Atividade Prática de Inteligência Artificial de 21/03/2022

Mundo do Pacman

Prof. Luan Garcia

Introdução

Como atividade prática, implementaremos algumas estratégias de Inteligência Artificial em um jogo do tipo Pacman.

Este projeto foi desenvolvido na universidade de Berkeley para disciplina de Inteligência Artificial (http://ai.berkeley.edu). O projeto consiste em uma implementação completa da interface gráfica do jogo e diversas classes e funções para lidar com o funcionamento dos agentes do jogo na linguagem Python. Este projeto é interessante porque permite o aluno focar apenas na implementação da IA do jogo ao mesmo tempo em que conta com um feedback visual.

O projeto está disponível no portal Ulife, dentro da Aula 3, em um arquivo zipado com o nome de search.zip. Dentro deste zip vocês encontrarão diversos arquivos Python que compõe o jogo.

Vocês modificarão apenas duas classes:

search.py	Arquivo onde ficarão seus algoritmos de busca
searchAgents.py	Arquivo onde ficarão seus agentes baseados em busca

Para entender melhor como o jogo funciona, você também deve olhar os seguintes arquivos (sem modificar!)

pacman.py	O arquivo main responsável por executar o jogo. Este arquivo descreve os estados do Pacman no tipo GameState, que será utilizado para desenvolver os seus algoritmos.
game.py	Onde a lógica por trás do mundo Pacman funciona. Aqui existem vários tipos utilizados no jogo, como AgentState, Agent, Direction e Grid.
util.py	Arquivo com diversas implementações úteis que podem ser aproveitadas, como estruturas de fila, pilha, fila de prioridade, etc.
commands.txt	Arquivo com exemplos de comandos para iniciar o jogo com diferentes configurações.

Leia os comentários dos arquivos com atenção, pois você precisa entender a lógica por trás das chamadas do jogo para poder implementar seus algoritmos.

O restante dos arquivos apenas dá suporte aos gráficos do jogo e outras facilidades e podem ser ignorados.

Executando o jogo

Para executar o jogo pela primeira vez, dentro de um terminal que esteja na pasta do projeto, utilize o seguinte comando:

```
python pacman.py
```

Este comando executa um jogo em que o agente é você mesmo!

O agente mais simples no arquivo *searchAgents.py* é o agente **GoWestAgent**, que sempre se move para a esquerda. Para executá-lo, digite o seguinte comando no terminal:

```
python pacman.py --layout testMaze --pacman GoWestAgent
```

No arquivo SearchAgent.py, você irá encontrar um SearchAgent completamente implementado, que planeja um caminho pelo mundo do Pacman e o executa passo a passo. Os algoritmos de busca para formular este planejamento ***não estão implementados***. Sua tarefa é implementar estes algoritmos. Primeiro, podemos testar o programa para ver se tudo está funcionando corretamente. Utilize o comando abaixo no terminal:

```
python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch
```

Este comando diz para o jogo executar com o layout "tinyMaze", utilizando um SearchAgent e utilizando uma função de busca "tinyMazeSarch". Esta função está implementada no arquivo ***search.py*.**

Sua tarefa será desenvolver algoritmos de busca para que o agente possa encontrar a única comida disponível no labirinto.

Você deverá fazer o upload do seu código completo no Ulife na tarefa existente na aula 3.

Dica: Utilize o algoritmo de busca em grafo presente nos slides, com uma estrutura para armazenar os nodos já visitados. Isto evitará que seu algoritmo fique em loop, sem encontrar uma solução.

Dica 2: Lembre que a diferença entre os algoritmos de busca é basicamente na forma de como os estados são armazenados. O algoritmo de busca em profundidade utiliza uma estrutura do tipo Pilha (Stack / Last in First Out / LIFO), que já possui uma implementação no arquivo **util.py**

Dica 3: Encontrar a solução não é suficiente, você deve também armazenar o caminho para chegar nesta solução.

Busca em profundidade em Python

```
def busca-em-profundidade(problema):

      fronteira = pilha()
```

```
estadoInicial = problema.estadoInicial()
fronteira.push(estadoInicial)
estadosExplorados = set() #não armazena repetidos

while not fronteira.isEmpty():
      nodoFolha = fronteira.pop()
      if(problema.estadoObjetivo(nodoFolha):
            return SOLUCAO
      else:
            if(nodoFolha not in estadosExplorados):
                  estadosExplorados.add(nodoFolha)
                  nodosFronteira =
                      problema.estadosFronteira(nodoFolha)
                  for nodo in nodosFronteira:
                        fronteira.push(nodo)

return FALHA
```

Atividades

1) Implementar o algoritmo de busca em profundidade.

Para testar se o seu código está funcionando, você pode executar o seguinte comando:

```
python pacman.py -l tinyMaze -p SearchAgent
```

Se o algoritmo tiver sido implementado corretamente, também deverá funcionar para o layout de labirinto “mediumMaze”.

2) Implementar o algoritmo de busca em largura

Para testar se o seu código está funcionando, você pode executar o seguinte comando:

```
python pacman.py -l tinyMaze -p SearchAgent -a f=bfs
```

Se o algoritmo tiver sido implementado corretamente, também deverá funcionar para o layout de labirinto “mediumMaze”.