Projeto Nº 2: Época Normal

Inteligência Artificial - Escola Superior de Tecnologia de Setúbal
2020/2021

Prof. Joaquim Filipe
Eng. Filipe Mariano

1. Problema do Quatro: Descrição

O projeto destina-se a resolver o problema baseado no jogo do Quatro. O jogo é constituído por um tabuleiro de 4 x 4 casas e um conjunto de peças (até 16 dependente dos problemas) que possuem traços característicos de forma e de cor: Existem peças brancas ou pretas; parte delas são altas, a outra parte são baixas; umas são quadradas outras são redondas. Algumas peças são cheias outras são ocas.

Figura 1: Exemplo do Problema do Quatro.

1.1. Tabuleiro

O jogo é constituído por um tabuleiro de 4 x 4 casas e um conjunto de 16 peças, que possuem traços característicos de forma e de cor. Todas as peças de jogo terão as seguintes características:

1. Branca ou Preta
2. Alta ou Baixa
3. Quadrada ou Redonda
4. Cheia ou Oca

O que significa que as 16 peças do jogo encontram-se divididas da seguinte forma, pelas características supramencionadas:

- Existem 8 peças brancas e 8 peças pretas
- Existem 8 peças altas e 8 peças baixas
- Existem 8 peças quadradas e 8 peças redondas
- Existem 8 peças cheias e 8 peças ocas

1.2. Desenrolar do Jogo / Jogada

À semelhança do que ocorreu na 1ª fase do Projeto, existe apenas um único movimento possível que é a colocação de uma peça da reserva, i.e. por colocar, no tabuleiro.

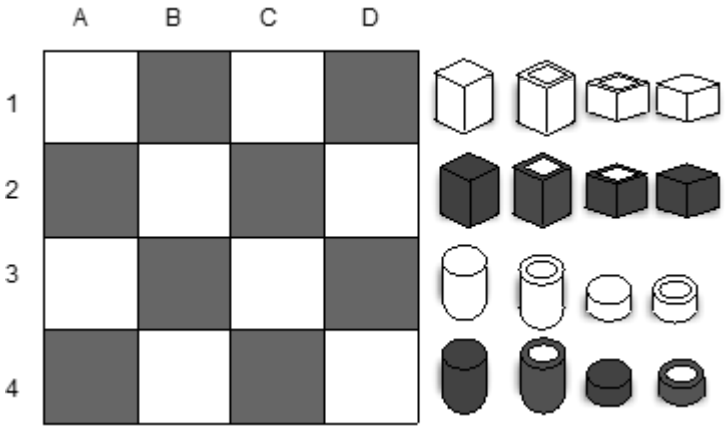

Figura 2: O tabuleiro vazio do problema "Quatro" com uma reserva de 16 peças.

- O jogo é sempre iniciado com um tabuleiro vazio.
- O jogo do Quatro joga-se com 2 jogadores em que cada jogador tem as mesmas peças para poder realizar uma jogada, ou seja, as peças disponíveis serão utilizadas para as jogadas do Jogador 1 e do Jogador 2.
- Cada jogada consiste na colocação de uma peça disponível (na reserva de peças) numa casa vazia.

1.3. Determinar o Vencedor

O jogo termina se um Jogador após colocar uma peça completar uma linha de 4 peças (na horizontal, na vertical ou na diagonal) que compartilhem pelo menos um traço em comum (4 peças pretas, ou 4 peças altas, ou 4 peças ocas, etc).

Caso não seja encontrado um vencedor e não existir mais peças para jogar (tabuleiro cheio), o jogo termina empatado.

2. Objetivo do Projeto: Versão Dois Jogadores

2.1. Objetivo

No âmbito deste projeto vamos considerar o Jogo do Quatro na versão de dois jogadores, possibilitando um enquadramento teórico– prático com os conhecimentos adquiridos no âmbito da Teoria de Jogos. Neste caso aplicam–se as regras de jogo descritas na Secção 1.

Tal como na 1ª fase do Projecto, pretende–se que os alunos desenvolvam um programa, em `Lisp`. O programa deverá implementar o algoritmo AlfaBeta ou Negamax com cortes alfa–beta, e as funções auxiliares que permitirão realizar as partidas do jogo.

