دانشکده مهندسی و علوم کامپیوتر

تمرین کامپیوتری اول

هوش مصنوعی و سیستم‌های خبره

تیم حل تمرین دکتر عبدوس – بهار ۱۴۰۴

مقدمه

در این تمرین از شما می‌خواهیم که چهار عامل مختلف را بر اساس الگوریتم‌های جستجوی BFS، Bi-Directional IDDFS، UCS و A* پیاده‌سازی کنید.

محیط مسئله به‌طور کامل پیاده‌سازی شده است و شما صرفا باید الگوریتم جستجوی ایجنت‌های آن را پیاده‌سازی کنید. جزئیات مربوطه و نحوه راه‌اندازی آن در ادامه آمده است.

شرح مسئله

عامل شما با شروع از نقطه آبی رنگ باید تلاش کند تا به نقطه سبز رنگ برسد. در این نقشه نقاط مشکی رنگ دیوار بوده و عامل نمی‌تواند از روی آن‌ها عبور بکند. حرکت عامل شما به صورت افقی، عمودی و مورب است. همچنین در نقشه تعدادی جفت تله‌پورتر(Teleporter) وجود دارد که با صرف هزینه‌ای می‌توانند شما را در محیط جابه‌جا کنند.

در الگوریتم‌های A* و UCS، حرکت مورب دارای هزینه ۱.۴، حرکات عمودی و افقی هزینه ۱ و هزینه تله‌پورت برابر مضربی رندوم از فاصله اقلیدسی بین دو سر تله‌پورتر است. در الگوریتم‌های دیگر هزینه تمامی حرکت‌ها و تله‌پورت یکسان و برابر ۱ است.

آموزش نصب و راه‌اندازی

مرحله اول – نصب پایتون ۳

آموزش قدم به قدم دانلود و نصب پایتون ۳.

مرحله دوم – نصب پکیج‌ها

پس از نصب پایتون، کافی‌ست پکیج PyGame را با یکی از دستورات زیر نصب کنید:

pip install pygame

python -m pip install pygame

مرحله سوم – اجرای برنامه

پس از پیاده‌سازی موارد خواسته‌شده که در ادامه توضیح داده خواهد شد، با اجرای فایل main.py برنامه اجرا می‌شود.

آشنایی با ساختار پروژه

توضیح فولدر و فایل‌ها

پروژه از فولدرها و فایل‌های مختلفی تشکیل شده است. مواردی که برای پیاده‌سازی و آشنایی بهتر با ساختار کد نیاز است، در ادامه توضیح داده شده است:

- implemented_agents.py 🐍: پیاده‌سازی تابع searching در کلاس عوامل مختلف در این فایل بر عهده‌ی شما و هدف این تمرین است. این توابع باید به ترتیب لیست نقاطی که مسیر را تشکیل می‌دهند و لیستی که شامل تمام نقاط بازدید شده است را برگردانند. شما میتوانید از توابع کمکی و ویژگی‌های کلاس انتزاعی که توضیح داده خواهد شد استفاده کنید. همچنین در صورت نیاز می‌توانید برای هر کلاس تابع‌های دیگری تعریف کنید و از آن‌ها استفاده کنید. به طور مثال نیاز است که برای عامل A* تابعی برای محاسبه تابع مکاشفه‌ای(heuristic) تعریف کنید تا بتوانید در تابع searching از آن برای محاسبه هزینه استفاده کنید. بعد از تکمیل کردن تابع جستجو می‌توانید عامل مورد نظر را انتخاب کرده و با اجرای فایل main.py نتیجه جستجوی عامل را مشاهده کنید.

فایل‌های زیر برای شما پیاده‌سازی شده‌اند و ارزیابی شما فقط براساس فایل بالا انجام می‌گردد.

