Inteligência Artificial 2023/24
Projeto: PIPEMANIA

17 de abril de 2024

1 Introdução

O projeto da unidade curricular de Inteligência Artificial (IArt) tem como objetivo desenvolver um programa em Python 3.8 que resolva uma adaptação do problema PIPEMANIA [1] utilizando técnicas de procura e resolução de problemas de Inteligência Artificial.

O jogo PIPEMANIA é um jogo de quebra-cabeças desenvolvido em 1989 pela The Assembly Line para o computador Amiga. Posteriormente foi adaptado para várias outras plataformas pela Lucasfilm Games, que lhe deu o nome de "Pipe Dream". O objetivo do jogo é construir um sistema de canalização funcional para evitar fugas de água.

2 Descrição do problema

O jogo PIPEMANIA decorre numa grelha quadrada, em que cada posição contém uma peça de tubagem. O objetivo é rodar as peças de modo a que todas fiquem conectadas e a água possa circular sem fugas.

A Figura 1a mostra um exemplo da disposição inicial de uma grelha. A Figura 1b mostra uma solução para essa mesma grelha. Podemos assumir que uma instância de PIPEMANIA tem uma solução única.

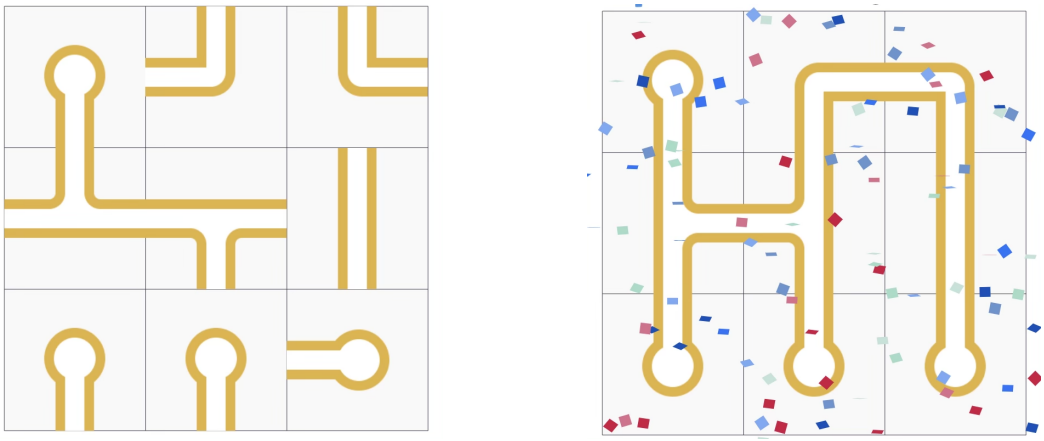
As imagens que constam no enunciado foram obtidas a partir da aplicação Pipes Puzzle Set desenvolvida por Denis Pushkarev para IOS [2] e Android [3].

3 Objetivo

O objetivo deste projeto é o desenvolvimento de um programa em Python 3.8.2 que, dada uma instância de PIPEMANIA, retorna a solução (única), i.e., todos os tubos conectados e sem fugas.

O programa deve ser desenvolvido num ficheiro `pipe.py`, que lê uma instância de PIPEMANIA a partir do *standard input* no formato descrito na secção 4.1. O programa deve resolver

[1] https://en.wikipedia.org/wiki/Pipe_Mania
[2] https://apps.apple.com/pt/app/pipes-puzzle-set/id6446434610?l=en-GB
[3] https://play.google.com/store/apps/details?id=com.dip.puzzle&pcampaignid=web_share



(a) Estado inicial (b) Objetivo/Solução

Figura 1: Exemplo de um jogo de PIPEMANIA.

o problema utilizando uma técnica à escolha e imprimir a solução para o *standard output* no formato descrito na secção 4.2.

```
python pipe.py <initial-state.txt>
```

4 Formato de input e output

Num tabuleiro de PIPEMANIA existem 4 tipos de peças:

1. Peças de fecho (F), com orientação cima (C), baixo (B), esquerda (E) e direita (D);
2. Peças de bifurcação (B), com orientação cima (C), baixo (B), esquerda (E) e direita (D);
3. Peças de volta (V), com orientação cima (C), baixo (B), esquerda (E) e direita (D);
4. Peças de ligação (L), com orientação horizontal (H) e vertical (V).

A Figura 2 contém a designação para cada uma das peças que podem existir num tabuleiro.

A título de exemplo, as peças da primeira linha da Figura 1a correspondem a FB / VC / VD, enquanto que as peças da segunda fila correpondem a BC / BB / LV.

