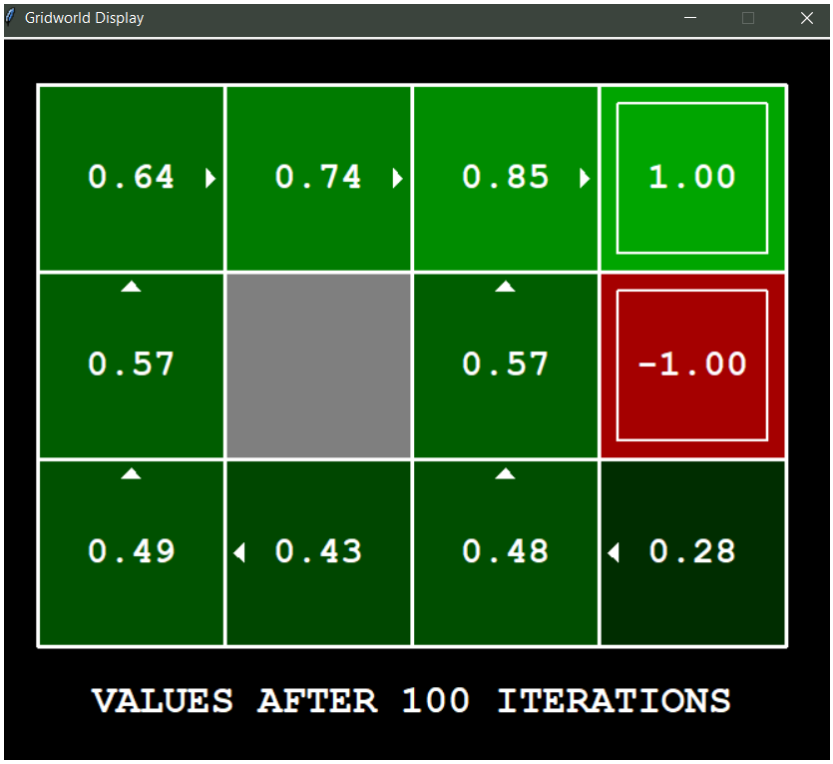
به نام خدا

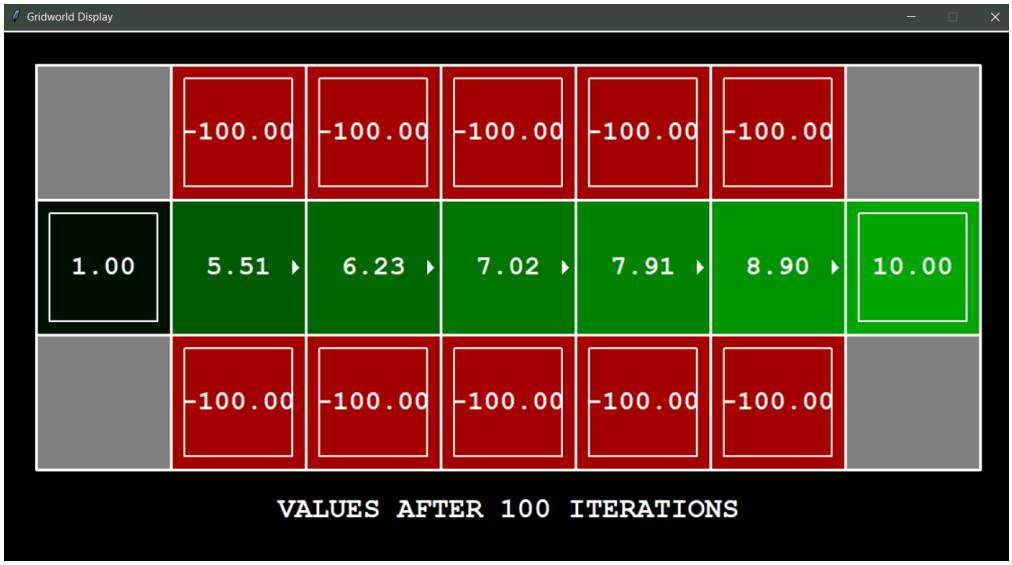
گزارش فاز سوم پروژه مبانی هوش مصنوعی

سوال اول)



در این بخش در هر تکرار ارزش کل استیت ها را آپدیت می کنیم به این صورت که می بینیم ماکسیمم qvalue آنها چقدر است که درون یک tmp_vaules نگهداری می شود و در انتهای آن دور به counter اصلی یعنی self.values منتقل می شود.