- agent.py: کلاس انتزاعی `AbstractSearchAgent` در این فایل پیاده‌سازی شده است. کلاس عامل شما باید از این کلاس ارث‌بری کند.
- generator.py: امکان اعمال تغییرات در موانع و گیت‌های تله‌پورتر و ذخیره در یک نقشه جدید را به شما می‌دهد.
- plotting.py: شامل پیاده‌سازی نمایش برنامه با استفاده از دستورات کتابخانه‌ی pygame می‌شود.
- env.py: کلاس محیط اجرایی برنامه را شامل می‌شود.
- main.py: این فایل برای اجرای پروژه به کار می‌رود و شامل تنظیمات اجرای برنامه است.
- فولدر Maps: فایل‌های json مپ‌ها در این فولدر قرار می‌گیرند.

توضیحات تکمیلی

- main.py:

نقطه‌ی شروع عامل را با متغیر start و نقطه‌ی پایان را با متغیر goal می‌توانید تنظیم کنید. نقشه مورد نظر در پوشه Maps را با تنظیم نام فایل در متغیر map_name می‌توانید انتخاب کنید. اگر می‌خواهید مکان تله‌پورترها تصادفی باشد use_random_teleports را True و تعداد جفت‌ها را با متغیر num_pairs تنظیم کنید. اگر می‌خواهید تمامی حرکت‌ها هزینه یکسان و برابر با ۱ داشته باشند، euclidean_cost را False (برای الگوریتم‌های BFS و Bi-directional IDDFS) و در غیر این صورت True تنظیم کنید.

```python
def main():
    map_name = "default"  # Choose the map file
    use_random_teleports = False  # Change to True to use random teleports
    num_pairs = 2  # Number of random teleport gates if enabled
    FPS = 60  # Frames per second for animation

    start = (5, 5)  # Start position
    goal = (45, 25)  # Goal position
    euclidean_cost = False  # True to use Euclidean distance as cost

    environment = Env(map_name, use_random_teleports, num_pairs)
    agent = BFSAgent(start, goal, environment, euclidean_cost) # TODO: your agent here
    path, visited = agent.searching()

    plot = Plotting(start, goal, environment, FPS)
    plot.animation(path, visited, agent.COST)
```

- agent.py:

دیوارهای بازی در متغیر self.obs و تله‌پورترها در متغیر self.teleports در کلاس انتزاعی موجود در این فایل ذخیره می‌شوند. از متغیرهای PARENT و COST و VISITED می‌توانید در پیاده‌سازی عوامل خود استفاده کنید. متغیر self.NEIGHBOR_COSTS فاصله بین هر خانه و همسایه‌هایش را

