0.1P - Introduction

Peregrin Ryan no. 220220421

Q0.

Name: Peregrin Ryan
Student ID: 220220421
Tutor: Mr. Xiangwen Yang
Class: SIT320 Advanced Algorithms
Intended Grade: Credit

	Module		Tasks	Submission deadline	Discussion deadline
1	Introduction	Must	Module Task 1	17 July	29 July
2	Advanced Trees	Pass	Module Task 2	21 July	2 August
3	Advanced Hashing and Sorting	Pass	Module Task 3	28 July	9 August
4	Advanced Algorithmic Complexity	Credit	Module Task 4	4 August	16 August
5	Graphs	Pass	Module Task 5	11 August	23 August
6	Dynamic Programing	Pass	Module Task 6	18 August	30 August
7	Greedy Algorithms	Pass	Module Task 7	25 August	6 September
8	Linear Programming	Pass	Module Task 8	1 September	13 September
9	Flow-based Algorithms	Credit	Module Task 9	8 September	20 September
10	**Portfolio**	**Must**		**13 October**	

Q1.

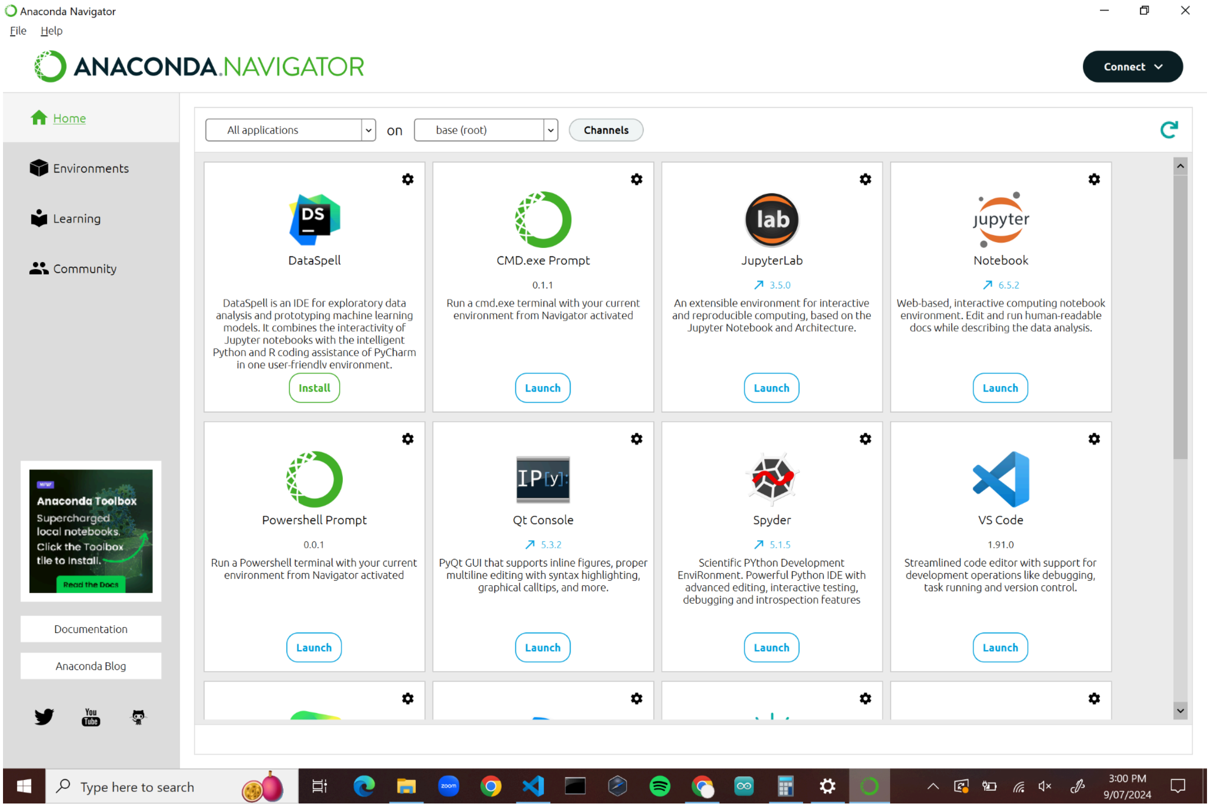
Pseudo code for tic tac toe algorithm:

