به نام خدا

درس هوش مصنوعی و سیستم‌های خبره

تمرین سری اول

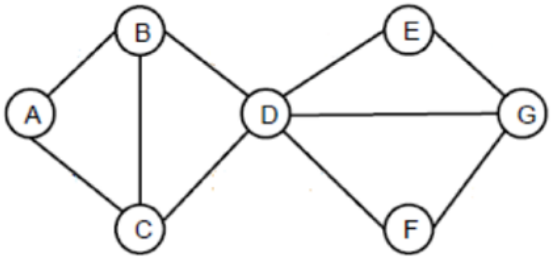
طراحان:
حوریه سبزواری، الناز رضایی

مدرس درس:
جناب آقای دکتر محمدی

مهلت ارسال: ۱۴۰۱/۰۷/۱۶

بخش تئوری

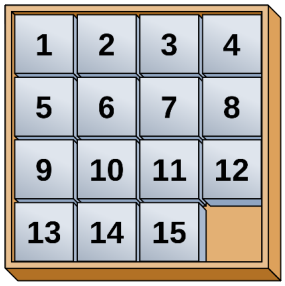
۱. گراف زیر را در نظر بگیرید. گره A گره شروع و G گره پایانی است.



ترتیب گره هاي expand شده و مسیر پیدا شده در هر کدام از روش هاي جستجوی BFS و DFS توسط جستجوی گرافی را مشخص کنید.

۲. درباره‌ی روش Iterative Deepening و کاربرد آن در هوش مصنوعی توضیح دهید.

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 هر یک از روش های BFS و DFS 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.



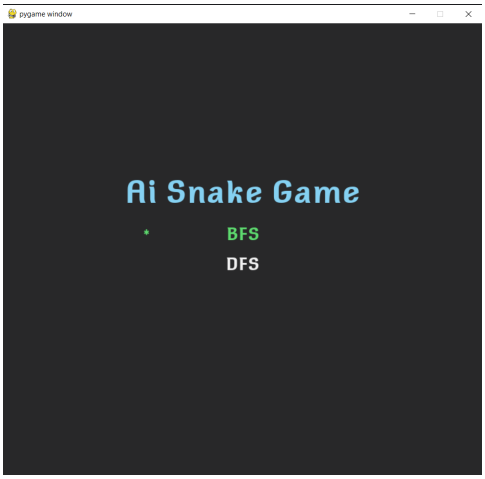
بخش عملی:

۱. در این سوال قصد داریم تا مار موجود در بازی snake را با استفاده از الگوریتم‌های BFS ، DFS را به fruit (مربع‌های قرمز) برسانیم. پیاده‌سازی این توابع (BFS DFS) به عهده شماست. بنابراین تنها فایل‌هایی که باید تغییر دهید، BFS.py و DFS.py هستند. همچنین برای تغییر این فایل‌ها 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 bfs.py و dfs.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.

قوانین:

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 [HW1_ID_NAME] در سامانه gradescope بارگذاری کنید.