دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر

«مبانی و کاربردهای هوش مصنوعی ترم پائیز ۱۴۰۴»

پروژه اول

در انجام پروژه‌ها به نکات زیر توجه فرمائید:

۱- پیاده‌سازی پروژه‌ها را به زبان برنامه‌نویسی پایتون انجام دهید.

۲- مطابق قوانین دانشگاه هر نوع کپی‌برداری و اشتراک کار دانشجویان غیرمجاز است و پاسخ به پروژه‌ها باید به صورت انفرادی و بدون استفاده از ابزارهای هوش مصنوعی انجام شود، در صورت مشاهده چنین مواردی با طرفین شدیدا برخورد خواهد شد.

۳- فایل پروژه را با فرمت StudentID_AI_P01.zip تا ساعت ۲۳:۵۹ روز ۱۴۰۴/۰۸/۱۶ فقط در بخش مربوطه در سایت درس آپلود نمائید.

۴- توجه نمائید پاسخ پروژه‌ها تنها در صورت آپلود در سامانه کورسز پذیرفته خواهد شد و ارسال پاسخ از طریق ایمیل یا تلگرام بررسی نخواهد شد.

۵- در مجموع برای پروژه‌ها ۱۰ روز تاخیر مجاز دارید که می‌توانید در طول ترم بسته به شرایط از آن استفاده نمائید، در صورت اتمام تاخیر مجاز، هر روز تاخیر منجر به کسر نمره از پروژه خواهد شد.

فهرست مطالب

مقدمه ۲
شرح دقیق قوانین بازی..... ۳
ساختار کلی پروژه و وظیفه‌ی شما..... ۵
گزارش کار و تحلیل نتایج ۷

مقدمه

یکی از بنیادی‌ترین مفاهیم در هوش مصنوعی، جست‌وجو در فضاهای حالت است. در بسیاری از مسائل هوش مصنوعی، از مسیریابی در نقشه‌ها گرفته تا حل پازل‌ها و برنامه‌ریزی خودکار، عامل هوشمند باید بتواند در میان حالت‌های ممکنِ مسئله جست‌وجو کند تا به یک وضعیت هدف برسد. این جست‌وجو ممکن است با روش‌های کور (مانند DFS و BFS) یا آگاهانه (مانند UCS، Greedy و A^* که از تابع هزینه و تابع Heuristic استفاده می‌کنند) انجام شود. هدف از این پروژه، درک عمیق‌تر این الگوریتم‌ها از طریق پیاده‌سازی عملی و مشاهده رفتارشان در یک محیط تعاملی است.

در این پروژه، شما با نسخه‌ای تغییریافته از بازی کلاسیک Sokoban کار خواهید کرد. در نسخه‌ی اصلی، بازیکن در یک انبار قرار دارد و باید با هل‌دادن جعبه‌ها، آن‌ها را روی موقعیت‌های هدف قرار دهد. در نسخه‌ی تغییریافته‌ی ما، علاوه بر جعبه‌ها و اهداف، عناصر جدیدی مانند سیب‌ها و سم‌ها نیز به محیط اضافه شده‌اند که به‌صورت پویا بر هزینه‌ی حرکات بازیکن تأثیر می‌گذارند. این تغییرات باعث می‌شود مسئله نه‌تنها از نظر هندسی و منطقی، بلکه از منظر مدل‌سازی هزینه و طراحی تابع ابتکاری نیز چالش‌برانگیز شود.

این پروژه به‌گونه‌ای طراحی شده است که شما در نقش یک عامل هوشمند قرار می‌گیرید که باید با استفاده از الگوریتم‌های مختلف جست‌وجو، بهترین مسیر را برای حل پازل پیدا کند. تفاوت در هزینه‌ها، وجود سیب‌ها و سم‌ها، و امکان گیر افتادن در موقعیت‌های بن‌بست باعث می‌شود که تحلیل و مقایسه‌ی روش‌های جست‌وجو معنای عمیق‌تری پیدا کند. به‌عنوان مثال، مشاهده خواهید کرد که چرا BFS همیشه مناسب نیست، چگونه UCS می‌تواند مسیر بهینه را تضمین کند و یا چرا طراحی یک تابع هیوریستیک خوب در عملکرد A^* نقش حیاتی دارد.

