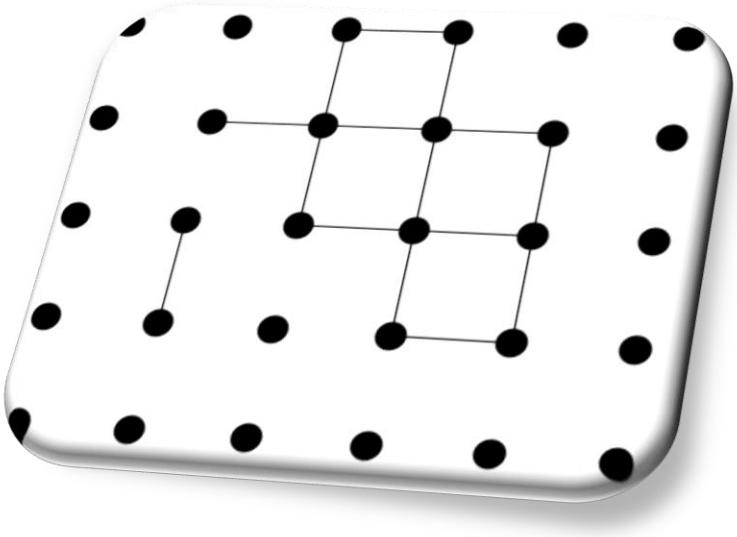
IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Manual Utilizador

Inteligência Artificial

2016/2017



Problema do Puzzle dos Pontos e das Caixas

Professores:

Joaquim Filipe
Cédric Grueau

Turma: 3º-Inf-ES2

140221017 – André Bastos
140221002 – Luís Mestre

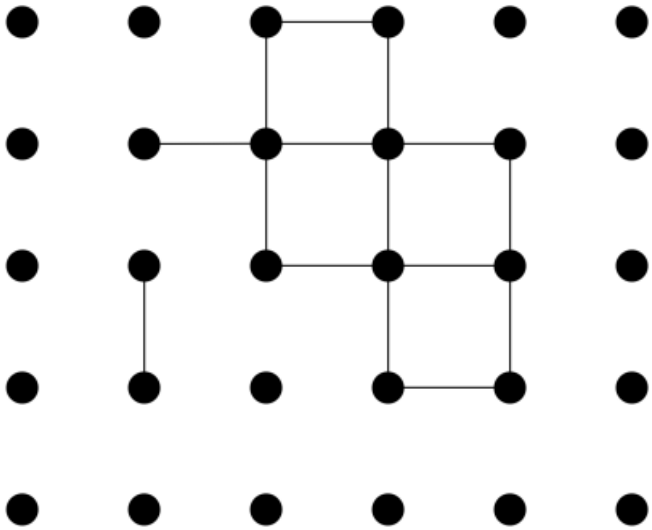
Índice

1. Introdução

Este manual irá servir como guia para o utilizador puder usar o nosso programa. Este manual destina-se mais em concreto para utilizadores que não estão familiarizados com a linguagem de programação LISP ou com o IDE LispWorks.

2. Jogo

O puzzle é constituído por um tabuleiro de **n*m** caixas. Cada caixa é delimitada por 4 pontos entre os quais é possível desenhar um arco. Quando os 4 pontos à volta de uma caixa estiverem conectados por 4 arcos, estes formam uma caixa considerada fechada. O espaço da solução é portanto constituido por n*m caixas, **(n + 1) * (m + 1)** pontos e **((n + 1) * m) + (n * (m + 1))** arcos. Na imagem abaixo é mostrado um exemplo de um puzzle com 20 caixas (n=4 e m=5), com 15 arcos conectados e 4 caixas fechadas:



