# **Projeto Nº 2**: Época Normal

Inteligência Artificial - Escola Superior de Tecnologia de Setúbal 2018/2019

Prof. Joaquim Filipe
Prof. Hugo Silva
Eng. Filipe Mariano

## 1. Descrição do Jogo - Adji-boto

O Adji-boto* (uma variante do Adji-boto e Oware, especialmente concebida para este projeto) é um jogo de estratégia da família dos jogos de tabuleiro Mancala, que hoje em dia ainda têm uma grande popularidade. Nesta versão do jogo Adji-boto*, existe um tabuleiro com 2 linhas e 6 buracos em cada linha e é iniciado com 8 peças em cada buraco.

Figura 1: Tabuleiro real de Mancala na versão Oware.

### 1.1. Regras

O jogo desenrola-se da seguinte forma:

- Cada jogador fica com uma das linhas de buracos;
- As jogadas são feitas à vez e, em cada turno, um jogador retira todas as peças de um dos buracos da sua linha e vai depositando cada uma dessas peças retiradas no buraco adjacente e em cada um dos buracos seguintes, no sentido contrário ao dos ponteiros do relógio;
- Quando ao depositar uma peça em cada buraco seguinte, se chegar ao buraco em que inicialmente se retirou as peças (foi efetuada uma volta completa), deve-se saltar essa casa colocando a(s) peça(s) que ainda sobra(m) na(s) casa(s) seguinte(s);
- Se a jogada acabar numa casa da linha do tabuleiro contrária aquela em que estavam inicialmente as peças, **ficando na casa final 1, 3 ou 5 peças, essas peças podem ser "capturadas", sendo retiradas do tabuleiro**.

### 1.2. Exemplo de Jogada com Direito a Remoção de Peças

Para tornar mais explícito como se desenrola o jogo, irá ser apresentado num esquema de duas imagens um tabuleiro antes de uma jogada e após uma jogada que termina na linha contrária aquela em que estavam inicialmente as peças:

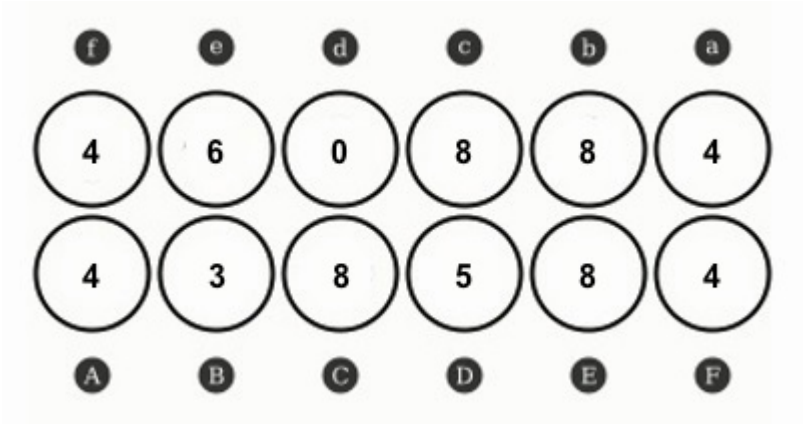


Figura 2: Tabuleiro antes da jogada.

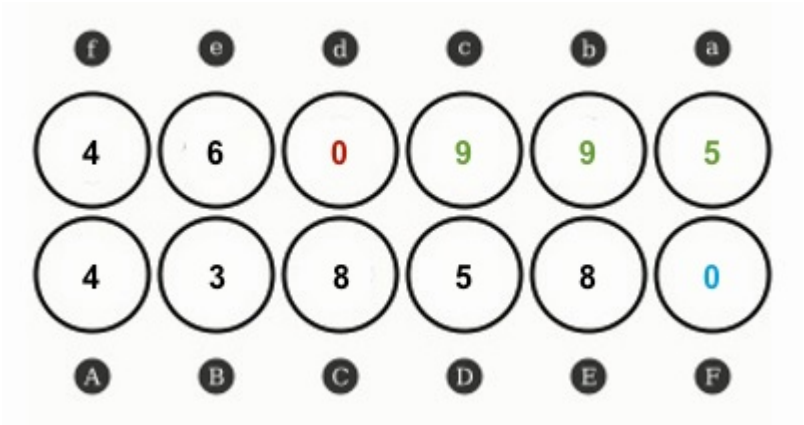


Figura 3: Tabuleiro após a jogada que termina na linha contrária aquela em que estavam inicialmente as peças.