سوال دوم)



مقدار نویز را به ۰.۰۰۱ تغییر دادم زیرا هرچقدر نویز کمتر باشد عامل می تواند با احتمال بیشتری در جهت سیاست انتخاب شده گام بردارد.

سوال سوم)

الف)


```
E:\uni\Term6\intelligence\project\3\AI_P3>python gridworld.py -a value -i 100 -g DiscountGrid --discount 0.9 --noise 0.1
 --livingReward -4
```

با منفی کردن هزینه زندگی عامل تمایل دارد زودتر به هدف برسد حتی اگر بمیرد.

ب)


```
E:\uni\Term6\intelligence\project\3\AI_P3>python gridworld.py -a value -i 100 -g DiscountGrid --discount 0.35 --noise 0.
3 --livingReward 0.01
```

با دادن هزینه زندگی بسیار کم (در حد صفر) و تخفیف کم، کاری می کنیم که بخواهد بیشتر زنده بماند اما به نزدیکترین جایزه برسد.

ج)

```
E:\uni\Term6\intelligence\project\3\AI_P3>python gridworld.py -a value -i 100 -g DiscountGrid --discount 0.9 --noise 0.1
 --livingReward -2
```

با دادن هزینه زندگی خیلی منفی و تخفیف زیاد، کاری می کنیم که بخواهد زودتر به پر جایزه ترین هدف برسد.

د)


```
E:\uni\Term6\intelligence\project\3\AI_P3>python gridworld.py -a value -i 100 -g DiscountGrid --discount 0.9 --noise 0.1
 --livingReward 0
```

هزینه زندگی را از بین می بریم که بخواهد با خیال راحت به طرف دورترین و پر امتیاز ترین هدف برود.

ه)

```
E:\uni\Term6\intelligence\project\3\AI_P3>python gridworld.py -a value -i 100 -g DiscountGrid --discount 0.9 --noise 0.3
 --livingReward 6
```

در ابتدا از آنجا که تخفیف زیاد است (نکته مثبت) عامل به طرف جایزه زیاد اما از مسیر کم خطر می رود سپس چون هزینه زندگی زیاد است نه می تواند به هدف برسد و نه جایی برای خودکشی باقی است در نتیجه اجرای بازی ناتمام می ماند

سوال چهارم)

خروجی با استفاده از Asynchronous value iteration:

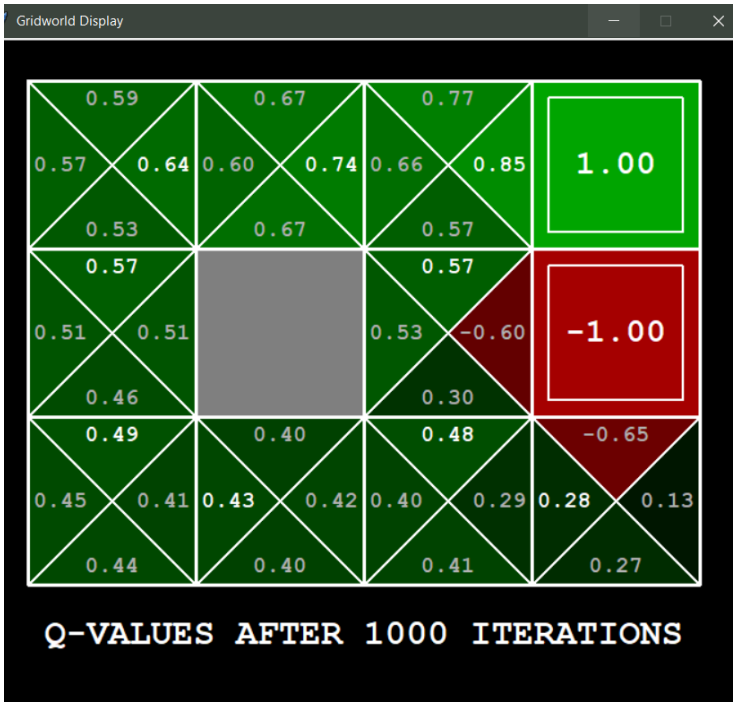
خروجی با استفاده از value iteration:

