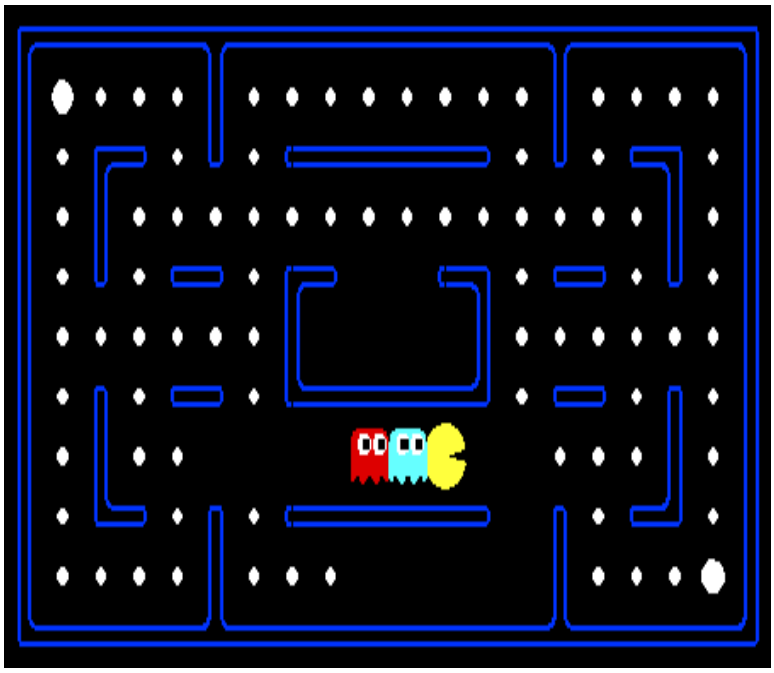
بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری اول هوش مصنوعی

پاییز ۱۳۹۶



معرفی

در این پروژه شما عاملان را برای پکمن می سازید که شامل ارواح (دشمنان) است و باید هر دو جست و جوی مینیماکس و اکسپکتی ماکس را پیاده سازی کنید.

اساس این کد با پروژه قبلی تفاوت زیادی نمیکند اما پیشنهاد میکنیم به جای ادغام با پروژه قبلی یک پروژه جدید بسازید.

همانند پروژه قبلی در این پروژه نیز از ابزار اتوگریدر برای امتیاز دهی استفاده میکنیم:

```
python autograder.py
```

این ابزار برای هر سوال به شیوه ی زیر عمل میکنید.مثلا برای نمره دهی سوال دو کد زیر را در ترمینال وارد کنید.

```
python autograder.py -q q2
```

هم چنین برای انجام یک تست خاص از دستور زیر استفاده میکنیم

```
python autograder.py -t test_cases/q2/0-small-tree
```

برای دانلود کدهای این پروژه بر روی اینجا کلیک کنید.

فایل هایی که باید ویرایش کنید:	
multiAgents.py	محلی که عاملان شما باید در ان قرار بگیرند.
فایل هایی که باید بخوانید ولی نیاز به ویرایش ندارند:	
pacman.py	فایل اصلی که بازی را اجرا میکند.هم چنین گیم استیت پکمن در اینجا تعریف شده است.
game.py	منطق بازی پکمن در اینجا قرار دارد.در این فایل نوع های مختلف مانند عاملان,حالت های عامل,جهت ها و گرید ها تعریف شده است.
util.py	دیتا استراکچرهای مفید برای پیاده سازی جست و جست ها در اینجا قرار دارد.
این فایل ها خواندشان الزامی ندارد	
graphicsDisplay.py	گرافیک بازی
graphicsUtils.py	فایل کمکی برای گرافیک پکمن
textDisplay.py	گرافیک ASCII برای پکمن
ghostAgents.py	عاملان کنترل روح ها
keyboardAgents.py	اینترفیس کیبورد برای کنترل پکمن
layout.py	کد هایی برای خواندن فایل های لایوت و ذخیره ی محتوای آن ها
autograder.py	سیستم نمره دهی خودکار پروژه
testParser.py	تجزیه آزمون نمره دهی خودکار و فایل های راه حل
testClasses.py	آزمون نمره دهی عمومی کلاس ها
test_cases/	مسیری که شامل موارد آزمون برای هر سوال است
multiagentTestClasses.py	سیستم نمره دهی مخصوص پروژه دوم

فایل هایی که نیاز به تغییرات دارند:

شما نیاز دارید که multiAgents.py را ویرایش کنید.شما باید فقط توکن ها را وارد کنید.لطفا از دستکاری بقیه فایل ها خودداری کنید.

