هوش مصنوعی

تمرین شماره‌ی ۱

مدرسین: **دکتر فدایی و دکتر یعقوب‌زاده**

طراحان: الهه خداوردی، علی حمزه‌پور، محمدمهدی صمدی، کسری حاجی حیدری

مهلت تحویل: دوشنبه ۲۰ اسفند ۱۴۰۳، ساعت ۲۳:۵۹

مقدمه

در این پروژه قرار است با استفاده از الگوریتم‌های جستجوی آگاهانه و ناآگاهانه که در درس هوش مصنوعی آموخته‌اید، راه حل مناسبی برای مسئله‌ای که در ادامه مطرح می‌شود بیابید و آن را پیاده‌سازی کنید.

توضیح مسئله

مایک وازوفسکی که در کارخانه هیولاها مشغول به کار است، موقعیت جدیدی به دست آورده و از این پس وظیفه دارد فریادهای جمع‌آوری‌شده را انبار کند تا بعداً از آن‌ها استفاده شود. از آنجا که کارخانه هیولاها بزرگ‌ترین کارخانه جمع‌آوری فریاد در مونستروپلیس است، نحوه انبار کردن کپسول‌های فریاد بسیار مهم است تا به بهترین نحو از ظرفیت انبار استفاده شود و دسترسی به آن‌ها در زمان استفاده مجدد نیز به بهترین حالت ممکن انجام گیرد. به این منظور، از سیستم انبارداری آشوب (Chaos) استفاده می‌شود. در این سیستم به مایک وازوفسکی گفته می‌شود که هر جعبه از کپسول‌ها را باید دقیقاً در کدام موقعیت قرار دهد. همچنین، از آنجا که عملکرد مایک در دوره کارآموزی بسیار خوب بوده، مدیر کارخانه حداکثر دو جفت پورتال در انبار قرار داده است تا مایک بتواند با استفاده از آن‌ها جعبه‌ها را راحت‌تر به مقصد برساند (با وارد شدن به هر پورتال، از پورتال دیگر در همان جهت وارد شده خارج می‌شود). برای حرکت دادن هر جعبه، مایک باید آن را در جهت دلخواه هل دهد. به دلیل جثه کوچک مایک، اگر بیش از یک جعبه پشت‌سر هم قرار بگیرند، او نمی‌تواند همه جعبه‌ها را همزمان هل دهد و در نتیجه جعبه‌ها حرکت نخواهند کرد.

در این مسئله، ما یک حالت اولیه برای موقعیت هر جعبه، مقصد آن‌ها و مایک داریم و می‌خواهیم بهینه‌ترین حالتی را پیدا کنیم که در آن مایک می‌تواند جعبه‌ها را به جایگاه‌های مناسبشان منتقل کند.

ورودی

ورودی برنامه یک نقشه است که در آن مکان هر یک از عناصر بازی مشخص شده است.

- Pi: نمایانگر پورتال i-ام است (ممکن است پورتالی وجود نداشته باشد و حداکثر 2 جفت پورتال در اختیار داریم.)
- Bi: نمایانگر جعبه i-ام است. هر جعبه iام باید به مکان Gi برود.
- W: نمایانگر دیوار است.
- H: نمایانگر موقعیت مایک است.
- نقطه: نمایانگر مکان‌های خالی است که مایک می‌تواند در آن‌ها حرکت کند.

نکته: ممکن است در بعضی از مکان‌ها دو عنصر داشته باشیم که با کاراکتر "/" از هم جدا شده‌اند؛ برای مثال G1/H نمایانگر مکانی است که جعبه 1ام باید در آن قرار بگیرد و در حال حاضر مایک در آن قرار دارد.

نمونه ورودی
W P1 H W W W W W W W G1 W W W W W W B1 W W W W G2 B2 . P1 W W W W W B3 W W W W W W G3 W W W W W W W W W W

خروجی

خروجی هر الگوریتم شامل رشته‌ای است که نشان‌دهنده‌ی دنباله‌ی حرکاتی است که مایک باید انجام دهد و تعداد استیت‌های دیده‌شده در عملیات جستجو است. اگر جوابی برای مسئله وجود نداشت در خروجی "No solution" داریم. دقت کنید که اگر برای یک مسئله چندین جواب وجود داشت، پیدا کردن یکی از جواب‌ها کافی است. برای الگوریتم‌های BFS و A*، لازم است جواب نهایی با کمترین حرکات ممکن باشد.