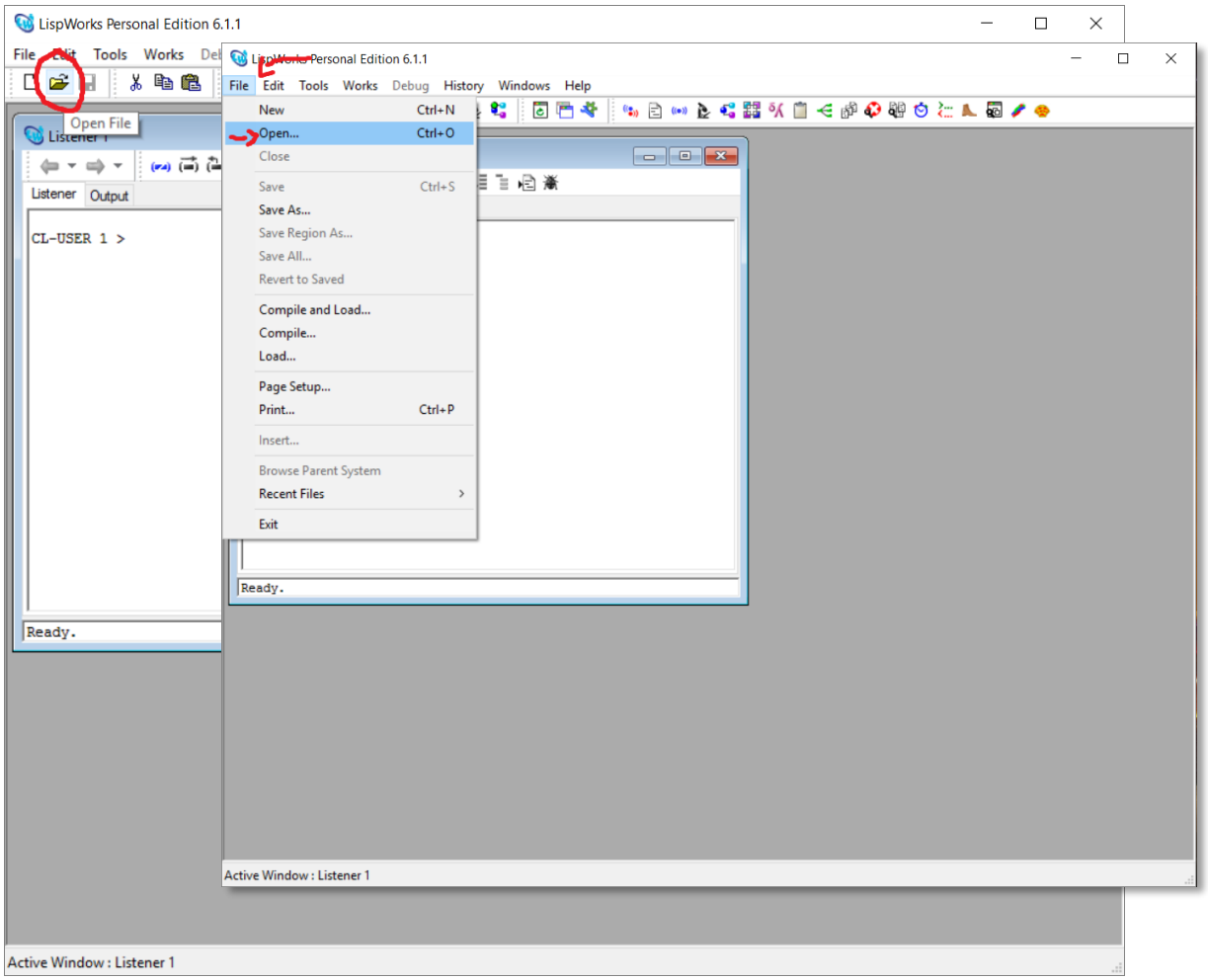
O objetivo do puzzle é fechar um determinado número de caixas estipuladas pelo utilizador a partir de um tabuleiro inicialmente configurado. Para atingir este objetivo, é possível desenhar um arco entre dois pontos adjacentes, na horizontal e na vertical. Quando o número de caixas fechadas é atingido, o puzzle está resolvido.

