نام مدرس: دکتر محمدی **دستیاران آموزشی مرتبط: زینب باقیان، کسری شریعتی** **مهلت تحویل:** 19 آبان	**تمرین عملی سری دوم** **درس هوش مصنوعی**	

بازی نجات گروگان

در این تمرین، شما یک بازی دو بعدی طراحی خواهید کرد که در آن باید یک گروگان را نجات دهید و در عین حال از موانع عبور کنید. با استفاده از یک الگوریتم جستجوی محلی، بازیکن به سمت گروگان حرکت می‌کند و همچنین مکانیزم‌هایی برای جلوگیری از گیر کردن در حلقه‌ها یا برخورد با موانع را پیاده‌سازی می‌کند.

نکته: در این بازی، از کتابخانه Pygame برای ایجاد رابط گرافیکی استفاده می‌شود و از یک سری تصاویر برای نمایش بازیکن، گروگان و موانع بهره گرفته شده است.

پیاده‌سازی فعلی

در اینجا، پیاده‌سازی اولیه بازی شامل موارد زیر است:

1. **راه‌اندازی بازی:**
   با استفاده از کتابخانه Pygame، پنجره‌ای برای بازی ایجاد می‌شود.
   ابعاد صفحه نمایش و اندازه خانه‌های شبکه تعریف شده‌اند.
   با اجرای تابع pygame.display.set_mode پنجره بازی ایجاد می‌شود و در ادامه عنوان بازی تنظیم می‌شود.
   تصاویر بازیکن، گروگان و موانع بارگذاری و به اندازه شبکه تغییر داده می‌شوند تا به درستی در خانه‌های شبکه قرار گیرند.

2. **تولید موانع:**

   با استفاده از تابع generate_obstacles، تعدادی مانع به صورت تصادفی در محیط بازی قرار داده می‌شود.

   موقعیت‌های موانع به شکلی انتخاب می‌شوند که همدیگر را نپوشانند و تداخل نداشته باشند.

   به هر مانع یک تصویر تصادفی از لیست تصاویر موانع اختصاص داده می‌شود.

   موانع در هر خانه‌ای از شبکه که به صورت تصادفی انتخاب شده‌اند، نمایش داده می‌شوند و کاربر با آنها در تعامل است.

3. **شروع بازی جدید**

   در هر بار شروع یک بازی جدید، تابع start_new_game اجرا می‌شود.

نام مدرس: دکتر محمدی **دستیاران آموزشی مرتبط: زینب باقیان، کسری شریعتی** **مهلت تحویل**: 19 آبان	**تمرین عملی سری دوم** **درس هوش مصنوعی**	

موقعیت بازیکن و گروگان به صورت تصادفی تولید می‌شود و موانع جدیدی در محیط بازی ایجاد می‌شوند.

فاصله بازیکن و گروگان باید بیشتر از 8 واحد باشد تا بازیکن فرصت کافی برای رسیدن به گروگان داشته باشد. (می‌توانید این مقدار را تغییر دهید تا بهتر بتوانید عملکرد کد خود را بررسی کنید).

موقعیت‌های اخیر بازیکن در یک لیست ذخیره می‌شوند تا برای تشخیص گیر کردن در حلقه‌ها استفاده شود.

4. **نمایش پیروزی و شروع دوباره:**

اگر بازیکن به موقعیت گروگان برسد، پیغام پیروزی نمایش داده می‌شود و یک افکت بصری (فلش سبز رنگ) برای چند لحظه بر روی صفحه نمایش داده می‌شود تا نشان دهد که گروگان نجات یافته است.
پس از نجات گروگان، بازی متوقف می‌شود و با کلیک روی دکمه‌ای که در صفحه نشان داده می‌شود، بازی جدید شروع خواهد شد.

5. **تکرار حلقه اصلی بازی**

بازی به طور مداوم در یک حلقه اجرا می‌شود تا زمانی که کاربر پنجره بازی را ببندد.
در هر فریم بازی:
صفحه بازی مجدداً رسم می‌شود.
موقعیت جدید بازیکن محاسبه می‌شود.
وضعیت پیروزی یا شکست بررسی می‌شود.
نرخ به‌روزرسانی بازی با تابع clock.tick کنترل می‌شود که سرعت اجرای بازی را تنظیم می‌کند (در اینجا 5 فریم بر ثانیه برای عملکرد روان‌تر).