رشته زیر یک نمونه از خروجی مربوط به دنباله حرکات مایک است.

نمونه خروجی
LUDDUL

- U: نمایانگر حرکت به سمت بالا
- D: نمایانگر حرکت به سمت پایین
- L: نمایانگر حرکت به سمت چپ

- R: نمایانگر حرکت به سمت راست

پیاده‌سازی مسئله

در این پروژه، شما باید مسئله را با سه روش جستجوی ناآگاهانه‌ی BFS، DFS و IDS و همچنین روش جستجوی آگاهانه‌ی A* حل و پیاده‌سازی کنید. هنگام پیاده‌سازی الگوریتم‌ها به بخش سوالات نیز دقت کنید زیرا در آن‌ها ایده‌ها و راهنمایی‌هایی برای پیاده‌سازی بهینه‌تر الگوریتم‌ها وجود دارد.

برای روش A*، حداقل یک هیوریستیک مناسب معرفی و پیاده‌سازی کنید. در انتها، weighted A* را با حداقل یک مقدار α پیاده‌سازی کرده و آن‌ها را با هم مقایسه کنید.

یکی از ایده‌ها برای هیوریستیک، استفاده از فاصله منهتن است. به این صورت که تابع را به صورت جمع فاصله منهتن بازیکن تا جعبه و فاصله منهتن جعبه تا مکان نهایی در نظر بگیریم. مشکل استفاده از این هیوریستیک در این مسئله چیست؟ چگونه می‌توان با تغییر این تابع آن را بهبود داد و به هزینه واقعی نزدیک کرد؟ حداقل یک تابع هیوریستیک دیگر برای این مسئله تعریف کنید. تابع هیوریستیکی که شما معرفی می‌کنید می‌تواند نسخه بهبود یافته هیوریستیک مربوط به فاصله منهتن باشد. دقت کنید که در نهایت الگوریتم A* باید پاسخ بهینه را پیدا کند.

الگوریتم Weighted A*

این الگوریتم نسخه‌ای از الگوریتم A* است که در آن از یک وزن (w) برای تسریع فرایند استفاده می‌کنیم. تفاوت این الگوریتم با A* در این است که تابع هیوریستیک را در یک ضریب ثابت ضرب می‌کنیم. با این کار تفاوت بین دو مقدار مختلف در هیوریستیک بیشتر شده و فرایند جستجو سریع‌تر خواهد شد.

در این الگوریتم تابع هزینه به شکل زیر تعریف می‌شود:

$$f(n) = w * h(n) + g(n)$$

سعی کنید با امتحان مقادیر مختلف w، بهترین مقدار ممکن برای هیوریستیکتان را پیدا کنید.

محدودیت زمانی اجرا

برای نقشه‌های 1 تا 5 محدودیت را 0.1 ثانیه در نظر بگیرید.

	map 6	map 7	map 8	map 9	map 10
BFS	0.5	5	1	-	2
DFS	-	-	-	-	-
IDS	-	-	5	-	-

A*	0.25	2.5	0.1	-	1.5
Weighted A*	0.25	2.5	0.1	10	1.5

توجه داشته باشید که محدودیت‌ها و معیارهای ارائه‌شده بر اساس نتایج دستیاران آموزشی تعیین شده‌اند و به عنوان یک راهنما برای ارزیابی عملکرد شما در نظر گرفته می‌شوند. رعایت دقیق این محدودیت‌ها الزامی نیست، اما بهتر است آن‌ها را به صورت تقریبی در نظر بگیرید و سعی کنید پیاده‌سازی خود را تا حد امکان به این معیارها نزدیک کنید. هدف این است که با در نظر گرفتن این محدودیت‌ها، بهبودهای لازم را در کد خود اعمال کنید تا به نتایج مطلوب و نزدیک به انتظارات دست یابید. **همچنین هنگام تحویل پروژه نقشه‌های جدیدی تست خواهند شد که از نظر سختی مشابه نقشه‌های داده شده خواهند بود.**

کلاس Game

برای اینکه درگیر پیاده‌سازی جزییات بازی نشوید، فایل game.py در اختیارتان قرار می‌گیرد که در آن یک کلاس شبیه‌ساز برای بازی وجود دارد. در زیر به توضیح آن می‌پردازیم. اجباری به استفاده از این کلاس وجود ندارد.