4.1 Formato do input

As instâncias do problema PIPEMANIA seguem o seguinte formato:

```
<pipe-piece-l1c1> ... <pipe-piece-l1cN>
<pipe-piece-l2c1> ... <pipe-piece-l2cN>
...
<pipe-piece-lNc1> ... <pipe-piece-lNcN>
```

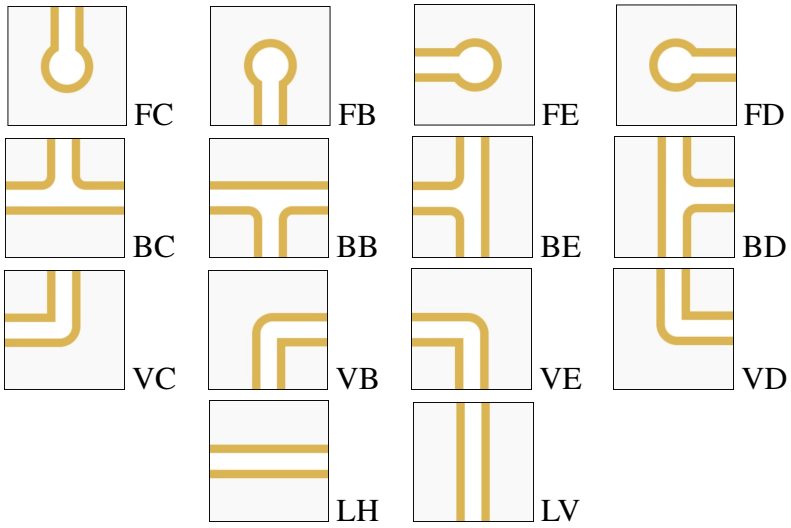


Figura 2: Designações para as peças de PIPEMANIA

Os valores possíveis para `<pipe-piece-*>` são strings de duas letras, em que a primeira tem como domínio a identificação da peça {F,B,V,L} e a segunda tem como domínio a orientação da peça {C,B,E,D,H,V}.

4.1.1 Exemplo

O ficheiro de input que descreve a instância da Figura 1a é o seguinte:

```
FB VC VD
BC BB LV
FB FB FE

FB\tVC\tVD\n
BC\tBB\tLV\n
FB\tFB\tFE\n
```

4.2 Formato do output

O output do programa deve descrever uma solução para o problema de PIPEMANIA descrito no ficheiro de input, i.e., uma grelha completamente preenchida que respeite as regras previamente enunciadas. O output deve seguir o seguinte formato:

- Uma linha por cada linha da grelha.
- Cada linha indica o conteúdo da respetiva linha da grelha.

4.2.1 Exemplo

O output que descreve a solução da Figura 1b é:

```
FB VB VE
BD BE LV
FC FC FC

FB\tVB\tVE\n
BD\tBE\tLV\n
FC\tFC\tFC\n
```

5 Implementação

Nesta secção é descrito o código que poderá ser usado no projeto e o código que deverá ser implementado no projeto.

5.1 Código a utilizar

Para a realização deste projeto devem ser utilizados os ficheiros Python, a ser disponibilizados na página da unidade curricular, que implementam os algoritmos de procura que irão ser dados ao longo da época letiva [4]. O mais importante é compreender para que servem e como usar as funcionalidades implementadas nestes ficheiros.

Estes ficheiros não devem ser alterados. Se houver necessidade de alterar definições incluídas nestes ficheiros, estas alterações devem ser feitas no ficheiro de código desenvolvido que contém a implementação do projeto.

Outras dependências não são permitidas, exceto o Python package `numpy`, que pode ser útil para representar a solução e ter acesso a operações sobre arrays.

5.1.1 Procuras

No ficheiro `search.py` estão implementadas as estruturas necessárias para correr os diferentes algoritmos de procura. Destacam-se:

- Classe `Problem`: Representação abstrata do problema de procura;
- Função `breadth_first_tree_search`: Procura em largura primeiro;
- Função `depth_first_tree_search`: Procura em profundidade primeiro;
- Função `greedy_search`: Procura gananciosa;
- Função `astar_search`: Procura A*.

[4] Este código é adaptado a partir do código disponibilizado com o livro *Artificial Intelligence: a Modern Approach* e que está disponível em https://github.com/aimacode.

5.1.2 Classe `State`

Esta classe representa os estados utilizados nos algoritmos de procura. O membro `board` armazena a configuração da grelha a que o estado corresponde.