ارزیابی : کد شما برای ارزیابی صحت فنی توسط سیستم نمره دهی خودکار بررسی خواهد شد .لطفا نام هیچ کدام از توابع و یا کلاس های داخل کد ها را تغییر ندهید.

ابزار چیت دیکودر: چیت دیکودری پروژه یک افزونه لاجیکال است که به سادگی هرگونه تقلب و کپی کد ها را تشخیص میدهد.این افزونه فقط مختص کپی بودن صرف کد ها نیست بلکه از لحاظ منطقی نیز مشابهت کدها را بررسی میکند و فریب دادن ان کار ساده ای نیست! پس کپی نکنید!

پکمن چند عامله

ابتدا پکمن را با دستور زیر اجرا کنید:

```
python pacman.py
```

سپس با `ReflexAgent` را با دستور زیر اجرا کنید.

```
python pacman.py -p ReflexAgent
```

توجه داشته باشید که حتی در طرح های ساده نیز بسیار ضعیف عمل میکند:

```
python pacman.py -p ReflexAgent -l testClassic
```

سوال ۱:

`ReflexAgent` را در `multiAgents.py` را بهبود ببخشید.برای بهبود باید عامل شما محل غذاها و ارواح را در نظر بگیرد.عامل شما باید به اسانی و قابل اعتماد در طرح `testClassic` اشکار باشد:

```
python pacman.py -p ReflexAgent -l testClassic
```

عامل خود را در حالت مدیوم کلاسیک با یک یا دو روح امتخان کنید

```
python pacman.py --frameTime 0 -p ReflexAgent -k 1
python pacman.py --frameTime 0 -p ReflexAgent -k 2
```

هزینه عامل شما چه قدر است؟ در این حالت معمولا با دو روح میمرید مگر اینکه تابع ارزیابی شما بسیار عالی بوده باشد.

نکته:به جای در نظر گرفتن تمام ویژگی ها فقط ویژگی های مهم مثل فاصله تا غذا را بررسی کنید.

برای ارزیابی سوال اول:

```
python autograder.py -q q1
```

و در حالت بدون گرافیک:

```
python autograder.py -q q1 --no-graphics
```

سوال ۲:

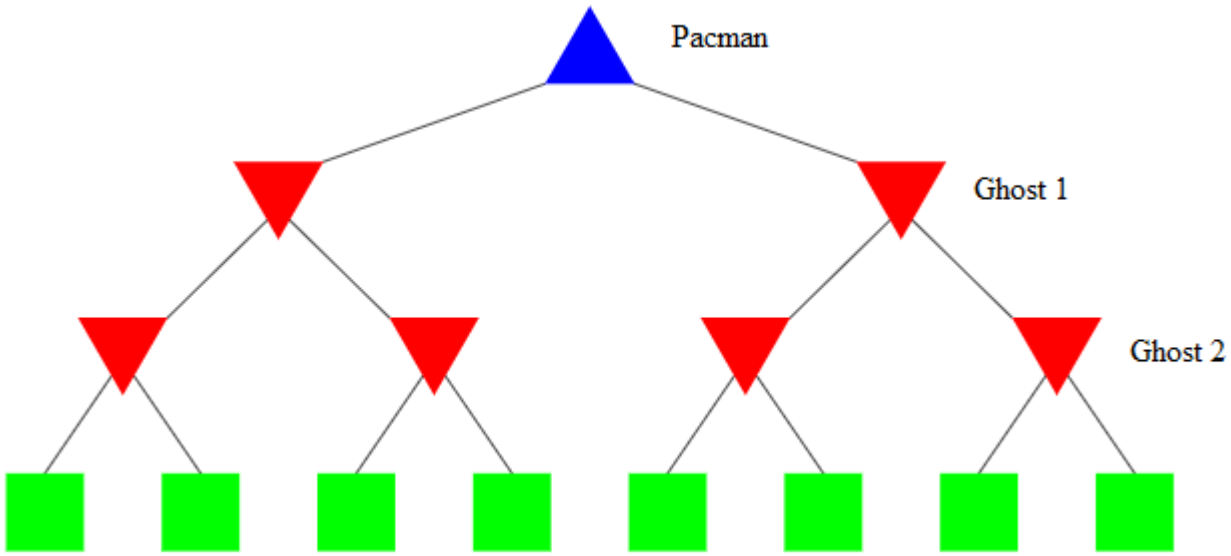
مینیماکس

در این قسمت شما در `multiAgents.py` و با کلاس `MinimaxAgent` یک عامل دشمن را ایجاد میکنید.الگوریتم شما باید با هر تعداد روح به خوبی باید کار بکند.درخت مینیمکس شما باید به ازای هر لایه مکس یک لایه چندگانه مینیموم داشته باشد.

هم چنین کد شما باید قابلیت این را داشته باشد که تا یک عمق دلخواه عملیات بسط را انجام دهد.نمره دهی با `self.evaluationFunction` انجام می شود که پیش فرضش `scoreEvaluationFunction` است.`MinimaxAgent` عامل `MultiAgentSearchAgent` را گسترش میدهد که اجازه دسترسی به `self.depth` و `self.evaluationFunction` را میدهد.

اطمینان حاصل کنید که کد مینیماکس شما به این دو متغیر اشاره میکند چون این متغیرها در پاسخ به گزینه های خط فرمان آپلود می شوند.

مهم: عمق در plies محاسبه می شود.در یک جست و جو یک لایه ای همه ی عامل ها(پکمن و همه ی روح ها) هر کدام یک اکشن را انجام میدهند.برای روشن سازی بیشتر در مثال زیر درختی را نشان می دهد که نود های عمق اول باید به وسیله ی پکمن گسترش یابند(ماکسیمایزر) و سپس دو روح(مینیمایزر) انجام گیرد.در لایه اخر به جای اینکه به پکمن اجازه نوبتش را انجام دهد.مینیماکس لایه ۱ تابع ارزیابی فراخوانی می شود(مربع های سبز).



برای نمره دهی از دو تابع زیر استفاده میکنیم:

```
python autograder.py -q q2
python autograder.py -q q2 --no-graphics
```

نکات:

ممکن است در طول بازی پکمن شکست بخورد که اشکالی ندارد!

تابع ارزیابی تست Pacman در این بخش قبلا نوشته شده است (self.evaluation Function). شما نباید این تابع را تغییر دهید.

برای چک کردن پیروزی یا شکست از GameState.isLose , GameState.isWin میتوانید استفاده کنید.هم چنین تابع ارزیابی را نباید فقط در زمانی که به ماکسیسمم عمق لایه رسیدیم استفاده کنید.بلکه باید در حالات پیروزی و شکست نیز این بررسی انجام شود.

ارزش مینیماکس در حالت ابتدایی در minimaxClassic layout ۹ و ۸ و ۷ و ۴۹۲- برای عمق های ۱ و ۲ و ۳ و ۴ است.

- `python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4`

در حالت بالا حدود ۶۶۵ بار از ۱۰۰۰ بار پکمن به پیروزی می رسد.

پکمن همیشه عامل ۰ است و عامل ها در پی افزایش ایندکس عامل هستند.
در صفحات بزرگ مانند openClassic یا mediumClassic ممکن است پکمن شکست نخورد اما به پیروزی هم نرسد!در این سوال مواجه شدن با این حالت اشکالی ندارد.برای حل این مشکل میتوانید تابع ارزیابی را بهبود ببخشید.
همه ی حالت ها در مینیماکس باید در GameStates باشند یا اینکه به getAction پاس شوند و یا در GameState.generateSuccessor ساخته شوند.

زمانی که پکمن مطمئن شد شکستش حتمی هست تلاش میکند در کوتاه ترین زمان بمیرد(خودکشی کند!)چون ادامه ی بازی برای او امتیاز منفی دارد.کار اشتباهی است که این کار را با یک روح رندوم انجام دهیم.با این حال مینیماکس بدترین حالت را در نظر میگیرد:

```
python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3
```

سوال ۳:

هرس آلفا – بتا :

در این سوال برای افزایش بازده ما از درخت هرس الفا بتا در AlphaBetaAgent استفاده میکنیم.در این بازی چالش اصلی برای حالت چند عامله ی مینیمایز است.
سرعت در این حالت ممکن از بیشتر از مینیماکس باشد.مثلا در عمق سوم الفا بتا سرعت ممکن از عمق دوم مینی ماکس بیشتر باشد.

```
python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic
```

مقدار مینیماکس AlphaBetaAgent وMinimaxAgent باید برابر باشند.اگرچه ممکن است کارهایی که انجام میدهند متفاوت باشد.