مقداردهی اولیه

کلاس با دریافت یک رشته که در قسمت ورودی توضیح داده شد، مقداردهی می‌شود.

مدیریت وضعیت بازی

با استفاده از set_player_position و set_box_position می‌توانید موقعیت بازیکن و جعبه‌ها را تغییر دهید.

برای بررسی کردن وضعیت فعلی بازی، مثلا اینکه بررسی کنید آیا بازیکن برنده شده است یا یک سری از حرکات معتبر هستند نیز می‌توانید از is_game_won و is_solution_valid استفاده کنید.

حرکت و تعاملات

بازیکن می‌تواند در چهار جهت اصلی حرکت کند. برای حرکت دادن بازیکن می‌توانید از apply_move یا apply_moves استفاده کنید.

همانطور که پیش‌تر توضیح داده شد، اگر بازیکن به یک جعبه برخورد کند، جعبه نیز در همان جهت حرکت می‌کند. اگر بازیکن یا جعبه به یک پورتال برخورد کند، به موقعیت متناظر پورتال دیگر منتقل می‌شود.

نمایش و دسترسی اطلاعات

برای نمایش دادن نقشه فعلی بازی به صورت متنی می‌توانید از display_map استفاده کنید.

همچنین می‌توانید به اطلاعاتی مانند نقشه یا موقعیت جعبه‌ها، اهداف، پورتال‌ها و موقعیت بازیکن از طریق getterهای مربوطه دسترسی داده باشید.

نحوه ذخیره‌سازی موقعیت اشیاء

برای استفاده از توابع getter و setter در این کلاس دقت کنید که هر کدام از اشیاء به صورت زیر ذخیره می‌شوند:

- به طور کلی هر موقعیت به صورت یک tuple ذخیره می‌شود: (row_index, column_index)
- جعبه‌ها: به صورت لیستی از موقعیت‌ها ذخیره می‌شود. به این صورت که در خانه اول، موقعیت جعبه اول و در خانه دوم موقعیت خانه دوم و به همین شکل تا انتها قرار می‌گیرند.
- اهداف: دقیقا مانند جعبه‌ها ذخیره می‌شوند.
- پورتال‌ها: به صورت لیستی از جفت موقعیت‌ها ذخیره می‌شوند. به این صورت که در خانه اول لیستی از دو موقعیت پورتال اول و در خانه دوم لیستی از دو موقعیت پورتال دوم قرار می‌گیرد (اگر تعداد پورتال‌های مسئله کمتر باشد، این لیست می‌تواند یک عضو داشته باشد یا حتی خالی باشد.):

[[Portal1Pos1, Portal1Pos2], [Portal2Pos1, Portal2Pos2]]

فایل GUI

برای مشاهده بازی به صورت گرافیکی، کافی است تا فایل gui.py را اجرا کنید. دقت کنید برای اجرای این کار نیاز به نصب کتابخانه raylib دارید. برای راهنمای نصب به این سایت مراجعه کنید.
همچنین در این فایل متغیری به نام SOLVERS وجود دارد که میتوانید به آن توابعی که پیاده‌سازی کرده‌اید را اضافه کنید تا بتوانید عملکرد الگوریتم خود را مشاهده کنید.