Abaixo é apresentado o código desta classe. Podem ser feitas alterações a esta classe, como por exemplo modificações ao método `__lt__(self, other)` para suportar funções de desempate mais complexas. No entanto, estas alterações devem ser devidamente justificadas com comentários no código.

```
class PipeManiaState:
    state_id = 0

    def __init__(self, board):
        self.board = board
        self.id = PipeManiaState.state_id
        PipeManiaState.state_id += 1

    def __lt__(self, other):
        """ Este método é utilizado em caso de empate na gestão da lista
        de abertos nas procuras informadas. """
        return self.id < other.id
```

5.2 Código a implementar

5.2.1 Classe `Board`

A classe `Board` é a representação interna de uma grelha de PIPEMANIA. A implementação desta classe e respectivos métodos é livre.

Pode, a título de exemplo, incluir os métodos para determinar valores adjacentes `adjacent_vertical_values` e `adjacent_horizontal_values` que recebem dois argumentos, as coordenadas na grelha (linha, coluna), e devolvem um tuplo com duas strings que correspondem aos valores adjacentes na vertical (acima, abaixo) e na horizontal (esquerda, direita), respectivamente. Caso não existam valores adjacentes, i.e. nas extremidades da grelha, devolvem `None`.

Pode também implementar outros métodos, como por exemplo um método `get_value` que retorne o valor preenchido numa determinada posição, ou um método `print` que imprime a grelha no formato descrito na secção 4.2.

Estes métodos poderão ser utilizados para fazer testes à restante implementação da classe.

```python
class Board:
    """ Representação interna de uma grelha de PipeMania. """

    def adjacent_vertical_values(self, row: int, col: int) -> (str, str):
        """ Devolve os valores imediatamente acima e abaixo,
        respectivamente. """
        # TODO
        pass

    def adjacent_horizontal_values(self, row: int, col: int) -> (str, str):
        """ Devolve os valores imediatamente à esquerda e à direita,
        respectivamente. """
        # TODO
        pass

    # TODO: outros metodos da classe
```

5.2.2 Função `parse_instance`

A função `parse_instance` é responsável por ler uma instância do problema no formato de input apresentado (secção 4.1) e devolver um objeto do tipo `Board` que a represente. Esta função deve ler a instância a partir do *standard input* (stdin).

```python
    @staticmethod
    def parse_instance():
        """Lê a instância do problema do standard input (stdin)
        e retorna uma instância da classe Board.

        Por exemplo:
            $ python3 pipe_mania.py < input_T01

            > from sys import stdin
            > line = stdin.readline().split()
        """
        # TODO
        pass
```

5.2.3 Classe `PipeMania`

A classe `PipeMania` herda da classe `Problem` definida no ficheiro `search.py` do código a utilizar e deve implementar os métodos necessários ao seu funcionamento.

O método `actions` recebe como argumento um estado e retorna uma lista de ações que podem ser executadas a partir desse estado. O método `result` recebe como argumento um estado e uma ação, e retorna o resultado de aplicar essa ação a esse estado.

Numa primeira abordagem, muito simples e pouco eficiente, pode considerar que uma ação corresponde a uma rotação de 90° de uma certa peça da grelha. Neste caso, cada ação pode ser representada por um tuplo com 3 elementos (índice da linha, índice da coluna, rotação 90° clockwise/anti-clockwise), em que a peça do canto superior esquerdo corresponde às coordenadas (0,0). Por exemplo, `(0, 1, False)` representa a ação "rodar a peça da linha 0 e coluna 1 anti-clockwise". Note que outras modelações, eventualmente mais complexas, deverão ser mais eficientes.

Para suportar as procuras informadas, nomeadamente a procura gananciosa e a procura A*, deve desenvolver uma heurística que consiga guiar da forma mais eficiente possível estas procuras. A heurística corresponde à implementação do método `h` da classe `PipeMania`. Esta função recebe como argumento um `node`, a partir do qual se pode aceder ao estado atual em `node.state`.

