نیم‌سال دوم سال ۹۷-۹۸

تمرین سری دوم: جست‌وجوی آگاهانه

لطفاً به نکات زیر توجه کنید:

- مهلت ارسال این تمرین تا ۲۸ اسفند است.
- در صورتی که به اطلاعات بیشتری نیاز دارید می‌توانید به صفحه‌ی تمرین در وب‌سایت درس مراجعه کنید.

- این تمرین شامل سوال‌های برنامه‌نویسی می‌باشد، بنابراین توجه کنید که حتماً موارد خواسته‌شده در سوال را رعایت کنید. در صورتی که به هر دلیلی سامانه‌ی داوری نتواند آن را اجرا کند مسئولیت آن تنها به عهده‌ی شماست.
- ما همواره هم‌فکری و هم‌کاری را برای حلِ تمرین‌ها به دانشجویان توصیه می‌کنیم. اما هر فرد باید تمامی سوالات را به تنهایی تمام کند و پاسخ ارسالی حتماً باید توسط خود دانش‌جو نوشته‌شده باشد. لطفاً اگر با کسی هم‌فکری کردید نام او را ذکر کنید. در صورتی که سامانه‌ی تطبیق، تقلبی را تشخیص دهد متأسفانه هیچ مسئولیتی بر عهده‌ی گروه تمرین نخواهد بود.
- لطفاً برای ارسال پاسخ‌های خود از راهنمای موجود در صفحه‌ی تمرین استفاده کنید.
- هر سوالی درباره‌ی این تمرین را می‌توانید از دستیاران حلِ تمرین بپرسید.

- آدرس گروه درس: https://groups.google.com/forum/#!forum/ai972
- صفحه تمرین: https://quera.ir/course/assignments/8388/problems

موفق باشید

سوال‌های عملی

۱. منطقه سمی (۱۰ نمره)

همانطور که می‌دانید، عامل پک‌من حتی الامکان باید از روح‌های نقشه دوری کند. یکی از راه های دوری کردن از روح‌ها، افزایش هزینه حرکت به خانه‌های مجاور با روح‌هاست. با این کار می‌توان با استفاده از الگوریتم‌های UCS و یا حتی الگوریتم‌های حریصانه از روح‌ها دوری کرد. شما در این سوال باید هزینه ورود به خانه‌های مجاور روح‌ها را بالا ببرید.

برای پاسخ به این سوال باید بدنه تابع getCostOfActions در کلاس GoalSearchProblem موجود در فایل searchAgents.py را تغییر دهید.

۲. جست‌وجو در فضای مسئله – A* (۲۰ نمره)

هدف از این سوال، پیاده‌سازی الگوریتم A* است. این الگوریتم باید مسیر بهینه برای رسیدن از حالت مبدا به حالت مقصد را پیدا کند. شما باید الگوریتم را جوری پیاده‌سازی کنید که عامل پک‌من بدون برخورد با روح‌ها به مقصد نهایی برسد. برای الگوریتم در این سوال از تابع ثابت صفر استفاده می‌کنیم تا الگوریتم مانند الگوریتم حریصانه کار کند. در نتیجه خروجی، تاثیر افزایش هزینه‌ی حرکت به خانه‌های مجاور روح‌ها در سوال اول را خواهید دید.

برای پاسخ به این سوال باید بدنه تابع aStarSearch(problem) را در فایل search.py پر کنید. خروجی تابع، دنباله ای از حرکت هاست. برای تست کد خود می‌توانید از دستورات زیر استفاده کنید:

```
$ python pacman.py -l dangMaze -p AStarGoalSearchAgent -g FixedGhost
$ python pacman.py -l bigMaze -p AStarGoalSearchAgent -g FixedGhost
```

۳. تعیین Heuristic و فرار از روح‌ها (۱۵ نمره)

یکی دیگر از راه حل‌های فرار از روح‌ها استفاده از Heuristic مناسب برای الگوریتم A* است. در این سوال عامل پک‌من باید علاوه بر اینکه با روح‌ها برخورد نکند، باید تمام غذاهای روی نقشه را نیز بخورد. برای این کار شما باید تابع Heuristic ای پیاده‌سازی کنید که با استفاده از الگوریتم A* پیاده‌سازی شده در سوال قبل بتواند بدون نیاز به افزایش هزینه‌ی خانه‌های مجاور روح‌ها، از روح ها فرار کرده و تمام غذاها را بخورد.

