دانشگاه اصفهان
دانشکده مهندسی کامپیوتر

گزارش پروژه درس هوش مصنوعی

مرحله اول: حل مسأله با جستجو

اعضای گروه: مقداد دهقان
شماره دانشجویی: ۹۸۳۶۵۳۰۰۱
نام گروه: بازیکن شماره ۴۵۶

پاییز ۱۴۰۰

من، مقداد دهقان، تعهد می‌نمایم که پروژه تحویل داده شده نتیجه کار من بوده و در هیچ یک از بخش‌های انجام شده از کار دیگران کپی برداری نشده است. در صورتی که مشخص شود که این پروژه کار من نبوده است، طبق ضوابط آموزشی با من برخورد شده و حق اعتراض نخواهم داشت.

پیاده‌سازی پروژه ابتدا توسط زبان پایتون شروع شد اما با توجه به تجربه‌ی کم استفاده از این زبان و پیشرفت کند پروژه تصمیم گرفته شد تا از زبان جاوا به جای پایتون استفاده شود. برای انجام مراحل بعدی پروژه درصورتی که نیاز به استفاده از کتابخانه‌های مخصوص زبان پایتون باشد، ابتدا این مرحله از پروژه به زبان پایتون تکمیل می‌شود و سپس سایر مراحل با این زبان و کتابخانه‌های مورد نیاز آن انجام خواهد شد.

بنابراین برای اجرای کد پروژه از کلاینت نوشته شده به زبان جاوا استفاده شود.

از آنجایی که در این مرحله از پروژه عامل دوم وجود ندارد بنابراین از کنش قرار دادن تله (trap) صرف نظر شده و در مسائل جستجو و تعیین راهبرد جمع‌آوری الماس‌ها استفاده از این کنش و بررسی آن در نظر گرفته نشده است. همچنین از امکان وجود تله در خانه‌های نقشه نیز صرف نظر شده است.

۱- مسأله مسیریابی

در این مسأله با داشتن مختصات مکان عامل در نقشه و نیز مختصات خانه‌ی الماس مورد نظر، هدف این است که کوتاه‌ترین مسیر ممکن بین عامل و خانه‌ی الماس مورد نظر مشخص شود تا عامل بتواند طی یک مجموعه‌ی متوالی از کنش‌ها به خانه‌ی هدف برسد. نحوه‌ی انتخاب الگوریتم مناسب برای رسیدن به این هدف با درنظر گرفتن محدودیت‌های موجود، در ادامه آورده شده است.

فضای حالت: هر خانه‌ی نقشه می‌تواند تنها یکی از انواع خانه‌های خالی (E)، دیوار (W)، سیاه چاله (T)، خانه‌ی شامل الماس زرد (۱)، خانه شامل الماس سبز (۲)، خانه‌ی شامل الماس قرمز (۳)، و خانه‌ی شامل الماس آبی (۴) باشد. اگر الماس موجود در خانه‌ای توسط عامل برداشته شود، آن خانه به خانه‌ی خالی تبدیل می‌شود. عامل می‌تواند در هر یک از خانه‌های بالا به جز خانه‌ی شامل دیوار قرار بگیرد. ۴ خانه‌ی گوشه‌ی نقشه همواره خالی هستند.

حالت اولیه: در ابتدای مسیریابی در هر مرحله (نه صرفاً در ابتدای شروع بازی)، عامل می‌تواند در هر یک از خانه‌های نقشه که امکان حضور در آن‌ها وجود دارد، قرار داشته باشد. بنابراین حالت اولیه‌ی عامل شامل یکی از خانه‌های خالی، سیاه چاله، خانه‌ی شامل الماس زرد، خانه‌ی شامل الماس سبز، خانه‌ی شامل الماس قرمز، و یا خانه‌ی شامل الماس آبی است. همچنین الماس هدف نیز می‌تواند یکی از خانه‌های شامل الماس باشد.

به عنوان مثال اگر عامل بخواهد از خانه‌ی با مختصات (۲، ۴) به خانه‌ای با مختصات دیگری برود، حالت اولیه عامل، خانه‌ی با مختصات (۲، ۴) خواهد بود که نوع این خانه هر یک از خانه‌های خالی یا سیاه چاله یا خانه‌های شامل الماس می‌تواند باشد.