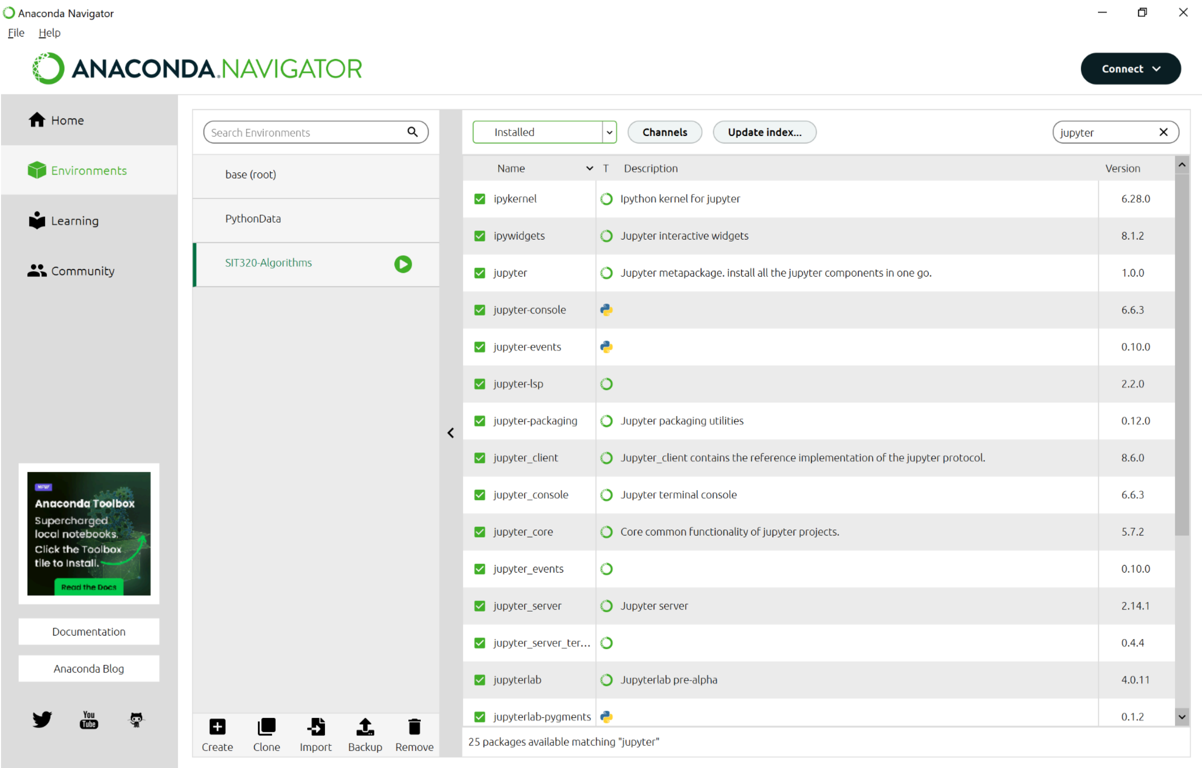
```
function minimax(boardState, turnsAhead, isPlayer):
   // Check if this is a player win or opponent win and returns the value of the win

   // if the player wins then we take 10 and subtract how long it took to get the win, since
we want the sooner win
   if boardState == player win:
       return 10 - turnsAhead
```

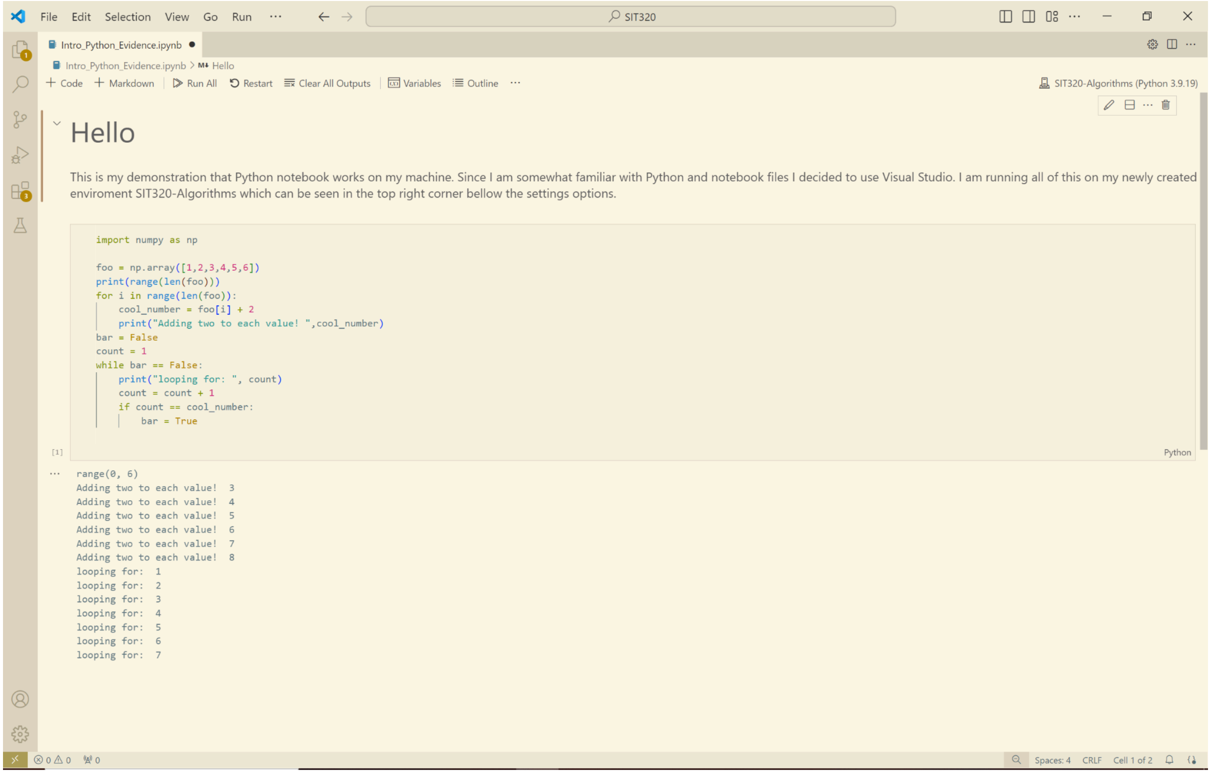
```
   // if opponent does win then we take -10 since its min and we add how many turns since we
need to show how close it is to happening
   if boardState == opponent win:
       return -10 + trunsAhead

   // We check if board has spaces available, if not the else statement below runs
   if boardState has open spaces
       // iterate through each open space
       for space in available spaces:
           // Check if it is the players turn to go for a max value
           if isPlayer == True :
               // create a new board situation that has the player taking the next available
space
               newBoardState = playerMove(space)
               // Continue to recursively use minmax until a win state is discovered
               potentialNewMaxMove = minimax(newBoardState, turnsAhead +1, false)
               // Compare win state score, if it is better than current option use that
               if currentMaxMove < potentialNewMaxMove:
                   currentMaxMove = potentialNewMaxMove
           // return back the best available move
           return currentMaxMove

           // If it is not player turn then go for min value
           if isPlayer == False:
               // create a new board situation that has the opponent taking the next available
space
               newBoardState = opponentMove(space)
               // Continue to recursively use minmax until a more pressing lose state is
discovered
               potentialNewMiniMove = minimax(newBoardState, turnsAhead +1, false)
               // Compare nearest min move to other more pressing one if it is use that result
as current
               if currentMiniMove > potentialNewMiniMove:
                   currentMiniMove = potentialNewMiniMove
           // return the mini move
           return currentMiniMove
   // if the board is full then we return 0 since it is a tie
   else
       return 0
```

