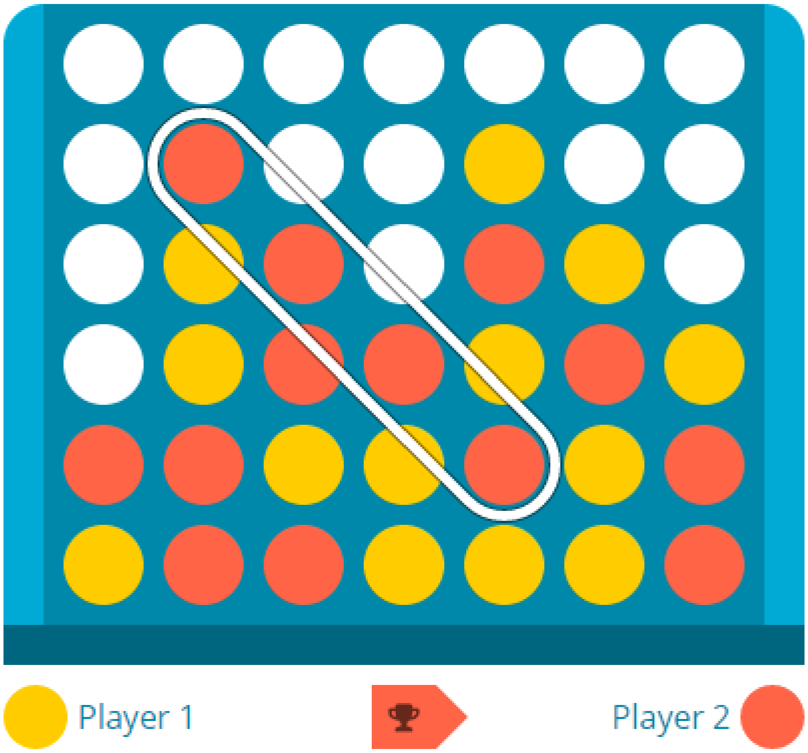


CONNECT-FOUR

Using minimax , alpha and beta pruning algorithm

ABSTRACT

Implementing connect-four game using minimax , alpha and beta pruning algorithm

Team:

- Marwa Adel
- Marwa Mostafa
- Raheeq Alaa
- Salma Samir

Project Overview:

The goal of this project is to implement a Connect 4 game using the Minimax algorithm with alpha-beta pruning. This requires you to design an appropriate Connect 4 board evaluation function to be used as the algorithm's utility function. Your game should allow a human player to play against your algorithm. The algorithm uses a depth-first strategy when exploring the game tree to ensure efficient memory usage.

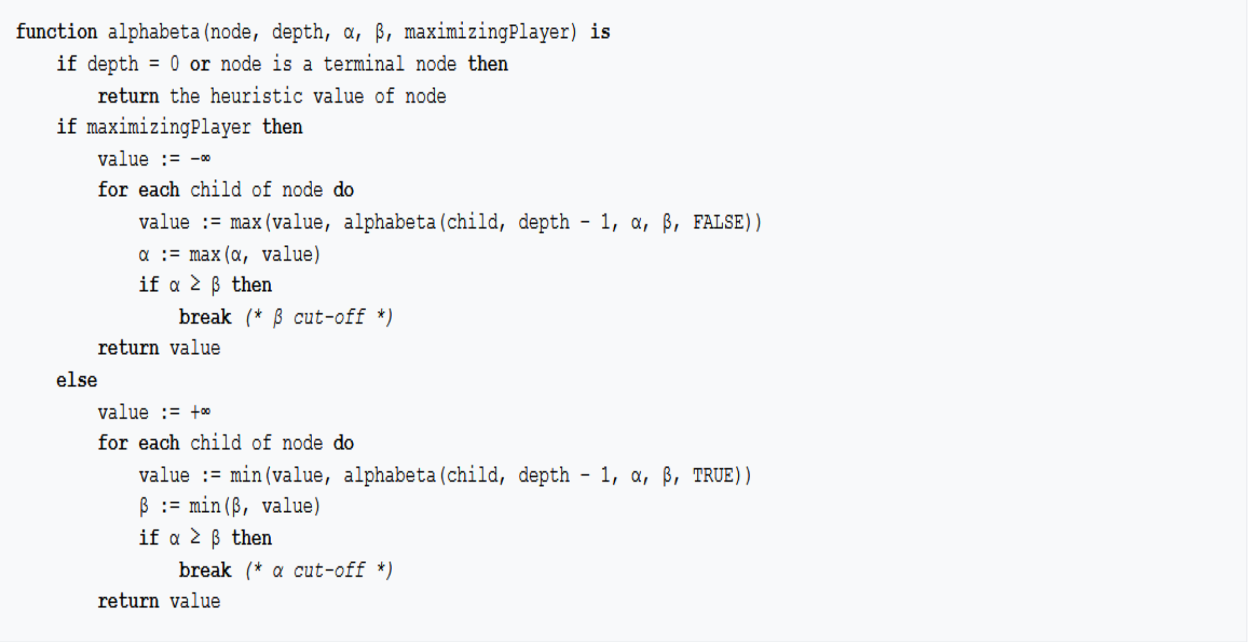
The project is implemented using any high-level language python, the game allow either players to start first. It responds with a move in 30 seconds or less.