همانطور که دیدیم بعد از ۱۰۰ اجرا نتیجه هر دو از نظر مقدار ارزش ها و سیاست ها یکی شده است. اما باید توجه داشته باشیم این الگوریتم برای تعداد اجراهای زیاد بهینه است و برای وقتی که می خواهیم با تعداد کمی اجرا ارزش و سیاست را ببینیم مناسب نیست برای مثال:

Asynchronous value iteration with 50 iteration:

value iteration with 50 iteration:

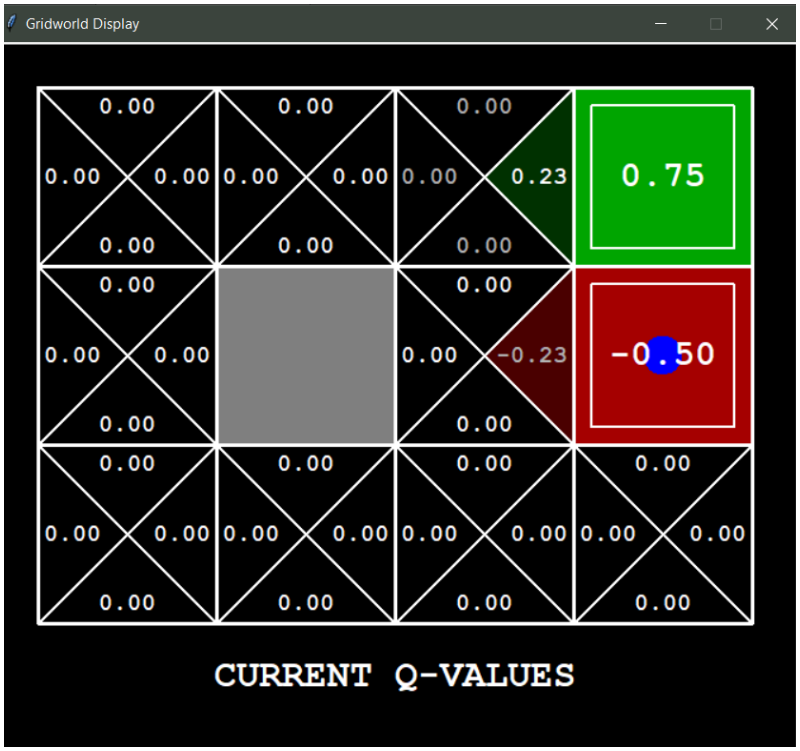
سوال پنجم)



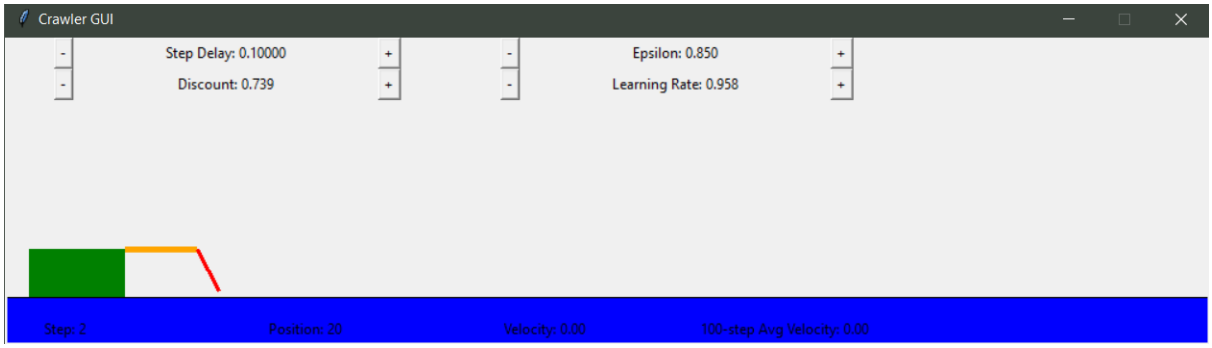
تمام سودوکد ها را انجام داده و نتیجه به صورت بالاست