در نهایت، هدف این پروژه صرفاً پیاده‌سازی الگوریتم‌ها نیست، بلکه درک شهودی از رفتار و کارایی آن‌ها در دنیای واقعی است. شما با مشاهده‌ی مسیرهای انتخاب‌شده توسط هر الگوریتم، تحلیل هزینه‌ها، و بررسی تعداد گره‌های بسط‌داده‌شده، درمی‌یابید که چگونه انتخاب یک روش جست‌وجو می‌تواند تفاوت چشمگیری در کیفیت، سرعت و بهینگی پاسخ ایجاد کند. این تجربه، نخستین گام عملی شما برای درک نحوه‌ی تصمیم‌گیری هوشمند در محیط‌های پیچیده است.

شرح دقیق قوانین بازی

محیط بازی به‌صورت یک صفحه‌ی دوبعدی از سلول‌ها تشکیل شده است که هر سلول می‌تواند یکی از عناصر بازی را در خود جای دهد. دیوارها با نماد # مشخص می‌شوند و هیچ موجودیتی، نه بازیکن و نه جعبه، نمی‌تواند از آن‌ها عبور کند. سلول‌های خالی که بازیکن می‌تواند روی آن‌ها حرکت کند، با فضای خالی ( ) نمایش داده می‌شوند. بازیکن با نماد @ مشخص شده و موقعیت فعلی او در نقشه را نشان می‌دهد. جعبه‌ها که باید توسط بازیکن هل داده شوند با $ نمایش داده می‌شوند و اهدافی که جعبه‌ها باید روی آن‌ها قرار گیرند، با . مشخص هستند. وقتی یک جعبه روی هدف قرار می‌گیرد، نماد آن به * تغییر پیدا می‌کند و اگر بازیکن روی یکی از موقعیت‌های هدف بایستد، به صورت + نمایش داده می‌شود. علاوه بر این عناصر اصلی، در نسخه‌ی تغییریافته‌ی بازی، دو عنصر جدید نیز وجود دارد: سیب‌ها که با حروف a یا A مشخص می‌شوند و سم‌ها که با حروف p یا P نمایش داده می‌شوند.

بازیکن در هر لحظه می‌تواند در یکی از چهار جهت بالا، پایین، چپ یا راست حرکت کند. حرکتی معتبر است که مقصد آن دیوار نباشد و اگر در مسیر حرکت، جعبه‌ای وجود دارد، پشت آن جعبه فضای خالی یا موقعیت هدف باشد تا امکان هل دادن آن وجود داشته باشد. بازیکن نمی‌تواند بیش از یک جعبه را هم‌زمان هل دهد و در صورتی که در جهت حرکتش، دو جعبه یا یک دیوار پشت جعبه قرار داشته باشد، آن حرکت غیرمجاز است. هدف نهایی بازی، رساندن تمام جعبه‌ها به موقعیت‌های هدف است، به طوری که در پایان، تمام آن‌ها به نماد * تبدیل شوند. وقتی همه‌ی جعبه‌ها روی اهداف خود قرار گرفتند، بازی به اتمام می‌رسد و بازیکن به هدف رسیده است.

هر حرکت در بازی با توجه به نوع آن دارای هزینه است. اگر بازیکن فقط روی زمین حرکت کند و جعبه‌ای را هل ندهد، هزینه‌ی حرکت برابر با یک واحد است. اما اگر در حرکت خود جعبه‌ای را هل دهد، هزینه‌ی آن در حالت معمولی پنج واحد خواهد بود. این هزینه‌ها پایه‌ی اصلی محاسبه‌ی مسیر بهینه هستند و در کنار عناصر خاص بازی مانند سیب و سم، باعث می‌شوند که هزینه‌ی مسیرها به‌صورت پویا تغییر کند. سیب‌ها عنصری هستند که با خوردن آن‌ها بازیکن برای مدتی قدرت بیشتری پیدا می‌کند. هنگامی که بازیکن روی یک سلول حاوی سیب قرار می‌گیرد، اثر سیب فعال می‌شود و از آن لحظه تا ده حرکت آینده، هزینه‌ی هل دادن جعبه‌ها به‌طور موقت کاهش می‌یابد. در این وضعیت،

هزینه‌ی هل دادن جعبه از پنج به یک کاهش پیدا می‌کند، در حالی که هزینه‌ی حرکت ساده همچنان برابر با یک باقی می‌ماند. پس از انجام هر حرکت، شمارنده‌ی اثر سیب یک واحد کاهش می‌یابد و پس از گذشت ده حرکت، اثر آن از بین می‌رود. اگر بازیکن در این مدت دوباره سیب دیگری بخورد، شمارنده‌ی اثر مجدداً از ابتدا تنظیم می‌شود.