De seguida é disponibilizado um protótipo da classe `PipeMania` que pode ser usado como base para a sua implementação.

```python
class PipeMania(Problem):
    def __init__(self, initial_state: Board, goal_state: Board):
        """ O construtor especifica o estado inicial. """
        # TODO
        pass

    def actions(self, state: State):
        """ Retorna uma lista de ações que podem ser executadas a
        partir do estado passado como argumento. """
        # TODO
        pass

    def result(self, state: State, action):
        """ Retorna o estado resultante de executar a 'action' sobre
        'state' passado como argumento. A ação a executar deve ser uma
        das presentes na lista obtida pela execução de
        self.actions(state). """
        # TODO
        pass

    def h(self, node: Node):
        """ Função heuristica utilizada para a procura A*. """
        # TODO
        pass
```

5.2.4 Exemplos de utilização

De seguida, são apresentados alguns exemplos da utilização do código a desenvolver, assim como o respetivo output. Estes exemplos podem ser utilizados para testar a implementação. Considere que o ficheiro `initial-state.txt` se encontra na diretoria "./boards/board-1/"e que contém a instância descrita na secção 4.1.

Exemplo 1:

```
# Ler grelha do figura 1a:
board = Board.parse_instance()

print(board.adjacent_vertical_values(0, 0))
print(board.adjacent_horizontal_values(0, 0))

print(board.adjacent_vertical_values(1, 1))
print(board.adjacent_horizontal_values(1, 1))
```

Output:

```
(None, BC)
(None, VC)
(VC, FB)
(BC, LV)
```

Exemplo 2:

```
# Ler grelha do figura 1a:
board = Board.parse_instance()

# Criar uma instância de PipeMania:
problem = PipeMania(board)

# Criar um estado com a configuração inicial:
initial_state = PipeManiaState(board)

# Mostrar valor na posição (2, 2):
print(initial_state.board.get_value(2, 2))

# Realizar ação de rodar 90° clockwise a peça (2, 2)
result_state = problem.result(initial_state, (2, 2, True))

# Mostrar valor na posição (2, 2):
print(result_state.board.get_value(2, 2))
```

Output:

```
FE
FC
```

Exemplo 3:

```
# Ler grelha do figura 1a:
board = Board.parse_instance()

# Criar uma instância de PipeMania:
problem = PipeMania(board)

# Criar um estado com a configuração inicial:
s0 = PipeManiaState(board)

# Aplicar as ações que resolvem a instância
s1 = problem.result(s0, (0, 1, True))
s2 = problem.result(s1, (0, 1, True))
s3 = problem.result(s2, (0, 2, True))
s4 = problem.result(s3, (0, 2, True))
s5 = problem.result(s4, (1, 0, True))
s6 = problem.result(s5, (1, 1, True))
s7 = problem.result(s6, (2, 0, False)) # anti-clockwise (exemplo de uso)
s8 = problem.result(s7, (2, 0, False)) # anti-clockwise (exemplo de uso)
s9 = problem.result(s8, (2, 1, True))
s10 = problem.result(s9, (2, 1, True))
s11 = problem.result(s10, (2, 2, True))

# Verificar se foi atingida a solução
print("Is goal?", problem.goal_test(s5))
print("Is goal?", problem.goal_test(s11))
print("Solution:\n", s11.board.print(), sep="")
```

Output:

```
Is goal? False
Is goal? True
Solution:
FB VB VE
BD BE LV
FC FC FC
```

Exemplo 4:

```
# Ler grelha do figura 1a:
board = Board.parse_instance()

# Criar uma instância de PipeMania:
problem = PipeMania(board)

# Obter o nó solução usando a procura em profundidade:
goal_node = depth_first_tree_search(problem)

# Verificar se foi atingida a solução
print("Is goal?", problem.goal_test(goal_node.state))
print("Solution:\n", goal_node.state.board.print(), sep="")
```

Output:

```
Is goal? True
Solution:
FB VB VE
BD BE LV
FC FC FC
```

O valor de retorno das funções de procura é um objeto do tipo `Node`. Do nó de retorno podem ser retiradas as diversas informaçãoes, por exemplo, estado final (`goal_node.state`), a ação que levou ao estado final `goal_node.action`, e o nó precedente `goal_node.parent`.

6 Avaliação

A nota do projecto será baseada nos seguintes critérios:

- Execução correcta (75% - 15 val.). Estes valores correspondem a testes realizados via submissão no Mooshak.
- Relatório (25% - 5 val.). O relatório será realizado em formato vídeo.

7 Condições de realização e prazos

- O projecto deve ser realizado em grupos de 2 alunos.
- Publicação do enunciado: até ao dia 19 de Abril
- Inscrições de grupos no Fénix: até às 17:00 de dia 24 de Abril
- Entrega do projeto (.py) no Mooshak: 24 de Maio, até às 17:00

- Entrega do relatório no Microsoft Teams: 24 de Maio, até às 23:59

As inscrições dos grupos para o projeto serão feitas através do Fénix; este passo é essencial para posterior acesso ao Mooshak. O código do projeto e relatório têm de ser entregues obrigatoriamente por via electrónica. O código do projeto será entregue através do sistema Mooshak e o relatório através do Microsoft Teams.