Pretende–se que o programa permita ao computador vencer o jogador humano ou um outro computador, pelo que deverá funcionar em dois modos:

1. Humano vs Computador
2. Computador vs Computador

No modo 1, no início de cada partida, o utilizador deve decidir quem começa, humano ou computador, e qual o tempo limite para o computador jogar, enquanto que no modo 2 apenas é necessário definir o tempo

limite.

Os jogadores, à vez, escolhem a peça que pretendem colocar no tabuleiro. Um jogador vence se ao colocar a peça consegue formar uma linha, coluna ou diagonal em que todas as peças tenham uma característica comum. O jogo termina quando não existir mais nenhuma peça a ser colocada no tabuleiro, terminando em empate se todo o tabuleiro estiver preenchido e não se tiver verificado nenhuma situação de vitória.

O Jogo do Quatro tem a particularidade de ser um jogo em que apenas existe um conjunto de peças para os dois jogadores em vez de um conjunto separado para cada jogador.

2.2. Funcionamento

No início de cada partida, o utilizador deve decidir quem começa, humano ou computador, e qual o tempo limite para o computador jogar, o qual será de `X` milissegundos. O valor de `X` deverá ser um valor compreendido entre 1000 e 5000 milissegundos.

A jogada do jogador computador é encontrada através da aplicação do algoritmo de jogo, o AlfaBeta ou Negamax com cortes alfa–beta.

No que diz respeito à jogada do jogador humano, o programa deverá ler a jogada inserida através do teclado (coordenadas da linha e coluna), apresentar a sua própria jogada no ecrã e repetir o procedimento até a partida terminar. O programa deverá ainda ir escrevendo num ficheiro `log.dat` o número de nós analisados, o número de cortes efetuados (de cada tipo), o tempo gasto em cada jogada e o tabuleiro atual.

3. Formulação do Problema

3.1. Tabuleiro

O tabuleiro é representado sob a forma de uma estrutura de `4 x 4` casas. A notação das colunas utiliza as letras `A, B, C, D` e a notação das linhas utiliza os números `1, 2, 3, 4`. Por exemplo, a casa B3 faz referência a casa situada na segunda coluna da terceira linha. A Figura 2 apresenta o tabuleiro vazio.

É assim possível representar em LISP o tabuleiro como uma lista de 4 listas compostas, cada uma composta por 4 elementos, em que cada elemento representa uma casa:

- O valor `0` (zero) representa uma casa vazia
- Uma lista de quatro elementos representa uma peça. Nesta lista, o **primeiro elemento representa a cor da peça** (`preta` ou `branca`), **o segundo elemento representa a forma da peça** (`redonda` ou `quadrada`), **o terceiro elemento representa a altura** (`alta` ou `baixa`) e **o quarto elemento representa a densidade** (`cheia` ou `oca`). Desta forma as 16 peças podem ser representadas com as 16 listas seguintes:

```
(branca redonda alta oca)
(preta redonda alta oca)
(branca redonda baixa oca)
(preta redonda baixa oca)
(branca quadrada alta oca)
(preta quadrada alta oca)
(branca quadrada baixa oca)
(preta quadrada baixa oca)
```

```
(branca redonda alta cheia)
(preta redonda alta cheia)
(branca redonda baixa cheia)
(preta redonda baixa cheia)
(branca quadrada alta cheia)
(preta quadrada alta cheia)
(branca quadrada baixa cheia)
(preta quadrada baixa cheia)
```

Tabuleiro com reserva de peças:

Para representar um tabuleiro completo, é necessário incluir a lista de peças por colocar no tabuleiro (até 16 no inicio da resolução do problema). Desta forma um tabuleiro e a reserva de peças podem ser representados por uma lista de duas listas, na qual a primeira lista representa o tabuleiro e a segunda lista a reserva. Um exemplo de representação em LISP do tabuleiro vazio com reserva de peças, seria da seguinte forma:

```
(
    (
        (0 0 0 0) (0 0 0 0) (0 0 0 0) (0 0 0 0)
    )
    (
        (branca redonda alta oca)
        (preta redonda alta oca)
        (branca redonda baixa oca)
        (preta redonda baixa oca)
        (branca quadrada alta oca)
        (preta quadrada alta oca)
        (branca quadrada baixa oca)
        (preta quadrada baixa oca)
        (branca redonda alta cheia)
        (preta redonda alta cheia)
        (branca redonda baixa cheia)
        (preta redonda baixa cheia)
        (branca quadrada alta cheia)
        (preta quadrada alta cheia)
        (branca quadrada baixa cheia)
        (preta quadrada baixa cheia)
    )
)
```