Passos:

- O buraco selecionado pelo jogador para iniciar a jogada foi o `F` na 2ª linha, por isso retirou todas as peças desse buraco - número a azul;
- As peças existentes no buraco anteriormente mencionado foram depositadas nos buracos seguintes adjacentes, uma peça por buraco (`a`, `b`, `c`) - número a verde - tendo sido colocada a última peça no buraco `d` - número a vermelho;
- Como o buraco `d` ficou com `1` peça, pelo facto de ter sido depositada uma peça proveniente do buraco `F`, e pertence à linha contrária ao mesmo, pode ser aplicada a regra de **se na casa final ficarem 1, 3 ou 5 peças, essas peças podem ser capturadas**, sendo retiradas do tabuleiro - número a vermelho.

**NOTA:** Apenas por essa razão é que o buraco `d` ficou com `0` peças, caso contrário tinha ficado com o número de peças igual ao antes da jogada, mais a peça proveniente do buraco `F` (como demonstrado no exemplo seguinte).

### 1.3. Exemplo de Jogada sem Direito a Remoção de Peças

Para complementar o desenrolar do jogo, irá ser apresentado num esquema de duas imagens um tabuleiro antes de uma jogada e após uma jogada que termina na mesma linha em que estavam inicialmente as peças:

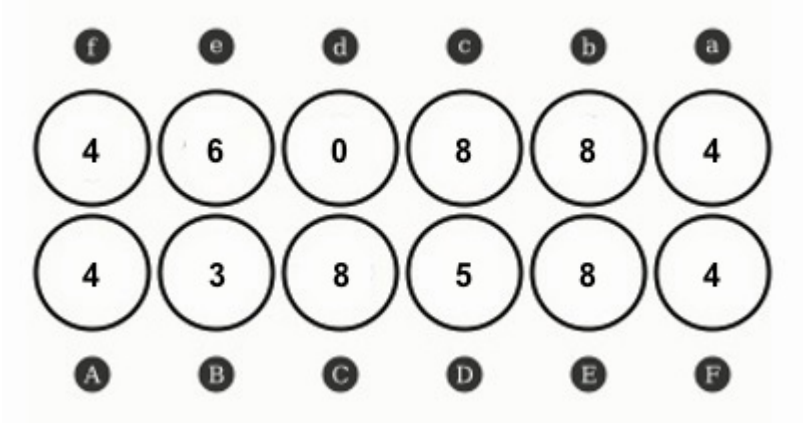


Figura 4: Tabuleiro antes da jogada.

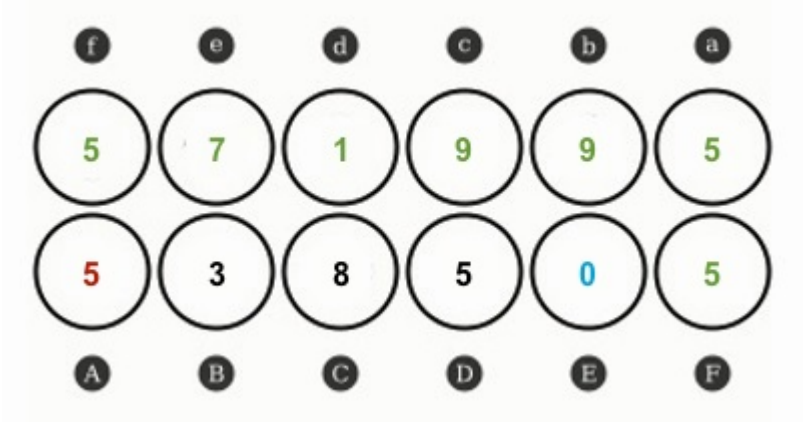


Figura 5: Tabuleiro após a jogada que termina na mesma linha em que estavam inicialmente as peças.

Passos:

- O buraco selecionado pelo jogador para iniciar a jogada foi o `E` na 2ª linha, por isso retirou todas as peças desse buraco - número a azul;
- As peças existentes no buraco anteriormente mencionado foram depositadas nos buracos seguintes adjacentes, uma peça por buraco (`F`, `a`, `b`, `c`, `d`, `e`, `f`) - número a verde - tendo sido colocada a última peça no buraco `A` - número a vermelho;
- O buraco `A` ficou com `5` peças, pelo facto de ter sido depositada uma peça proveniente do buraco `E`, contudo pertence à mesma linha, razão pela qual não pode ser aplicada a regra de **se na casa final ficarem 1, 3 ou 5 peças, essas peças podem ser capturadas**, e as peças não são retiradas do tabuleiro - número a vermelho.

### 1.4. Determinar o Vencedor

O jogo termina quando não existirem peças no tabuleiro, sendo o vencedor o jogador que capturar **mais** peças.

## 2. Objetivo do Projeto

No âmbito deste projeto vamos considerar o Adji-boto 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 1.1.

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.

O jogo é iniciado pelo Jogador 1 seleccionando um buraco, na sua linha, a partir do qual serão retiradas peças para distribuição pelo tabuleiro. O processo é análogo para o Jogador 2, que selecciona um buraco da linha contrária à do Jogador 1.

