UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Introduction to Artificial Intelligence
Exercise 1 - reflex agent

September 21, 2022

1. REFLEX AGENT

The task is to create a reflex agent that will play the TicTacToe game (and Gomoku).
This task is for **1 point.**

Reflex agent is a simple type of agent (player), which decides based only on the actual state of its world. It does not remember preceding development of the game and **does not plan the upcoming moves**.

Program:

In the file *games.py* is defined a class *Game* and from that derived *TicTacToe* and *Gomoku*, which you do not need to modify. You will only work with *player.py*, where is defined an abstract class *Player* with an obligatory function *choose_move*. There is a class *MyPlayer*, which you need to program. At the end of the file you can choose which version of the game you want to play.

Your job is to finish the function **MyPlayer.choose_move(game, state)** in a way, that it makes "reasonable" move in the game of TicTacToe. Function gets two arguments:

- **game** - instance of class **Game** in which are already implemented all the functions regarding the game itself (see below)
- **state** - actual state of the game (state of the board, player on move, ...), on which we want to respond with the best possible action (best move). You don't need to parse the state, only use it as an argument into already prepared functions (see below).

Output of the function is an action (a move, which is to be played) – an integer from list of possible actions **game.actions(state)** for a given state **state**.

A few functions of class **Game**, which can be helpful:

- **game.player_at_turn(state)** – returns, which player is currently (in state *state*) on move ('X' or 'O')
- **game.other_player(player)** – as an argument gets a letter of a player and returns the letter of the other player

- **game.board_in_state(state)** – returns the state of the board *(list(list(char)))* in the state *state*
- **game.w** – width of the board
- **game.h** – height of the board
- **game.k** – the amount of characters in a row a player needs to have in order to win the game
- **game.actions(state)** – returns a list *(list(int))* of all possible actions (valid actions) from state *state*
- **game.state_after_move(state, a)** – returns new state, which arises from performing action *a* from state *state*
- **game.is_terminal(state)** – tests, if the state *state* is terminal (if it is a win/loss situation or a draw)
- **game.utility(state, player)** – returns the 'utility' of state *state* for player *player* : 1 if won, -1 if lost, 0 otherwise. Keep in mind, that the individual utilities are known only at the end of the game. There is no sense in asking for utility in state *state*, until it is not terminal.

Simple examples of these functions are included in codes.

At the end of the code the games are launched – uncomment whatever you will need. When debugging we recommend playing computer vs human. For the assignment evaluation we will use the last two rows (*play_n_games*). Your agent should be convincingly winning. The grading will consist of four runs of *play_n_games* - 2x against a random player and 2x against a player designed by us (albeit a weak one).
Beating the random player in Gomoku is really simple, however when playing TicTacToe in a standard layout (3x3 grid) it is expected that your score after n games is at least 0.4.

We will not grade the agent's runtime, however if it takes too much time to choose a move we can penalise the homework. It is expected for your agent to make move in TicTacToe almost instantly (and in a second about 100 games can be played), on the other hand when playing Gomoku one game can take up to several seconds.

After submitting this homework from all of you we will run a tournament of your agents in TicTacToe and Gomoku. The best three players in individual games will be rewarded with **0.6**, **0.4** and **0.2** bonus points (for 1st, 2nd and 3rd place respectively).