Minimax, alpha and beta pruning algorithm:

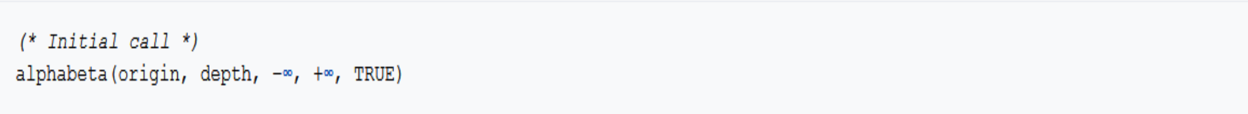
1. What is it?

   It's an optimization technique for minimax algorithm. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

2. Pseudo code

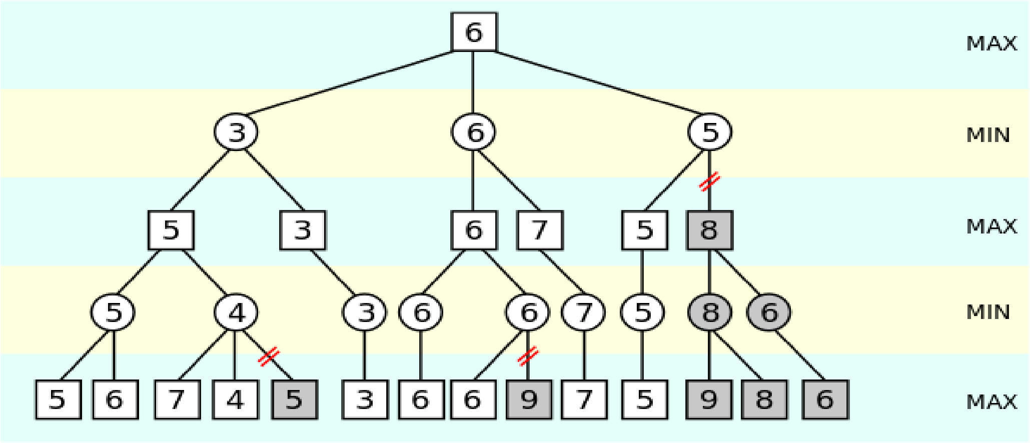


```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if α ≥ β then
                break (* α cut-off *)
        return value
```



```
(* Initial call *)
alphabeta(origin, depth, −∞, +∞, TRUE)
```

3. Tree look



Bonus feature:

- Adding level of difficulty (depth) up to 10.

Utility function used:

The function depends on coins that form horizontal , vertical or diagonal line with same color/identity (C or P) .

-If only two coins form the line ,thus utility function incremented by three.

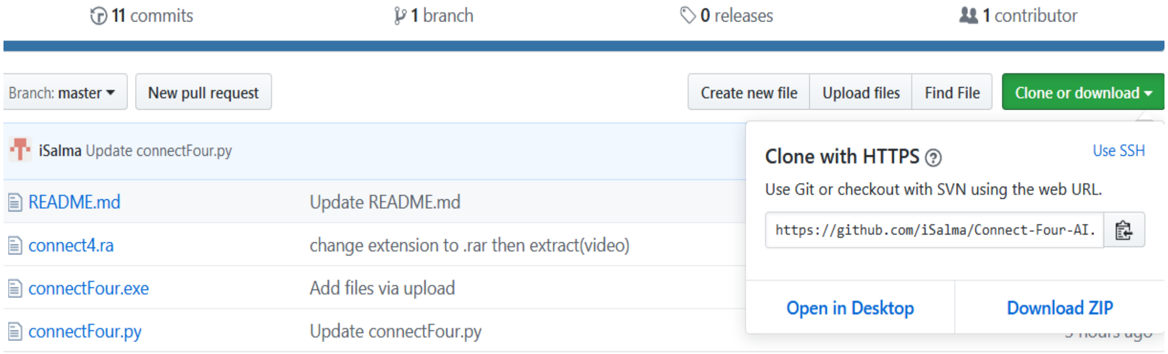
-If three coins form the line ,thus utility function incremented by ten.