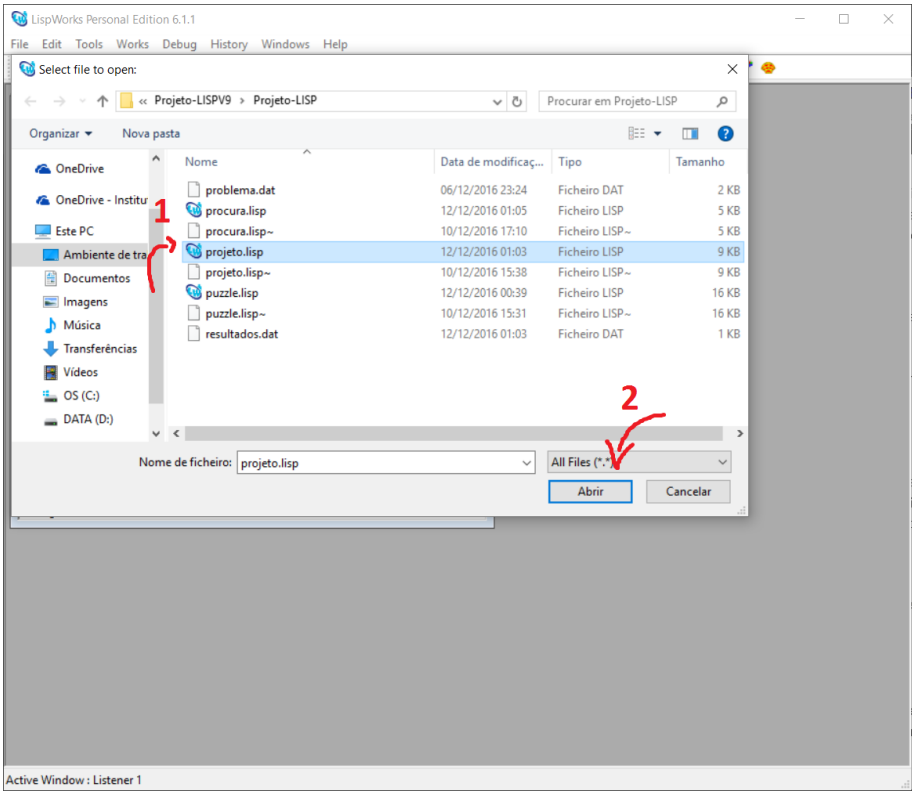
3. Funcionamento

Para poder executar o programa é necessário ter instalado no seu computador um **IDE** que consiga interpretar a linguagem de programação **LISP**. Para as instruções que irão ser dadas iremos utilizar o **LispWorks**, por isso recomendamos que seja utilizado o mesmo.

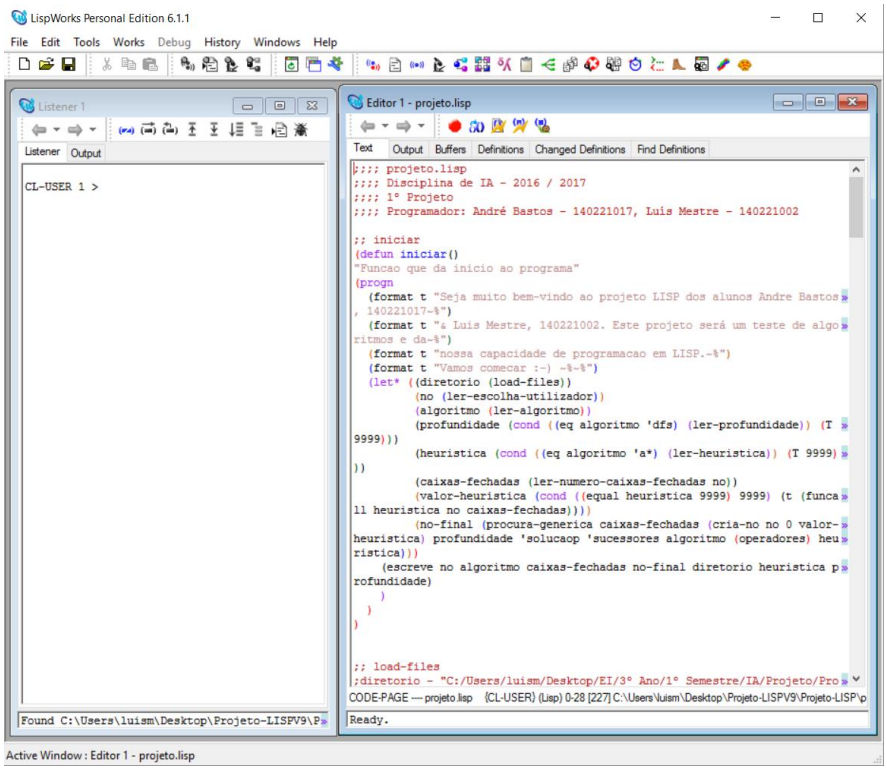
Executar

Para executar o programa é necessário abrir o ficheiro **projeto.lisp**, para isso basta no LispWorks ir a menu e carregar em **Open File**, que também pode ser acedido através do atalho Ctrl+O em Windows, e localizar o ficheiro projeto.lisp no seu computador.