برای پاسخ به این سوال باید بدنه تابع foodHeuristic(state, problem) را در فایل searchAgents.py پر کنید. لازم به ذکر است که شما در این سوال حق تغییر کلاس FoodSearchProblem را ندارید. برای تست کد خود می‌توانید از دستور زیر استفاده کنید:

```
$ python pacman.py -l heavySearch -p AStarFoodSearchAgent -g FixedGhost
```

۴. حل پازل (۲۵ نمره)

در این سوال از فریم‌ورک دیگری استفاده خواهیم کرد. این فریم‌ورک برای حل پازل اعداد است. پازل به گونه ای است که در یک پازل ۹ تایی، ۸ خانه با عدد پر شده‌اند و یک خانه از پازل خالی است. حال باید با تکان دادن خانه خالی بر ترتیب خاصی از اعداد در پازل رسید. شما در این سوال باید پازل را با پیاده‌سازی الگوریتم A* در این فریمورک و مشخص کردن Heuristic مناسب برای آن، حل کنید. راهنمای استفاده از فریم‌ورک حل پازل به شکل زیر است:

برای اجرای فریم‌ورک باید دستور زیر را برای نصب وابستگی‌های فریم‌ورک اجرا کنید:

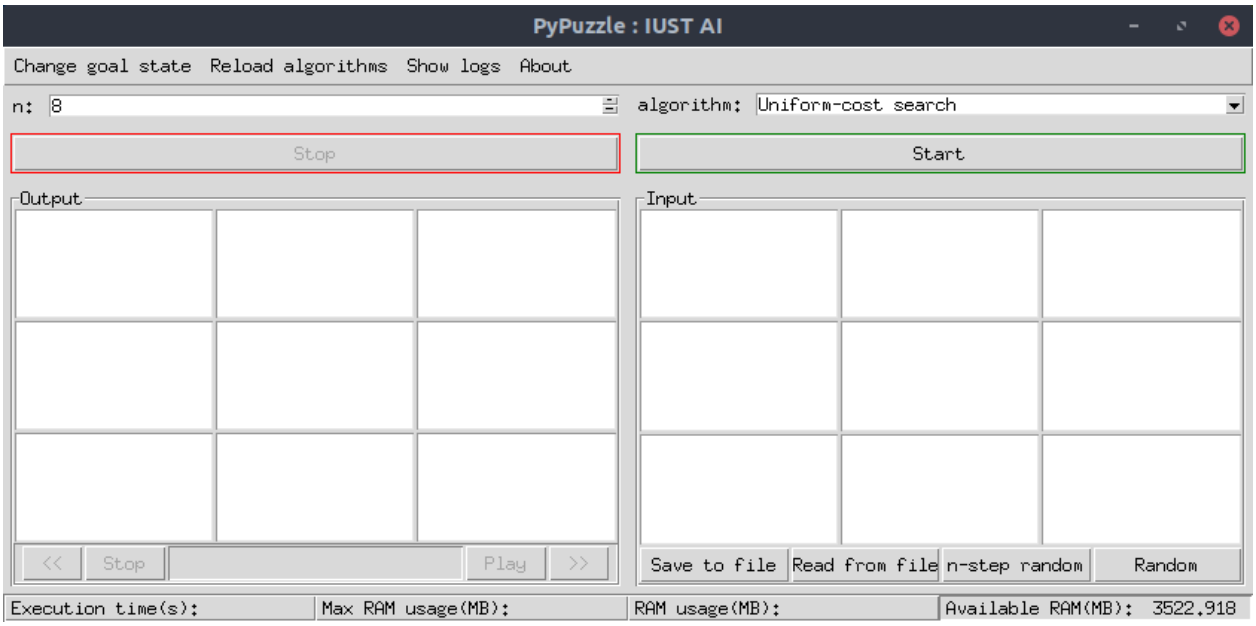
```
$ sudo apt-get install python3-tk python3-psutil
```

همانطور که مشاهده میکنید برای پیاده‌سازی راه‌حل این مسئله باید از نسخه ۳,۵ یا بالاتر پایتون استفاده کنید.

