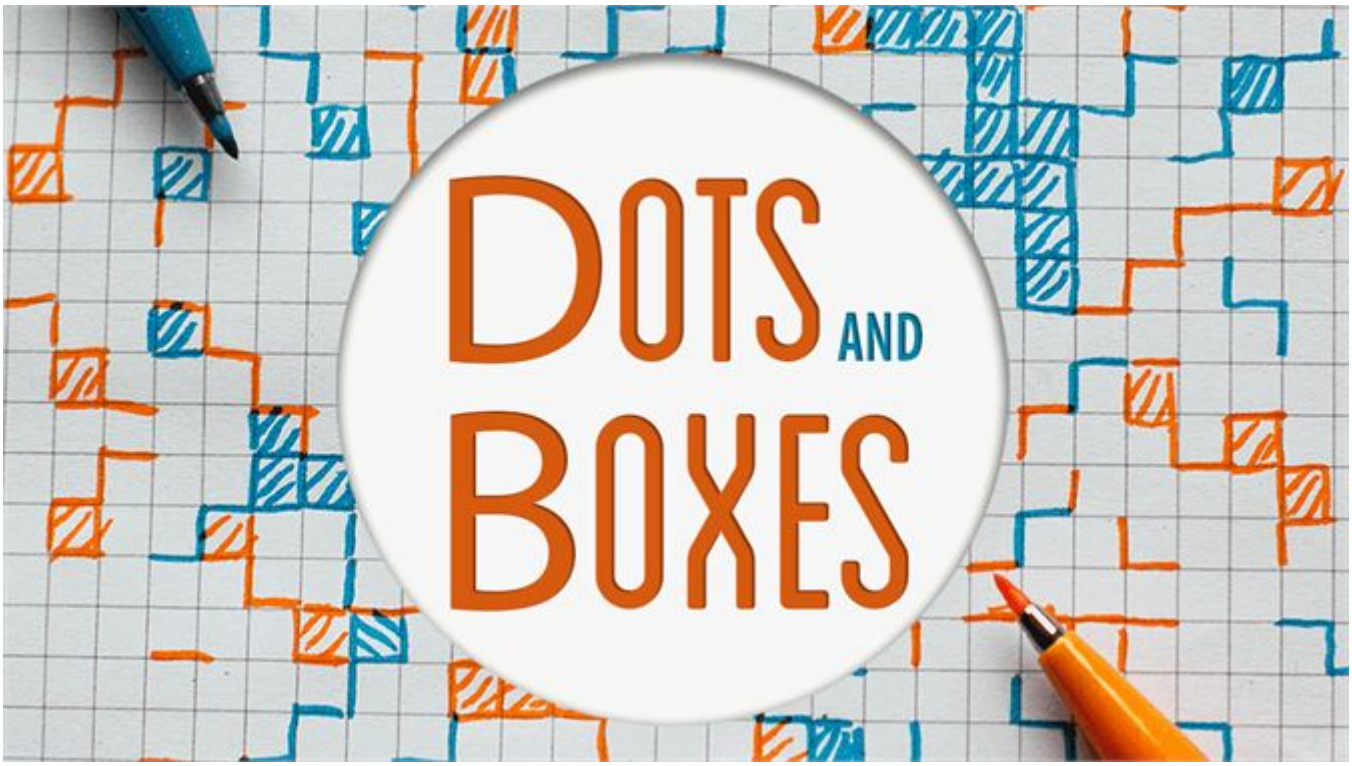
Projeto Nº 2: Época Normal



Inteligência Artificial - Escola Superior de Tecnologia de Setúbal
2022/2023

Prof. Joaquim Filipe
Eng. Filipe Mariano

1. Descrição do Jogo

O `Dots and Boxes` é um jogo de 2 jogadores criado por Édouard Lucas em 1889. Possui várias denominações, tais como **dots and dashes**, **game of dots**, **dot to dot grid**, **boxes**, entre outros

É um jogo constituído por um tabuleiro de `n` * `m` caixas (`n` linhas de caixas e `m` colunas de caixas). Cada caixa é delimitada por 4 pontos entre os quais é possível desenhar um arco. Quando os quatro pontos à volta de uma caixa tiverem conectados por 4 arcos, a caixa é considerada fechada. O espaço da solução é portanto constituído por `n` * `m` caixas, `(n + 1) * (m + 1)` pontos e `(m * (n + 1)) + (n * (m + 1))` arcos.

