Comp 202 - Fall 2021
Homework #2
Due date - 23:59 28/11/2021

In this homework, your task is to complete an implementation of the Monte Carlo Tree Search (MCTS) algorithm for the Tic-Tac-Toe game. MCTS is a search algorithm that is specialized in solving board or card games, in which the search tree size increases rapidly. In order to implement this algorithm we are going to use a tree data structure. You are given the main class, a class that describes the board of the Tic-Tac-Toe game, and the node class that denotes a node in the tree. The main class executes a number of iterations of the MCTS algorithm so that the program 'learns' how to play it. You do not have to modify this class as it should function as a way for you to test the methods you will implement. The MCTS algorithm makes use of a tree data structure to store the states in the board game, so every combination of the gameplay can be achieved if the tree is exhausted. Each of the nodes of the tree stores the board configuration for that node, the total accumulated score achieved by passing through that node, and the number of games that have passed through this configuration. We firstly start with an empty Tic-Tac-Toe board. From here, there are 9 possible moves the first player can make, one for each of the squares in the 3 by 3 grid. An example of the root node of the tree and its children for the Tic-Tac-Toe game is shown in Figure 1. In the beginning

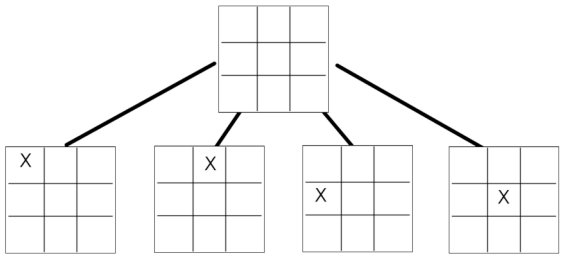

Figure 1: Tic-Tac-Toe tree

since none of the possible games have been tried before we just choose one at random. After that the program simulates the gameplay between to players that always choose a random square to play in. When the simulation is finished we record the score achieved in it by updating the scores of the last node before the simulation begins and all of the nodes in the path from the root to that node, since they are needed configurations to arrive to the last node. By performing this process many times some of the nodes in the tree gain better scores than the others, indicating that they are the best moves for a certain configuration in the game.
In order to complete the homework successfully you should implement the following methods which are also marked by TODO in the source code provided to you.

- Tree class:
  - **ArrayList<Node> getNodesAtLevel(int level)** Increasing the number of iterations of the algorithm improves its performance in playing the game, but also increases the memory used by the nodes of the tree. To address that, you are given a method called **pruneTree(int level)** that deletes the node with the worst score at the specified level. This method uses **ArrayList<Node> getNodesAtLevel(int level)** to get a list of nodes at the given level. You are going to implement the **getNodesAtLevel** method which takes the level as parameter and returns an ArrayList of nodes at the specified level. The root node is assumed to be at level 0 in the tree. For example, in Figure 2, the grey nodes: 4, 5, an 6 are the result of calling **getNodesAtLevel(2)** in the shown tree. You can add any helper method if needed.

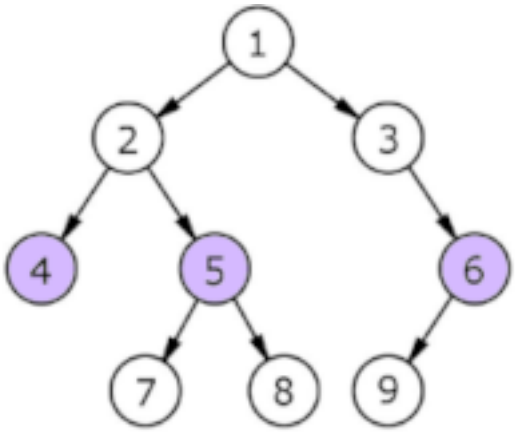

Figure 2: Nodes at level 2

- **Node select():** This method is needed to implement the first step of the MCTS algorithm which is selection. The method should select a leaf node or a node with a non-empty list of possible moves. Starting from the root of the tree you will generate a random number as explained in the source code of the **select** method. If that number is greater than a given threshold and if the current node has possible moves (there is at least one child that is null) you should return the current node from the function. Otherwise you should select one of the children of the current node using the **selectChild** method of the Node class and repeat the same process until you arrive at either a leaf node or return from the method as mentioned above. A simple example of this process is given in Figure 3. In the figure the order of visiting the nodes is shown by a bold border. This is only one of the possible cases since we are using a random number to define which child to choose. In Figure 3, assuming the maximum number of children 3, node 6 may be a possible return value from the function if the shown path is followed.

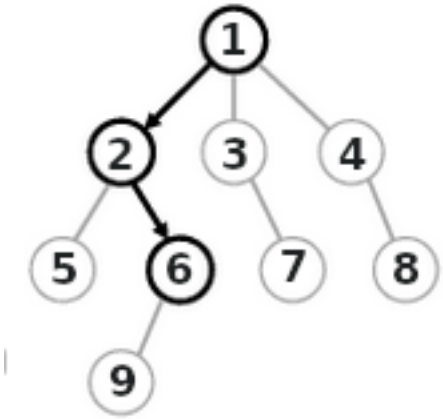

Figure 3: Selection step

- Node class:
  - **void backPropagate(int score):** In this method you will implement the backpropagation process of the algorithm. This method will be called after simulating a random game using the **simulateGame** method. The score parameter of the method corresponds to the resulting score of the simulated game. You should traverse the nodes in the path from this node to the root node of the tree (root node included) in order to update the **totalScore** and **gamesPlayed** attributes of each of the nodes in the path. For all of the nodes visited their total score will be increased by the **score** parameter of the method and their gamesPlayed attribute will be incremented by one. Figure 4 shows the nodes that are visited as a result of calling the method from the node labelled 's'.

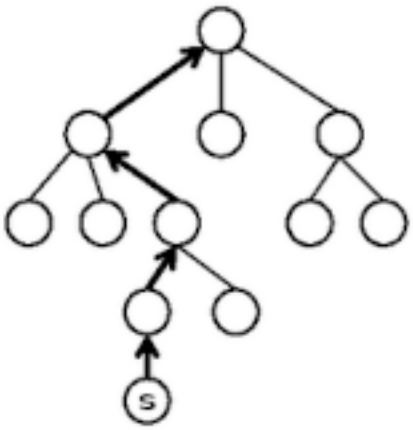

Figure 4: Backpropagation

Submission Details

Usage of github classroom and submission details:

1. Accept the invitation using this link `https://classroom.github.com/a/Tdu_T5jh`.
2. Choose your student ID from the list.
3. A personal repository with the starter code will be created. You can directly edit the code on github's page, clone it to your local storage and edit it there, or use the online IDE VS Code. Ensure that your repositories are private.
4. Make sure your changes are committed and pushed to the repository.
5. Check if you have successfully passed the tests in the "Actions" tab of your repository. Note that these automatic tests do help you understand whether or not your solution works up to some level, but passing them does not guarantee that you will receive a full Grade.
6. To make sure your code is received and avoid any potential problems on github's side, submit a copy of java files on Blackboard as well. Submit all files as a single .zip file, named as:

"ID_NAME_SURNAME.zip"

Grading Criteria

You are going to receive 30 points for each of the select and backPropagate methods and 40 points for the getNodesAtLevel method.
For your further questions about the homework send an email to: emerkuri20@ku.edu.tr