-If its the computer winner node , returned cost is 10000

-If its the player winner node , returned cost is -10000

user guide and snapshots

1. Open https://github.com/iSalma/Connect-Four-AI and download .zip file



iSalma Update connectFour.py

README.md	Update README.md
connect4.ra	change extension to .rar then extract(video)
connectFour.exe	Add files via upload
connectFour.py	Update connectFour.py

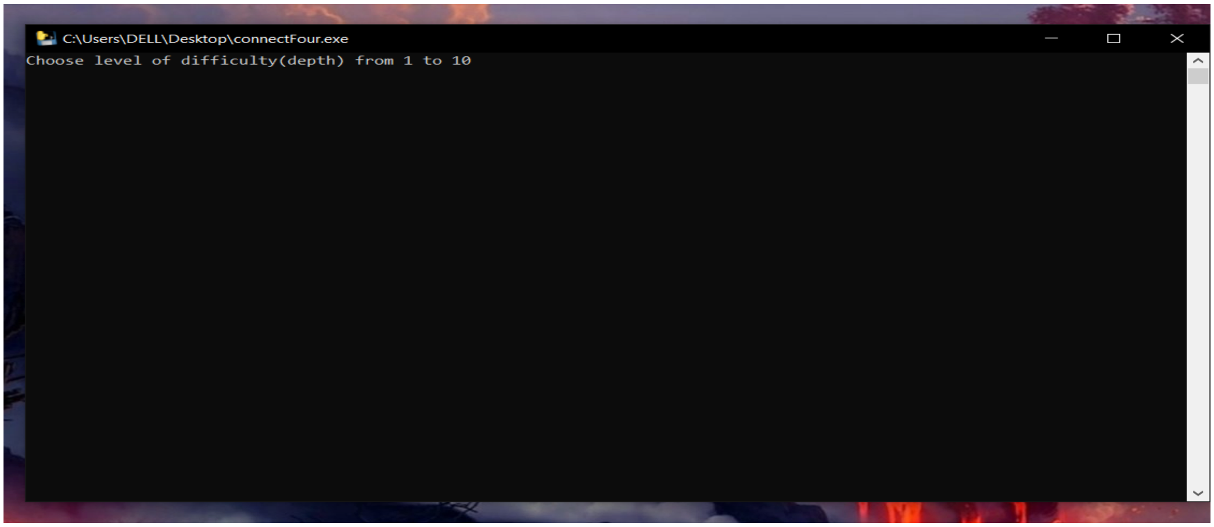
Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/iSalma/Connect-Four-AI.
```

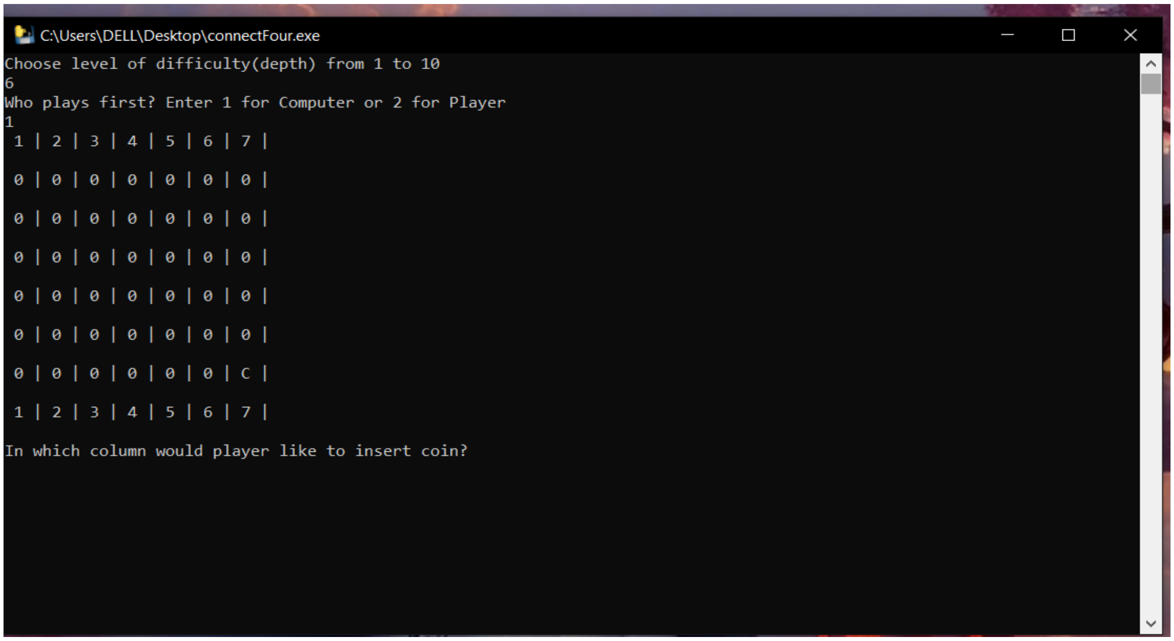
2. Extract .zip file.
3. Run exe program.

4. Enter level of difficulty.



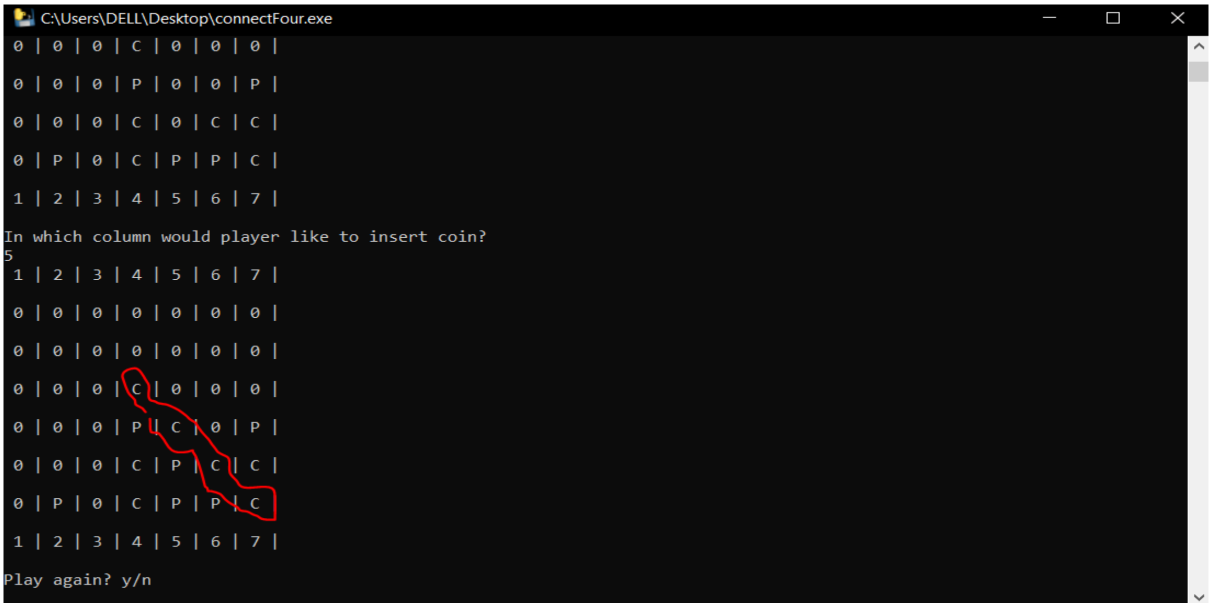
```
C:\Users\DELL\Desktop\connectFour.exe
Choose level of difficulty(depth) from 1 to 10
```

5. Choose who plays first, 1 for computer and 2 for player. (letter ‘C’ for computer and ‘P’ for player)



