ESPM TECH 1^o semestre de 2020

Roteiro para o laboratório de Algoritmos de Busca em Espaço de Estados

Inteligência Artificial Aplicada Prof. Fabrício J. Barth

1 Infra-estrutura necessária para a execução dos exercícios

Para a execução dos exercícios propostos neste documento você terá que ter em sua máquina:

- **Python 3**: de preferência, instale a última versão disponível para o sistema operacional da sua máquina. Para instalar o Python em sua máquina você precisa acessar o site https://www.python.org/downloads/ e baixar a última versão do Python. Independente do sistema operacional da sua máquina, o processo de instalação é via uma interface gráfica estilo *next button, next, next, ...*
- **Um Ambiente Integrado de Desenvolvimento (IDE)**: sugiro fazer uso **Visual Studio Code**. Para instalar o Visual Studio Code na sua máquina acesse o site https://code.visualstudio.com/, faça o download do mesmo e instale na sua máquina (também através de um processo next, next, next button).

Opcionalmente, você poderá ter em sua máquina:

- **Git**: os códigos utilizados neste laboratório estão todos no GitHub. Uma das formas para acessar o código e utilizá-lo na sua máquina é através de comandos do próprio Git, como por exemplo, *git clone*. No entanto, esta ferramenta é *opcional*. Você poderá baixar o código através da interface web do GitHub.

Você terá que copiar o código do projeto **https://github.com/fbarth/ai-espm** para a sua máquina. Esta cópia poderá acontecer de duas formas:

- através do comando **git clone https://github.com/fbarth/ai-espm.git**, ou;
- clicando no botão de **Download ZIP** como apresentado na imagem 1.

Ao fazer a cópia do projeto para a sua máquina você deverá ter um diretório chamado **ml-espm** com a seguinte estrutura:

- README.md: arquivo README com algumas informações sobre o projeto;
- docs: diretório com alguns arquivos do tipo PDF, incluindo os slides utilizados em sala de aula;
- games: diretório com implementações que serão utilizadas quando discutirmos o assunto busca competitiva;
- reinLearn: diretório com implementações que serão utilizadas quando discutirmos o assunto aprendizagem por reforço, e;
- search: diretório com vários arquivos *.py que serão utilizados nas implementações sobre algoritmos de busca em espaço de estados.

Além disso, ao abrir em uma IDE o projeto, você deverá ver uma estrutura similar ao apresentado na figura 2.

