1. Inteligência Artificial

JOGO DO BISPO - PARTE 2

Docente:

- Filipe Mariano

Alunos:

- Diogo Venâncio - 160221076
- André Gonçalves - 170221015

Indíce

Objetivo do projecto

O principal objetivo desta fase do projecto é o enquadramento teórico–prático com os conhecimentos adquiridos no âmbito da Teoria de Jogos.

1. Jogador vs PC

O jogo é iniciado quando o jogador escolhe se o pc ou o mesmo executa a primeira jogada. Após a escolhe o jogo vai se desenvolvendo, mostrando ao jogador os valores das casas que o mesmo pode jogar o seu bispo. O jogo termina quando o jogador e o pc não conseguirem efetuar mais jogadas.

2. PC vs PC

O jogo é iniciado e termina quando já não houver casas possiveis para se jogar um dos bispos.

Divisão do projecto

1. **interact.lisp**

> Carrega os outros ficheiros de código, escreve e lê ficheiros, e trata da interação com o jogador

2. **jogo.lisp**

> Código relacionado com o problema.

3. **algoritmo.lisp**

> Contem a implementação dos algoritmos

Interact

Interecção com o jogador

> Função ao qual o jogador executa para iniciar o programa

```
;Mostra o menu inicial
(defun inicial-menu()
  (progn
    (format t "~% ------------------------------------")
    (format t "~%              Jogo do Bispo          ")
    (format t "~%                                     ")
    (format t "~%           1 - Humano vs PC          ")
    (format t "~%             2 - PC vs PC            ")
    (format t "~%                                     ")
```

```
    (format t "~%                s - Sair                ")
    (format t "~% ---------------------------------------~%~%> "))
)

;Iniciar programa
(defun start()
  (progn
    (inicial-menu)
      (let ((opt (read)))
        (cond
         ((eq opt 's) (format t "Até á próxima!"))
         ((or (< opt 1) (> opt 2)) (progn (format t "Insira uma opção válida")
(start)))
         ((eq opt '1) (humano-vs-pc))  ;Chamar função humano vs pc
         ((eq opt '2) (pc-vs-pc)))))  ;Chamar função pc vs pc
)
```

> Mensagem para o jogador selecionar se quer efetuar a 1ªjogada

```
(defun definir-jogador()
  (format t "~% ---------------------------------------------------")
  (format t "~%  Jogo do Bispo - Escolha quem inicia a 1ª jogada  ")
  (format t "~%                                                   ")
  (format t "~%                  1 - Humano                       ")
  (format t "~%                   2 - PC                          ")
  (format t "~%                                                   ")
  (format t "~%                  s - Sair                         ")
  (format t "~% ---------------------------------------------------~%~%> ")
)
```

Jogo

Código relacionado com o problema

> Função responsável por retornar todas as posições que o bispo do jogador pode ir numa certa diagonal

Esta função recebe um board, o jogador e verifica a posição atual do bispo do jogador.

Verifica se o bispo já foi colocado no board, senão coloca o bispo na casa da primeira linha ou ultima com maior valor e chama novamente a função.

Verifica se as posições estão dentro dos limites do board e se os valores não contêm 0 nas posicoes.

Se tudo for válido, adiciona a posição á lista e percorre o resto do board.

```
(defun diagonal-1-pos(board jogador &optional (bispo-pos (posicao-jogador board
jogador)))
  (cond
   ((null bispo-pos) (diagonal-1-pos (coloca-jogador (create-no board nil)
jogador) jogador))
   ((or (< (car bispo-pos) 0) (< (cadr bispo-pos) 0) (> (car bispo-pos) 8) (>
(cadr bispo-pos) 8) (equal (car bispo-pos) 0) (equal (cadr bispo-pos) 0)) nil)
   (t (cons bispo-pos (diagonal-1-pos (cdr board) jogador (list (- (car bispo-pos)
1) (+ (cadr bispo-pos) 1))))))
)
```