کنش‌های ممکن: کنش‌های قابل انجام توسط عامل شامل حرکت به سمت بالا (up)، حرکت به سمت پایین (down)، حرکت به سمت راست (right)، حرکت به سمت چپ (left)، عبور از سیاه چاله (trap)، و کنش بدون حرکت (noop) است.

هزینه کنش: هزینه‌ی هر یک از کنش‌های حرکت به سمت بالا، حرکت به سمت چپ، حرکت به سمت راست، حرکت به سمت پایین، و کنش بدون حرکت برابر با یک است. همچنین با یک بار انجام کنش عبور از سیاه چاله نیز هزینه‌ای برابر با یک واحد از امتیاز عامل کم می‌شود اما در حل مسأله جستجو، برای پیدا کردن کوتاه‌ترین

مسیر تا هدف، هزینه کنش عبور از سیاه چاله برابر با یک واحد کمتر از تعداد کل سیاه چاله‌ها در نظر گرفته شده است. زیرا هنگامی که در یک خانه‌ی شامل سیاه چاله کنش عبور از سیاه چاله انجام می‌شود، احتمال بیرون آمدن از هر یک از سیاه چاله‌های دیگر با هم برابر است. مثلاً اگر ۵ سیاه چاله در نقشه وجود داشته باشد و عامل ما در یکی از این سیاه چاله‌ها قرار داشته باشد، با عبور از آن سیاه چاله، احتمال بیرون آمدن از هر یک از سیاه چاله‌های دیگر برابر با یک چهارم است. بنابراین اگر عامل بخواهد با عبور از یک سیاه چاله به سیاه چاله‌ی دیگری منتقل شود باید حداقل ۴ بار کنش عبور از سیاه چاله را انجام دهد تا مطمئن شود به سیاه چاله‌ی مورد نظر منتقل شده است.

مدل انتقال: با انجام کنش بدون حرکت، عامل در مکان قبلی خود باقی می‌ماند. با انجام کنش‌های حرکت به سمت بالا، حرکت به سمت چپ، حرکت به سمت راست، و حرکت به سمت راست عامل به خانه‌ی مورد نظر منتقل می‌شود. با انجام این ۴ کنش، در صورتی که خانه‌ی هدف شامل دیوار باشد یا در نقشه قرار نداشته باشد کنش بدون حرکت انجام می‌شود. با انجام کنش عبور از سیاه چاله به یکی از سیاه چاله‌های دیگر موجود در نقشه منتقل می‌شویم. البته کنش عبور از سیاه چاله زمانی امکان پذیر است که در خانه‌ی شامل سیاه چاله باشیم. زمانی که در خانه‌ای غیر از سیاه چاله باشیم و کنش عبور از سیاه چاله را انجام دهیم، کنش بدون حرکت انجام می‌شود.

آزمایش هدف: زمانی که مختصات عامل با متخصات خانه‌ای که الماس هدف در آن قرار دارد برابر شود، به حالت هدف رسیده‌ایم.

برای پیدا کردن یک الگوریتم مناسب برای مسأله مسیریابی ابتدا از یک نقشه‌ی ساده و بدون وجود هیچ‌گونه دیوار و سیاه چاله‌ای استفاده می‌کنیم تا بتوانیم مناسب‌ترین الگوریتم را مرحله به مرحله تا در نظر گرفتن تمامی موجودیت‌های حاضر در نقشه انتخاب کنیم.

بنابراین در ابتدا فرض کنید که در نقشه صرفاً عامل ما و الماس هدف موجود است. در این صورت مناسب‌ترین الگوریتم برای پیدا کردن کوتاه‌ترین مسیر در کمترین زمان ممکن، الگوریتم A^* می‌باشد. چراکه توسط این الگوریتم و با استفاده از یک تابع اکتشافی ساده که هزینه‌ی رسیدن عامل به الماس را برابر با مجموع فواصل سطری و ستونی عامل از الماس در نظر می‌گیرد، می‌توان خانه‌هایی را که فاصله‌ی آن‌ها تا الماس هدف بیشتر از فاصله‌ی عامل تا الماس است را در نظر نگرفت و بسیار سریع‌تر به هدف نزدیک شد.