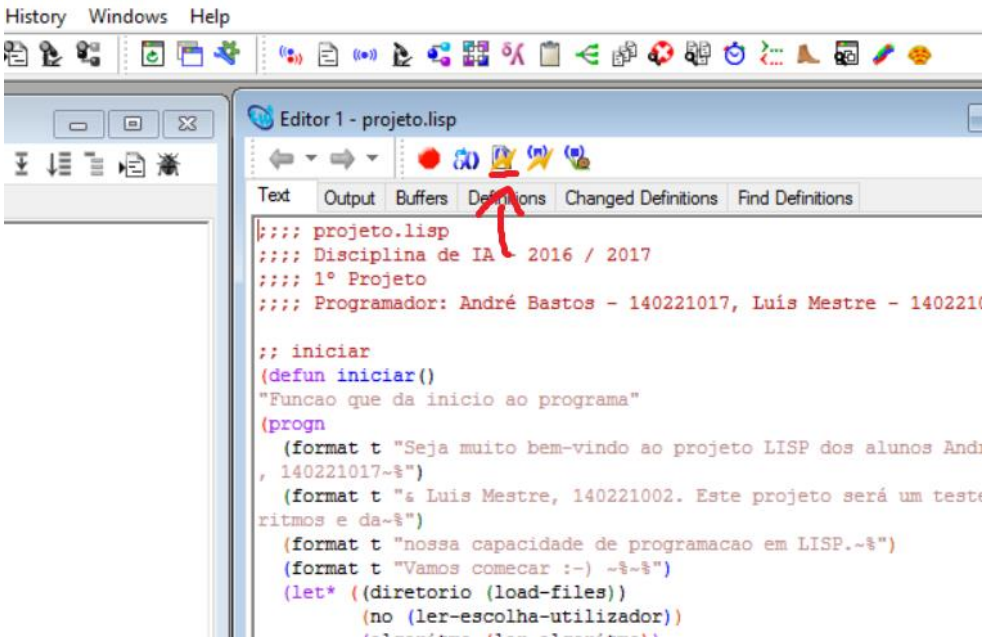




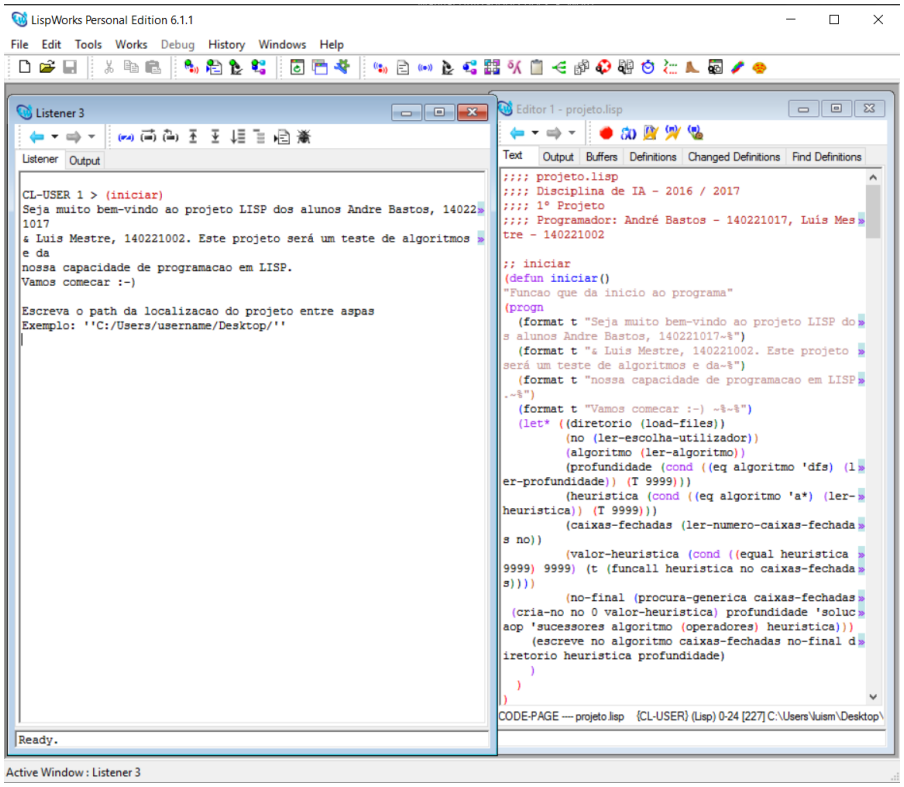
Depois de abrir o ficheiro, o seu IDE terá um aspeto semelhante ao seguinte:



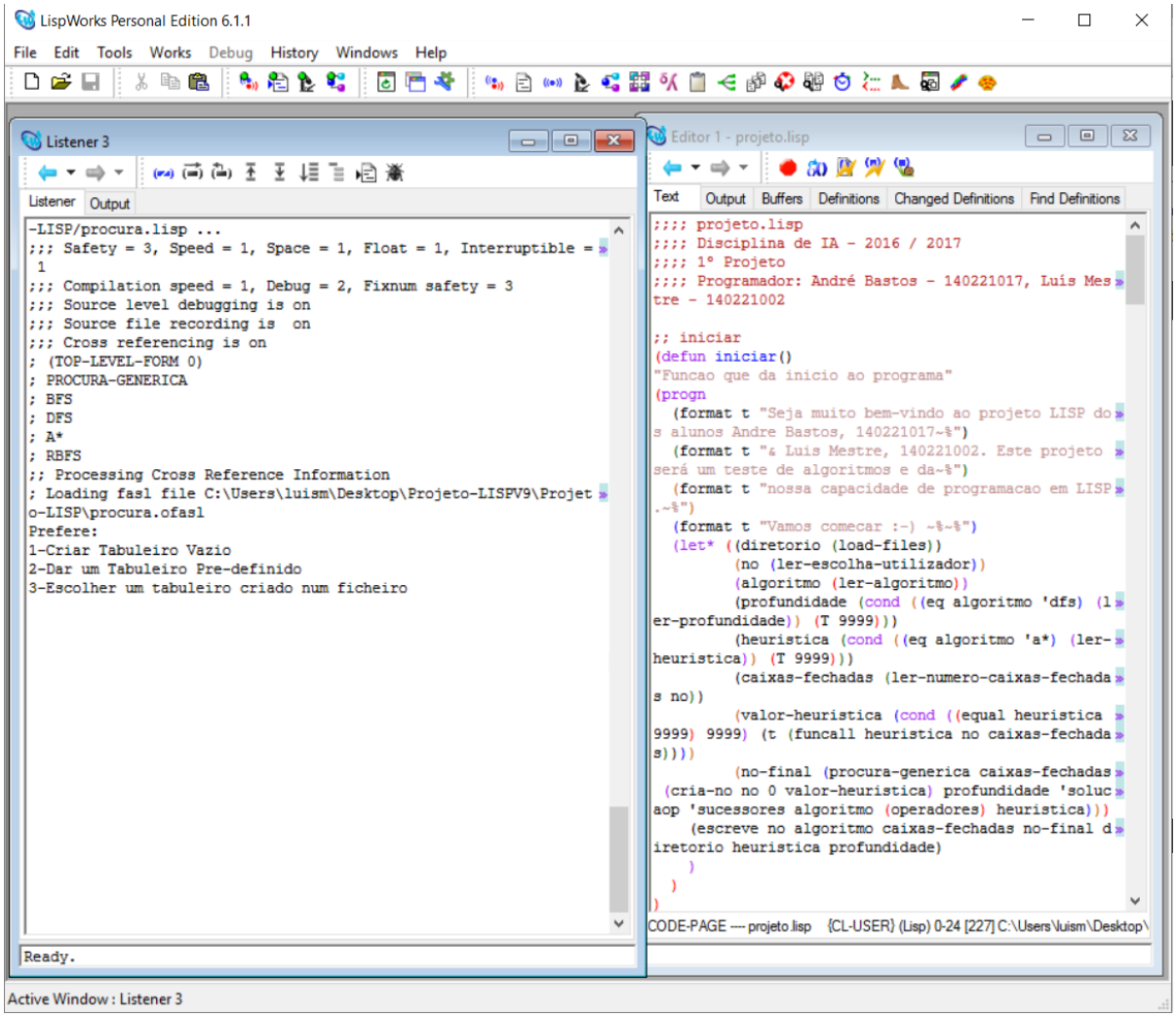
Antes de poder executar o programa pela primeira vez é importante que compile o mesmo antes de o poder executar, para isso basta carregar no botão o indicado na imagem em baixo:



Após ter compilado o ficheiro, na zona do listener irá executar o programa ao escrever (iniciar). O nosso programa começa por cumprimentar o utilizador e pede para este inserir o path do programa, isto é, o diretório de onde se encontram os ficheiros do programa (formato "C:/Users/username/Desktop/").