در مقابل، سم‌ها اثر منفی دائمی دارند. وقتی بازیکن روی سلولی که حاوی سم است قدم بگذارد، از آن لحظه به بعد، تمام حرکات او شامل جریمه‌ی هزینه خواهند شد. این جریمه به مقدار سه واحد است و به تمام حرکات اضافه می‌شود، چه بازیکن فقط حرکت کند و چه جعبه‌ای را هل دهد. اثر سم برخلاف سیب موقتی نیست و تا پایان بازی باقی می‌ماند. نکته‌ی جالب این است که اثر سیب و سم می‌توانند به‌طور هم‌زمان فعال باشند؛ در چنین حالتی هزینه‌ها با یکدیگر ترکیب می‌شوند. برای مثال، اگر بازیکن در حالتی باشد که هر دو اثر فعال هستند، هل دادن جعبه هزینه‌ای برابر با چهار واحد خواهد داشت (یعنی یک برای هل دادن و سه به‌دلیل سم)

در هر اکشن، محاسبه‌ی هزینه بر اساس ترتیب مشخصی انجام می‌شود. ابتدا نوع حرکت مشخص می‌شود، سپس هزینه‌ی پایه با توجه به نوع آن تعیین می‌گردد. اگر بازیکن مسموم باشد، جریمه‌ی سم به هزینه اضافه می‌شود. پس از انجام حرکت، تایمر سیب (در صورت فعال بودن) یک واحد کاهش می‌یابد، و اگر بازیکن در موقعیت جدید روی سیب یا سم قرار گیرد، اثر مربوطه فعال می‌شود. در طول بازی ممکن است موقعیت‌هایی ایجاد شود که دیگر حل پازل از آن‌ها ممکن نباشد؛ این موقعیت‌ها بن‌بست نام دارند. برای نمونه، اگر جعبه‌ای در گوشه‌ای از نقشه قرار گیرد که هدفی در آن نقطه وجود ندارد و از دو طرف با دیوار محصور شده است، دیگر هیچ حرکتی نمی‌تواند آن را از گوشه خارج کند. چنین حالت‌هایی باید در جست‌وجو نادیده گرفته شوند، زیرا تنها باعث اتلاف زمان و حافظه خواهد شد.

برای درک بهتر، فرض کنید بازیکن در حالت عادی حرکت می‌کند. در این وضعیت، حرکت ساده چهار واحد هزینه دارد و هل دادن جعبه پنج واحد. اگر بازیکن سیبی بخورد، تا ده حرکت آینده هزینه‌ی هل دادن به یک کاهش می‌یابد. حال اگر در این مدت بازیکن روی سم هم برود، از آن لحظه به بعد هر حرکت او سه واحد گران‌تر می‌شود. بنابراین حرکت ساده در حالت مسموم برابر با هفت واحد هزینه دارد و هل دادن جعبه در حالی که اثر سیب فعال است، چهار واحد هزینه خواهد داشت. پس از پایان اثر سیب، هل دادن جعبه در وضعیت مسموم به هشت واحد افزایش می‌یابد.

