Documentação Tic-Tac-Toe Rubik's Cube

Aluno: Eduardo Cunha Campos

1. Procedimentos Implementados

bestMove/2

Calcula a melhor próxima posição a partir de uma posição Pos informada utilizando o algoritmo alpha-beta.

firstMove/1

Verifica se é o primeiro movimento do computador. Em caso afirmativo, retorna verdadeiro. Caso contrário, retorna falso.

generateRandomMove/3

O primeiro movimento do computador é escolhido aleatoriamente. Para posições diferentes no tabuleiro, o valor de utilidade não muda. Ou seja, é gerado uma posição aleatória entre 1 e 64 no tabuleiro. Este procedimento é responsável por gerar a primeira jogada do computador.

gravaJogada/2

Responsável por imprimir na tela a jogada do computador e chamar o procedimento **gravaJogadaNoArquivo/2**.

gravaJogadaNoArquivo/2

Responsável por gravar a jogada de um determinado jogador (humano ou computador) na posição correta do arquivo “game.txt”.

converte_caractere/2

Este procedimento apenas converte o jogador (x ou o) para o caractere ('X' ou 'O') que será gravado no arquivo “game.txt”.

play/0

Responsável por iniciar o jogo.

playAskColor/0

Solicita ao jogador humano para que ele escolha uma marca válida no jogo (i.e., x ou o) para indicar suas jogadas.

retornaJogadaMaquina/4

Dado o tabuleiro antes do movimento e o tabuleiro após o movimento do computador, calcula qual a última jogada do computador, ou seja, a última posição entre 1 e 64 em que o computador jogou. Este procedimento é necessário para poder registrar a jogada do computador no arquivo "game.txt".

recuperaValor/1

Dada uma lista com apenas um elemento, retorna este elemento. É um procedimento auxiliar do procedimento **retornaJogadaMaquina/4.**

retornaListaJogadas/3

Retorna uma lista contendo as posições escolhidas por um jogador (humano ou computador).

humanMove/3

Responsável por ler a posição do jogador humano e realizar o movimento desejado.

set1/4

Procedimento que marca a jogada de um jogador (humano ou computador) em uma determinada posição do tabuleiro, que é informada para este procedimento.

show/1

Responsável por imprimir o tabuleiro na tela. O tabuleiro é formado por quatro quadros, com cada quadro possuindo quatro linhas e quatro colunas. Este tabuleiro é composto por 64 posições no total.

show2/1

Procedimento auxiliar de **show/1** que substitui 0 por espaço (' ').

indexOf/3

Dada uma lista e um elemento desta lista, encontra a posição do elemento dentro da lista.

move/2

Retorna verdadeiro se existe um movimento legal (conforme as regras definidas) de uma posição Pos para uma próxima posição NextPos.

nextPlayer/2

Procedimento para passar a vez do jogo para o próximo jogador.

move_aux/3

Retorna verdadeiro se o próximo tabuleiro é o tabuleiro antigo que teve uma posição vazia substituida pela marcação (x ou o) de um jogador.

utility/2

Responsável por retornar a utilidade para uma jogada. A utilidade é um valor entre -4 e +4. Para o jogador MAX, o maior valor será +4 (que corresponde a uma vitória do humano) . Já para o jogador MIN, o maior valor será -4 (que corresponde a uma vitória do computador). Esta função heurística é utilizada pelo algoritmo alpha-beta para que seja possível decidir qual a melhor jogada naquele momento para um determinado jogador.

retornaUtilidade/3

Retorna o menor elemento da lista (caso em que o jogador é o computador) ou o maior elemento da lista (caso em que o jogador é o humano). Lembrando que o humano é o jogador MAX e o computador é o jogador MIN.

avaliaUtilidade/3

Responsável por analisar a quantidade de x's e a quantidade de o's que existem em uma dada jogada. Retorna o valor da utilidade (positivo, negativo ou nulo) após a realizar a análise da quantidade.

count/3

Conta quantas vezes um dado elemento aparece em uma lista informada.

calculaUtilidade/2

Responsável por calcular a utilidade para uma dada jogada. Cada jogada é uma lista de quatro posições. Invoca os procedimentos **count/3** e **avaliaUtilidade/3**.

delta1/3

Calcula a utilidade considerando o intervalo de 1 entre as posições do tabuleiro.

delta4/3

Calcula a utilidade considerando o intervalo de 4 entre as posições do tabuleiro.

delta5/3

Calcula a utilidade considerando o intervalo de 5 entre as posições do tabuleiro.

delta3/3

Calcula a utilidade considerando o intervalo de 3 entre as posições do tabuleiro.

delta17/3

Calcula a utilidade considerando o intervalo de 17 entre as posições do tabuleiro.

delta19/3

Calcula a utilidade considerando o intervalo de 19 entre as posições do tabuleiro.

delta20/3

Calcula a utilidade considerando o intervalo de 20 entre as posições do tabuleiro.

delta21/2

Calcula a utilidade considerando o intervalo de 21 entre as posições do tabuleiro.

delta13/2

Calcula a utilidade considerando o intervalo de 13 entre as posições do tabuleiro.

drawPos/2

Retorna verdadeiro se o jogo está empatado.

winPos/2

Este procedimento contém todas as jogadas vencedoras possíveis no jogo. São levados em consideração todos os deltas mencionados acima.

equal/5

Retorna verdadeiro se uma tupla é formada por 5 elementos iguais.

min_to_move/1

Retorna verdadeiro se o próximo jogador a jogar for o MIN.

max_to_move/1

Retorna verdadeiro se o próximo jogador a jogar for o MAX.

alphabeta/6

Algoritmo alpha-beta prunning retirado do livro do **Ivan Bratko** e modificado para ir até uma profundidade desejada. Caso isto não fosse feito, iria ocorrer estouro da pilha uma vez que a quantidade de estados possíveis é muito grande.

2. Grafo de Relacionamentos entre os Procedimentos