امیر بهنام -۹۸۳۱۱۳۳-گزارش سوم

سوال اول

در این سوال در تابع computeActionFromValues در ابتدا تمامی حالت های ممکن برای مسئله توسط تابع self.mdp.getPossibleActions(state) دریافت می شود سپس در صورتی که لیست دریافت شده خالی باشد و یا حالت مسئله پایان پذیرفته باشد مقدار none برگشته برگشت داده میشود

همچنین در این مسئله از تابع کمکی کانتر استفاده شده است

این تابع یک تابع کمکی برای شمارش تعداد در یک دیکشنری است :

A counter keeps track of counts for a set of keys.

سپس با توجه به مقادیر داخل لیست دریافت شده بیشترین مقدار در دیکشنری را با توجه به تابع argmax پیدا کرده و به عنوان خروجی برمیگردانیم.

در تابع ابتدا تمامی حالات ممکن را از فانکشن self.mdp.getTransitionStatesAndProbs(state, action)

دریافت کرده و با توجه به مقادیر تخفیف و جایزه که درون متغیر self قرار دارند مقدار q را با یک حلقه بر روی حالات ممکن جمع بسته و در نهایت مقدار را برمی گردانیم

سوال دوم

در این سوال نیاز داریم تا پارامترهای مسئله را با تغییر فقط یکی از پارامترهای پیش فرض تخفیف یا نویز با آزمون و خطا به جواب مناسب برسیم در این مسئله مقادیر به دست آمده برای تخفیف برابر با مقدار پیش فرض و برای مقدار صفراست سیاست از پل عبور می کنم

سوال سوم

در این سوال مانند سوال قبل نیاز است تا با تغییر پارامتر ها سیاست بهینه را پیدا کنیم. این پارامتر ها با استفاده از آزمون و خطا به دست آمده اند

سوال چهارم

با توجه به نیاز مسئله که کامل کردن تابع runValueIteration است، در این تابع مقدار تکرار را که از قبل در سازنده تابع دریافت شده است را به عنوان تعداد تکرار در نظر میگیریم. سپس تمامی حالات مساله را پیدا کرده و با توجه به تمامی اکشن های ممکن در این حالت مقدار هر مسیر را به روز رسانی می کند.

منظور از تعدا حالات خروجی تابع

self.mdp.getStates()

و منظور از از اکشن ها خروجی تابع

self.mdp.getPossibleActions(state)

است. بدیهی است به ازای هر حالت در مسئله ممکن است چند اکشن وجود داشته باشد. در این تابع برای هر اکشن مقدار qvalueرا محاسبه کرده و برای هر حالت بیشترین مقدار اکشن ها در qvalue را محاسبه می‌کنیم. پس از محاسبه بیشترین مقدار، آن را در متغیر

self.values

قرار می دهیم. لازم به ذکر است که این مقدار یک دیکشنری است که مقادیر همه حالات را در خود نگه می دارد.

سوال پنجم

برای حل این مسئله با توجه به صورت سوال، ابتدا یک صفحه خالی را برای نگهداری اولویت ها تعریف می کنیم

priorityQueue = util.PriorityQueue()

سپس برای هر حالت مقدار diffرا با توجه به صورت سوال پیدا کرده و به سر اولویت ها با مقدار منفی اضافه می کندو چون ثبت ما مینیموم است، این مقدار را با علامت منفی اضافه می کند سپس تا زمانی که صفحه اولویت ها خالی نباشد، مقدار q را بیشترین این مقدار را در حالت های ممکن دارد را محاسبه میکنیم .فرض کنید مقدار متناظر با q حالت pاست .سپس در صورتی که diff<theta

حالت p را با اولویت diff- به صف اضافه می کنیم .

سوال ششم

برای تکمیل تابع

getQValue

فقط بررسی می کنیم در صورتی که در داخل دیکشنری

values

مقدار متناظر با این حالت و اکشن وجود نداشته باشند مقدار صفر و در غیر این صورت مقدار آن را برمی گردانیم

در تابع update پس از دریافت مقدار از تابعgetQValue

سعی می کنیم با اضافه و کم کردن مقادیر تخفیف جایزه و آلفا مقدار آن را به روز کنید

در این مسئله برای تکمیل تابعcomputeActionFromQValues

ابتدا تمامی اکشن های ممکن را دریافت کرده و سپس بیشترین مقدار را با توجه به تابع

getQValue

پیدا می کنیم. در صورتی که مقدار بهترین مقدار یافت شده تاکنون برابر باشد می توانیم به صورت رندوم و به این تابع خاتمه دهیم در نهایت سایت بهترین اکشن را با توجه به بالاترین مقدار برمیگردد

سوال 7:

در این مسئله تابع getAction را با انداختن سکه با شانس اپسیلون اجرا میکنیم. در صورتی که سکه true باشد از بین حالات یکی را به صورت رندوم انتخاب می کنیم. در غیر این صورت مقدار اکشن را از تابع computeActionFromQValues محاسبه ld کنیم

سوال :8

در این سوال با آزمون و خطا سیاست بهینه با احتمال خیلی بالا را به دست می‌آورد. جواب به دست آورده شده برای دو پارامتر اپسیلون و و نرخ یادگیری برابر با نمی باشد

سوال 10:

در این سوال دو فانکشن getQValue و update پیاده سازی شده اند

در فانکشن اول مقدار مورد نیاز با جمع کردن ضرب برداری وزن های یک حالت و بردار ویژگی ها به دست می آید

در تابع دوم مقدار وزن ها بر اساس ضرب برداری وزن در ویژگی متناظر با این وزن وزن و مقدار آلفا و مقدار اختلاف به روز می شود

self.weights[x] = self.weights[x] + self.alpha * difference * y