O jogo inicia com um tabuleiro vazio em que os jogadores alternadamente vão colocando um arco horizontal ou vertical. Quando o arco colocado por um jogador fecha uma caixa, essa caixa conta como 1 ponto para o jogador que colocou o arco e esse jogador deve jogar novamente.

O jogo termina quando todas as caixas tiverem fechadas, ou seja, não existirem mais arcos para colocar, ganhando o jogador que fechou mais caixas.

A figura 1 apresenta um exemplo do puzzle com 30 caixas `(n=5 e m=6)`, com 51 arcos conectados, 8 caixas fechadas para o jogador vermelho e 4 caixas fechadas para o jogador azul.

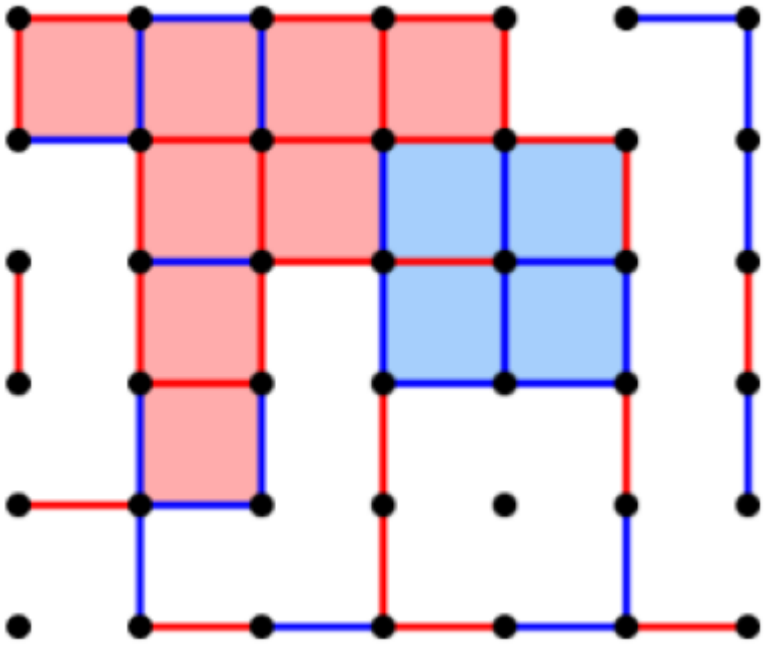


Figura 1: Exemplo de um jogo com o tabuleiro `5` * `6`.

As regras a contemplar nesta 2ª fase do Projeto são:

- O jogo é disputado entre `2` jogadores.
- Para esta versão do jogo, irá se utilizar um tabuleiro de `30 caixas`, em que `n=5` e `m=6` tal como ilustrado na figura 1.
- As jogadas são feitas à vez e, em cada turno, o jogador coloca um arco horizontal ou arco vertical numa posição vazia.
- **O jogador que coloque o último arco de uma caixa, ganha 1 ponto e joga novamente**. Se na jogada seguinte fechar novamente um arco, poderá continuar a jogar até que o arco colocado não feche nenhuma caixa.
- O jogo termina quando nenhum dos jogadores consegue colocar mais arcos.
- Quando o jogo termina, os jogadores contam o número de pontos obtidos ao fechar caixas no tabuleiro, e o jogador que tiver **o maior** número de caixas fechadas é o vencedor.

Nota: Para conhecer melhor as regras do jogo poderá visitar o site http://dotsandboxes.org/

1.2 Modo de Funcionamento

O programa deverá funcionar em 2 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.

O jogo é iniciado pelo Jogador 1 colocando um arco (horizontal ou vertical) numa posição vazia. Por sua vez, o Jogador 2 inicia o seu jogo colocando também um arco (horizontal ou vertical) numa posição vazia. Esta

sucessão de arcos alternados entre jogadores só é interrompida quando um jogador consegue fechar uma caixa, podendo jogar um novo arco.

O jogo prossegue até que nenhum jogador consiga colocar mais nenhum arco, contando o número de caixas que cada jogador fechou, havendo a hipótese de existir empate.