> Função que coloca o bispo do jogador na nova casa jogada

- Esta função recebe uma posicao jogavel pelo bispo, o board e o jogador.
- Verifica se a posicao e o board têm valores.
- Guarda a posição pretendida, a posição atual do bispo e o valor da casa da nova posição.
- Verifica se a posição está vazia.
- Verifica se a o valor da nova posição se encontra com algum jogador e se está vazia.
- De seguida, guarda o valor da casa para o qual vai corresponder á posição no board.
- Guarda o board com a posição atual do bispo como NIL.
- Guarda o board com o bispo na nova posição.
- Guarda a posição do valor simétrico da nova posição.
- Se não existir o simétrico, simplesmente retorna o board atualizado.
- Se existir simétrico, altera a casa do valor simetrico para NIL.

```
(defun jogada(lista board jogador)
  (cond
   ((and (null lista) (null board)) nil)
   (t (let* ((posicao lista)
         (bispo-pos (posicao-jogador board jogador))
         (pos-new (verifica-pos (car posicao) (cadr posicao) board)))
     (cond
      ((null posicao) nil)
      ((or (equal jogador pos-new) (null pos-new) (equal (muda-jogador jogador)
pos-new)) nil)
      (t (let ((valor (get-celula (car posicao) (cadr posicao) board)))
        (cond
         ((null valor) nil)
         (t (let* ((board-t (substituir (car bispo-pos) (cadr bispo-pos) board
'NIL))
```

```
                (board-t-1 (substituir (car posicao) (cadr posicao) board-t
jogador))
                (simetrico (simetrico-pos valor board-t-1)))
           (cond
            ((null simetrico) board-t-1)
            (t (substituir (car simetrico) (cadr simetrico) board-t-1
'NIL)))))))))))
)
```

Algoritmo

Implementação do algoritmo

> Algoritmo Negamax com cortes Alfa-Beta

NEGAMAX - FAIL-HARD

1. Guardar a lista de sucessores ordenados por ordem crescente a partir do no recebido e dependendo do jogador
2. Se a profundidade for igual a 0, a lista de sucessor estiver vazia e o tempo inicial de processamento for maior ou igual ao tempo limite, cria no solução
3. Caso contrário, chama a função auxiliar do negamax para os sucessores de um no.

SUCESSORES NEGAMAX

1. Se o size dos sucessores for igual a 1, chama a função do negamax com o no da lista de sucessores, trocando o jogador, alterando a profundidade, o valor de alfa e beta, e adicionando +1 aos nos gerados
2. Guarda o no solucao com os respetivos dados: no, nos gerados, cortes e tempo
3. Guarda o no
4. Guarda o melhor valor de f entre o valor do no solucao e o no pai
5. Guarda o novo valor para alfa entre o maior valor entre o valor atual de alfa e o melhor valor de f
6. Se o novo valor de alfa for maior ou igual ao valor do beta, faz o corte e cria o no solução
7. Caso contrário, chama a função recursivamente com:

- no pai,
- o resto da lista de sucessores,
- tempo limite,
- função dos sucessores,
- o jogador,
- o novo valor de alfa,
- valor de beta,
- tempo inicial,
- o valor de nos gerados até ao no solução,
- o valor de cortes até ao no solução

```
(defun negamax(no time-limite sucessor &optional (jogador -1) (profun 50) (alfa
most-negative-fixnum) (beta most-positive-fixnum)
```

```
                  (time-inicial (get-internal-real-time)) (nos-gerados 0) (cortes
0))
  (let* ((lista (posicoes-jogadas-possiveis (jogadas-possiveis (estado-no no)
jogador) (estado-no no)))
         (sucessores (sucessores-order-negamax (funcall sucessor no jogador lista)
jogador)))
    (cond
     ((or (= profun 0) (null sucessores) (>= (- (get-internal-real-time) time-
inicial) time-limite))
      (create-no-solucao no nos-gerados cortes time-inicial))
     (t (sucessores-negamax no sucessores time-limite sucessor jogador profun alfa
beta time-inicial nos-gerados cortes))))
)

(defun sucessores-negamax(no sucessores time-limite sucessor jogador profun alfa
beta time-inicial nos-gerados cortes)
  (cond
   ((= (length sucessores) 1)
    (negamax (car sucessores) time-limite sucessor (muda-jogador jogador) (1-
profun) (- alfa) (- beta) time-inicial (1+ nos-gerados) cortes))
   (t (let* ((solucao (negamax (car sucessores) time-limite sucessor (muda-jogador
jogador) (1- profun) (- alfa) (- beta) time-inicial (1+ nos-gerados) cortes))
             (top-no (car solucao))
             (best-valor (f-calcula-best (f-no top-no) (f-no no)))
             (alfa-novo (max alfa best-valor)))
        ;A procura pode ser descontinuada abaixo de qualqer no MAX com o valor
alfa >= beta de qualquer dos seus antecessores MIN (corte beta)
        (if (>= alfa-novo beta) ;Efetua o corte
          (create-no-solucao no (nth 0 (cadr solucao)) (1+ (nth 1 (cadr solucao)))
time-inicial)
         (sucessores-negamax no (cdr sucessores) time-limite sucessor jogador
profun alfa-novo beta time-inicial (nth 0 (cadr solucao)) (nth 1 (cadr
solucao)))))))
)
```

Análise

Humano vs PC

Jogador Humano -> -1

Jogador PC -> -2

> 1º Jogada

```
   (37 62 55 31 73 58 51 47)
   (35 44 38 12 41 83 74 68)
   (36 33 24 27 18 75 88 61)
```

```
   (54 43 32 34 81 78 87 77)
   (56 13 15 86 42 71 21 46)
   (85 28 45 11 64 14 63 22)
   (17 57 16 66 82 47 72 48)
   (53 67 23 76 25 84 65 26)

Escolha uma coluna da 1ªlinha para colocar o bispo - 5

   (37 62 55 31 -1 58 51 47)
   (35 44 38 12 41 83 74 68)
   (36 33 24 27 18 75 88 61)
   (54 43 32 34 81 78 87 77)
   (56 13 15 86 42 71 21 46)
   (85 28 45 11 64 14 63 22)
   (17 57 16 66 82 47 72 NIL)
   (53 67 23 76 25 -2 65 26)

   Pontos feitos pelo Jogador 2 -PC-: 84
   Nº de Nos: 5746
   Nº Cortes: 897
   Tempo Execução: 1015
```

> Pontos totais

```
----------------------------------------
        Humano VS PC
----------------------------------------
1º Jogador: HUMANO
Pontos Jogador 1: 624
Pontos Jogador 2: 1220
```

PC vs PC

> 1º Jogada de cada jogador

```
   (NIL 62 55 31 -1 58 51 47)
   (35 44 38 12 41 83 74 68)
   (36 33 24 27 18 75 88 61)
   (54 43 32 34 81 78 87 77)
   (56 13 15 86 42 71 21 46)
   (85 28 45 11 64 14 63 22)
   (17 57 16 66 82 47 72 48)
   (53 67 23 76 25 84 65 26)

   Pontos feitos pelo Jogador 1 -PC-: 73
   Nº de Nos: 5707
   Nº Cortes: 1028
   Tempo Execução: 1015
```

```
    (NIL 62 55 31 -1 58 51 47)
    (35 44 38 12 41 83 74 68)
    (36 33 24 27 18 75 88 61)
    (54 43 32 34 81 78 87 77)
    (56 13 15 86 42 71 21 46)
    (85 28 45 11 64 14 63 22)
    (17 57 16 66 82 47 72 NIL)
    (53 67 23 76 25 -2 65 26)

    Pontos feitos pelo Jogador 2 -PC-: 84
    Nº de Nos: 6045
    Nº Cortes: 1075
    Tempo Execução: 1013
```

> Pontos totais

```
---------------------------------------
        PC VS PC
---------------------------------------
Pontos Jogador 1: 1198
Pontos Jogador 2: 1795
```

Requisitos não implementados

1. O bispo não pode ser jogado para uma casa que esteja ameaçada pelo bispo adversário.