برای مثال اگر نقشه زیر نمایی از نقشه‌ی بازی باشد که شامل عامل در خانه‌ی شماره ۱۰ و الماس در خانه‌ی شماره ۳۴ است، تنها با گسترش خانه‌هایی که با رنگ سبز مشخص شده‌اند، بهترین مسیر توسط الگوریتم A* پیدا می‌شود. این مسیر می‌تواند با عبور از خانه‌های ۱۷، ۱۸، ۲۵، ۲۶، ۲۷، و ۳۴ بدست آید.

۱	۲	۳	۴	۵	۶	۷
۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴
۱۵	۱۶	۱۷	۱۸	۱۹	۲۰	۲۱
۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸
۲۹	۳۰	۳۱	۳۲	۳۳	۳۴	۳۵
۳۶	۳۷	۳۸	۳۹	۴۰	۴۱	۴۲

نقشه ۱ – الگوریتم A*

حال فرض کنید که علاوه بر عامل و الماس هدف، خانه‌های شامل دیوار نیز در نقشه وجود داشته باشند. در این صورت باز هم الگوریتم A* نسبت به سایر الگوریتم‌ها بهینه‌تر است. چراکه کوتاه‌ترین مسیر را در کوتاه‌ترین زمان بدست می‌آورد.

برای مثال در نقشه زیر خانه‌های خاکستری رنگ دیوار هستند و خانه‌هایی که توسط الگوریتم A* گسترش پیدا می‌کنند تا بهترین مسیر پیدا شود، به رنگ سبز نشان داده شده‌اند. در این نقشه بهترین مسیری که توسط الگوریتم A* پیدا می‌شود با عبور از خانه‌های ۱۷، ۲۴، ۳۱، ۳۸، ۳۹، ۴۰، ۴۱، و ۳۴ بدست می‌آید.

۱	۲	۳	۴	۵	۶	۷
۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴
۱۵	۱۶	۱۷	۱۸	۱۹	۲۰	۲۱
۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸
۲۹	۳۰	۳۱	۳۲	۳۳	۳۴	۳۵
۳۶	۳۷	۳۸	۳۹	۴۰	۴۱	۴۲

نقشه ۲ – الگوریتم A*

اما زمانی که سیاه چاله نیز به خانه‌های موجود در نقشه اضافه شود الگوریتم A* تضمین نمی‌کند که همواره کوتاه‌ترین مسیر را پیشنهاد دهد.

برای مثال اگر در همان نقشه قبل دو سیاه چاله در خانه‌های ۹ و ۴۲ اضافه شوند و نقشه‌ی بازی به صورت زیر درآید، الگوریتم A* همان مسیر قبلی را پیشنهاد می‌دهد درصورتی که بهترین مسیر رسیدن به الماس، با عبور از دو سیاه چاله بدست می‌آید.

۱	۲	۳	۴	۵	۶	۷
۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴
۱۵	۱۶	۱۷	۱۸	۱۹	۲۰	۲۱
۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸
۲۹	۳۰	۳۱	۳۲	۳۳	۳۴	۳۵
۳۶	۳۷	۳۸	۳۹	۴۰	۴۱	۴۲

نقشه ۳ – الگوریتم A*

بنابراین برای رسیدن به کوتاه‌ترین مسیر نیاز است که سایر خانه‌های نقشه نیز گسترش داده شوند. در این صورت اگر سیاه چاله‌هایی در نقشه وجود داشته باشند که بتوانند سریع‌تر عامل را به هدف برسانند در نظر گرفته می‌شوند. پس با این وجود الگوریتم بهترین اول مناسب‌ترین گزینه به نظر می‌رسد. زیرا شرط بهینگی را حفظ می‌کند، هر چند از نظر زمانی نسبت به الگوریتم A* از مرتبه‌ی زمانی کمتری برخوردار نیست. البته از آنجایی که ابعاد نقشه‌ی بازی محدود هستند و حداکثر ابعاد آن ۲۰*۲۰ می‌باشد، استفاده از الگوریتم بهترین اول می‌تواند محدودیت زمان تصمیم‌گیری عامل را رعایت کند.

برای مثال توسط الگوریتم بهترین اول نحوه‌ی گسترش خانه‌های نقشه قبل به صورت زیر خواهد بود که منجر به پیدا کردن بهترین مسیر با عبور از خانه‌های ۹، ۴۲، ۳۵، و ۳۴ می‌شود.