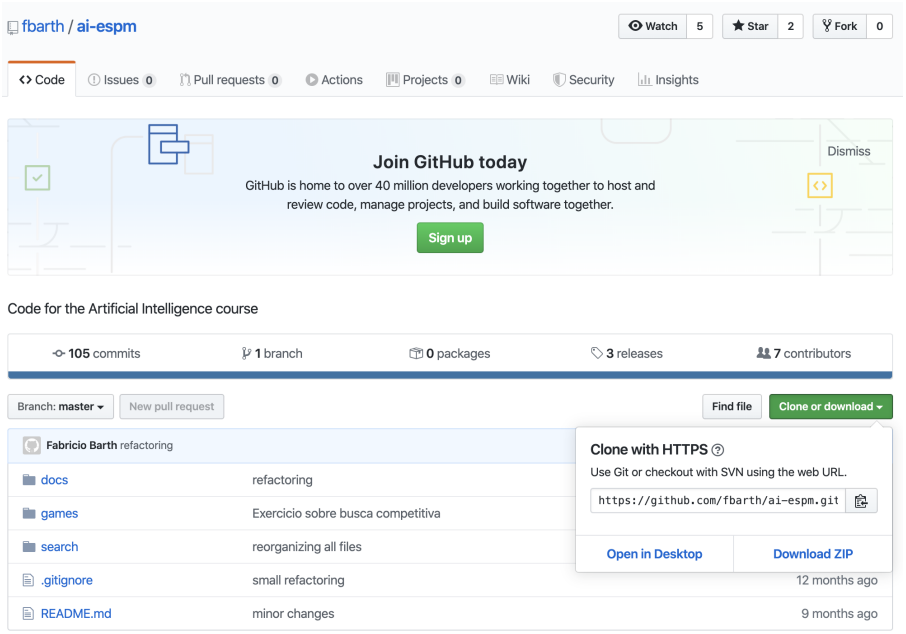

Figura 1: Página principal do projeto da disciplina no GitHub

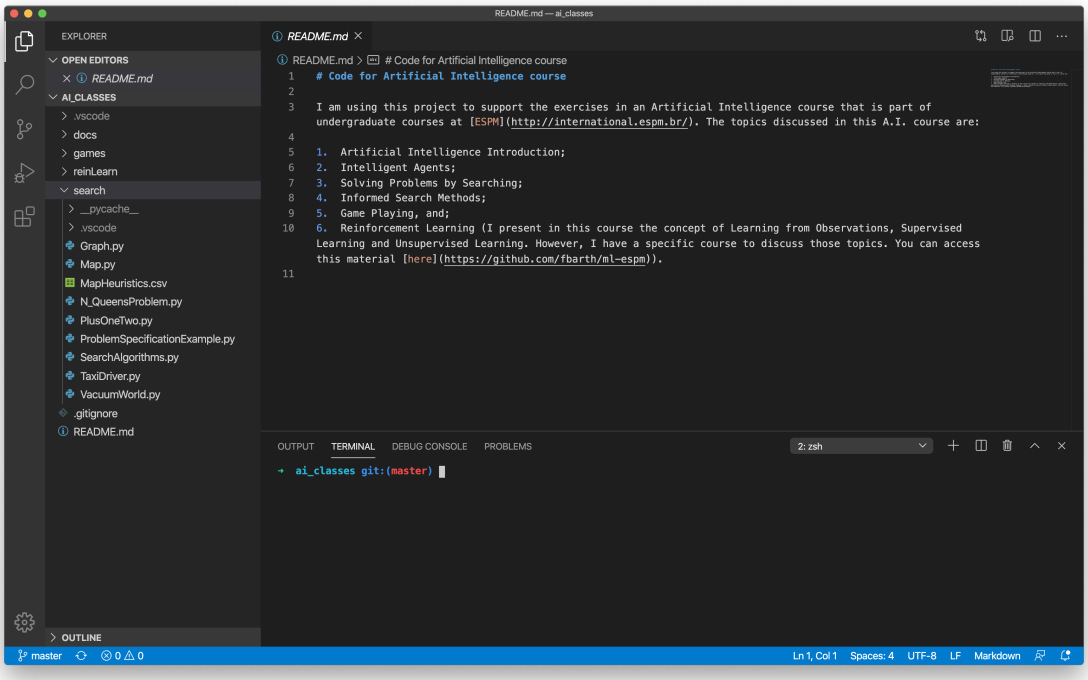

Figura 2: Página principal do projeto da disciplina no GitHub

2 Registro da atividade

Nas próximas páginas serão apresentadas diversas questões. Faça o roteiro descrito utilizando um documento a parte onde você deverá colocar as respostas para as perguntas. Ao final do exercício este documento deverá ser submetido no canvas na atividade informada no mesmo.

3 Exercício: Aspirador de Pó

Considere o exemplo do aspirador de pó. Uma possível solução está implementada no arquivo **search/VacuumWorld.py**. Considerando esta implementação, execute as seguintes ações:

- acesse o diretório **search** e execute o comando **python3 VacuumWorld.py**. (*Q1*) Qual foi a resposta fornecida pelo programa? O que esta resposta representa?
- Acesse o código **VacuumWorld.py**. (*Q2*) Quais são os métodos definidos na classe VacuumWorld? Descreve o que você acredita que cada método realiza.
- Vá até a função **main()** do arquivo **VacuumWorld.py**. (*Q3*) Esta função faz uso de algum algoritmo de busca? Quais?
- Altere o estado inicial e o estado objetivo descritos no arquivo **VacuumWorld.py**, execute novamente o comando **python3 VacuumWorld.py** e veja o que acontece.

3.1 Espaço com 3 quartos

Vamos implementar um Aspirador de Pó com 3 quartos? Três quartos lado-a-lado (esquerda, centro e direita)?

Para esta atividade já temos um código pré-pronto: **VacuumWorldQuestion.py**. Neste código falta completar o código de alguns métodos da classe (figura 3).

```
class VacuumWorldQuestion(State):

    def __init__(self, vacuumPosition, isLeftRoomClean, isCenterRoomClean, isRightRoomClean, op):
        #TODO

    def sucessors(self):
        sucessors = []
        #TODO
        return sucessors

    def is_goal(self):
        #TODO
        return False

    def description(self):
        return "Problema do aspirador de pó, contendo três (3) salas"

    def cost(self):
        return 1

    def print(self):
        return str(self.operator)
```

Figura 3: Esqueleto para a solução do VacuumWorld com três quartos

- Complete os métodos da classe e execute os dois algoritmos que estão na função main(): Busca em Largura e Busca em Profundidade. (*Q4*) Liste as soluções encontradas por ambos os algoritmos.

- O arquivo **SearchAlgorithms.py** implementa diveros algoritmos de busca, inclusive o BPI. Altere o código do arquivo **VacuumWorldQuestion.py** para executar também o algoritmo de busca em profundidade iterativo (BPI). (*Q5*) Apresente o resultado encontrado pelo algoritmo BPI para este problema.

4 Exercício: soma +1 e soma +2

Considere o problema onde o estado inicial é o número 1, o estado meta é o número 10. Existem duas operações de geração de sucessores: adicionar 1 ao número do estado atual; adicionar 2 ao número do estado atual. Utilizando o código disponibilizado no arquivo **PlusOneTwo.py**, faça as seguintes atividades:

- (*Q6*) Qual a solução apresentada pelos algoritmos de busca em largura, busca em profundidade e busca em profundidade interativa? Quais foram ótimos? Qual foi mais rápido?
- (*Q7*) Mude a meta para 1000 e repita as avaliações acima. Quais foram os resultados? Qual a justificativa para tal resultado?