Caso um dos jogadores não possua peças em nenhum dos buracos da sua linha, cede a vez ao jogador oposto. O jogo termina quando não existem peças disponíveis em nenhum dos buracos do tabuleiro.

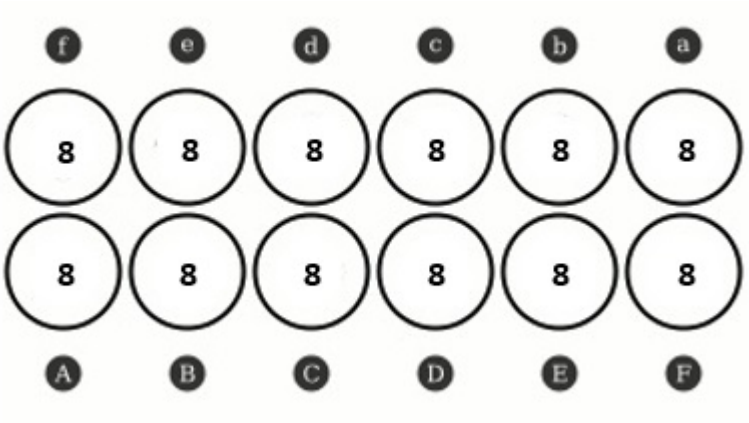


Figura 6: Estado inicial.

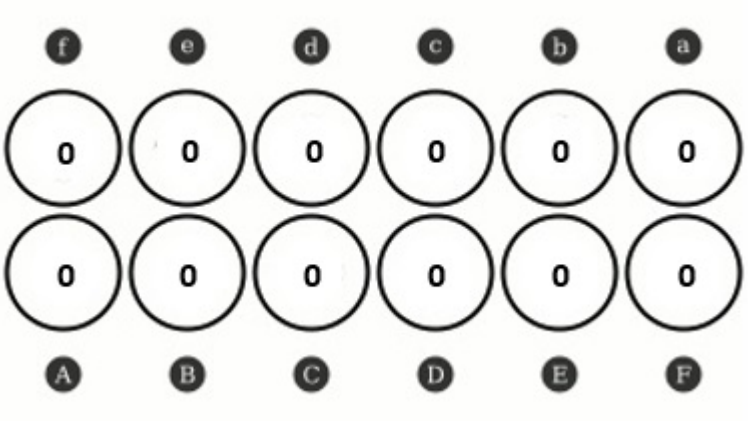


Figura 7: Estado final.

## 3. Formulação do Problema

### 3.1. Tabuleiro

O tabuleiro é representado sob a forma de uma lista composta por `2` outras listas, cada uma delas com `6` átomos. Cada uma das listas representa uma linha do tabuleiro, enquanto que cada um dos átomos representa um buraco dessa linha. Por outras palavras, o tabuleiro é representado por uma lista de listas em que `0` representa um buraco vazio e `#n` representa um buraco com `n` peças.

Abaixo mostram-se respectivamente a representação do tabuleiro inicial (Figura 6) e do tabuleiro finalizado apresentado na Figura 7.

```
(
 (8 8 8 8 8 8)
 (8 8 8 8 8 8)
)
```

Figura 8: Representação do tabuleiro inicial (Figura 6).

```
(
 (0 0 0 0 0 0)
 (0 0 0 0 0 0)
)
```

Figura 9: Representação do tabuleiro finalizado (Figura 7).

### 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 Adji-boto (dependente do domínio de aplicação).

Na parte 2 deverá ser definida uma função com o nome jogar com os parâmetros posicao e tempo, e que devolva uma lista com a posição em que o deverá ser feita a próxima 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 posição onde jogou, 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 a partida terminar.

> O tempo limite para o computador jogar será de X milissegundos. O valor de X deverá ser um valor compreendido entre 1 e 5 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, 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, 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**: 29 de Janeiro de 2019, até às 09:00 da manhã.

## 7. 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_P1`.

### 7.1. 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.

### 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 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).

## 8. Avaliação

Tabela 1: Grelha de classificação.

| **Funcionalidade** | **Valores** |
|---|---|
| Algoritmo AlfaBeta ou Negamax com Cortes alfa-beta | 5 |
| Função de Avaliação | 2 |
| Implementação do limite de tempo para o computador | 2 |
| Procura quiescente | 1 |
| Memoização | 1 |
| Operadores do jogo | 2 |
| Ordenação dos nós | 1 |
| Qualidade do código | 2 |
| Interação com o utilizador | 1 |
| Manuais (utilizador e técnico) | 3 |
| **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<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 `posicao`, `tempo` e que devolva `uma lista com a jogada e com a posição resultante`.

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 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.