CS4023D- Artificial Intelligence

Submitted By- Vishnu Sajith, B180474CS

1. The environment is considered to be 2 cells A and B with unknown initial dirt distribution and agent location. I have considered all possible scenarios and ran the code for 3 lifetime steps for the purpose of taking a snapshot of the output and then ran it for 1000 lifetime steps.

   Assumptions

   - The environment is considered as an array or a list of length 2. **(Cells A and B).**
   - The values in the cells indicate the state of the cell, **0 as dirty and 1 as clean.**
   - The starting position of the Agent is unknown.
   - The position of **cell A is given as 0 and cell B is given as 1.**

   Design

   - To generate all possible scenarios, I have used three nested loops - The outer loop for the start position of the agent, the inner loop for the state of cell A, and the innermost loop for the state of cell B. **Eight different scenarios** are generated.
   - At a particular state, the agent will check if the current position of the agent is dirty or not. If it's found to be dirty the agent will **SUCK** and clean the location and then move to a random location. After each **SUCK** operation the agent is awarded a **performance score of 1** to the existing performance score. If it is not dirty, then the agent will move to a random location.
   - If the agent is at **cell A** and the random move is **LEFT** or if the agent is at **cell B** and the random move is **RIGHT**, an alert message is displayed saying, **“Bot is already here!”.**
   - These steps are run for **1000 lifetime steps for all 8 scenarios.**

