هوش مصنوعی

تمرین شماره‌ی ۱

مدرسین: **دکتر فدایی و دکتر یعقوب‌زاده**

طراحان: بابک حسینی محتشم، آرتین توسلی، محمد طاها مجلسی، محمد مهدی صمدی

مهلت تحویل: پایان روز یک‌شنبه، ۴ آبان.

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

در اعماق یک هزارتوی تاریک و بی‌پایان، Pacman به دنبال غذا می‌گردد. در این هزارتو، دو نوع میوه A و B و از هر کدام چند عدد قرار دارند. Pacman که به تازگی رژیم گرفته، به توصیه دکتر نباید میوه‌های انواع مختلف را همراه هم بخورد. پس تصمیم دارد ابتدا همه میوه‌های A را بخورد و سپس شروع به خوردن میوه‌های B کند. در این مسیر روح‌هایی قرار دارند که خودشان از میوه خسته شده‌اند و قصد خوردن Pacman را دارند. هر روح تنها در راستای عمودی یا افقی و تا شعاع معینی که در این بازی 2 می‌باشد حرکت می‌کند. پس هر روح از مرکز حرکت شروع کرده و 2 خانه در راستای حرکتش می‌رود، سپس به مرکز بازگشته و 2 خانه در همان راستا و جهت مخالف می‌رود و این روند ادامه دارد. هدف نهایی Pacman خوردن تمام میوه‌ها با رعایت ترتیب گفته شده و برخورد نکردن به روح‌ها است. او که مشخص نیست تا پیدا کردن وعده غذای بعدی چقدر باید صبر کند، می‌خواهد در کوتاه‌ترین زمان و کم‌ترین تعداد حرکت این میوه‌ها را بخورد تا کارش تمام شود. حل این مسئله برایش دشوار است و از شما خواسته تا با الگوریتم‌های جستجو مسیری نشان او دهید.

ورودی

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

- P: نمایانگر موقعیت شروع Pacman است.

- H, V: نمایانگر روح‌های بازی هستند. روح‌های نوع H حرکت افقی و نوع دیگر حرکت عمودی دارند.
- W: نمایانگر دیوار است.
- A, B: نمایانگر میوه‌های بازی هستند. میوه نوع A نسبت به دیگری اولویت دارد.
- فضالی خالی: نمایانگر فضای خالی است که Pacman و روح‌ها می‌تواند در آن‌ها حرکت کنند.

نمونه ورودی
WWWWWWW WB PW W W W V W W W WA AW WWWWWWW

خروجی

خروجی هر الگوریتم شامل لیستی از دنباله‌ی حرکاتی است که Pacman باید انجام دهد به همراه اطلاعات موجودیت‌های بازی. برای اطلاعات بیشتر به تابع PacmanGame.get_info مراجعه کنید. دقت کنید که اگر برای یک مسئله چندین جواب بهینه وجود داشت، پیدا کردن یکی از آنان کافی است.

لیست زیر یک نمونه از خروجی مربوط به دنباله حرکات Pacman برای ورودی نمونه بالا می‌باشد.

نمونه خروجی
['L', 'D', 'R', 'D', 'D', 'D', 'L', 'L', 'L', 'L', 'U', 'U', 'U', 'U']

- 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 دو heuristic مناسب معرفی و پیاده‌سازی کنید. دقت کنید که در نهایت الگوریتم A* باید پاسخ بهینه را پیدا کند پس به admissible و consistent بودن heuristic دقت کنید. در انتها، weighted A* را با حداقل دو مقدار α پیاده‌سازی کرده و آن‌ها را با هم و حالت بدون وزن مقایسه کنید.

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 خروجی تابع heuristic را در w ضرب می‌کنیم و بدین صورت وزن هزینه تخمین‌زده شده بیشتر از هزینه ابتدا تا الان شده و فرایند جستجو حریصانه و سریع‌تر خواهد شد.

در این الگوریتم تابع هزینه به شکل زیر تعریف می‌شود:

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

محدودیت زمانی اجرا

در جدول زیر زمان تقریبی اجرای هر الگوریتم روی هر نقشه داده شده است. خانه‌هایی که با '-' مشخص شده‌اند به معنای این است که نیازی نمی‌باشد برای آن تست در زمان کم به جواب برسید.

	map 1	map 2	map 3	map 4	map 5
BFS	0	0	1	140	15
DFS	0	0	-	-	-
IDS	25	0	-	-	-
A*	0	0	0	100	5
Weighted A*	0	0	0	50	1

	map 6	map 7	map 8	map 9	map 10
BFS	7	5	105	-	-
DFS	-	-	-	-	-
IDS	-	-	-	-	-
A*	6	8	85	190	150
Weighted A*	6	6	90	150	90

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.**

توضیحات کد

برای اینکه درگیر پیاده‌سازی جزییات بازی نشوید، فایل‌هایی در اختیارتان قرار گرفته است که توضیحات لازم در هر فایل کامنت شده است. در زیر به بررسی کلی عملکرد آن‌ها می‌پردازیم. توجه کنید که اجباری به استفاده از این کدها وجود ندارد.