7.1 Planeamento

Sugere-se que cada grupo planeie atempadamente a realização do projeto, por exemplo tendo em conta as seguintes metas:

- **Até ao dia 26 de Abril**: Leitura de input / instância de exemplo.
- **Até ao dia 3 de Maio**: Modelação do problema e definição do que é uma ação no contexto do projeto.
- **Até ao dia 10 de Maio**: Implementação das funções de geração de ações e resultados de aplicar uma ação a um estado.
- **Até ao dia 17 de Maio**: Experimentar diferentes procuras e obter soluções corretas para instâncias iniciais (validar formato de output).

7.2 Mooshak

A avaliação da execução do código do projecto será feita automaticamente através do sistema Mooshak [5]. Após o prazo de inscrição no Fénix e quando notificado pelo corpo docente siga as seguintes instruções para registar e submeter no Mooskak:

- **A partir do dia 3 de Maio**, as credênciais de acesso ao Mooshak poderão ser obtidas no seguinte URL utilizando o número de grupo:
  `http://acm.tecnico.ulisboa.pt/~mooshak/cgi-bin/ia2223p4getpass`.
  A senha ser-lhe-á enviada para o email que tem configurado no Fénix. A senha pode não chegar de imediato; aguarde.
- Após ter recebido a sua senha por email, deve efetuar o login no sistema através da página:
  `http://acp.tecnico.ulisboa.pt/~mooshak/`.
  Preencha os campos com a informação fornecida no email.
- Deverá ser submetido o ficheiro *pipe_mania.py* contendo o código do seu projecto. O ficheiro de código deve conter em comentário, nas primeiras linhas, o número e o nome dos alunos.
- Utilize o botão "Choose file", selecione o ficheiro com extensão .py contendo todo o código do seu projeto. O seu ficheiro .py deve conter a implementação das funções

[5] A versão de Python utilizada nos testes automáticos é a 3.8.2.

pedidas no enunciado. De seguida clique no botão “Submit” para efetuar a submissão. Aguarde para que o sistema processe a sua submissão!

- Quando a submissão tiver sido processada, poderá visualizar na tabela o resultado correspondente.

Submeta o seu projeto atempadamente, dado que as restrições seguintes podem não lhe permitir fazê-lo no último momento:

- Até ao prazo de entrega poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a **última entrega efectuada**.
- Os testes considerados para efeitos de avaliação podem incluir ou não os exemplos disponibilizados, além de um conjunto de testes adicionais.
- O tempo de execução de cada teste está limitado, bem como a memória utilizada.
- Existe um intervalo mínimo entre submissões de **15 minutos**.
- Só são permitidas 10 submissões em simultâneo no sistema, pelo que uma submissão poderá ser recusada se este limite for excedido. Nesse caso tente mais tarde.

7.3 Relatório em formato de vídeo

Deve produzir um relatório em formato de vídeo com a **duração máxima de 3 minutos**. No vídeo devem aparecer todos os alunos que fazem parte do grupo.

O vídeo deve ser estruturado, claro e perceptível, e deve fazer uso de material de apoio ilustrativo que não tem de ser sofisticado (por exemplo: slides, imagens, esquemas, demos). Seguem-se algumas sugestões para o conteúdo do vídeo que podem ser usadas se parecerem adequadas:

- Breve descrição do problema (regras e objetivo).
- Descrição da ideia geral para abordar o problema.
- Identificação do que é uma ação no contexto do problema.
- Descrição do que é o resultado de aplicar uma ação num estado.
- Caso exista algum tipo de pre-processamento ou métodos de inferência, estes devem ser mencionados.
- Pequena avaliação experimental comparando diferentes procuras com as instâncias disponibilizadas.
- Identificação da procura final selecionada, quais as suas características (heurística, por exemplo), e motivo da escolha.

Grupos de alunos que não tenham os meios básicos necessários para a realização do vídeo (i.e. computador com zoom, microfone e câmara) deverão contactar o corpo docente.

8 Cópias

Projectos iguais, ou muito semelhantes, originarão a reprovação na disciplina e, eventualmente, o levantamento de um processo disciplinar. Para o efeito será usada a ferramenta Moss da Universidade de Stanford [6].

Os programas entregues serão também testados em relação a soluções existentes na web. As analogias encontradas com os programas da web serão tratadas como cópias.

[6] https://theory.stanford.edu/~aiken/moss/