GOMOKU

Nesta tarefa, você implementará um mecanismo de IA simples (e imperfeito) para o jogo Gomoku, jogado em um tabuleiro 8 × 8. Em Gomoku, existem dois jogadores. Um jogador joga com pedras pretas e o outro jogador joga com pedras brancas. Um jogador se move colocando uma pedra em um quadrado vazio do tabuleiro. O jogador que joga com pedras pretas sempre se move primeiro. Após o primeiro movimento, os jogadores se alternam. Um jogador ganha se ele colocar cinco de suas pedras em uma sequência, seja horizontalmente, verticalmente ou diagonalmente. Por favor leia: https://pt.wikipedia.org/wiki/Gomoku. Estaremos jogando a variante padrão.

O código inicial

O arquivo gomoku.py contém o código inicial para o jogo Gomoku e a IA. O programa funciona da seguinte maneira. O computador (que joga com as pedras pretas) sempre se move primeiro. Após o primeiro movimento, os movimentos do usuário e do computador se alternam. O computador determina seu movimento encontrando o movimento que maximiza o valor de retorno da função de pontuação (que é fornecida).

As funções em gomoku.py aceitam **tabuleiro** como um de seus argumentos. Esta é a representação do tabuleiro Gomoku. O quadrado (y, x) no tabuleiro é armazenado no **tabuleiro[y][x]**. O valor do quadrado é:

- " ", se o quadrado estiver vazio;
- "p", se o quadrado tiver uma pedra preta, e;
- "b", se o quadrado tiver uma pedra branca.

Consulte a função **imprime_tabuleiro** para obter um exemplo de como usar o **tabuleiro**.

Uma parte importante do mecanismo de IA do Gomoku é encontrar sequências contíguas de pedras da mesma cor no tabuleiro do Gomoku. Existem quatro direções possíveis para uma sequência: da esquerda para a direita, de cima para baixo, da esquerda para a direita inferior e da direita para a esquerda inferior. Observe que não consideramos, por exemplo, a direção direita para esquerda, uma vez que uma sequência da direita para a esquerda pode ser representada como uma sequência da esquerda para a direita. A direção de uma sequência pode ser representada por um par de números (d_y, d_x) como segue:

- (0,1): direção da esquerda para a direita. Por exemplo, a sequência de pedras da mesma cor nas coordenadas (5,2), (5,3), (5,4), (5,5) é uma sequência na direção da esquerda para a direita. Observe que dizemos que a última pedra na sequência está na localização (5,5), não na localização (5,2).
- (1,0): direção de cima para baixo. Por exemplo, uma sequência de pedras da mesma cor nas coordenadas (3,1), (4,1), (5,1) é uma sequência de cima para baixo. Observe que dizemos que a última pedra na sequência está no local (5,1), não no local (3,1).
- (1,1): direção superior-esquerda-inferior-direita. Por exemplo, uma sequência de pedras da mesma cor nas coordenadas (2,3), (3,4), (4,5) é uma sequência superior esquerda para inferior direita. Observe que dizemos que a última pedra na sequência está no local (4,5), não no local (2,3).

- (1, -1): direção superior direita para inferior esquerda. Por exemplo, uma sequência de pedras da mesma cor nas coordenadas (5,5), (6,4), (7,3) é uma sequência superior direita para inferior esquerda. Observe que dizemos que a última pedra na sequência está na localização (7,3), não na localização (5,5).

Uma sequência pode ser:

- aberta: uma pedra pode ser colocada em um quadrado em qualquer lado da sequência;
- fechada: a sequência é bloqueada em ambos os lados para que nenhuma pedra seja colocada. Isso pode ocorrer porque a sequência inicia/termina perto da borda do tabuleiro ou porque há uma pedra de cor diferente na localização vizinha ao início/fim da sequência.
- semiaberta: a sequência nem está aberta e nem fechada.

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4		b						
5		b						
6		b						
7								

Figura 1: (1,0)

	0	1	2	3	4	5	6	7
0								
1								
2								
3				b	b	b	p	
4								
5								
6								
7								

Figura 2: (0,1)

	0	1	2	3	4	5	6	7
0								
1								
2								
3	b	b	b	b	p			
4								
5								
6								
7								

Figura 3: (0,1)

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5	p							
6		p						
7			p					

Figura 4: (1,1)

	0	1	2	3	4	5	6	7
0							p	
1						p		
2					p			
3				p				
4								
5								
6								
7								

Figura 5: (1,-1)

Aqui estão alguns exemplos de sequências e suas classificações:

- Figura 1: uma sequência aberta de três pedras brancas. A direção é (1,0). A última pedra está na localização (6,1).
- Figura 2: uma sequência semiaberta de três pedras pretas. A direção é (0,1). A última pedra está em (3,5).
- Figura 3: uma sequência fechada de quatro pedras brancas. A direção é (0,1). A última pedra está em (3,3).
- Figura 4: uma sequência fechada de três pedras pretas. A direção é (1,1). A última pedra está em (7,2).
- Figura 5: uma sequência semiaberta de quatro pedras brancas. A direção é (1,-1). A última pedra está em (3,3).