2. Objetivo do Projeto

Pretende-se desenvolver um programa, em LISP, para jogar o jogo **Dots and Boxes**. O programa deverá implementar o algoritmo AlfaBeta ou Negamax com cortes alfa-beta e as funções auxiliares que irão permitir realizar as partidas do jogo. Pretende-se que o programa permita ao computador vencer o jogador humano ou um outro computador.

3. Formulação do Problema

3.1. Tabuleiro

O tabuleiro é representado sob a forma de uma lista composta por 2 listas.

- A primeira lista representa os arcos horizontais. Este lista é composta por `6` listas, em que cada uma delas é composta por `6` elementos. O valor de cada elemento é `0` se não existir arco nessa posição, `1` se tiver sido o **Jogador 1** a colocar o arco e `2` se tiver sido o **Jogador 2** a colocar o arco.
- A segunda lista representa os arcos verticais. Esta lista é composta por `7` listas, em que cada uma delas composta por `5` elementos. O valor de cada elemento é `0` se não existir arco nessa posição, `1` se tiver sido o **Jogador 1** a colocar o arco e `2` se tiver sido o **Jogador 2** a colocar o arco.

De seguida, mostram-se respectivamente a representação do tabuleiro inicial (sem arcos colocados) e do tabuleiro final apresentado na Figura 1.

```
;tabuleiro inicial
(
 (;arcos horizontais
    (0 0 0 0 0 0)
    (0 0 0 0 0 0)
    (0 0 0 0 0 0)
    (0 0 0 0 0 0)
    (0 0 0 0 0 0)
    (0 0 0 0 0 0)
 )
 (;arcos verticais
    (0 0 0 0 0)
    (0 0 0 0 0)
    (0 0 0 0 0)
    (0 0 0 0 0)
    (0 0 0 0 0)
    (0 0 0 0 0)
    (0 0 0 0 0)
 )
)
```

```
;tabuleiro figura 1
(
 (;arcos horizontais
    (1 2 1 1 0 2)
    (2 1 1 1 1 0)
    (0 2 1 1 2 0)
    (0 1 0 2 2 0)
    (1 2 0 0 0 0)
    (0 1 2 1 2 1)
 )
 (;arcos verticais
    (1 0 1 0 0)
    (2 1 1 2 2)
    (2 1 1 2 0)
    (1 2 2 1 1)
    (1 2 2 0 0)
    (0 1 2 1 2)
    (2 2 1 2 0)
 )
)
```

Uma posição contendo o valor `0` simboliza um arco vazio. Caso contenha o valor `1` significa que está colocado um arco do **Jogador 1**, e caso contenha o valor `2` significa que está colocado um arco do **Jogador 2**.

3.2. Representação do Estado

Nesta fase do projeto, o estado deverá ser representado por uma lista com o tabuleiro e outra lista com o número de caixas que cada jogador já fechou, sendo o primeiro elemento o número de caixas fechadas pelo **Jogador 1** e o segundo elemento o número de caixas fechadas pelo **Jogador 2**.

```
(;estado
    (;tabuleiro figura 1
        (;arcos horizontais
            (1 2 1 1 0 2)
            (2 1 1 1 1 0)
            (0 2 1 1 2 0)
            (0 1 0 2 2 0)
            (1 2 0 0 0 0)
            (0 1 2 1 2 1)
        )
        (;arcos verticais
            (1 0 1 0 0)
            (2 1 1 2 2)
            (2 1 1 2 0)
            (1 2 2 1 1)
            (1 2 2 0 0)
            (0 1 2 1 2)
            (2 2 1 2 0)
        )
    )
```

```
    (8 4);8 caixas fechadas pelo jogador1 e 4 caixas fechadas pelo jogador2
)
```

3.3. Jogada

Uma jogada é uma lista composta por três elementos: o par de coordenadas da linha e da coluna e o tipo de arco colocado (horizontal ou vertical).

3.4. 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 (jogo.lisp).
3. E a terceira parte corresponde aos operadores do jogo (puzzle.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 Dots and Boxes (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 em que o primeiro elemento é uma jogada realizada e o segundo elemento o novo estado do jogo resultante da aplicação da jogada.