```
Choose level of difficulty(depth) from 1 to 10
6
Who plays first? Enter 1 for Computer or 2 for Player
1
 1 | 2 | 3 | 4 | 5 | 6 | 7 |

 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 0 | 0 | 0 | 0 | 0 | 0 | C |

 1 | 2 | 3 | 4 | 5 | 6 | 7 |

In which column would player like to insert coin?
```

6. Insert the column you want to put the coin in then continue playing.



```
 0 | 0 | 0 | C | 0 | 0 | 0 |

 0 | 0 | 0 | P | 0 | 0 | P |

 0 | 0 | 0 | C | 0 | C | C |

 0 | P | 0 | C | P | P | C |

 1 | 2 | 3 | 4 | 5 | 6 | 7 |

In which column would player like to insert coin?
5
 1 | 2 | 3 | 4 | 5 | 6 | 7 |

 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 0 | 0 | 0 | C | 0 | 0 | 0 |

 0 | 0 | 0 | P | C | 0 | P |

 0 | 0 | 0 | C | P | C | C |

 0 | P | 0 | C | P | P | C |

 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Play again? y/n
```

7. When game is over choose if you want to play again or exit. (Enter ‘y’ for yes and ‘n’ for no)

Demo video:

YouTube:

- https://www.youtube.com/watch?v=zJ5cnkcZl64

Work flow:

- Salma Samir : main function, winner function and exe program.

- Raheeq Alaa: alphabeta function, maxvalue and minvalue function.

- Marwa Adel: is full function and cost function.

- Marwa Mostafa: addcoin function and printboard function.