۱	۲	۳	۴	۵	۶	۷
۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴
۱۵	۱۶	۱۷	۱۸	۱۹	۲۰	۲۱
۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸
۲۹	۳۰	۳۱	۳۲	۳۳	۳۴	۳۵
۳۶	۳۷	۳۸	۳۹	۴۰	۴۱	۴۲

نقشه ۴ – الگوریتم بهترین اول

بنابراین برای حل مسأله جستجو در پروژه از الگوریتم بهترین اول استفاده شده است تا عامل بتواند کوتاه‌ترین مسیر ممکن تا الماس هدف را بدست بیاورد.

۲- مسأله تعیین راهبرد جمع‌آوری الماس‌ها

در این مسأله هدف این است که یک الماس بهینه از میان سایر الماس‌های موجود در نظر گرفته شود تا عامل بتواند با صرف کمترین هزینه بیشترین امتیاز را در پایان بازی بدست بیاورد.

راه‌حل فراگیر این مسأله که منجر به پیدا شدن جواب بهینه می‌شود به این صورت است که باید کوتاه‌ترین مسیر از هر الماس تا سایر الماس‌ها و نیز عامل را بدست بیاوریم و سپس گراف کاملی را تشکیل دهیم که در آن همه الماس‌ها و نیز عامل به هم متصل هستند (در صورت وجود مسیر بین هر دو الماس با هم و نیز با عامل) و هزینه هر یال برابر با هزینه کوتاه‌ترین مسیر از رئوس دو سر آن یال به هم می‌باشد. حال با استفاده از یک الگوریتم جستجو مانند A^* یا بهترین اول می‌توان کوتاه‌ترین مسیری که از حداکثر تعداد الماس‌ها عبور می‌کند و بیشترین امتیاز را بدست می‌آورد را پیدا کرد. مشکل این روش در این است که برای بدست آوردن کوتاه‌ترین مسیر از هر الماس تا سایر الماس‌ها و نیز عامل، پیچیدگی زمانی زیادی مورد نیاز است و حتی در صورت پیدا کردن این مسیرها برای جلوگیری مجدد از پیدا کردن دوباره باید آن مسیرها را ذخیره کرد که همین امر باعث زیاد شدن پیچیدگی حافظه نیز خواهد شد.

بنابراین برای حل این مسأله از الگوریتم جستجوی محلی استفاده می‌شود و در هر مرحله الماسی را به عنوان الماس هدف انتخاب می‌کنیم که دارای کمترین فاصله‌ی تخمینی تا عامل باشد. در صورتی که چندین الماس دارای کمترین فاصله‌ی تخمینی تا عامل باشند، الماس با امتیاز بیشتر انتخاب می‌شود.

برای بدست آوردن فاصله‌ی تخمینی یک الماس تا عامل از مجموع فواصل سطری و ستونی استفاده می‌کنیم. از آنجایی که این روش تنها فاصله‌ی سطری و ستونی تا الماس را در نظر می‌گیرد، ممکن است در عمل الماسی انتخاب شود که هیچ مسیری از آن تا عامل وجود نداشته باشد. در این صورت توسط الگوریتم بهترین اول هیچ مسیری بین عامل و الماس پیدا نمی‌شود و عامل نمی‌تواند به الماس برسد. حال اگر این الماس همچنان جزو گزینه‌های قابل انتخاب توسط عامل باشد، این الماس در دورهای بعد نیز به عنوان بهترین گزینه توسط عامل انتخاب خواهد شد و عامل در این نقطه باقی می‌ماند و نمی‌تواند به سراغ سایر الماس‌ها برود. برای رفع چنین مشکلی یک لیست از الماس‌های غیرقابل دسترسی (blockedGems) در نظر گرفته شده است که در صورت پیدا نشدن مسیری به یک الماس توسط الگوریتم بهترین اول، این الماس به لیست blockedGems اضافه می‌شود و دوباره مسأله تعیین راهبرد جمع‌آوری الماس‌ها بدون در نظر گرفتن الماس‌های لیست

blockedGems اجرا می‌شود. حال عامل گزینه‌ی دیگری را انتخاب می‌کند که می‌تواند از آن را از نقطه‌ی قبلی که در آن گرفتار شده بود نجات دهد.

با این توضیحات می‌توان تعریف مسأله تعیین راهبرد جمع‌آوری الماس‌ها را به این صورت آورد:

فضای حالت: فضای حالت این مسأله همانند مسأله جستجو می‌باشد با این تفاوت که در این مسأله چندین الماس می‌توانند به عنوان هدف مورد بررسی قرار بگیرند.