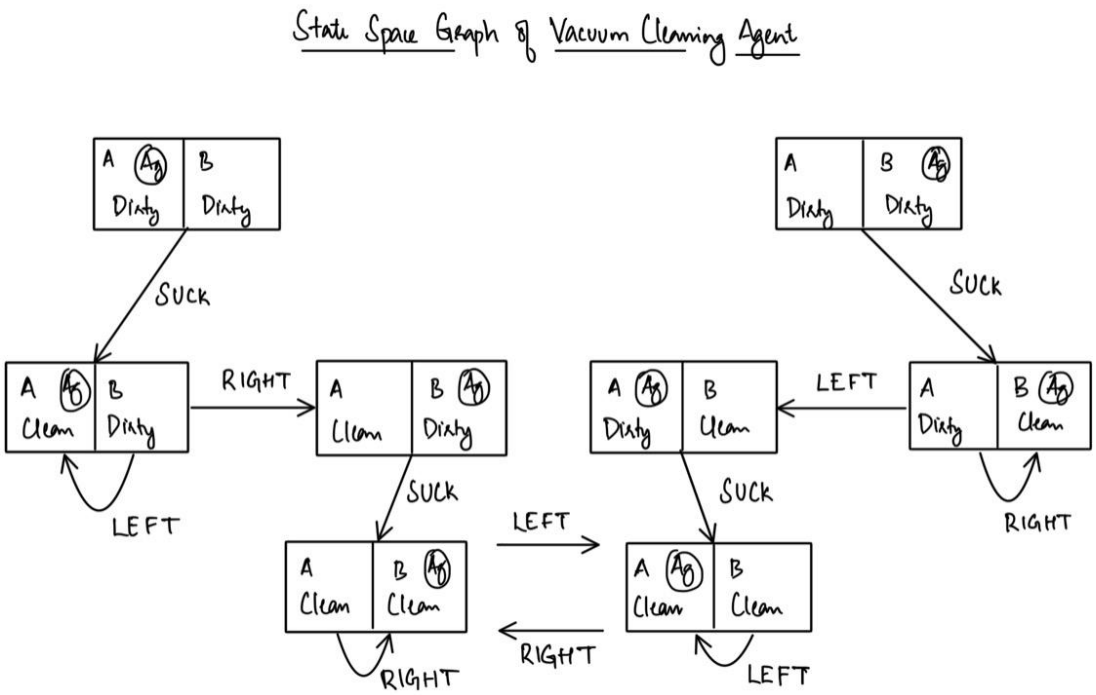
   Snapshot of the Output

   - For the purpose of this report I have restricted the lifetime to 3 steps.
   - The output of the simulation for 1000 lifetime steps is given as **output.txt** in the zip folder

```
$ C:/Python39/python.exe g:/NITC/S7/AI/AI-Assignment/cleaning_agent.py
CELL Value: 0 ==> DIRTY
CELL Value: 1 ==> CLEAN
Agent Position 0 ==> Agent at CELL A
Agent Position 1 ==> Agent at CELL B
A=0;B=1;Agent@A ==> Agent is at postion A, with cell A DIRTY and cell B CLEAN

----------Initial State of Environment --> [0, 0] ; Initial Position of agent --> 0----------
Action: Suck
Current State of Environment: [1, 0] ; Current Position of agent --> 0 ; Performance score: 1
Action: Right
Current State of Environment: [1, 0] ; Current Position of agent --> 1 ; Performance score: 1
Action: Suck
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 2

---STATE SPACE GRAPH-----
 ===> A=0;B=0;Agent@A ===> A=1;B=0;Agent@A ===> A=1;B=0;Agent@B ===> A=1;B=1;Agent@B

----------Initial State of Environment --> [0, 1] ; Initial Position of agent --> 0----------
Action: Suck
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 1
Action: Left
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 1
Action: Left
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 1

---STATE SPACE GRAPH-----
 ===> A=0;B=1;Agent@A ===> A=1;B=1;Agent@A ===> A=1;B=1;Agent@A ===> A=1;B=1;Agent@A

----------Initial State of Environment --> [1, 0] ; Initial Position of agent --> 0----------
Action: Right
Current State of Environment: [1, 0] ; Current Position of agent --> 1 ; Performance score: 0
Action: Suck
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 1
Action: Left
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 1

---STATE SPACE GRAPH-----
 ===> A=1;B=0;Agent@A ===> A=1;B=0;Agent@B ===> A=1;B=1;Agent@B ===> A=1;B=1;Agent@A

----------Initial State of Environment --> [1, 1] ; Initial Position of agent --> 0----------
Action: Left
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 0
Action: Left
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 0
Action: Left
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 0

---STATE SPACE GRAPH-----
 ===> A=1;B=1;Agent@A ===> A=1;B=1;Agent@A ===> A=1;B=1;Agent@A ===> A=1;B=1;Agent@A
```

```
----------Initial State of Environment --> [0, 0] ; Initial Position of agent --> 1----------
Action: Suck
Current State of Environment: [0, 1] ; Current Position of agent --> 1 ; Performance score: 1
Action: Right
Bot is already here!
Current State of Environment: [0, 1] ; Current Position of agent --> 1 ; Performance score: 1
Action: Right
Bot is already here!
Current State of Environment: [0, 1] ; Current Position of agent --> 1 ; Performance score: 1

---STATE SPACE GRAPH-----
 ===> A=0;B=0;Agent@B ===> A=0;B=1;Agent@B ===> A=0;B=1;Agent@B ===> A=0;B=1;Agent@B

----------Initial State of Environment --> [0, 1] ; Initial Position of agent --> 1----------
Action: Left
Current State of Environment: [0, 1] ; Current Position of agent --> 0 ; Performance score: 0
Action: Suck
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 1
Action: Right
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 1

---STATE SPACE GRAPH-----
 ===> A=0;B=1;Agent@B ===> A=0;B=1;Agent@A ===> A=1;B=1;Agent@A ===> A=1;B=1;Agent@B

----------Initial State of Environment --> [1, 0] ; Initial Position of agent --> 1----------
Action: Suck
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 1
Action: Right
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 1
Action: Left
Current State of Environment: [1, 1] ; Current Position of agent --> 0 ; Performance score: 1

---STATE SPACE GRAPH-----
 ===> A=1;B=0;Agent@B ===> A=1;B=1;Agent@B ===> A=1;B=1;Agent@B ===> A=1;B=1;Agent@A

----------Initial State of Environment --> [1, 1] ; Initial Position of agent --> 1----------
Action: Right
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 0
Action: Right
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 0
Action: Right
Bot is already here!
Current State of Environment: [1, 1] ; Current Position of agent --> 1 ; Performance score: 0

---STATE SPACE GRAPH-----
 ===> A=1;B=1;Agent@B ===> A=1;B=1;Agent@B ===> A=1;B=1;Agent@B ===> A=1;B=1;Agent@B


VISHNU SAJITH@MSI MINGW64 /g/NITC/S7/AI
$
```

State Space Graph



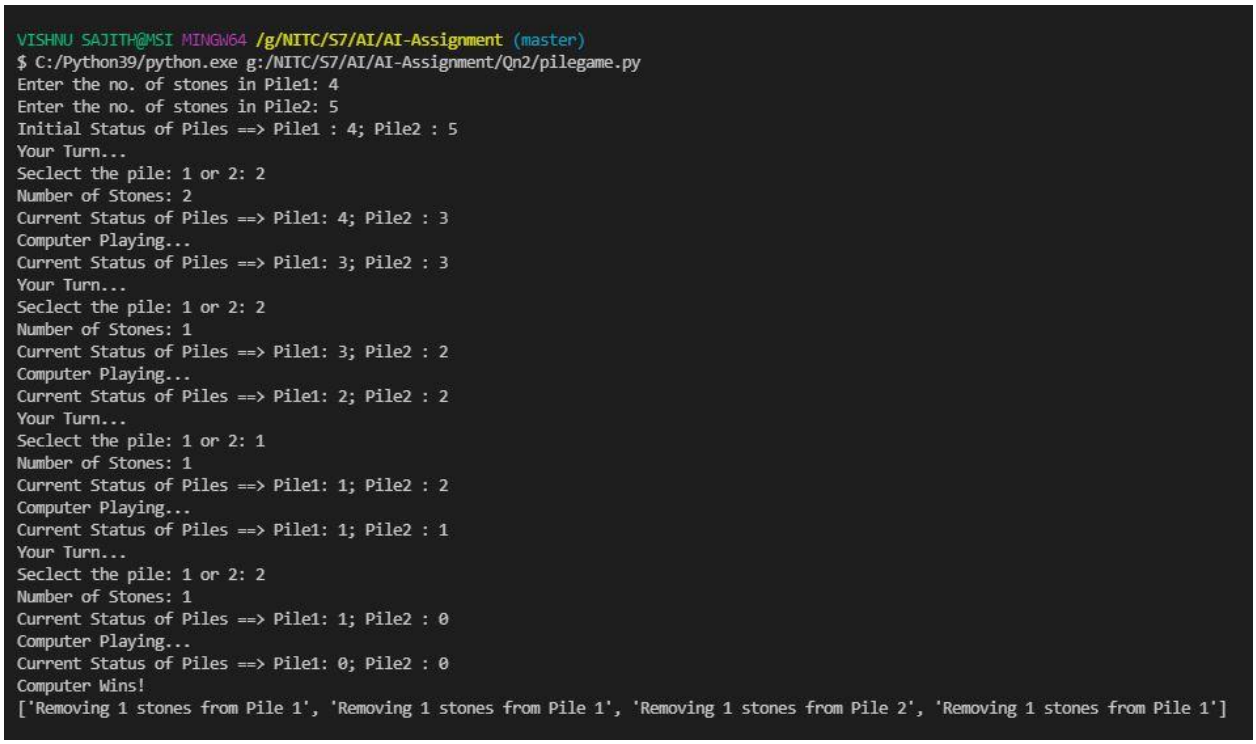
2. **Assumptions**
    - It is assumed that the **Human Player plays first.**

