به نام خدا

درس هوش مصنوعی و سیستم‌های خبره

تمرین سری دوم

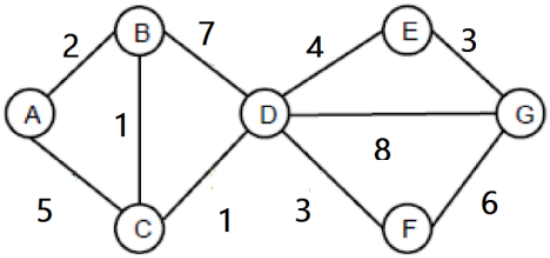
طراحان:
حوریه سبزواری، الناز رضایی

مدرس درس:
جناب آقای دکتر محمدی

مهلت ارسال: ۱۴۰۱/۰۷/۲۳

بخش تئوری

۱. گراف زیر را در نظر بگیرید. گره A گره شروع و G گره پایانی است.



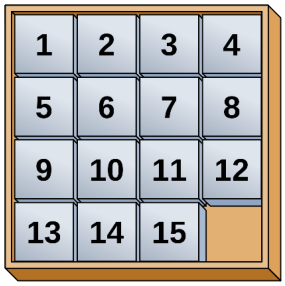
Node	h1	h2
A	9.5	10
B	9	12
C	8	10
D	7	8
E	1.5	1
F	4	4.5
G	0	0

- مشخص کنید هر کدام از توابع heuristic آیا consistent هستند یا خیر.
- ترتیب گره هاي expand شده و مسیر پیدا شده در روش A^* جستجوي را مشخص کنید.

۲. درستی یا نادرستی هر یک از عبارات زیر را با ذکر دلیل مشخص کنید.

- بازی سودوکو یک بازی Stochastic است.
- جستجوی DFS پیچدگی فضایی کمتری نسبت به جستجوی BFS دارد.
- اگر یک تابع heuristic ،ویژگی admissible بودن را داشته باشد می توان گفت که آن تابع consistent است.
- اگر یک تابع heuristic ،ویژگی consistent بودن را داشته باشد می توان گفت که آن تابع admissible است.

۳. در پازل زیر در ابتدا اعداد در جای خود قرار ندارند. بعد از حل شدن پازل، هر عدد باید مانند شکل زیر در جای خود قرار بگیرند. برای روش Informed search توضیح دهید که آیا با استفاده از این روش می‌توان این پازل را حل کرد یا نه؟ در صورت پاسخ مثبت، پیچیدگی محاسباتی و جزئیات آن ارائه دهید.



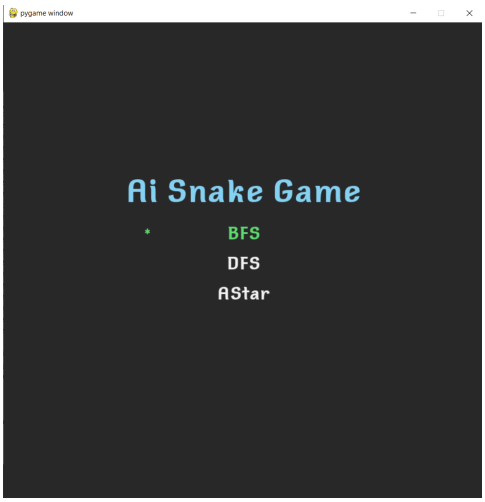
بخش عملی:

۱. در این سوال قصد داریم تا مار موجود در بازی snake را با استفاده از الگوریتم A^* به fruit (مربع‌های قرمز) برسانیم. پیاده‌سازی این تابع (ASTAR) به عهده شماست. بنابراین تنها فایلی که باید تغییر دهید، A_STAR.py هست. همچنین برای تغییر این فایل می‌توانید از توابع موجود در Algorithm.py استفاده کنید. برای نصب پکیج‌های لازم، دستورات زیر را اجرا کنید.

```
pip install pygame
pip install numpy
```

برای دیدن محیط و اجرای بازی، یکی از دستورات زیر را اجرا کنید.

```
python Main.py
python3 Main.py
```



تصویر محیط بازی

منظور از snake در کدهای داده شده، آرایه‌ای از مختصات بدن مار است که index صفر آن مربوط به سر مار می‌باشد. همچنین مار به هر خانه‌ی قرمزی که می‌رسد، یک واحد به طولش اضافه می‌شود.
برای نگهداری هایnode بازدید شده و مجموعه‌ای که از بین آن یک node برای گسترش انتخاب می‌شود، از self.explored_set و self.frontier استفاده کنید.
برای بدست آوردن state اولیه و state نهایی می‌توانید از تابع
self.get_initstate_and_goalstate() از فایل algorithm.py استفاده کنید. برای پیدا کردن همسایه‌های یک خانه می‌توانید از تابع self.get_neighbors() استفاده کنید.

برای چک کردن اینکه آیا یک خانه روی بدن مار قرار دارد یا خیر از تابع self.inside_body() استفاده کنید. همچنین برای چک کردن خارج نشدن از صفحه‌ی بازی از تابع self.outside_boundary() استفاده کنید. خروجی این دو تابع بصورت True و False هستند.

خروجی تابع run_algorithm() در فایل A_STAR.py مسیری است که توسط آن الگوریتم از state اولیه به state نهایی می‌رسیم. برای این کار می‌توانید در هر بار بررسی هر node، آن را با state نهایی مقایسه کرده و در صورت یکی بودن، از تابع self.get_path() استفاده کرده و مسیر را برگردانید.

```
return self.get_path(node)
```

توجه داشته باشید برای استفاده از این تابع در هر نوبت بررسی باید parent هر node مشخص شود.

هر node سه متغیر f و g و h دارد که در این الگوریتم باید استفاده شوند. g وزن هر یال است که یک در نظر بگیرید. h هیوریستیکی است که میتوانید از تابع manhattan_distance() موجود در algorithm.py استفاده کنید. f هم مجموع h و g هر node است.

قوانین:

$$f = g + h$$

۱. تمرین ها به‌صورت فردی انجام شوند و حل گروهی تمرین‌ها مجاز نیست.

۲. برای تحویل تمرین یک فایل zip شامل فایل اولیه تغییر داده شده توسط خودتان، با نام [HW2_ID_NAME] در سامانه gradescope بارگذاری کنید.