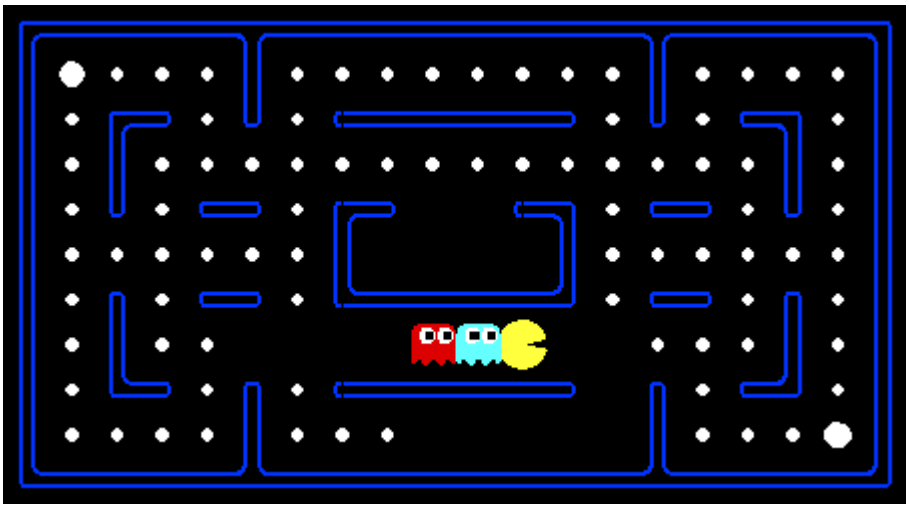
Pacman



توضیح کلی

هدف این پروژه پیاده‌سازی یک عامل هوشمند است که بتواند در محیط بازی Pacman به فعالیت پرداخته و بیش‌ترین امتیاز را کسب کند.

محیط

در این فاز از پروژه از محیط Pacman که توسط دانشگاه UC Berkeley تهیه شده است استفاده می‌کنیم. در این محیط عامل شما در یک هزارتو می‌گردد و تعداد زیادی نقطه‌ی کوچک و چند نقطه‌ی درشت را می‌خورد. در این بازی هدف این است که نقطه‌ها را بخورید و در عین حال از برخورد به اشباح پرهیز کنید. با خوردن نقطه‌های درشت، شرایط به نفع شما تغییر می‌کند: برای مدتی کوتاه می‌توانید اشباح را هم بخورید و امتیاز کسب کنید.

```
multiagent.zip
├── multiAgents.py
├── pacman.py
├── game.py
└── util.py
```

جهت اجرای بازی در ابتدا، ترمینال را در پوشه محیط باز کرده و دستور زیر را اجرا کنید. جهت حرکت عامل می‌توانید از کلیدهای جهت‌دار کیبورد یا دکمه‌های زیر استفاده کنید.

کنش	کلید
چپ	a
راست	d
بالا	w
پایین	s
ایست	q

```
python pacman.py -k 1
```

کنش‌های ممکن

عامل می‌تواند کنش‌های بالا (`North`)، راست (`East`)، پایین (`South`)، چپ (`West`) و ایست (`Stop`) را درصورتی که حرکت آن مجاز باشد، انتخاب نماید. کنش‌های عامل قطعی هستند.

اتمام بازی

بازی در حالات زیر پایان می‌یابد:

- در صورتی که روح عامل را بخورد!
- درصورتی که تمامی نقاط داخل محیط خورده شوند.

پیاده‌سازی

هر مرحله از بازی توسط یک `GameState` تعریف می‌شود. با فراخوانی توابعی که برای این کلاس تعریف شده است می‌توانید وضعیت بازی را به‌دست آورید.

- `gameState.getLegalActions(agentIndex)` : با فراخوانی این تابع، لیستی از حرکات مجاز عامل در این وضعیت بازگردانده می‌شود. این لیست زیرمجموعه‌ای از موارد زیر می‌باشد:

```
North - South - East - West - Stop
```

- `gameState.generateSuccessor(agentIndex, action)` : با فراخوانی این تابع، یک `GameState` جدید بازگردانده می‌شود که نشان‌دهنده اعمال `action` بر روی عامل `agentIndex` می‌باشد.
- کد شما باید درخت بازی را تا عمق مشخص شده گسترش دهد. امتیاز برگ های درخت `Minimax` خود را با `self.evaluationFunction` ارائه شده امتیاز دهید. می‌توانید تابعی برای امتیازدهی به دلخواه باتوجه به شرایط بازی تعریف کنید.
- توجه کنید که زمان پاسخگویی الگوریتم شما به ازای هر کنش نباید بیشتر از `self.time_limit` ثانیه باشد.
- دانلود کد جهت پیاده‌سازی

multiAgents.py

```python
class AIAgent(MultiAgentSearchAgent):
    def getAction(self, gameState: GameState):
        """
        Here are some method calls that might be useful when implementing
        gameState.getLegalActions(agentIndex):
        Returns a list of legal actions for an agent
        agentIndex=0 means Pacman, ghosts are >= 1
        gameState.generateSuccessor(agentIndex, action):
        Returns the successor game state after an agent takes an action
        gameState.getNumAgents():
        Returns the total number of agents in the game
        gameState.isWin():
        Returns whether or not the game state is a winning state
        gameState.isLose():
        Returns whether or not the game state is a losing state
        """
```

```
        "*** YOUR CODE HERE ***"
```

جهت تست الگوریتم خود، ترمینال را در پوشه محیط باز کرده و دستور زیر را اجرا کنید.

```
python pacman.py -p AIAgent -k 1 -a depth=4
```

مستندات Pacman

ویژگی‌های مستندات:

۱. نحوه کار الگوریتم را برای حل این مسئله توضیح دهید.

۲. چنان چه از منبعی به غیر از اسلایدهای درس و کتاب مرجع استفاده کرده‌اید حتما نام آن منبع را ذکر کنید.

۳. چنانچه از کتابخانه‌ای استفاده کرده‌اید (مطابق با شرایط ذکر شده در اطلاعیه) نام آن را ذکر نمایید.

۴. فایل آپلود شده به فرمت PDF ، دارای مشخصات دانشجویان نظیر نام و نام خانوادگی و شماره دانشجویی و شماره تیم باشد.