فایل Config:

در ابتدای این فایل توابع solver خود را به SOLVERS اضافه کنید. همچنین برای جلوگیری از معطلی بیش از حد در هنگام اجرای الگوریتم‌ها time limit ست می‌شود که آن را برای هر الگوریتم در این فایل ست کنید.

کلاس MapLoader:

کلاس با دریافت یک مسیر فایل نقشه بازی، اطلاعاتی را بازمی‌گرداند که شامل موقعیت دیوارها، بازیکن، آبجکت روح و میوه‌ها است. نیاز به انجام کاری برای آن نیست.

کلاس PacmanGame:

این کلاس اطلاعات بازی را در هر لحظه نگه می‌دارد. توضیح توابع آن در فایل کد نوشته شده است و شما باید بخشی از آن را کامل کنید که شامل تعریف state بازی و پیدا کردن state های مجاور در گراف جستجو است.

بخش Solvers:

شما در این بخش باید الگوریتم‌های خواسته شده را پیاده‌سازی کنید. در هر تابع solver پارامتر time limit تعریف شده که برای جلوگیری از اجرای طولانی‌مدت الگوریتم است. برای پیاده‌سازی آن در شروع الگوریتم تایم را ثبت کرده و در هر بار اجرای حلقه زمان گذشته را چک کنید. توجه کنید که برخی نقشه‌ها طبق جدول زمان زیادی ممکن است بگیرند، پس مقدار time limit را بسته به نیاز عوض کنید.

فایل Tester:

این فایل تمام الگوریتم‌ها را به روی هر مپ اجرا کرده و نتیجه را نشان می‌دهد. از آن برای بررسی عملکرد الگوریتم‌های خود استفاده کنید.

بخش گرافیک:

در این پروژه از کتابخانه pygame استفاده شده است که میتوانید از این لینک نصب کنید. برای اجرا و مشاهده بازی به صورت گرافیکی فقط کافیست فایل `main.py` را ران کنید.

سوالات

1. در این مسئله تعریف state را چه می‌توان در نظر گرفت؟ بدیهی ترین تعریف برای آن نقشه بازی در هر لحظه است. ایراد این تعریف چیست؟ چگونه می‌توان آن را رفع کرد؟
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 هرس 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. سپس تفاوت‌ها و مزیت‌های هر یک نسبت به دیگری (مثلا مدت زمان جستجو، تعداد state های دیده شده و میزان حافظه مصرفی) را قید کرده و عنوان کنید که کدام الگوریتم‌ها جواب بهینه تولید می‌کنند و چرا.
    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. یک ایده ساده برای heuristic، استفاده از مجموع فاصله منهتن Pacman از تمام خوراکی‌ها است. مشکل استفاده از این heuristic چیست؟ چگونه می‌توان آن را اصلاح کرد؟
8. توابع heuristic که در بخش جستجوی آگاهانه معرفی کردید توضیح داده و آن‌ها را از نظر admissible و consistent بودن بررسی کنید.
9. سپس، الگوریتم را با استفاده از تمام heuristic هایی که معرفی کردید اجرا کرده و نتایج آن‌ها را با یکدیگر مقایسه کنید.
10. در مسئله جستجوی ما، برای heuristic آیا admissible بودن کافی‌ست یا حتما باید consistent هم باشد؟
11. در میان تمام heuristic هایی که شرط سوال 10 را برقرار می‌کنند، کدام یک به سرعت الگوریتم جستجو کمک بیشتری می‌کند و باعث می‌شود تعداد گره‌های دیده‌شده به شدت کاهش یابد.
12. به ازای هر الگوریتم، طبق جدول زمان‌های تقریبی، نقشه‌ها را اجرا کرده و میانگین زمان اجرای هر الگوریتم را برای هر تست‌کیس ثبت کنید. سپس برای هر تست‌کیس Dataframe خروجی تابع `run_test` فایل `tester` را ذخیره کنید.

*** 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. حجم توضیحات گزارش شما هیچ گونه تاثیری در نمره نخواهد داشت و تحلیل 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.
- فایل‌های خود را در قالب یک فایل فشرده با فرمت زیر در سامانه ایلرن بارگذاری کنید. به طور مثال AI_CA1_[stdNum].zip؟ AI_CA1_810102123.zip.
- محتویات پوشه باید شامل گزارش و کدهای شما باشد.
- 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.