سوالات

1. در این مسئله تعریف state را چه می‌توان در نظر گرفت؟ بدیهی ترین تعریف برای آن نقشه انبار در هر لحظه است. ایراد این تعریف چیست؟ چگونه می‌توان آن را بهبود داد؟
2. با توجه به تعریفی که برای state در نظر گرفته‌اید actionها را چگونه تعریف می‌کنید؟
3. نحوه مدل کردن مسئله شامل تعریف action، goal state، initial state و ... را به طور دقیق توضیح دهید.
4. چگونه می‌توان این مسئله را بهبود داد و فضای سرچ را بهینه کرد؟ آیا در الگوریتم‌های جستجو برای این مسئله بایستی تمام حالات ممکن که می‌توانیم در قدم بعدی حرکت کنیم را گسترش (expand) دهیم؟ این کار چه نتایجی دارد؟ توضیح دهید برای بهبود این مسئله چه اقدامات و ملاحظاتی می‌توان در نظر گرفت.
5. هر یک از الگوریتم‌های پیاده‌سازی شده را توضیح دهید و تفاوت‌ها و مزیت‌های هر یک نسبت به دیگری را قید کرده و عنوان کنید که کدام الگوریتم‌ها جواب بهینه تولید می‌کنند.
    a. مشکل الگوریتم DFS چیست؟ با این وجود چرا همچنان از این الگوریتم استفاده می‌شود؟
    b. با توجه به مزیت‌های الگوریتم‌های DFS و BFS نسبت به یکدیگر، چگونه الگوریتم IDS این دو الگوریتم را ترکیب می‌کند و به دنبال حل چه مشکلاتی است؟
6. الگوریتم‌های قید شده را پیاده‌سازی کنید و پاسخ‌های سوال 5 را با نتایج خود توجیه کنید.
7. هیوریستیک‌هایی که در بخش جستجوی آگاهانه معرفی کردید توضیح داده و آن‌ها را از نظر admissible بودن و consistent بودن بررسی کنید.
8. سپس، الگوریتم را با استفاده از تمام هیوریستیک‌هایی که معرفی کردید اجرا کرده و نتایج آن‌ها را با یکدیگر مقایسه کنید.
9. به ازای هر الگوریتم، تست‌کیس‌ها را اجرا کرده و میانگین زمان اجرای هر الگوریتم را برای هر تست‌کیس ثبت کنید (هر الگوریتم را چند بار بر روی یک تست‌کیس اجرا کنید). سپس برای هر تست‌کیس جدول زیر را در گزارش خود کامل کنید.

	تعداد stateهای دیده شده	پاسخ مسئله (خروجی الگوریتم‌ها)	زمان اجرا
BFS			
DFS			
IDS			
A*			

Weighted A*			

*** جواب این سوالات و سوالاتی که پیش‌تر عنوان شده را به صورت کامل در گزارش خود بنویسید.**

نکات پایانی

- دقت کنید که کد شما باید به نحوی زده شده باشد که نتایج قابلیت بازتولید داشته باشند.
- توضیحات مربوط به هر بخش از پروژه را بطور خلاصه و در عین حال مفید در گزارش خود ذکر کنید. از ابزارهای تحلیل داده مانند نمودارها استفاده کنید. حجم توضیحات گزارش شما هیچ گونه تاثیری در نمره نخواهد داشت و تحلیل و نمودارهای شما بیشترین ارزش را دارد.
- سعی کنید از پاسخ‌های روشن در گزارش خود استفاده کنید و اگر پیش‌فرضی در حل سوال در ذهن خود دارید، حتما در گزارش خود آن را ذکر نمایید.
- پس از مطالعه کامل و دقیق صورت پروژه، در صورت وجود هرگونه ابهام یا سوال با طراحان پروژه در ارتباط باشید.
- نتایج، گزارش و کدهای خود را در قالب یک فایل فشرده با فرمت AI_CA1_[stdNumber].zip در سامانه ایلرن بارگذاری کنید. به طور مثال AI_CA1_810102111.zip
- محتویات پوشه باید شامل فایل jupyter-notebook، خروجی html و فایل‌های مورد نیاز برای اجرای آن باشد. از نمایش درست خروجی‌های مورد نیاز در فایل html مطمئن شوید.
- توجه کنید این تمرین باید به صورت تک‌نفره انجام شود و پاسخ‌های ارائه شده باید نتیجه فعالیت فرد نویسنده باشد. در صورت مشاهده تقلب به همه افراد مشارکت‌کننده، نمره تمرین 100- و به استاد نیز گزارش می‌گردد. **همچنین نوشته شدن کدها توسط هوش مصنوعی نیز بررسی می‌شود!**

موفق باشید