O arquivo gomoku.py contém a implementação das funções seguintes:

- imprime_tabuleiro(tabuleiro)
  - Imprime o tabuleiro de gomoku.
- pontuacao(tabuleiro)
  - Computa e retorna a pontuação para uma posição no tabuleiro. Assume movimento do jogador com pedra preta anteriormente.
- jogar_gomoku(tamanho_tabuleiro)
  - Permite um usuário jogar contra um computador em um tabuleiro de tamanho **tamanho_tabuleiro x tamanho_tabuleiro.** Ela interage com a IA chamando a função **busca_max()**, que você escreverá.
- por_seq_no_tabuleiro(tabuleiro, y, x, d_y, d_x, comprimento, cor)
  - Função auxiliar que adiciona a sequência de pedras de **cor** e **comprimento** ao **tabuleiro**, iniciando na posição **x** e **y**, movendo-se na direção **(d_y, d_x)**.
- analise(tabuleiro)
  - Analisa a posição no tabuleiro computando o número de sequências abertas e semiabertas de ambas as cores.

Não modifique essas funções!

A IA

Escreva uma IA para o jogo Gomoku implementando as seguintes funções.

- esta_vazia(tabuleiro)
  - Esta função retorna True se não houver pedras no **tabuleiro**.
- esta_limitada(tabuleiro, y_fim, x_fim, comprimento, d_y, d_x)
  - Esta função analisa a sequência de **comprimento** que termina na localização **(y_fim, x_fim)**. A função retorna "**ABERTA**" se a sequência estiver aberta, "**SEMIABERTA**" se a sequência for semiaberta e "**FECHADA**" se a sequência for fechada. Suponha que a sequência seja completa (ou seja, você não recebe apenas uma subsequência) e é válida e contém pedras de apenas uma cor.
- detecta_linha(tabuleiro, cor, y_ini, x_ini, comprimento, d_y, d_x)
  - Esta função analisa a linha (vamos chamá-la de L) de quadrados que começa no local **(y_ini, x_ini)** e vai na direção **(d_y, d_x)**. Observe que este uso da palavra linha é diferente de “uma linha em uma tabela”. Aqui, a palavra linha significa uma sequência de quadrados, que são adjacentes horizontalmente, verticalmente ou diagonalmente. A função retorna uma **tupla** cujo primeiro elemento é o número de sequências abertas de **cor** e de **comprimento** na linha L, e cujo segundo elemento é o número de sequências semiabertas de **cor** e de **comprimento** na linha L.
  - Suponha que (y_ini, x_ini) está localizado na borda do tabuleiro. Somente sequências completas contam. Por exemplo, a coluna 1 na Fig. 1 é considerada como contendo uma linha aberta de comprimento 3, e nenhuma outra linha.
  - Suponha que o **comprimento** seja um número inteiro maior ou igual a 2.
- detecta_linhas(tabuleiro, cor, comprimento)
  - Esta função analisa o tabuleiro. A função retorna uma tupla, cujo primeiro elemento é o número de sequências abertas de **cor** e de **comprimento** em

todo o tabuleiro e cujo segundo elemento é o número de sequências semiabertas de **cor** e de **comprimento** em todo o tabuleiro.
    - Somente sequências completas contam. Por exemplo, a Fig. 1 é considerada como contendo uma linha aberta de comprimento 3, e nenhuma outra linha.
    - Suponha que o **comprimento** seja um número inteiro maior ou igual a 2.
- busca_max(tabuleiro)
    - Esta função usa a função pontuacao() (fornecida) para encontrar o movimento ideal para o preto. Ele encontra a localização (y, x), de modo que (y, x) esteja vazio e colocando uma pedra preta em (y, x) maximiza a pontuação do tabuleiro conforme calculado pela pontuação(). A função retorna uma tupla (y, x) de forma que colocar uma pedra preta nas coordenadas (y, x) maximiza a pontuação potencial (se houver várias tuplas, você pode retornar qualquer uma delas). Após o retorno da função, o conteúdo do tabuleiro deve permanecer o mesmo.
- e_vitoria(tabuleiro)
    - Esta função determina o estado atual do jogo e retorna um de ["Venceram as brancas", "Venceram as pretas", "Empate"], dependendo do status atual no tabuleiro. A única situação em que "Empate" é retornado é quando o tabuleiro está cheio.