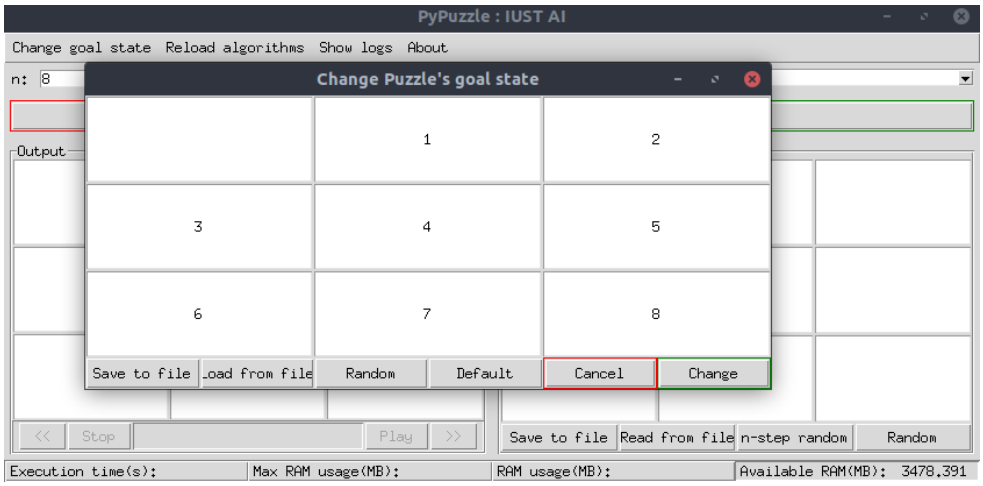
پس از نصب پیش‌نیاز های فریم‌ورک، می‌توانید با اجرای دستور زیر، در محل فایل های قرار داده شده، فریم‌ورک را اجرا کنید:

```
$ python3 pypuzzler.py
```

پس از اجرای دستور بالا با پنجره‌ای مانند عکس زیر مواجه خواهید شد:



با فشردن دکمه Change goal state در گوشه‌ی سمت چپ صفحه می‌توانید حالت نهایی و هدف مورد نظر خود را برای فریم‌ورک تعیین کنید. دقت داشته باشید که یکی از خانه ها باید خالی بماند.



پس از تعیین حالت نهایی و فشردن دکمه change، از منوی سمت راست، باید الگوریتم مورد نظر برای حل مسئله را انتخاب کنید. الگوریتم های BFS,DFS و UCS از قبل برای شما پیاده‌سازی شده‌اند. شما باید تابع مربوط به الگوریتم A* را که در ادامه توضیح داده خواهد شد تکمیل کنید.

سپس باید حالت شروع مسئله در سمت راست صفحه به فریم‌ورک به عنوان ورودی داده شود. با فشردن دکمه Start، فریم‌ورک شروع به حل مسئله می‌کند. دقت داشته باشید که از هر حالت شروع، لزوما نمی‌توان به حالت پایانی مسئله رسید. لذا باید حالت شروع مسئله به شکلی باشد که راهی برای حل مسئله وجود داشته باشد.

پس از پایان حل مسئله می‌توانید از قسمت چپ پنجره، راه حل خود را مشاهده کنید.

برای پاسخ به این سوال شما باید بدنه‌ی تابع search در فایل a-star-search.py واقع در پوشه‌ی algorithms را کامل کنید. برای اعمال Heuristic می‌توانید در همان فایل، تابع مورد نظر خود را تعریف کرده و در تابع اصلی استفاده کنید.

