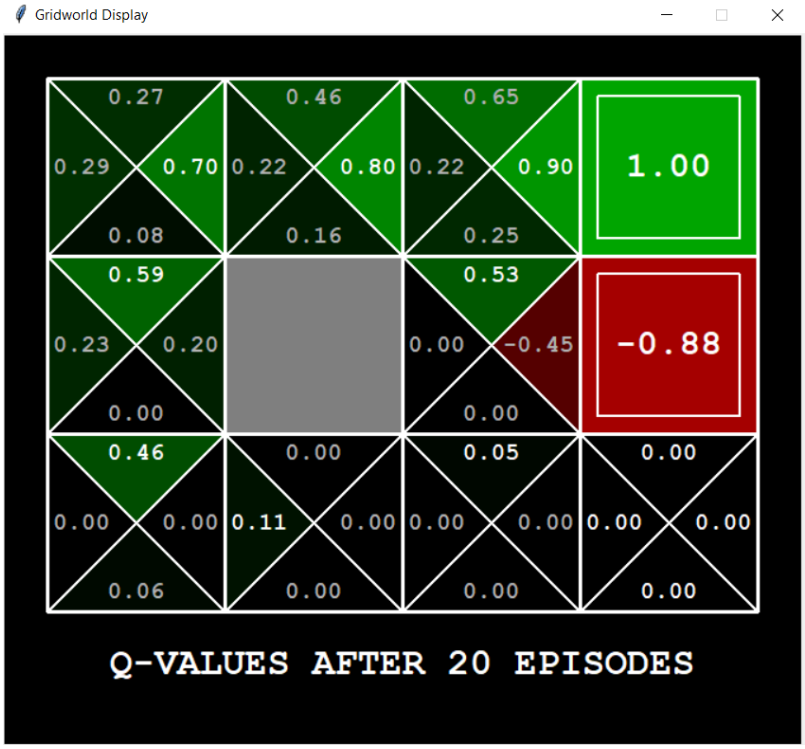
در بخش عملی ما 5 تا متد را پیاده سازی میکنیم:

1) getQValue() : در این متد از دیکشنریcount استفاده میکنیم که key آن یک تاپل از state و action است. در صورتی که استیتی رو ندیده باشیم و در فور چک شده 0.0 خروجی میدهیم و در غیر این صورت مقدار ذخیره شده را میدهیم.
2) computeValueFromQValues() : در این متد ما از بین تمام Q ها ماکسیمم را بدست آورده و در صورتی که Q ای نداشته باشیم 0.0 خروجی میدهد.
3) getACtion() : در این تابع مشخص میشود که agent چقدراز تجربه خودش و مقادیر بدست آمده با احتمال epsilon استفاده کند. که برای این کار با تابع flipCoin زمانی که خروجی از مقدار epsilon کمتر باشد یک عدد رندوم و در غیر این صورت از مقادیر تجربه اش استفاده کند.
4) ComputeActionFromQValues() : در این متد قصد داریم بهترین اکشن را انتخاب کنیم بنابراین ایتدا بهترین Q را بدست آورده و در صورتی که یک qvalues ای داشته باشیم که مقدارش با value برابر باشد، اکشنش را ریترن میکنیم.
5) Update() : در این تابع با استفاده از فرمولی که در اسلاید ها بود مقادیر را محاسبه کرده و خروجی میدهیم.

سواال 1: علت متفاوت بودن حرکت agent در قطعی نبودن حرکت آن است زیرا حرکت به هر سمت یک احتمالی دارد.

سوال 2: با توجه به لاگ متوجه میشویم که مسیر بالا راست بهتر است یعنی در مقادیر بدست آمده ما را به این نتیجه میرساند که احتمال برای این حرکت بالاتر است و حالت هایی که باعث رفتن به مقصد منفی است احتمال کمتری میگیرند و با بیشتر شدن تعداد iteration ها این مقادیر همین نتایج را میدهند و باعث میشوند از انتخاب خود مطمئن تر شویم.



Crawler :

Learning rate : عددی است بین 0 و 1 که مقدار یادگیری ما از نمونه های جدید را نشان میدهد. هرچقدر این عدد به 1 نزدیک تر باشد باعث تغییرات سریع Q میشود و کم تر تجارب قبلی اش را در نظر میگیرد و از نمونه های جدید استفاده میکند.

Epsilon:

این مقدار میزان استفاده ما از action های رندوم یا action های خوبی که داریم را مشخص میکند. نزدیک بودن این عدد به 1 نشان دهنده ی این است که از تجربه های قبلی خود استفاده کمتر میکنیم و از مقادیر رندوم بیشتر استفاده میکنیم. وبرای نزدیک بودن به صفر هم برعکس همین قضیه هست.