Design

- The player is asked to select the pile from which he/she wants to remove the stones and also the number of stones that need to be removed.
- The number of stones is then removed from the corresponding pile. A function declareWinner is called which accepts the piles and the number of turns played in the game to determine if the game should continue or who the Winner is. The Winner is declared if the maximum number of stones in both the piles is 0. If the number of turns is odd then Computer wins and if it's even the Human Player wins as Human Player starts the game.
- If the game is not over the number of turns is incremented and the next player is given the chance.

- When the computer is playing, the computer tries to simulate the player for all different moves that are available for the computer and selects the best score.
- The computer simulates the player moves and the corresponding computer moves recursively and **maximises** the computer score and **minimises** the human player score to select the optimal move to be played. **(MinMax)**
- The history of moves for both the human and computer is stored and the winning moves are displayed after the game.

Snapshots of the Output



```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

$ C:/Python39/python.exe g:/NITC/S7/AI/AI-Assignment/Qn2/pilegame.py
Enter the no. of stones in Pile1: 1
Enter the no. of stones in Pile2: 0
Initial Status of Piles ==> Pile1 : 1; Pile2 : 0
Your Turn...
Seclect the pile: 1 or 2: 1
Number of Stones: 1
Current Status of Piles ==> Pile1: 0; Pile2 : 0
You Win!
['Removing 1 stones from Pile 1']

VISHNU SAJITH@MSI MINGW64 /g/NITC/S7/AI/AI-Assignment (master)
$ C:/Python39/python.exe g:/NITC/S7/AI/AI-Assignment/Qn2/pilegame.py
Enter the no. of stones in Pile1: 1
Enter the no. of stones in Pile2: 0
Initial Status of Piles ==> Pile1 : 1; Pile2 : 0
Your Turn...
Seclect the pile: 1 or 2: 1
Number of Stones: 0
Current Status of Piles ==> Pile1: 1; Pile2 : 0
Computer Playing...
Current Status of Piles ==> Pile1: 0; Pile2 : 0
Computer Wins!
['Removing 1 stones from Pile 1']

VISHNU SAJITH@MSI MINGW64 /g/NITC/S7/AI/AI-Assignment (master)
$ C:/Python39/python.exe g:/NITC/S7/AI/AI-Assignment/Qn2/pilegame.py
Enter the no. of stones in Pile1: 3
Enter the no. of stones in Pile2: 4
Initial Status of Piles ==> Pile1 : 3; Pile2 : 4
Your Turn...
Seclect the pile: 1 or 2: 2
Number of Stones: 1
Current Status of Piles ==> Pile1: 3; Pile2 : 3
Computer Playing...
Current Status of Piles ==> Pile1: 2; Pile2 : 3
Your Turn...
Seclect the pile: 1 or 2: 2
Number of Stones: 1
Current Status of Piles ==> Pile1: 2; Pile2 : 2
Computer Playing...
Current Status of Piles ==> Pile1: 1; Pile2 : 2
Your Turn...
Seclect the pile: 1 or 2: 2
Number of Stones: 1
Current Status of Piles ==> Pile1: 1; Pile2 : 1
Computer Playing...
Current Status of Piles ==> Pile1: 0; Pile2 : 1
Your Turn...
Seclect the pile: 1 or 2: 2
Number of Stones: 1
Current Status of Piles ==> Pile1: 0; Pile2 : 0
You Win!
['Removing 1 stones from Pile 2', 'Removing 1 stones from Pile 2', 'Removing 1 stones from Pile 2', 'Removing 1 stones from Pile 2']
```