Depois de o utilizador inserir esse diretório press Enter, o programa faz o load dos restantes métodos, incluindo o método de procura e os algoritmos implementados nos programa. Após isto é pedido ao utilizador o que ele prefere para a criação do tabuleiro, aparecem as seguintes opções:



3.1.Menu Principal

Seja bem-vindo ao problema do puzzle dos pontos e caixas. Este será o aspeto do menu principal da aplicação, onde poderá encontrar as seguintes opções:

```
Prefere:
1-Criar Tabuleiro Vazio
2-Dar um Tabuleiro Pre-definido
3-Escolher um tabuleiro criado num ficheiro
```

1-**Criar Tabuleiro Vazio** -> o utilizador insere as dimensões de comprimento e largura respetivamente e internamente é criado um tabuleiro vazio.

2-**Dar um Tabuleiro Pre-definido** -> o utilizador insere um tabuleiro criado por ele.

3-**Escolher um tabuleiro criado num ficheiro** -> o utilizador escreve o diretório onde se localiza o ficheiro que criou com uma lista de tabuleiros. É de se destacar que este diretório tem de incluir o nome do ficheiro e a sua respetiva extensão.

Criar Tabuleiro Vazio

Caso o utilizador escolha a **opção 1)** será pedido ao utilizador para inserir as dimensões de Número de Linhas e Número de Colunas do tabuleiro. A imagem abaixo é um exemplo da criação de um tabuleiro vazio 2*2:

```
1-Criar Tabuleiro Vazio
2-Dar um Tabuleiro Pre-definido
3-Escolher um tabuleiro criado num ficheiro
1
Insere a dimensao das linhas do tabuleiro
2
Insere a dimensao das colunas do tabuleiro
2
```

Dar um Tabuleiro Pre-Definido

Caso o utilizador escolha a **opção 2)** será pedido ao utilizador para inserir um tabuleiro. Caso o utilizador não saiba a estrutura de um tabuleiro o programa apresenta um exemplo de um tabuleiro que possa inserir, como é representado na imagem abaixo:

```
Prefere:
1-Criar Tabuleiro Vazio
2-Dar um Tabuleiro Pre-definido
3-Escolher um tabuleiro criado num ficheiro
2
Insira o tabuleiro definido:
'(((nil nil nil) (nil nil t) (nil t t) (nil nil t)) ((nil nil nil) (n
il t nil) (nil nil t) (nil t t)))
```

Escolher um tabuleiro criado num ficheiro

Caso o utilizador escolha a **opção 3)** será pedido ao utilizador para inserir o diretorio de onde se localiza o ficheiro com uma lista de tabuleiros. Após inserir o diretorio terá de escolher qual o tabuleiro que pretende usar. Em conjunto com o nosso projeto disponibilizamos um ficheiro com tabuleiros, caso preferir usar este existente. Na imagem abaixo é representado o exemplo com o ficheiro referido.

```
Prefere:
1-Criar Tabuleiro Vazio
2-Dar um Tabuleiro Pre-definido
3-Escolher um tabuleiro criado num ficheiro
3
Escreva o nome do ficheiro juntamente com o seu path
Exemplo ''C:/Users/username/Desktop/problema.dat''
"C:/Users/luism/Desktop/Projeto-LISPV9/Projeto-LISP/problema.dat"
Escolha o tabuleiro que quer usar (insira um numero)
2
```

3.2. Algoritmo

Após o tabuleiro ter sido escolhido o utilizador terá que escolher o algoritmo que quer que seja executado durante a resolução do puzzle.

```
Que algoritmo quer utilizar?
1 - Procura na largura (Breadth First)
2 - Procura em profundidade (Depth First)
3 - Procura com heuristica (A*)
4 - Procura recursive best first search (RBFS)
```

1-**Procura em largura (Breathd First)** -> Algoritmo de procura em largura, encontra sempre a melhor solução mas demora muito tempo em casos de tabuleiros grandes.