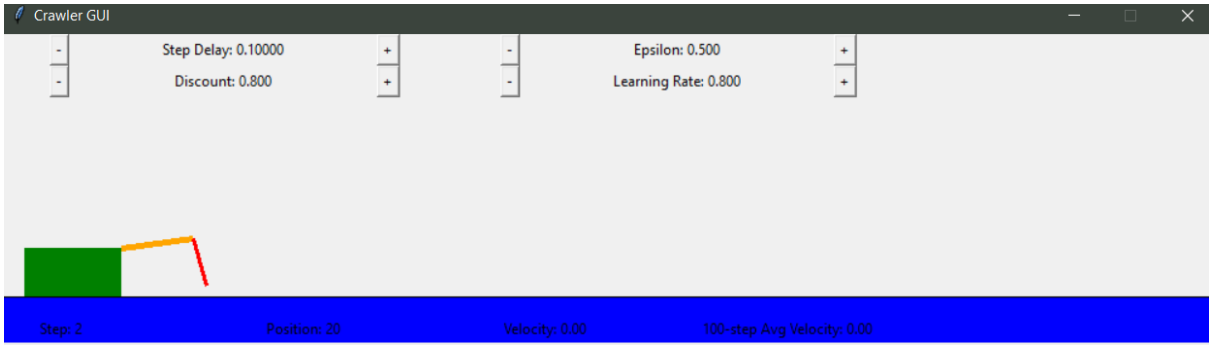
سوال ششم)

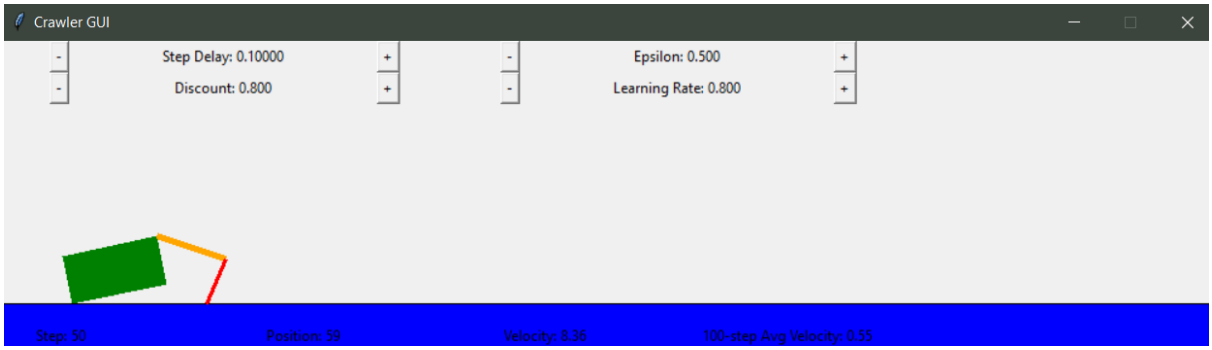
با انجام چند مرحله از آپدیت مقادیر qvalue:

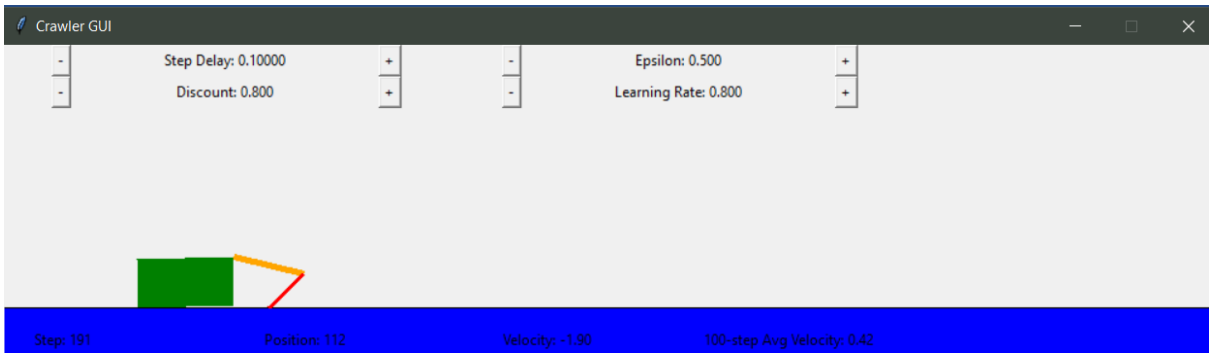


تصویر ربات خزنده:



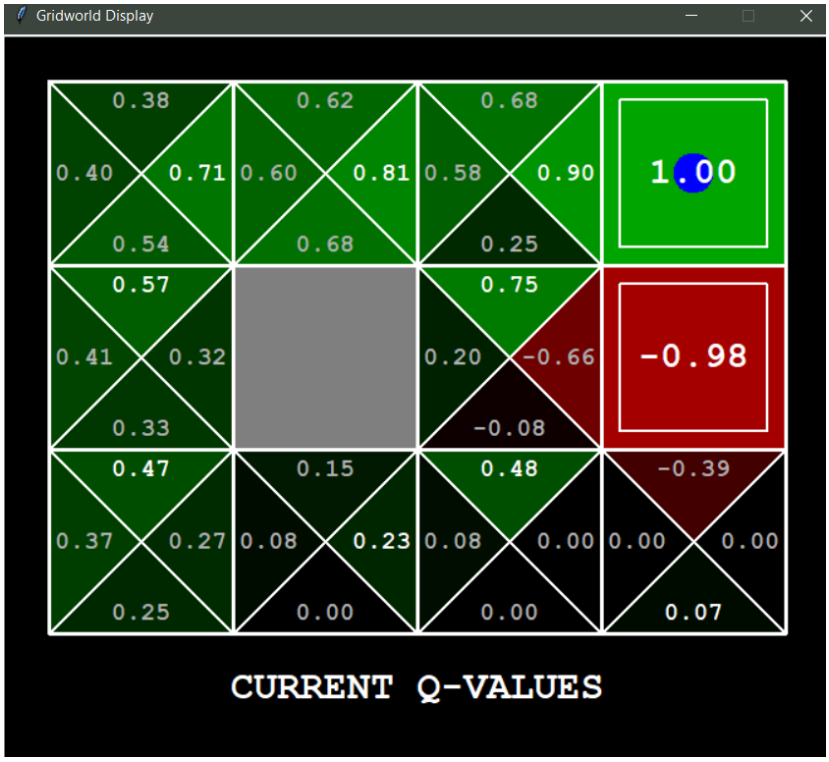






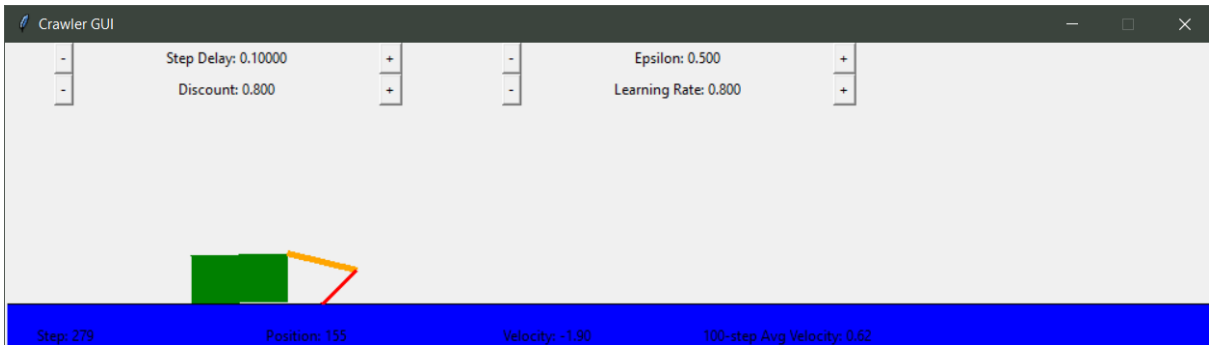
سوال هفتم)

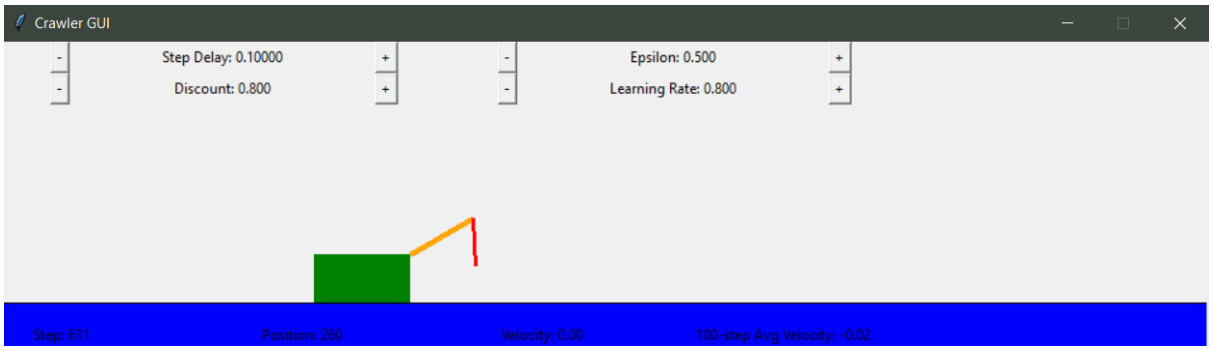
در شکل های زیر چند آپدیت از انتخاب اکشن ها به واسطه اپسیلون حریصانه را می بینیم (اپسیلون = ۰.۳)


Gridworld Display
0.38
0.40
0.71
0.54
0.62
0.60
0.81
0.68
0.68
0.58
0.90
0.25
1.00
0.57
0.41
0.32
0.33
0.75
0.20
-0.66
-0.08
-0.98
0.47
0.37
0.27
0.25
0.15
0.08
0.23
0.00
0.48
0.08
0.00
0.00
-0.39
0.00
0.00
0.07
CURRENT Q-VALUES


Gridworld Display
0.60
0.59
0.70
0.60
0.61
0.63
0.81
0.64
0.66
0.65
0.90
0.44
1.00
0.60
0.52
0.55
0.50
0.76
0.38
-0.57
0.36
-1.00
0.54
0.47
0.38
0.39
0.30
0.24
0.52
0.29
0.47
0.32
0.24
0.15
-0.65
0.34
0.13
0.06
Q-VALUES AFTER 100 EPISODES


Gridworld Display
0.60
0.59
0.70
0.60
0.61
0.63
0.81
0.64
0.66
0.65
0.90
0.44
1.00
0.60
0.52
0.55
0.50
0.76
0.38
-0.57
0.36
-1.00
0.54
0.47
0.38
0.39
0.30
0.24
0.52
0.29
0.47
0.32
0.24
0.15
-0.65
0.34
0.13
0.06
Q-VALUES AFTER 100 EPISODES


Crawler GUI
Step Delay: 0.10000
Discount: 0.800
Epsilon: 0.500
Learning Rate: 0.800
Step: 279
Position: 155
Velocity: -1.90
100-step Avg Velocity: 0.62


Crawler GUI
Step Delay: 0.10000
Discount: 0.800
Epsilon: 0.500
Learning Rate: 0.800
Step: 611
Position: 260
Velocity: 0.00
100-step Avg Velocity: -0.02

سوال هشتم)

همانطور که در شکل های بالا مشاهده می کنیم با استفاده از یک Q-learner کاملا تصادفی و با ۵۰ اپیزود نمی توان سیاست بهینه ای پیدا کرد.