4. Algoritmo

O algoritmo de jogo a implementar é o AlfaBeta ou Negamax com cortes alfa-beta. Como forma de verificação das diversas jogadas realizadas durante um jogo, antes de devolver cada resultado, o programa deverá escrever num ficheiro `log.dat` e no ecrã qual a jogada realizada, o novo estado, o número de nós analisados, o número de cortes efetuados (de cada tipo) e o tempo gasto. Caso tenha sido escolhido o modo **Humano vs Computador**, o programa deverá ler cada jogada do jogador humano inserida através do teclado e repetir o procedimento até um dos jogadores ganhar a partida.

> O tempo limite para o computador jogar será de X milissegundos. O valor de X deverá ser um valor compreendido entre 1 e 20 segundos que deverá ser fornecido pelo utilizador do jogo, nomeadamente lido do teclado caso seja um jogo contra um humano, ou recebido por parâmetro na função `jogar`, caso seja contra um computador (vide subsecção Campeonato na secção 8).

5. Grupos

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

Os grupos deverão ser os mesmos que realizaram a 1ª fase do Projeto, seguindo as regras anteriormente estipuladas em que deve ser constituido por alunos que frequentam a mesma turma e, apenas em casos

excepcionais e com aprovação dos docentes, é que poderão existir grupos com elementos provenientes de turmas diferentes.

6. Datas

Entrega do projeto: 27 de Janeiro de 2023, até às 23:00.

7. Documentação a entregar

A entrega do projeto e da respectiva documentação deverá ser feita através do Moodle, na zona do evento "entrega do segundo 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 `P2_nomeAluno1_numeroAluno1_nomeAluno2_numeroAluno2`.

7.1. Código fonte

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

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

puzzle.lisp Código relacionado com o problema.

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

7.2. 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, é necessário elaborar e entregar 2 manuais (o manual de utilizador e o manual técnico), em formato `PDF`, incluidos 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;
- Uma análise crítica dos resultados das execuções do programa, onde deverá transparecer a compreensão das limitações do projeto;
- 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).

8. Avaliação

Tabela 1: Grelha de classificação.

Funcionalidade	**Valores**
Algoritmo AlfaBeta ou Negamax com Cortes alfa-beta	5
Operadores do jogo	2
Função de Avaliação	2
Jogada Humano	1
Apresentar estatísticas a cada jogada (ecrã e ficheiro)	2
Implementação do limite de tempo para o computador	1
Procura quiescente	1
Memoização	1
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 aspetos:

- Data de entrega final – Existe uma tolerância de 4 dias em relação ao prazo de entrega, com a penalização de 1 valor por cada dia de atraso. Terminado 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<numeros>** em que **<numeros>**::=**<numero de um elemento do grupo>-<numero do outro elemento do grupo>** | **<numero do unico elemento do grupo>**.
b) seja definida uma função com o nome `jogar` com os parâmetros `estado` e `tempo` (em milissegundos) e que devolva `uma lista com a jogada e com o novo estado`.

Exemplo de input:

```
(jogar
    (;estado
        (
            (
                (1 2 1 1 0 2)
                (2 1 1 1 1 0)
                (0 2 1 1 2 0)
                (0 1 0 2 2 0)
                (1 2 0 0 0 0)
                (0 1 2 1 2 1)
            )
            (
                (1 0 1 0 0)
                (2 1 1 2 2)
                (2 1 1 2 0)
                (1 2 2 1 1)
                (1 2 2 0 0)
                (0 1 2 1 2)
                (2 2 1 2 0)
            )
        )
        (8 4);8 caixas fechadas pelo jogador1 e 4 caixas fechadas pelo jogador2
    )
    5000 ;tempo da jogado em ms
)
```

Exemplo de output:

```
(
    (3 1 arco-horizontal);jogada
    (;novo estado
        (
            (
                (1 2 1 1 0 2)
                (2 1 1 1 1 0)
                (1 2 1 1 2 0)
                (0 1 0 2 2 0)
                (1 2 0 0 0 0)
                (0 1 2 1 2 1)
            )
            (
                (1 0 1 0 0)
```

```
                (2 1 1 2 2)
                (2 1 1 2 0)
                (1 2 2 1 1)
                (1 2 2 0 0)
                (0 1 2 1 2)
                (2 2 1 2 0)
            )
        )
        (8 4);8 caixas fechadas pelo jogador1 e 4 caixas fechadas pelo jogador2
    )
)
```

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.

9. 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, sem ser no contexto de *closures*, é fortemente desincentivada dado que denota normalmente uma baixa qualidade técnica.

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.