program always selects the right-most available column.

You can test your program by:

- Playing with your program through standard input and standard output. However, you will have to implement a way to check whether your program decides each move within the time limit and within the memory limit.
- A simple version of the grading program, *judge.py*, is provided. It takes seven command line arguments:

  python judge.py <path-to-programA> <path-to-programB> <N> <M> <X> <F> <T>

  In this case, F = 1 means program A moves first.
    - For example, *python test.py rand.py left.py 6 7 4 1 2*

If you run your program on Windows, please use *judge_Windows.py*

If you run your program on Mac, please use judge_Mac.py

Grading Criteria:

Part 1: Evaluating against the baseline (500 points)

Your program will be played against baseline programs. You will receive score if your program consistently win or tie with the baseline programs

- (150 points) The board size is 4 x 7, time limit is 1 second. The baseline uses a basic minimax algorithm.
- (150 points) Vary board size and time limit. The baseline uses a minimax algorithm with iterative deepening and the baseline heuristic*.
- (200 points) Vary board size and time limit. The baseline uses a minimax algorithm with alpha-beta pruning, iterative deepening and the baseline heuristic*.

The baseline heuristic is similar to the example we saw in Problem 5.9 from the textbook.

Part 2: Evaluating against the other students' programs (500 points)

A tournament will be hosted, and students' programs are the participants. Your program will earn a score based on your ranking and the number of wins/ties/losses.

Submission:

As automatic scripts will be used for grading, to ensure that your submission will be graded, please follow the instructions below:

- Compile the following files into **one .zip file**
    - Your ***hw2.py*** file (do not rename the file), with your name(s) and ID at the top.
        - Please make sure your file does not import functions from other local files (e.g. judge.py, rand.py, …); otherwise, it will not run during the grading.
    - README file explaining about the algorithm / heuristic function that you use, with your name and ID at the top
- Name your zipped file as ***<ID>.zip*** (without a hyphen)***.*** For example, ***1234567890.zip .***

Important Notes:

- You are encouraged to discuss with your friends about how to solve this assignment, but please do NOT share your code, or tell others what to type.
- You may not use a code generator tool, such as ChatGPT, GitHub Copilot, or other similar software. The purpose of this assignment is to develop your understanding and implementation skills in search algorithms; relying on automated code generation undermines the learning process and creative problem-solving skills.

If you have any questions, please feel free to contact the instructor
sarunpa@kku.ac.th