حالت پایه مقدار مینیماکس minimaxClassic layout برابر است با ۹ و ۸ و ۷و۴۹۲- که در عمق های ۱ و ۲ و ۳ و ۴ هستند.

هم چنین در برای متچ کردن تابع ارزیابی ما نباید روی مقادیر مساوی عملیات هرس انجام دهید!

شبه کد زیر برای پیاده سازی الگوریتم کمکتان خواهد کرد.

Alpha-Beta Implementation

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v > β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v < α return v
        β = min(β, v)
    return v
```

برای تست و دیباگ کد خود از دستور زیر استفاده کنید

```
python autograder.py -q q3
```

و برای حالت بدون گرافیک:

```
python autograder.py -q q3 --no-graphics
```

در درخت هرس الفا بتا ,پکمن برخی از تست ها را انجام نمیدهد.اما اشکالی ندارد!

سوال ۴:

اکسپکتیماکس

مینیماکس و هرس الفا بتا هر دو بسیار خوب هستند اما هر دو ان ها گمان میکنند که دشمنان شما همیشه بهینه ترین حالت را انتخاب میکنند.در این سوال شما `ExpectimaxAgent` را پیاده سازی میکنید که از مدل سازی احتمالاتی برای رفتارهای عامل ها برای رسیدن به انتخاب `suboptimal` استفاده میکند.

برای ارزیابی در حالت `small` از دستور زیر استفاده میکنیم

```
python autograder.py -q q4
```

دیباگ کردن در حالت اسمال و تست های قابل مدیریت به شما در دیباگ اسان تر کمک بسیار زیادی خواهد کرد.توجه کنید که هنگام میانگین گیری از نوع float استفاده کنید تا مقادیر کسری از بین نرود.

زمانی که شما از درخت کوچکی استفاده میکنید میتوانید موفقیت پکمن را به سادگی مشاهده کنید.استفاده رندوم از روح ها نمیتواند نتیجه ی بهینه ای را برای شما فراهم بیاورد.ExpectimaxAgent دیگر نمیخواهد که مینیمم را روی همه ی روح ها پیدا کند بلکه انتظاراتی که دارد را در نظرمیگیرد.برای سادگی کد فرض کنید شما فقط در برابر یک دشمن عمل خواهید کرد که در انتخابات خود به طور تصادفی از getLegalActionS انتخاب می شود.

برای دیدن اینکه ExpectimaxAgent چه طور رفتار میکند دستور زیر را انجام دهید

```
python pacman.py -p ExpectimaxAgent -l minimaxClassic -a depth=3
```

اکنون باید در روابط نزدیک با ارواح، یک رویکرد سرگردان تر را مشاهده کنید.زمانی که پکمن بفمد که ممکن است شکست بخورد به سمت غذاهای بیشتر میرود.در این زمان سناریوهای زیر ممکن است اتفاق بیفتد

```
python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3
-q -n 10
```

```
python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3
-q -n 10
```

شما باید متوجه بشید که ExpectimaxAgent شما حدود نیمی از زمان ها را می برد در حالی که `AlphaBetaAgent` همیشه شکست میخورد.

همانند سوال های قبل پکمن ممکن است در بعضی از تست ها شکست بخورد که اشکالی ندارد

تمرینها باید از طریق ایمیل تحویل داده شود . موضوع ایمیل ارسالی حتما به فرم مثال زیر باشد در غیر اینصورت ۲۰ درصد نمره تمرین کسر خواهد شد:

<StudentNo>-<FisrtName>-<LastName>-HW<No>-Programming-Kharazmi-AI-Fall96

تمرینها به آدرس ai.kharazmi.fall96@gmail.com ارسال شود.

مهلت تحویل: تا ۱۴ ابان

در صورت تاخیر تا ۲ روز ۲۰ درصد , تا یک هفته ۵۰ درصد و بیشتر از از یک هفته ۱۰۰ درصد نمره را از دست میدهید.