ساختار کلی پروژه و وظیفه‌ی شما

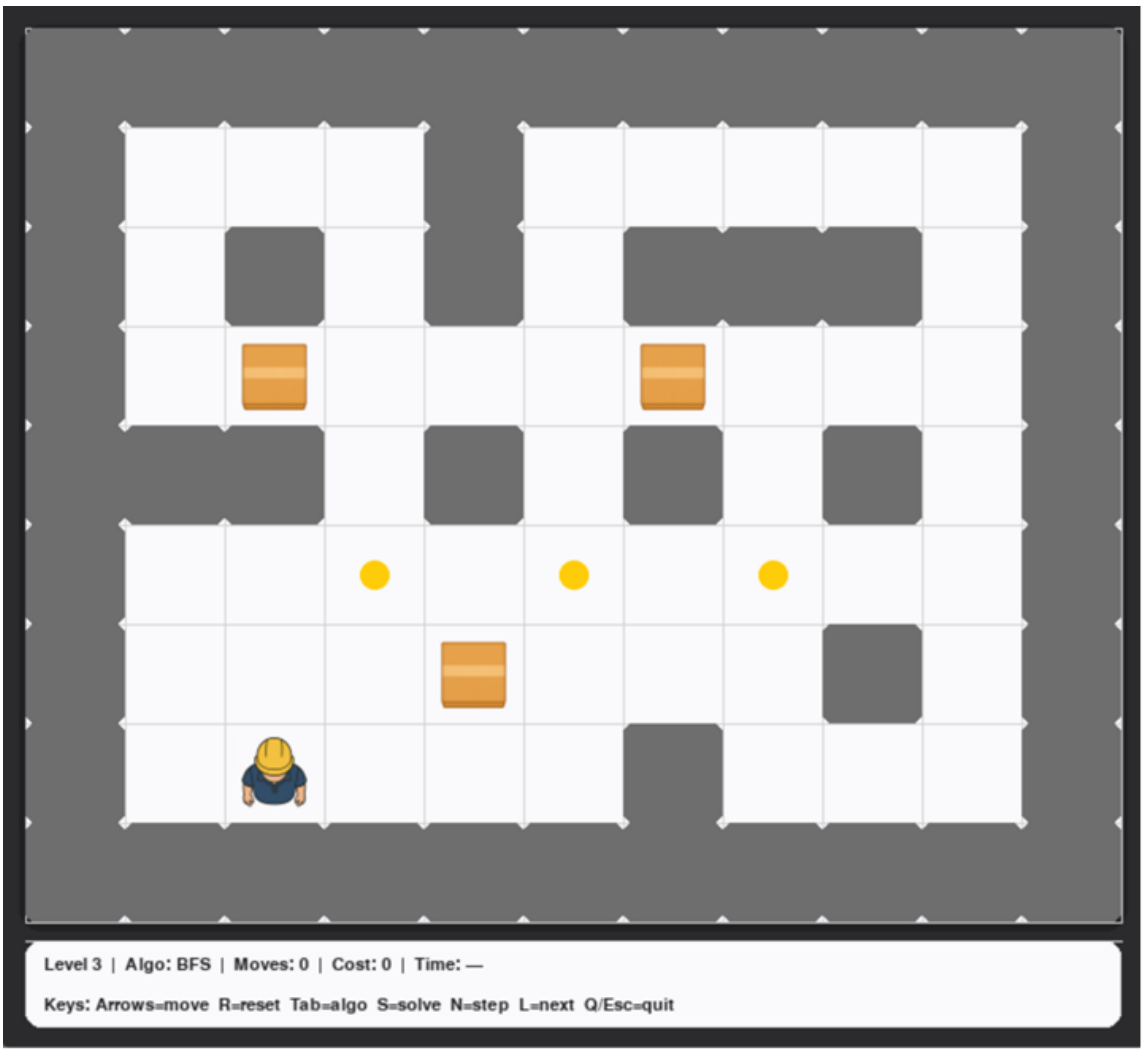
در این پروژه، رابط گرافیکی به‌صورت کامل پیاده‌سازی شده است و نیازی به تغییر یا ویرایش در آن وجود ندارد. این رابط وظیفه‌ی نمایش نقشه، اجرای حرکات بازیکن، پخش مسیر خروجی، و نمایش هزینه‌ی تجمعی و آمار عملکرد الگوریتم را برعهده دارد. **تمام فایل‌های مربوط به طراحی محیط، بارگذاری نقشه‌ها و نمایش گرافیکی حرکت‌ها در اختیار شما قرار داده شده است و شما نباید آن‌ها را تغییر دهید.**

وظیفه‌ی اصلی شما در این پروژه، پیاده‌سازی بخش هوش مصنوعی بازی است؛ یعنی بخشی که تصمیم می‌گیرد بازیکن چگونه باید حرکت کند تا در نهایت تمام جعبه‌ها روی موقعیت‌های هدف قرار گیرند. این بخش شامل تعریف فضای حالت، تولید حالت‌های جانشین، محاسبه‌ی هزینه‌ی حرکات و در نهایت پیاده‌سازی الگوریتم‌های جست‌وجو است.

به‌طور مشخص، شما باید پنج الگوریتم جست‌وجو را در این پروژه پیاده‌سازی کنید: جست‌وجوی عمق‌اول (DFS)، جست‌وجوی عرض‌اول (BFS)، جست‌وجوی یکنواخت (UCS)، جست‌وجوی حریصانه (Greedy Search)، و الگوریتم A^*. هر الگوریتم باید بر اساس تعریف استاندارد خود، فضای حالت بازی را کاوش کرده و در صورت یافتن مسیر تا هدف، فهرستی از حرکات را بازگرداند. در بخش مربوط به A^* و Greedy لازم است از تابع ابتکاری استفاده کنید تا بتوانید میزان «خوش‌بینی» نسبت به فاصله‌ی باقیمانده تا هدف را بسنجید. طراحی و ارزیابی این هیوریستیک بخش مهمی از پروژه است.

