



TIAGO ANDRÉ MENDES DE ALMEIDA RIBEIRO

TIAGO ALEXANDRE QUARESMA ALVES

Adji‒boto - Manual de Utilizador

Inteliêngia Artificial, Engenharia de Software

Ano letivo 2018 / 2019

Docentes

Professor, Joaquim Filipe

Professor, Hugo Silva

Engenheiro, Filipe Mariano

Índice

[1. Introdução 3](#_Toc532652583)

[2. Instalação 4](#_Toc532652584)

[3. Implementação técnica 5](#_Toc532652585)

# Introdução

Este documento é um guia de como utilizar a aplicação desenvolvida em lisp no âmbito do projeto de Inteligência Artificial

A aplicação 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 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.

O projeto implementado visa implementar os algoritmos BFS, DFS e A\*, para a resolução autónoma do jogos.

* BFS(breadth-first Search): Algoritmo de procura em largura, explora todos os sucessores de um nó antes de passar para o nível seguinte, segue a ordem crescente dos nós da figura no topo da secção.
* DFS(Depth-first Search): Algoritmo de procura em profundidade, explora sempre o primeiro sucessor de um nó, atingindo a máxima profundidade possível até encontrar uma solução, caso atinja o limite sem encontrar solução, percorre, utilizando os mesmos métodos, o sucessor mais próximo.
* A\*: Algoritmo de procura informada utiliza uma função heurística para calcular o valor de um nó e apenas explora os nós com o menos valor heurístico, garantindo assim, a melhor solução possível.

# Instalação

Para a aplicação correr é necessário no que no ficheiro projeto.lisp, se indique o caminho onde o projeto se encontra na função get-curr-dir.

Exemplo windows: C:/Users/${user}/Documents/projeto

Exemplo Mac: /Users/${user}/Documents/projeto

# Implementação técnica

O jogo começa por ser chamar a função (start-game), a mesma vai carregar as dependências necessárias e dar início ao jogo apresentando o menu de jogo.

Para escolher uma opção do menu é necessário introduzir o número da opção e carregar no Enter, e é levado para o menu que escolheu.

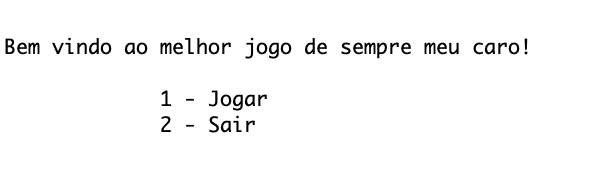


Figura 1 - Menu inicial

Ao escolher a opção jogar é apresentado um menu com os problemas que estão guardados no ficheiro problemas.dat, é necessário que escolha um.

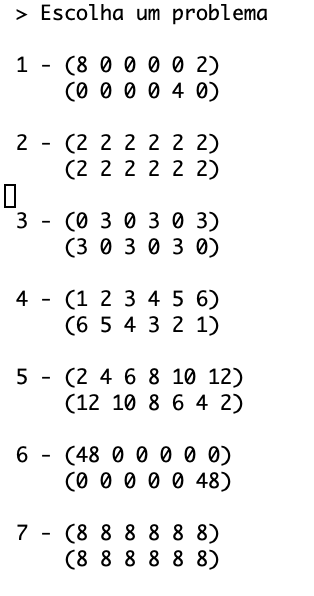


Figura 2 - Escolha do problema

Depois de ter sido o escolhido o problema é necessário escolher que algoritmo utilizar para aplicar na resolução do problemas

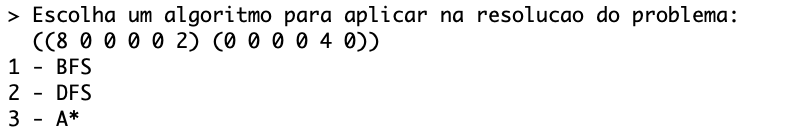


Figura 3 - Escolha do algortimo

Na escolha do A\*, ainda tem a opção de escolher que heurística usar

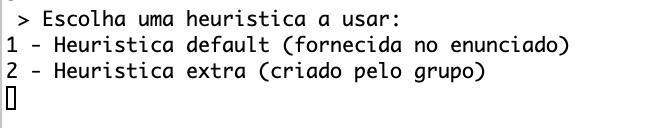


Figura 4 - Escolha de heuristica

No final do algoritmo, o resultado é guardar para o ficheiro resultados.dat e é apresentado no ecrã o resultado.

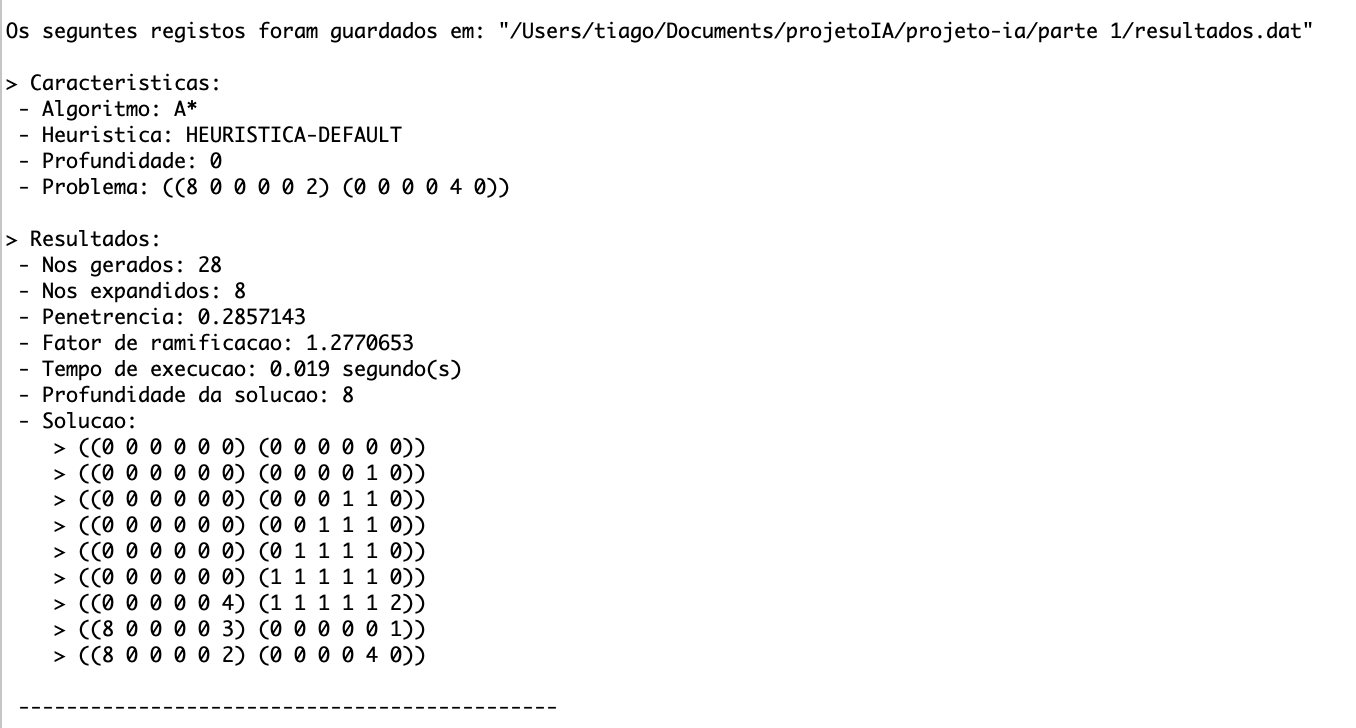


Figura 5 - Resultado final