حال وقتی اپسیلون را صفر قرار می دهیم از آنجا که مقدار qvalue برابر با بهترین مقداری که از گذشته به دست آورده ایم می باشد، از وقتی ارزش ۱ را پیدا میکند، همیشه در خانه ۱ باقی می ماند:

اپسیلون = ۰.۹ و آلفا = ۰.۹ می تواند انتخاب های مناسبی جهت تضمین سیاست بهینه باشند به این دلیل که به انتخاب های تصادفی حق دخالت می دهند اما انتخاب های بر مبنای تصمیم گیری سنگینی بیشتری دارند

سوال نهم)

در این بخش با ترکیب یادگیری تقویتی و پکمن توانستیم کدی که در قسمت Q-learning به درستی اجرا کرده بودیم را روی پکمن پیاده کنیم. یعنی پکمن با گذر زمان درمورد اکشن ها و امتیازات یاد می گرفت و در نهایت هم توانست به برد کامل برسد. (از آنجا که کد به درستی کار کرد و امتیاز کامل هم گرفت تغییرات خاصی صورت نگرفت)

برای مثال با اجرا دستور با آرگومان های پیش فرض:

```
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 502
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Average Score: 499.7
Scores:        499.0, 503.0, 495.0, 503.0, 499.0, 502.0, 495.0, 503.0, 503.0, 495.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
```

اما اگر اپسیلون = ۰.۱، آلفا = ۰.۳ و گاما = ۰.۷ بدهیم، دو باخت خواهیم داشت که نشان دهنده این است که پکمن نتوانسته بازی خوبی بکند

```
E:\uni\Term6\intelligence\project\3\AI_P3>python pacman.py -p PacmanQAgent -x 2000 -n 2010 -l smallGrid -a epsilon=0.1,a
lpha=0.3,gamma=0.7
```

```
----------------------------------------
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman died! Score: -502
Pacman died! Score: -504
Average Score: 298.2
Scores:        495.0, 495.0, 495.0, 503.0, 503.0, 499.0, 503.0, 495.0, -502.0, -504.0
Win Rate:      8/10 (0.80)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Loss, Loss

E:\uni\Term6\intelligence\project\3\AI_P3>
```

سوال دهم)

```
E:\uni\Term6\intelligence\project\3\AI_P3>python pacman.py -p ApproximateQAgent -a extractor=SimpleExtractor -x 50 -n 60
 -l mediumGrid
Beginning 50 episodes of Training
Training Done (turning off epsilon and alpha)
----------------------------------------
Pacman emerges victorious! Score: 525
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 527
Pacman emerges victorious! Score: 527
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 527
Pacman emerges victorious! Score: 527
Average Score: 527.8
Scores:        525.0, 529.0, 527.0, 527.0, 529.0, 529.0, 529.0, 529.0, 527.0, 527.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win

E:\uni\Term6\intelligence\project\3\AI_P3>
```

در این قسمت با پیاده سازی دو تابع getQValue که در آن

$$Q(s,a) = \sum_{i=1}^{n} f_i(s,a) w_i$$

را پیاده سازی کردیم. و در update بعد از محاسبه diff به این صورت:

$$difference = (r + \gamma \, max_{a'} Q(s', a')) - Q(s,a)$$

وزن ها را با فرمول زیر آپدیت کردیم:

$$w_i \leftarrow w_i + \alpha \cdot difference \cdot f_i(s, a)$$