Q2.

Here is evidence of my successful installation of Anaconda as well as creating an environment and an example of it successfully running a python notebook in visual studio.


Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Connect
Home
Environments
Learning
Community
All applications
on
base (root)
Channels
DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.
Install
CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch
JupyterLab
3.5.0
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.5.2
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch
Qt Console
5.3.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
Spyder
5.1.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
VS Code
1.91.0
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch
Anaconda Toolbox
Supercharged local notebooks. Click the Toolbox tile to install.
Documentation
Anaconda Blog
Type here to search
3:00 PM
9/07/2024


Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Connect
Home
Environments
Learning
Community
Search Environments
base (root)
PythonData
SIT320-Algorithms
Installed
Channels
Update index...
jupyter
Name
T
Description
Version
ipykernel Ipython kernel for jupyter 6.28.0
ipywidgets Jupyter interactive widgets 8.1.2
jupyter Jupyter metapackage. install all the jupyter components in one go. 1.0.0
jupyter-console 6.6.3
jupyter-events 0.10.0
jupyter-lsp 2.2.0
jupyter-packaging Jupyter packaging utilities 0.12.0
jupyter_client Jupyter_client contains the reference implementation of the jupyter protocol. 8.6.0
jupyter_console Jupyter terminal console 6.6.3
jupyter_core Core common functionality of jupyter projects. 5.7.2
jupyter_events 0.10.0
jupyter_server Jupyter server 2.14.1
jupyter_server_ter... 0.4.4
jupyterlab Jupyterlab pre-alpha 4.0.11
jupyterlab-pygments 0.1.2
Create
Clone
Import
Backup
Remove
25 packages available matching "jupyter"
Anaconda Toolbox
Supercharged local notebooks. Click the Toolbox tile to install.
Documentation
Anaconda Blog



Q4.

Solving the tic tac toe using an algorithm falls into a NP (non-polynomial) problem. This is because the problem is complex enough for it to not be categorised as a P problem due to the amount of potential board states and responses. But it is absolutely an NP problem since tic tac toe is a largely solved game.

Q5.

The problem indicated in the question was that " it will not choose an action which results in immediate winning of the game, and instead at each step explore the entire tree." To fix this I had decided to utilise the previously unused depth variable within the minimax function.

The first change is to add depth in calculating the effectiveness of a move. If the opponent player is to make a move that is dangerous the depth is added to the result as to indicate how immediate the response must be. And when checking a bot win it becomes less valuable with how many iterations it will need to use to reach it. As seen here:

```
def minimax(board, depth, isMaximizing):

    if (chkMarkForWin(bot)):
        return 10 - depth
    elif (chkMarkForWin(player)):
        return -10 + depth
    elif (chkDraw()):
        return 0
```

The second is to ensure that depth actually matters. This shows that the depth increases its value for each iteration it goes through when evaluating the score. The depth will increase by one meaning it will affect the results more when calculating them.

```
        for key in board.keys():
            if board[key]==' ':
                board[key]=bot
                score = minimax(board, depth + 1, False)
                board[key]= ' '
                if (score>bestScore):
                    bestScore = score
        return bestScore
```