Tabuleiro com uma peça colocada pelo Jogador 1:

O Jogador 1 e o Jogador 2 dispõem das mesmas peças de reserva que podem colocar no tabuleiro e, como jogam alternadamente, não é necessário registar qual o jogador que colocou uma determinada peça, visto que o critério para determinar o vencedor é verificado sempre que um jogador faz a sua jogada. Assim, a colocação de uma peça no tabuleiro tem a mesma representação da 1ª fase do projeto.

```
(
    (
```

```
        ((branca redonda alta oca)  0 0 0) (0 0 0 0) (0 0 0 0) (0 0 0 0)
    )
    (
        (preta redonda alta oca)
        (branca redonda baixa oca)
        (preta redonda baixa oca)
        (branca quadrada alta oca)
        (preta quadrada alta oca)
        (branca quadrada baixa oca)
        (preta quadrada baixa oca)
        (branca redonda alta cheia)
        (preta redonda alta cheia)
        (branca redonda baixa cheia)
        (preta redonda baixa cheia)
        (branca quadrada alta cheia)
        (preta quadrada alta cheia)
        (branca quadrada baixa cheia)
        (preta quadrada baixa cheia)
    )
)
```

3.2. Estrutura do Programa

O programa deverá estar dividido em três partes, cada uma num ficheiro diferente:

1. Uma parte para o algoritmo AlfaBeta ou Negamax (`algoritmo.lisp`).
2. Outra que deve conter as funções que permitem escrever e ler em ficheiros e tratar da interação com o utilizador (`interact.lisp`).
3. E a terceira parte corresponde aos operadores do jogo (`jogo.lisp`).

Enquanto que a primeira parte do programa deverá ser genérica para qualquer jogo que recorre ao algoritmo AlfaBeta ou Negamax com cortes alfa–beta (independente do domínio de aplicação), a segunda e a terceira parte são específicas do Jogo do Quatro (dependente do domínio de aplicação).

Na parte 2 deverá ser definida uma função com o nome `jogar` com os parâmetros `estado` e `tempo`, e que devolva uma lista com o tabuleiro em que o deverá ser feita a próxima jogada.

4. Grupos

Os projetos deverão ser realizados em grupos de, no máximo, duas pessoas sendo contudo sempre sujeitos a avaliação oral individual para confirmação da capacidade de compreensão dos algoritmos e de desenvolvimento de código em Lisp.

Os grupos deverão ser os mesmos que realizaram a 1ª fase do Projeto.

5. Datas

Entrega do projeto: 29 de Janeiro de 2021, até às 23:00.
Discussão do projeto: Inicio de Fevereiro de 2021.

6. Documentação a Entregar

A entrega do projeto e da respetiva documentação deverá ser feita através do Moodle, na zona do evento "Entrega do 2º Projeto". Todos os ficheiros a entregar deverão ser devidamente arquivados num ficheiro comprimido (`ZIP` com um tamanho máximo de 5Mb), até à data acima indicada. O nome do arquivo deve seguir a estrutura `nomeAluno1_numeroAluno1_nomeAluno2_numeroAluno2_P2`.

6.1. Código Fonte

Os ficheiros de código devem ser devidamente comentados e organizados da seguinte forma:

interact.lisp Carrega os outros ficheiros de código, escreve e lê de ficheiros e trata da interação com o utilizador.

jogo.lisp Código relacionado com o problema.

algoritmo.lisp Deve conter a implementação do algoritmo de jogo independente do domínio.

6.2. Manuais

No âmbito da Unidade Curricular de Inteligência Artificial pretende–se que os alunos pratiquem a escrita de documentos recorrendo à linguagem de marcação ***Markdown***, que é amplamente utilizada para os ficheiros ***ReadMe*** no ***GitHub***. Na Secção 4 do guia de Laboratório nº 2, encontrará toda a informação relativa à estrutura recomendada e sugestões de ferramentas de edição para ***Markdown***.

