UB Social

CS50AI with Python

2.Tic-Tac-Toe

Mateus Schwede

Problemática

- Utilizando Minimax, implemente IA para jogar Jogo da Velha (Tic-Tac-Toe) da melhor maneira possível;
- Há 2 arquivos principais neste projeto: runner.py e tictactoe.py. tictactoe.py contém toda lógica para jogar e para fazer movimentos ideais. runner.py foi implementado para você e contém todo código para executar interface gráfica do jogo. Depois de concluir todas funções necessárias em tictactoe.py, você deve conseguir executar python runner.py para jogar contra sua IA;
- No arquivo tictactoe.py, primeiro, define-se 3 variáveis: X, O e EMPTY, para representar possíveis movimentos do tabuleiro;
- Função initial_state retorna estado inicial do tabuleiro. Para este problema, opta-se representar tabuleiro como lista de 3 listas (representando 3 linhas do tabuleiro), onde cada lista interna contém 3 valores que são X, O ou EMPTY. Seguinte, serão funções para ser implementadas.

Instruções

- Baixe código de https://cdn.cs50.net/ai/2023/x/projects/0/tictactoe.zip e descompacte-o.

Especificações

- Complete as implementações de player, actions, result, winner, terminal, utility, e minimax;
- Função player deve receber estado do tabuleiro como entrada e retornar a rodada do jogador (X ou O);
    - No estado inicial do jogo, X faz 1º movimento. Posteriormente, jogador alterna com cada movimento adicional;
    - Qualquer valor de retorno é aceitável se placa (board) de terminal for fornecida como entrada (ou seja, jogo já terminou).
- Função actions deve retornar conjunto de todas ações possíveis que podem ser tomadas em determinado quadro;
    - Cada ação deve ser representada como tupla (i, j), onde i corresponde à linha do movimento (0, 1 ou 2) e j corresponde à célula na linha, que corresponde ao movimento (também 0, 1 ou 2);
    - Movimentos possíveis são quaisquer células no tabuleiro que ainda não tenham X ou O;
    - Qualquer valor de retorno é aceitável se placa de terminais for fornecida como entrada.

Especificações

- Função result recebe quadro e ação como entrada e deve retornar novo estado do quadro, sem modificar quadro original;
    - Se ação não for válida para conselho, programa deve gerar uma exceção (try except);
    - Estado do tabuleiro retornado deve ser tabuleiro que resultaria da tomada do tabuleiro de entrada original e da permissão para que jogador, cuja vez, é fazer seu movimento na célula indicada pela ação de entrada;
    - Tabuleiro original deve ser deixado inalterado: Minimax acabará exigindo consideração de muitos estados diferentes do tabuleiro durante sua computação. Ou seja, simplesmente atualizar célula no próprio tabuleiro não é implementação correta da função result. Pode-se fazer cópia (deep copy) do tabuleiro antes de fazer qualquer alteração.

Especificações

- Função winner deve aceitar tabuleiro como entrada e retornar vencedor do tabuleiro, se houver;
    - Se jogador X venceu jogo, função deve retornar X. Se jogador O venceu jogo, função deve retornar O;
    - É possível vencer jogo com 3 movimentos seguidos na horizontal, vertical ou diagonal;
    - Haverá no máximo 1 vencedor (ou seja, nenhum tabuleiro terá ambos jogadores com 3 em 1 fileira, pois isso seria estado de tabuleiro inválido);
    - Se não houver vencedor (seja pelo jogo em andamento ou porque terminou empatado), função deve retornar None.

Especificações

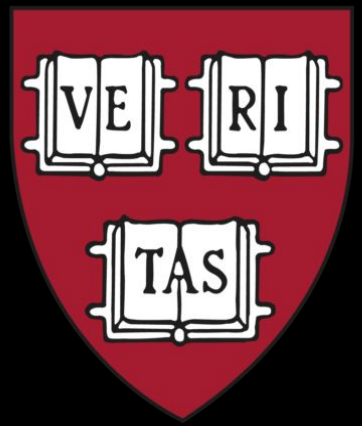
- Função terminal deve aceitar tabuleiro como entrada e retornar valor booleano indicando se jogo acabou;
    - Se jogo terminar, seja porque alguém ganhou jogo ou porque todas células foram preenchidas sem que ninguém tenha vencido, função deve retornar True;
    - Caso contrário, função deve retornar False se jogo ainda estiver em andamento.
- Função utility deve aceitar placa de terminais como entrada e gerar utilidade da placa;
    - Se X ganhou jogo, utility é 1. Se O ganhou jogo, utility é -1. Se jogo terminou empatado, utility é 0;
    - Utility só será chamado em placa se terminal (placa) for True.
- Função minimax deve receber tabuleiro como entrada e retornar movimento ideal para jogador mover naquele tabuleiro;
    - Movimento retornado deve ser ação otimizada (i, j) uma das ações permitidas no tabuleiro. Se múltiplos movimentos forem igualmente otimizados, qualquer um desses movimentos é aceitável;
    - Se placa for placa de terminais, função minimax deverá retornar None.

Especificações

- Para todas funções que aceitam tabuleiro como entrada, um tabuleiro válido (ou seja, lista que contem 3 linhas, cada uma com 3 valores de X, O ou EMPTY). Não deve-se modificar declarações de função (ordem ou nº de argumentos para cada função) fornecidas;
- Com todas funções implementadas corretamente, pode-se executar arquivo runner.py e jogar contra a IA.

Submissão

- Visual Studio Code online: https://cs50.dev
- Testar precisão da lógica do algoritmo: check50 ai50/projects/2024/x/tictactoe
- Testar estilização do código: style50 tictactoe.py
- Para submissão:
  - Em https://submit.cs50.io/invites/d03c31aef1984c29b5e7b268c3a87b7b, entre com GitHub e autorize CS50;
  - Instale pacote Git, Python 3 (e pip), instalando pacotes: pip3 install style50 check50 submit50
  - Submeta o projeto: submit50 ai50/projects/2024/x/tictactoe
- Verificar avaliação: https://cs50.me/cs50ai.



UB Social

Mateus Schwede
HBS ID 202400167108 - DCE ID @00963203