پیاده‌سازی مفاهیم اصلی

حالا نوبت شماست که بازی را با پیاده‌سازی ویژگی‌های زیر بهبود ببخشید. در این تمرین سه الگوریتم مختلف برای جستجوی مسیر با هم مقایسه خواهند شد و به تدریج بهبودهایی به آنها اضافه خواهد شد.

1. **تپه نوردی (Hill Climbing)**

نام مدرس: دکتر محمدی **دستیاران آموزشی مرتبط: زینب باقیان، کسری شریعتی** **مهلت تحویل:** 19 آبان	**تمرین عملی سری دوم** **درس هوش مصنوعی**	

ابتدا یک نسخه ساده از الگوریتم تپه نوردی را پیاده‌سازی کنید. در این نسخه، بازیکن به طور مستقیم به سمت گروگان حرکت می‌کند و از موانع دوری می‌کند. سپس مراحل زیر را اضافه کنید:
نسخه ساده: در این مرحله بازیکن نزدیک‌ترین حرکت به سمت گروگان را انتخاب می‌کند و هر بار به موقعیت همسایه‌ای که فاصله کمتری با گروگان دارد حرکت می‌کند.
اضافه کردن مکانیزم حرکت تصادفی در بن‌بست: در این بخش، مکانیزمی را پیاده‌سازی کنید که اگر بازیکن در بن‌بست قرار گرفت (هیچ حرکتی که او را به سمت گروگان نزدیک‌تر کند وجود نداشت)، یک حرکت تصادفی معتبر انجام دهد. این کار به جلوگیری از گیر کردن در شرایط بدون پیشرفت کمک می‌کند.
تشخیص و جلوگیری از گیر افتادن در لوپ: در این مرحله، بازیکن می‌تواند زمانی که در حال تکرار مسیرهای قبلی خود است (یعنی در یک لوپ گیر کرده است)، این شرایط را تشخیص داده و به جای ادامه مسیر، یک حرکت تصادفی معتبر انجام دهد تا از لوپ خارج شود. برای این منظور، شما باید موقعیت‌های اخیر بازیکن را ذخیره کنید و در صورت تکرار شدن بیش از حد موقعیت‌ها، یک حرکت تصادفی را اعمال کنید.

2. Simulated Annealing

ابتدا یک نسخه ساده از الگوریتم Simulated Annealing را پیاده‌سازی کنید که بر اساس دما، حرکت‌های مختلفی را می‌پذیرد و به تدریج به سمت بهینه حرکت می‌کند. سپس مراحل زیر را به آن اضافه کنید:
نسخه ساده: در این نسخه، بازیکن می‌تواند حتی حرکت‌هایی که او را از گروگان دورتر می‌کند را با احتمالی که به دما وابسته است، بپذیرد. دما با گذر زمان کاهش می‌یابد، و به این ترتیب در ابتدا حرکت‌های بیشتری را می‌پذیرد و در نهایت به سمت بهینه محلی حرکت می‌کند.
اضافه کردن مکانیزم حرکت تصادفی در بن‌بست: در این مرحله، مکانیزمی اضافه کنید که در صورت برخورد با بن‌بست (یعنی زمانی که هیچ حرکت معتبری که بازیکن را به هدف نزدیک‌تر کند وجود ندارد)، یک حرکت تصادفی انجام شود تا بازیکن بتواند از بن‌بست خارج شود.البته باید توجه کنید که نیاز به حرکت تصادفی به شکل سنتی‌ای که در تپه نوردی وجود دارد، کمتر احساس می‌شود. دلیلش این است که در Simulated Annealing ، الگوریتم به طور طبیعی می‌تواند در طول زمان حتی حرکت‌هایی که بدتر از وضعیت فعلی هستند را با احتمال تصادفی و متناسب با دما بپذیرد. این ویژگی Simulated Annealing ، به آن اجازه می‌دهد از بهینه‌های محلی عبور کند و به طور طبیعی نیازی به مکانیزم حرکت تصادفی برای خروج از بن‌بست ندارد. اما حرکت تصادفی زمانی که دما بسیار پایین شده و الگوریتم قفل

نام مدرس: دکتر محمدی **دستیاران آموزشی مرتبط: زینب باقیان، کسری شریعتی** **مهلت تحویل:** 19 آبان	**تمرین عملی سری دوم** **درس هوش مصنوعی**	