Para além de entregar os ficheiros de código e de problemas, é necessário elaborar e entregar 2 manuais (o manual de utilizador e o manual técnico), em formato `PDF` juntamente com os sources em `MD`, incluídos no arquivo acima referido:

ManualTecnico.pdf O Manual Técnico deverá conter o algoritmo implementado, devidamente comentado e respetivas funções auxiliares; descrição dos tipos abstratos usados no programa; identificação das limitações e opções técnicas. Deverá ser apresentada uma análise crítica dos resultados das execuções do programa, onde deverá transparecer a compreensão das limitações do projeto. Deverão fazer uma análise estatística acerca de uma execução do programa contra um adversário humano, mencionando o limite de tempo usado e, para cada jogada: o respetivo valor, a profundidade do grafo de jogo e o número de cortes efetuado no processo de análise. Poderão utilizar os dados do ficheiro `log.dat` para isso.

ManualUtilizador.pdf O Manual do Utilizador deverá conter a identificação dos objetivos do programa, juntamente com descrição geral do seu funcionamento; explicação da forma como se usa o programa (acompanhada de exemplos); descrição da informação necessária e da informação produzida (ecrã/teclado e ficheiros); limitações do programa (do ponto de vista do utilizador, de natureza não técnica).

7. Avaliação

Tabela 1: Grelha de classificação.

Funcionalidade	Valores
Algoritmo AlfaBeta ou Negamax com Cortes alfa–beta e determinar Jogada do Computador	5
Jogada Humano	1
Apresentar Estatísticas a cada Jogada (ecrã e ficheiro)	2

Funcionalidade	Valores
Função de Avaliação	2
Implementação do limite de tempo para o computador	1
Procura quiescente	1
Memoização	1
Operadores do jogo	2
Ordenação dos nós	1
Qualidade do código	2
Manuais (utilizador e técnico)	2
Total	20

A avaliação do projeto levará em linha de conta os seguintes aspectos:

- Data de entrega final – Existe uma tolerância de 3 dias em relação ao prazo de entrega, com a penalização de 1 valor por cada dia de atraso. Findo este período a nota do projeto será 0.
- Correção processual da entrega do projeto – (Moodle; manuais no formato correto). Anomalias processuais darão origem a uma penalização que pode ir até 3 valores.
- Qualidade técnica – Objetivos atingidos; Código correto; Facilidade de leitura e manutenção do programa; Opções técnicas corretas.
- Qualidade da documentação – Estrutura e conteúdo dos manuais que acompanham o projeto.
- Avaliação oral – Eficácia e eficiência da exposição; Compreensão das limitações e possibilidades de desenvolvimento do programa. Nesta fase poderá haver lugar a uma revisão total da nota de projeto.

Campeonato

Após a entrega dos projetos, será realizado um campeonato entre os programas de cada grupo (os alunos deverão manifestar a sua intenção de participar, fazendo um registo no Moodle por ocasião da entrega do projeto 2). Para participar, será necessário que:

a) o programa esteja totalmente inserido num package com a designação **p<numeroAluno1>-<numeroAluno2>**.
b) seja definida uma função com o nome `jogar` com os parâmetros `estado` e `tempo` e que devolva uma lista com as coordenadas da jogada e o novo estado.

Todos os participantes neste campeonato recebem um bónus na nota final do projeto 2, com destaque para os dois primeiros lugares:
a) 3 valores para o grupo vencedor.
b) 2 valores para o grupo que acabar na 2ª posição.
c) 1 valor para os restantes grupos que se inscreverem no campeonato e em que o programa esteja corretamente estruturado para participar.

8. Recomendações Finais

Com este projeto pretende–se motivar o paradigma de programação funcional. A utilização de variáveis globais, de instruções de atribuição do tipo set, setq, setf, de ciclos, de funções destrutivas ou de quaisquer funções com efeitos laterais é fortemente desincentivada dado que denota normalmente uma baixa qualidade técnica. Exceptua–se a utilização de setf em conjugação com gethash.

ATENÇÃO: Suspeitas confirmadas de plágio serão penalizadas com a anulação de ambos os projetos envolvidos (fonte e destino), e os responsáveis ficam sujeitos à instauração de processo disciplinar