

# Manual Técnico - Jogo de Tabuleiro com Cavalos

## Objetivo

Desenvolver um programa em Common Lisp para resolver um problema específico relacionado a um jogo de tabuleiro com cavalos. O jogo ocorre em um tabuleiro 10x10, e os cavalos devem ser movidos de acordo com regras específicas. O programa implementa algoritmos de busca para encontrar soluções para diferentes configurações do tabuleiro.

## Funcionalidades

### Leitura de Tabuleiros

O programa é capaz de ler tabuleiros a partir de um arquivo de dados. Os tabuleiros são representados por listas de listas, onde cada elemento é um número inteiro.

### Menu Principal

O programa apresenta um menu principal com as seguintes opções:

1. Escolher um tabuleiro específico.
2. Recarregar tabuleiros.
3. Sair.

### Escolha de Tabuleiro

O utilizador pode escolher um tabuleiro específico a partir de uma lista apresentada no menu. O programa exibe o tabuleiro escolhido.

### Algoritmos de Busca

O programa implementa três algoritmos de busca:

1. BFS (Breadth-First Search)
2. DFS (Depth-First Search)
3. A\* (A-star)

Cada algoritmo tem um menu específico para configuração de parâmetros.

### Estatísticas de Desempenho

O programa fornece estatísticas sobre o desempenho de cada algoritmo:

- Número de nós expandidos.
- Número de nós gerados.
- Ramificação média.
- Penetrância.

## **Persistência de Resultados**

O utilizador pode optar por salvar os resultados em um arquivo.

## **Implementação**

### **Estrutura do Código**

Ficheiro `procura.lisp`

Contém a implementação dos algoritmos de busca, definição de estruturas de dados e funções auxiliares.

Ficheiro `projeto.lisp`

Implementa a lógica da aplicação, incluindo menus, interação com o utilizador e chamadas aos algoritmos de busca.

### **Funções Principais**

`init_bfs`

Inicia o algoritmo BFS com a opção de especificar ou não a posição inicial. Retorna os resultados da busca.

`init_dfs`

Inicia o algoritmo DFS com a opção de especificar ou não a posição inicial e a profundidade máxima. Retorna os resultados da busca.

`init_a_star`

Inicia o algoritmo A\* com a opção de especificar ou não a posição inicial e a função heurística. Retorna os resultados da busca.

`result_to_string`

Converte os resultados da busca em uma representação de string amigável. Inclui informações sobre o caminho, número de nós expandidos, número de nós gerados, ramificação média e penetrância.

`app_start`

Função principal que inicia a aplicação e exibe o menu principal.

`app_console`

Menu principal da aplicação. Permite escolher um tabuleiro, recarregar tabuleiros ou sair.

`board_console`

Menu para escolher um tabuleiro específico. Permite visualizar o tabuleiro escolhido ou iniciar um algoritmo de busca.

`algorithm_console`

Menu para escolher um algoritmo de busca. Permite configurar parâmetros específicos para cada algoritmo.

`bfs_menu`, `dfs_menu`, `astar_menu`

Menus específicos para cada algoritmo de busca. Coletam parâmetros específicos do utilizador e iniciam a busca.

### Funções Auxiliares

- `load_boards`: Carrega tabuleiros a partir de um arquivo.
- `show_question`: Exibe uma pergunta ao utilizador e retorna a resposta.
- `read_file`: Lê o conteúdo de um arquivo.
- `split_file_into_lists`: Converte o conteúdo do arquivo em listas de listas.
- `print_board`: Mostra um tabuleiro na tela.
- `print_each_board`: Mostra todos os tabuleiros disponíveis.
- `get_board`: Obtém um tabuleiro específico.
- `format_line`: Formata uma linha de tabuleiro.
- `log_result`: Salva os resultados em um arquivo.
- `append-to-file`: Adiciona texto ao final de um arquivo.

### Algoritmos de Busca

#### BFS (Breadth-First Search)

Função `init_bfs`

A função `init_bfs` inicia o algoritmo BFS. Permite ao utilizador especificar ou não a posição inicial e retorna os resultados da busca.

#### Exemplo:

```
(init_bfs (state_constructor board) 2 '(7 8))
```

Neste exemplo, o BFS é iniciado a partir do estado inicial gerado pelo construtor de estado `state_constructor`. A posição inicial é especificada como (7, 8).

#### DFS (Depth-First Search)

Função `init_dfs`

A função `init_dfs` inicia o algoritmo DFS. Permite ao utilizador especificar ou não a posição inicial e a profundidade máxima. Retorna os resultados da busca.

#### Exemplo:

```
(init_dfs (state_constructor board) nil '(3 4) 10)
```

Neste exemplo, o DFS é iniciado a partir do estado inicial gerado pelo construtor de estado `state_constructor`. A posição inicial não é especificada, e a profundidade máxima é definida como 10.

### **A\* (A-star)**

Função `init_a_star`

A função `init_a_star` inicia o algoritmo A\*. Permite ao utilizador especificar ou não a posição inicial e a função heurística. Retorna os resultados da busca.

### **Exemplo:**

```
(init_a_star (state_constructor board) 1 '(1 2) 'heuristic_function)
```

Neste exemplo, o A\* é iniciado a partir do estado inicial gerado pelo construtor de estado `state_constructor`. A posição inicial é especificada como (1, 2), e a função heurística utilizada é denominada '`heuristic_function`'.