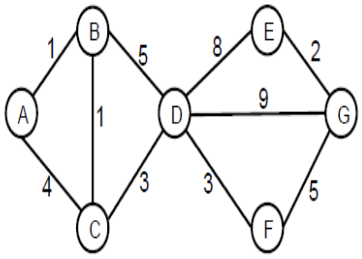
برای اینکه محسبات شما ساده‌تر شود، می‌توانید از کلاس Node موجود در فایل tree_search.py واقع در پوشه‌ی util استفاده کنید. توابع و متغیرهای مورد نیاز شما در کلاس Node به شرح زیر است:

توابع و متغیرهای کلاس Node	
is_goal(self, goal_state)	این تابع در صورت یکسان بودن حالت فعلی با حالت پایانی True و در غیر اینصورت False بازمی‌گرداند.
expand(self)	پس از صدا زدن این تابع، متغیر children آبجکت کنونی، با حالت‌های ممکن بعدی پر می‌شود. این تابع مقدار بازگشتی ندارد.
parents(self)	این تابع پدران یک حالت را باز می‌گرداند.
children	پس از صدا زدن تابع expand، لیستی از حالت‌های ممکن بعدی به این متغیر اختصاص داده می‌شود.

برای آشنایی بیشتر، می‌توانید کد توابع پیاده‌سازی شده، مانند BFS و ... را مطالعه کنید.

سوال‌های تئوری

۱. با در نظر گرفتن گراف زیر پاسخ دهید:



Node	h_1	h_2
A	9.5	10
B	9	12
C	8	10
D	7	8
E	1.5	1
F	4	4.5
G	0	0

الف) با در نظر گرفتن هیوریستیک h1 مسیر طی شده توسط الگوریتم A* را بنویسید.

ب) با در نظر گرفتن هیوریستیک h2 مسیر طی شده توسط الگوریتم A* را بنویسید.

ج) با در نظر گرفتن هیوریستیک h2 مسیر طی شده توسط الگوریتم Greedy را بنویسید.

Node	A	B	C	D	E	F	G
h_3	10	?	9	7	1.5	4.5	0

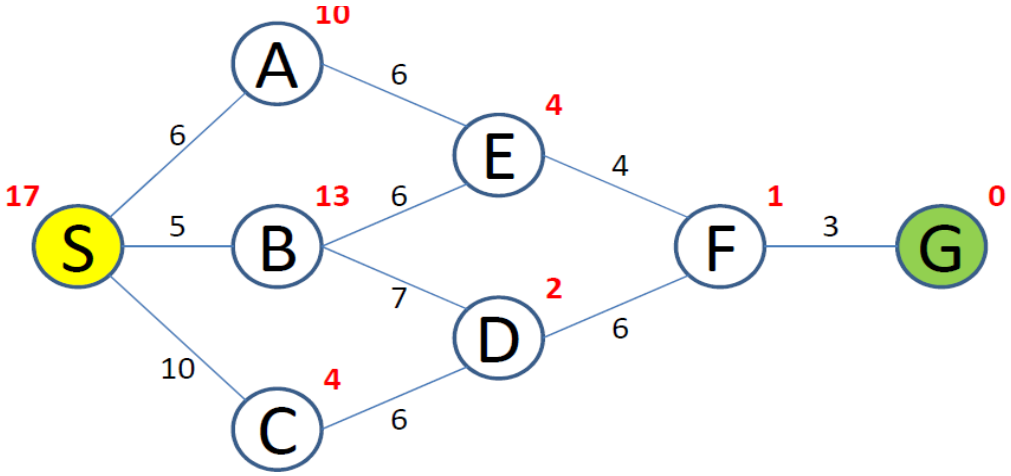
با در نظر گرفتن هیوریستیک h3 پاسخ دهید:

د) چه مقادیری از $h_3(B)$ باعث قابل قبول شدن h3 می‌شود.

و) چه مقادیری از $h_3(B)$ باعث قابل سازگار شدن h3 می‌شود.

د) چه مقادیری از $h_3(B)$ باعث گسترش node ها به ترتیب A و B و C و D و در الگوریتم A* می‌شود.

۲. گراف زیر را در نظر بگیرید:



الف) الگوریتم A* را برای این گراف نوشته و در هر مرحله وضعیت و node های در صف را مشخص کنید و درخ جست‌وجو را نیز رسم کنید.

ب) سازگاری و قابل‌قبول بودن آن‌را بررسی نمایید.

ج) با الگوریتم هزینه یکسان مسیری از مبدا به مقصد طی کنید.

۳. درخت زیر را با A* و Greedy پیمایش کنید و در هر مرحله node های داخل صف را بنویسید.