در طراحی بخش هوش مصنوعی، باید تمام قوانین بازی شامل هزینه‌های حرکت، اثر سیب و سم و شرایط بن‌بست را در نظر بگیرید. یعنی در هر حرکت باید تعیین شود که آیا حرکت مجاز است یا خیر، هزینه‌ی آن چقدر است و در صورت فعال بودن سیب یا سم، هزینه چگونه تغییر می‌کند. همچنین باید دقت کنید که در فرآیند جست‌وجو حالت‌هایی که از نظر منطق بازی غیرقابل‌حل هستند (مانند جعبه‌ی گیرکرده در گوشه‌ی بدون هدف) شناسایی و گسترش داده نشوند.

در مجموع، رابط گرافیکی آماده است و شما فقط باید مغز تصمیم‌گیرنده‌ی بازی را بسازید. پیاده‌سازی درست و بهینه‌ی الگوریتم‌های جست‌وجو و طراحی هوشمندانه‌ی تابع Heuristic، کلید موفقیت در این پروژه است.



تصویری از بازی

گزارش کار و تحلیل نتایج

در پایان پروژه، هر دانشجو باید گزارشی از کار خود تهیه و ارسال کند. هدف از این گزارش، توضیح روند پیاده‌سازی، تحلیل رفتار الگوریتم‌ها و بررسی تجربی نتایج به‌دست‌آمده بر روی مراحل مختلف بازی است. گزارش باید به‌صورت منظم، فنی و با استدلال نوشته شود و شامل بخش‌های زیر باشد.

در ابتدا لازم است توضیح دهید که هر قسمت از پروژه را چگونه پیاده‌سازی کرده‌اید. برای مثال، باید مشخص کنید که ساختار داده‌ای مورد استفاده برای نمایش حالت‌ها چگونه طراحی شده است، تابع تولید جانشین‌ها (successors) چه اطلاعاتی برمی‌گرداند، تابع هزینه به چه صورت محاسبه می‌شود و در پیاده‌سازی هر الگوریتم از چه ساختارهایی مانند صف، پشته یا صف اولویت‌دار استفاده کرده‌اید. علاوه بر این، باید به نحوه‌ی مدیریت اثر سیب و سم در منطق بازی اشاره کنید و توضیح دهید که چگونه تغییر هزینه‌ها بر تصمیم‌گیری الگوریتم‌ها اثر می‌گذارد. هدف این بخش، نشان دادن درک دقیق شما از نحوه‌ی کار الگوریتم‌ها و نحوه‌ی مدل‌سازی مسئله است.

در ادامه، لازم است درباره‌ی تابع Heuristic که در الگوریتم‌های Greedy و A^* استفاده کرده‌اید، توضیح دقیقی ارائه دهید. باید بیان کنید که چرا این هیوریستیک را انتخاب کرده‌اید، چه ویژگی‌هایی دارد، آیا خوش‌بینانه (admissible) است یا خیر و اگر از چند هیوریستیک استفاده کرده‌اید، مقایسه کنید کدام‌یک عملکرد بهتری دارد و چرا.

در بخش بعد، باید تمام الگوریتم‌ها (DFS، UCS، BFS، Greedy و A^*) را بر روی تمامی لول‌های داده‌شده اجرا کنید و نتایج آن‌ها را گزارش دهید. پس از ارائه‌ی نتایج عددی، لازم است این نتایج را تحلیل و تفسیر کنید. برای مثال، توضیح دهید چرا در لول‌های بزرگ‌تر، اختلاف زمان اجرای الگوریتم‌ها بیشتر می‌شود. در پایان گزارش، از شما خواسته می‌شود که رفتار غیرمنتظره‌ی الگوریتم‌ها را توضیح دهید. برای مثال، ممکن است در اجرای بعضی از الگوریتم‌ها مشاهده کنید که بازیکن در مسیر خود سم را هم می‌خورد، در حالی که در ظاهر این کار باعث افزایش هزینه می‌شود.

در مجموع، گزارش کار باید تصویری روشن از فرآیند تفکر، طراحی، آزمایش و تحلیل شما ارائه دهد. صرفاً ذکر نتایج عددی کافی نیست.