2-**Procura em profundidade (Depth First)** -> Algoritmo de procura em profundidade, tenta encontrar uma solução através de uma profundidade estabelecida pelo utilizador. Este algoritmo não garante encontrar uma solução na profundidade inserida.

3-**Procura com heuristica (A*)** -> Algoritmo de procura pelo menor custo, garante encontrar a melhor solução no caminho mais curto possível. Este encontrará uma solução à mesma profundidade do Breathd First, mas esta solução poderá não ser mesma, caso exista mais do que uma solução à mesma profundidade.

4-**Procura recursive best first search (RBFS)** -> Algoritmo de procura recursiva pelo melhor caminho. Este algoritmo garante o melhor caminho para a solução. Este algoritmo tem a vantagem de não ocupar muito espaço de memória durante a sua execução, mas devido a ser uma pesquisa recursiva este tende a demorar muito tempo.

No caso de ser escolhido o Depth First, o utilizador terá de dizer qual a profundidade máxima que o algoritmo pode percorrer ao tentar encontrar a solução.

No caso de ser escolhido o A* ou o RBFS, o utilizador terá de dizer qual a heurística a ser usada na procura da solução. Terá de escolher entre 3 das disponíveis.

```
Que algoritmo quer utilizar?
1 - Procura na largura (Breadth First)
2 - Procura em profundidade (Depth First)
3 - Procura com heuristica (A*)
4 - Procura recursive best first search (RBFS)
4
Prefere qual das heuristicas?:
1-Heuristica do enunciado
2-Heuristica nova
3-Heuristica nova 2
```

Após a escolha do algoritmo o utilizador terá de seguida dizer qual o objetivo que os algoritmos terão de encontrar no puzzle.

```
Qual o numero de caixas fechadas quer?
7
```

3.3.Solução

No final, quando o algoritmo acabar a sua execução irá ser mostrado ao utilizador as estatísticas. Nestas estatísticas está incluído a apresentação do problema-inicial, o objetivo, o algoritmo escolhido, a solução encontrada, os nós gerados, os nós expandidos, a profundidade atingida, a Heurística, o tempo que demorou a execução, em segundos, o fator de ramificação, a penetrancia e o nó pai da solução (este é a representação do caminho feito até ao nó solução).

```
Problema: (((NIL NIL T NIL) (T T T T) (NIL NIL T T) (NIL NIL T T) (NIL NIL T T)) ((NIL»
 NIL T T) (NIL NIL T T) (T T T T) (T NIL T T) (NIL T T T)))
Algoritmo Escolhido: A*
Numero de Caixas Fechadas (Objetivo): 7
Heuristica Escolhida: HEURISTICA
| Linhas: ((NIL NIL T NIL) (T T T T) (NIL NIL T T) (NIL NIL T T) (NIL NIL T T)) | Colu»
nas: ((NIL NIL T T) (NIL NIL T T) (T T T T) (T T T T) (NIL T T T)) | G: 1 | H: -1 |
| Tempo de Execucao (segundos): 0 | Fator-Ramificacao: 31457265/2097152 |
| Nos-Gerados: 16 | Nos-Expandidos: 1 | Penetrancia: 1/16 |
| Pai: ((((NIL NIL T NIL) (T T T T) (NIL NIL T T) (NIL NIL T T) (NIL NIL T T)) ((NIL N»
IL T T) (NIL NIL T T) (T T T T) (T NIL T T) (NIL T T T))) 0 1 NIL)
```

4. Limitações

Eficiência dos algoritmos – devido à memória que os algoritmos ocupam, pode haver problemas em certo tabuleiros devido ao facto de estes chegarem a ocupar demasiada memória ou demorarem demasiado tempo devido ao número de iterações necessárias para conseguir chegar à solução. Estes tipos de problemas encontram-se mais com tabuleiros vazios de dimensões grandes.

Limite de memória do IDE – em alguns casos. devido ao facto do LispWorks ter um limite de memória pode haver tabuleiros onde não se chega a encontrar a solução por ultapassar o limite de memória disponível pelo IDE.