می‌شود و در اواخر روند Simulated Annealing ، ممکن است به کار بیاید. این مکانیزم به طور میانگین چه تاثیری روی الگوریتم میگذارد؟ وجود چنین مکانیزمی در این بازی نیاز است؟
تشخیص و جلوگیری از گیر افتادن در لوپ: مانند تپه نوردی، این مکانیزم را اضافه کنید که بازیکن اگر در یک مسیر تکراری گیر کرد (یعنی در حال تکرار موقعیت‌های قبلی خود است)، یک حرکت تصادفی انجام دهد تا از لوپ خارج شود و بازی ادامه پیدا کند. به نظر شما این استراتژی در این الگوریتم استفاده خواهد شد؟ آیا ممکن است در این الگوریتم در لوپ گیر کنیم؟

3. الگوریتم ژنتیک (Genetic Algorithm)

در این بخش، یک نسخه ساده از الگوریتم ژنتیک را پیاده‌سازی کنید. الگوریتم ژنتیک مبتنی بر تولید جمعیتی از مسیرهای مختلف است که بهترین‌ها انتخاب شده و ترکیب (Crossover) و جهش (Mutation) در آنها صورت می‌گیرد. سپس مراحل زیر را بهبود دهید:
نسخه ساده: در این نسخه، بازیکن مجموعه‌ای از مسیرها (ژن‌ها) را در جمعیت اولیه تولید می‌کند. سپس از طریق انتخاب بهترین مسیرها و ترکیب آنها، نسل‌های جدید ایجاد می‌شود تا به مرور به مسیر بهینه برسد.
اضافه کردن مکانیزم حرکت تصادفی در بن‌بست: در این مرحله، مکانیزم جهش در مسیرها را بهبود دهید تا اگر مسیری به بن‌بست خورد (یعنی بازیکن به نقطه‌ای رسید که نمی‌تواند به گروگان نزدیک‌تر شود)، جهش بیشتری در مسیر اتفاق بیفتد تا مسیر تصادفی جدیدی انتخاب شود.به نظر شما پیاده سازی چنین مکانیزمی نیاز است؟ این مکانیزم به طور میانگین باعث بهبود الگوریتم میشود؟
تشخیص و جلوگیری از گیر افتادن در لوپ: اگر در نسل‌های جدید، ژن‌ها (مسیرها) تمایل به تکرار داشتند یا بازیکن در یک مسیر لوپ گرفتار شد، با استفاده از جهش تصادفی مسیرها را تغییر دهید تا از تکرار مسیرهای ناموفق جلوگیری شود. بازیکن باید بتواند از موقعیت‌های تنگ با انتخاب یک موقعیت همسایه تصادفی که توسط مانع اشغال نشده، فرار کند. فکر میکنید این مکانیزم به طور میانگین باعث بهبود الگوریتم میشود؟

پس از پیاده سازی نتایج را باهم بررسی کرده و به سوالات زیر پاسخ دهید:

- در کدام یک از این الگوریتم‌ها احتمال گیر افتادن در بهینه محلی بیشتر است؟
- هر الگوریتم چگونه با این مشکل مقابله می‌کند؟ آیا تپه نوردی با مکانیزم تصادفی به درستی از گیر افتادن در لوپ جلوگیری می‌کند؟
- کدام الگوریتم‌ها در یک محیط پیچیده و پر از موانع عملکرد بهتری دارند؟
- کدام یک از الگوریتم‌ها سریع‌تر به یک نتیجه نهایی می‌رسد؟
- کدام الگوریتم شانس بیشتری برای پیدا کردن بهینه جهانی دارد؟

نام مدرس: دکتر محمدی **دستیاران آموزشی مرتبط: زینب باقیان، کسری شریعتی** **مهلت تحویل:** 19 آبان	**تمرین عملی سری دوم** **درس هوش مصنوعی**	

- در هر الگوریتم، چگونه تنظیمات پارامترها (مانند نرخ جهش در الگوریتم ژنتیک یا دما در Simulated Annealing بر عملکرد آن تأثیر می‌گذارد؟ میتوانید این مورد را به صورت عملی پیاده سازی کرده و نتایج را باهم مقایسه کنید.

در نهایت تعداد گام‌های لازم برای رسیدن به گروگان، زمان اجرای الگوریتم و تعداد برخورد با موانع برای 3 الگوریتم را مقایسه کنید.(در صورت نیاز میتوانید پارامتر های مربوط به ساخت بازی را نیز تغییر دهید.)