حالت اولیه: حالت اولیه این مسأله شامل مختصات خانه‌ای است که عامل در آن قرار دارد و نیز مختصات سایر الماس‌هایی که در نقشه باقی مانده‌اند و قابل دستیابی هستند.

کنش‌های ممکن: از آنجایی که در این مسأله تنها فاصله تخمینی تا الماس‌ها بدست می‌آوریم و از وجود سیاه چاله‌ها در این تخمین صرف نظر می‌کنیم، می‌توان کنش عبور از سیاه چاله را در نظر نگرفت و صرفاً کنش‌های حرکت به سمت بالا، حرکت به سمت چپ، حرکت به سمت راست، حرکت به سمت پایین، و کنش بدون حرکت را در نظر گرفت. علت در نظر گرفتن کنش بدون حرکت این است که در زمان‌هایی که هیچ انتخاب مناسبی برای عامل وجود ندارد از آن استفاده کنیم. هرچند که در این مرحله از پروژه که عامل دوم وجود ندارد، استفاده از سایر کنش‌ها فرقی با کنش بدون حرکت ندارد.

هزینه کنش: با درنظر گرفتن کنش‌های ذکر شده، هزینه هر یک از این کنش‌ها برابر با یک واحد خواهد بود.

مدل انتقال: همانطور که گفته شد در این مسأله از فاصله‌ی تخمینی میان عامل تا الماس‌ها استفاده می‌شود. در این فاصله‌ی تخمینی علاوه بر سیاه چاله‌ها از دیوارها نیز می‌توان صرف نظر کرد. بنابراین نتیجه‌ی هر یک از کنش‌های حرکت به سمت بالا، حرکت به سمت چپ، حرکت به سمت راست، و حرکت به سمت پایین را در صورتی که باعث خارج شدن عامل از نقشه نشوند، می‌توان انتقال به خانه‌ی مورد نظر دانست. همچنین نتیجه‌ی کنش بدون حرکت باقی ماندن در خانه‌ی قبلی می‌باشد.

آزمایش هدف: از آنجایی که در این مسأله از الگوریتم جستجوی محلی استفاده شده است بجای استفاده از آزمایش هدف، از تابع هدف استفاده می‌شود که توسط این تابع مناسب‌ترین الماس برای انتخاب بعدی عامل در نظر گرفته می‌شود. این الماس باید کمترین فاصله تخمینی تا عامل را داشته باشد و جزو لیست blockedGems نباشد. در صورتی که چندین الماس دارای این شرایط باشند الماس با بیشترین امتیاز انتخاب می‌شود.

۳- نحوه لحاظ کردن محدودیت زمان تصمیم‌گیری برای عامل

از آنجایی که عامل برای تصمیم‌گیری در انتخاب کنش بعدی محدود به یک زمان مشخص (۱ ثانیه) است باید این زمان را در مدت تصمیم‌گیری عامل لحاظ کرد و درصورت عبور از این زمان یک کنش توسط عامل انجام شود. با توجه به اینکه فضای حالت این مسأله محدود بوده و الگوریتم‌های استفاده شده در آن، در زمان بسیار کمتر از ۱ ثانیه به نتیجه می‌رسند، نیاز به در نظر گرفتن زمان تصمیم گیری عامل از ابتدای الگوریتم تعیین راهبرد جمع‌آوری الماس‌ها تا انتهای الگوریتم جستجو نیست. تنها زمانی که عامل در حال بررسی کنش بعدی براساس پاسخ الگوریتم جستجوی بهترین اول است می‌توان این محدودیت زمانی را لحاظ کرد. چراکه اگر تعداد زیادی از الماس‌ها قابل دسترسی نباشند عامل باید مسیرهای موجود تا تمامی آن‌ها را بررسی کند تا بالاخره بتواند یک الماس قابل دسترسی پیدا کند. در این حالت اگر عامل نتواند تا زمان تعیین شده کنش خود را انتخاب کند، یک کنش بدون حرکت انجام می‌شود تا در مرحله‌ی بعد بتواند سایر الماس‌ها را بررسی کند به امید اینکه در مراحل بعدی بتواند الماس‌های قابل دسترسی را پیدا کند.