# Manual do Utilizador - Projeto Adji-Boto

## Introdução

O projeto **Adji-Boto** é uma aplicação desenvolvida em **Common Lisp** para resolver problemas de tabuleiro baseados no jogo **Oware**. Utiliza algoritmos de pesquisa (BFS, DFS e A*) para encontrar a melhor sequência de movimentos, seja para capturar o maior número de peças ou esvaziar o tabuleiro.

Este manual destina-se aos utilizadores finais e fornece orientações sobre como utilizar a aplicação, incluindo exemplos, saídas e limitações.

## Objetivo

Resolver tabuleiros do jogo Adji-Boto utilizando algoritmos de pesquisa que exploram diferentes estratégias para alcançar o estado final do jogo (tabuleiro vazio).

## Requisitos

- Um ambiente Lisp compatível (ex.: SBCL, CLISP ou equivalente).
- Ficheiros do projeto:
  - `procura.lisp`
  - `puzzle.lisp`
  - `projeto.lisp`
  - Ficheiro de entrada: `problemas.dat` (contendo tabuleiros pré-definidos).
- Sistema operativo compatível (Windows, Linux ou macOS).

## Iniciar o Programa

1. Certifique-se de que todos os ficheiros necessários estão na mesma pasta.
2. Abra um terminal ou interpreter de Lisp.
3. Carregue o ficheiro principal:

```
(load "projeto.lisp")
```

4. Inicie o programa:

```
(init)
```

# Interação com o Programa

## Menu Principal

Após iniciar o programa, o seguinte menu será apresentado:

| Opção | Descrição |
|---|---|
| 1 | Resolver Jogo |
| 2 | Mostrar Entradas Disponíveis |
| 0 | Sair |

## Resolver um Tabuleiro

Selecionar a **opção 1** leva ao menu de seleção de algoritmos:

1. **Escolha um Algoritmo**:
    - **1 - BFS (Pesquisa em Largura)**: Explora os nós na ordem em que são descobertos, garantindo o caminho mais curto.
    - **2 - DFS (Pesquisa em Profundidade)**: Explora o caminho mais profundo primeiro, com um limite de profundidade definido pelo utilizador.
    - **3 - A (Pesquisa A)***: Utiliza heurísticas para priorizar soluções promissoras.
2. **Selecionar uma Entrada de Tabuleiro**:
    - Será apresentada uma lista numerada de tabuleiros disponíveis.
    - Introduza o número correspondente ao tabuleiro que deseja resolver.
3. **Execução do Algoritmo**:
    - O programa tentará resolver o tabuleiro utilizando o algoritmo selecionado.
    - Se utilizar A*, o utilizador deverá escolher uma heurística:
        - **1 - Heurística Padrão** (baseada nas peças restantes).
        - **2 - Heurística Personalizada** (baseada na distribuição das peças).
4. **Resultados**:
    - Se uma solução for encontrada:
        - O caminho de execução será apresentado, incluindo estados do tabuleiro, profundidade e valores heurísticos.
    - Se não houver solução:
        - O programa notificará o utilizador da impossibilidade de resolução.

# Mostrar Tabuleiros Disponíveis

Escolher **opção 2** apresentará todas as entradas de tabuleiro disponíveis no ficheiro `problemas.dat`. Cada tabuleiro terá um número associado, permitindo ao utilizador rever as configurações iniciais antes de selecionar uma para resolver.

# Exemplo de Execução

## **Passos**:

1. Iniciar o programa e escolher **opção 1** (resolver um tabuleiro).
2. Selecionar **A*** como algoritmo de pesquisa e escolher a heurística padrão.
3. Escolher o tabuleiro **1** da lista apresentada.

## **Saída**:

```
=== Adji-boto* ===
1. Solve Game
2. Show Available Entries
0. Exit
Select an option: 1

=== Adji-boto* ===
1. BFS
2. DFS
3. A*
0. Exit
Select an option: 3

=== Available Game Entries ===
1: ((0 0 0 0 0 2) (0 0 0 0 4 0))
2: ((2 2 2 2 2 2) (2 2 2 2 2 2))
3: ((0 3 0 3 0 3) (3 0 3 0 3 0))
4: ((1 2 3 4 5 6) (6 5 4 3 2 1))
5: ((2 4 6 7 10 12) (12 10 8 6 4 2))
6: ((48 0 0 0 0 0) (0 0 0 0 0 48))
7: ((8 8 8 8 8 8) (8 8 8 8 8 8))

Done.

Choose an entry: 1

1. Provided Heuristic
2. Custom Heuristic
Select an option: 1

Starting A*...

(((0 0 0 0 0 0) (0 0 0 0 0 0)) 6 6 -6)
(((0 0 0 0 0 0) (0 0 0 0 0 1)) 5 5 -4)
(((0 0 0 0 0 1) (0 0 0 0 0 1)) 4 4 -2)
(((0 0 0 0 1 1) (0 0 0 0 0 1)) 3 3 0)
(((0 0 0 0 0 0) (0 0 0 0 4 0)) 2 2 2)
(((0 0 0 0 1 0) (0 0 0 0 4 0)) 1 1 4)
(((0 0 0 0 0 2) (0 0 0 0 4 0)) 0 0 6)
NIL
```

# Informação Gerada

## Durante a Execução:

- **Caminho de Execução**: Sequência de estados do tabuleiro.
- **Profundidade**: Número de movimentos realizados.
- **Valor Heurístico**: Estimativa da qualidade do estado atual.
- **Solução Encontrada**: Estado final do tabuleiro (se existir solução).

## Mensagens de Erro:

- Entrada inválida (quando se introduz uma opção não listada no menu).
- Nenhuma solução encontrada (quando o algoritmo falha na resolução do tabuleiro).

# Limitações

1. **Entrada e Interação do Utilizador**:
    - O programa suporta apenas entrada via texto.
    - Não há feedback visual gráfico.
2. **Casos Complexos**:
    - Os algoritmos de pesquisa podem demorar demasiado tempo ou falhar para tabuleiros grandes.
    - A heurística pode não ser ótima para determinadas configurações.
3. **Análise Estatística**:
    - O programa não gera automaticamente métricas detalhadas de desempenho, como tempo de execução ou fator de ramificação.

# Conclusão

A aplicação **Adji-Boto** resolve problemas de tabuleiro utilizando algoritmos clássicos de pesquisa. Este manual fornece todas as instruções necessárias para operar o programa, selecionar estratégias de pesquisa e interpretar os resultados.