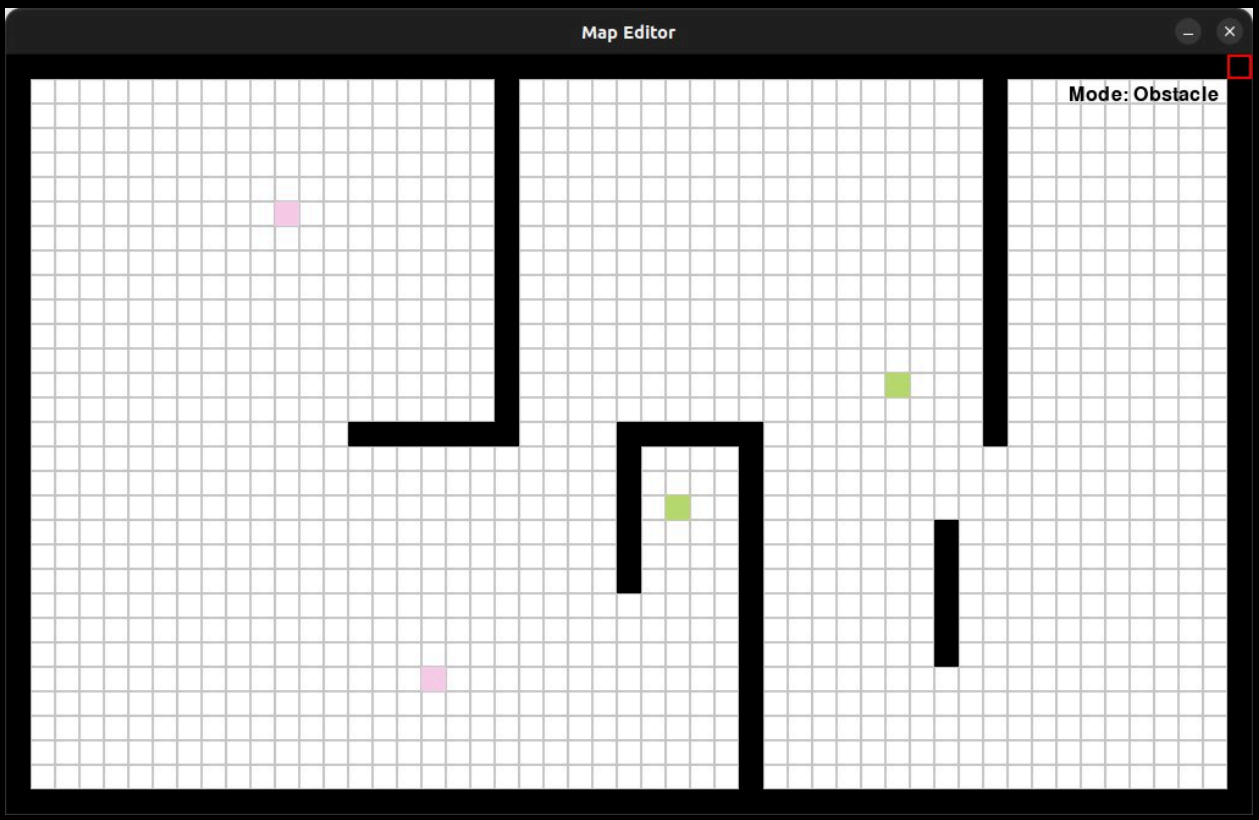
precompute و ذخیره می‌کند. با استفاده از این متغیر می‌توانید به فاصله هر دو خانه همسایه دسترسی داشته باشید.

متد get_neighbors با گرفتن یک خانه، تمام خانه‌های معتبر کناری (اعم از همسایه افقی، عمودی، مورب و تله‌پورت) را برمی‌گرداند.

متد extract_path با استفاده از لیست PARENT راه پیدا شده از مبدا به مقصد را برمی‌گرداند.

- generator.py:

با اجرای این فایل پنجره‌ای مانند زیر جهت تغییر نقشه برای شما باز می‌شود:

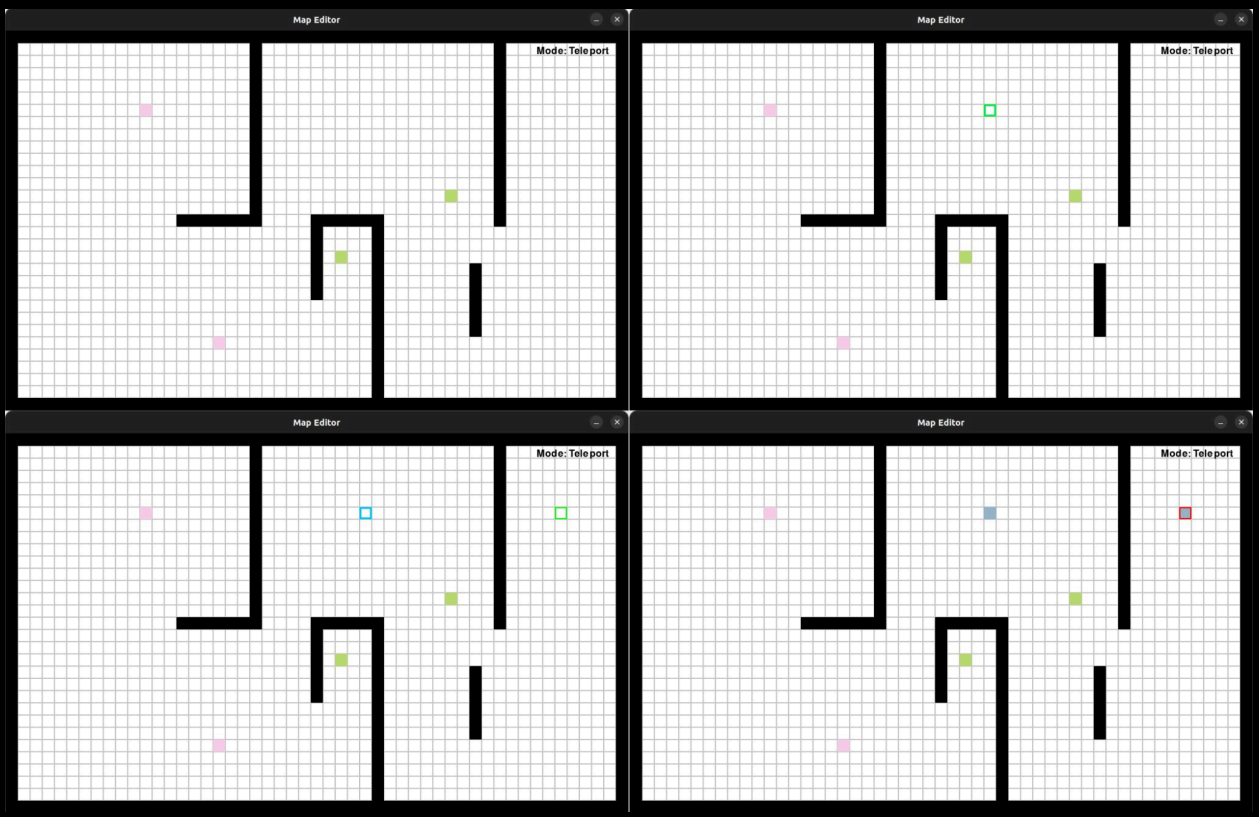


این برنامه دو حالت(Mode) دارد که در هر لحظه، در سمت راست بالای صفحه نمایش داده می‌شود: Obstacle و Teleport. با فشردن کلید t می‌توانید بین این دو حالت جابه‌جا شوید.

در حالت Obstacle، با کلیک‌کردن روی هر خانه مجاز، می‌توانید موانع را اضافه یا حذف کنید.

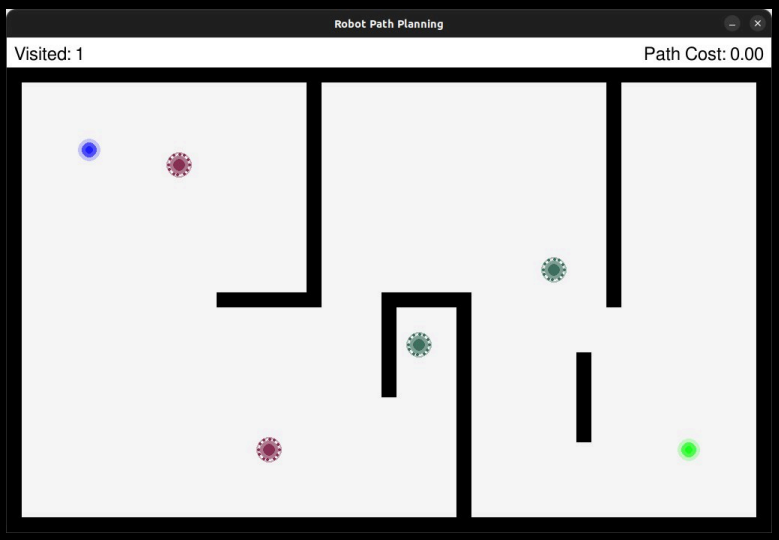
در حالت Teleport، دو حالت وجود دارد:

- برای حذف یک جفت گیت خاص، با کلیک روی هر کدام از آن‌ها، جفت مربوطه حذف می‌شود.
- برای اضافه‌کردن یک جفت گیت جدید، ابتدا گیت اول را با کلیک بر روی خانه مدنظر مشخص کنید. سپس گیت دوم را به همین روش اضافه کنید. خواهید دید که رنگ هر دوی آن‌ها به صورت رندوم انتخاب شده و نمایش داده می‌شود. برای جزئیات بیشتر به تصاویر زیر دقت کنید:

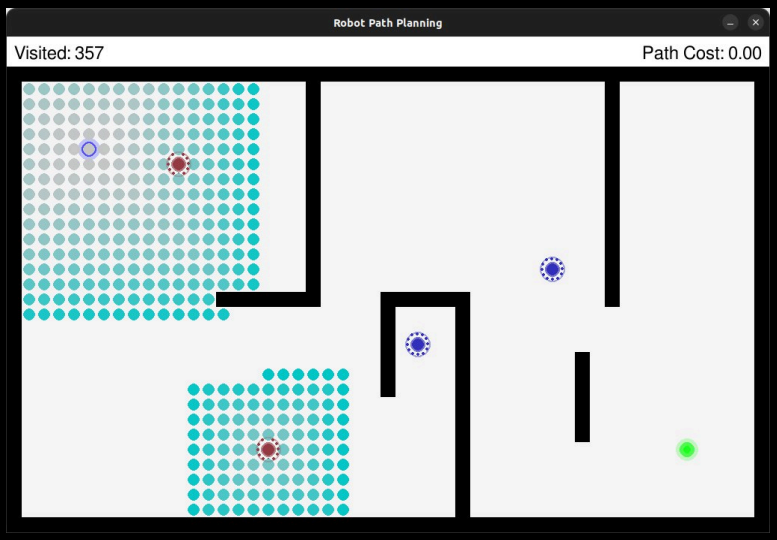


در نهایت با بستن برنامه، نقشه جدید با نامی که در map_name مشخص کرده‌اید ذخیره می‌شود.

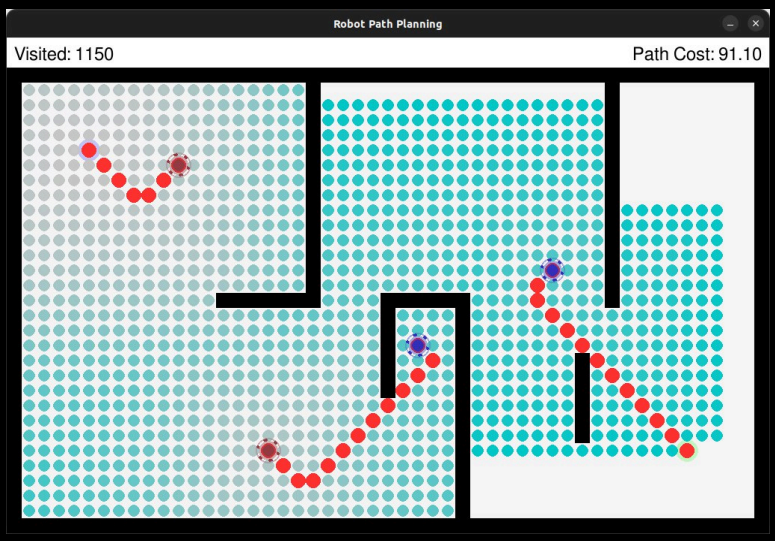
اجرا و مشاهده نتایج



پس از پیاده‌سازی مسئله، با اجرای فایل main.py پنجره زیر باز می‌شود که وضعیت اولیه محیط را نشان می‌دهد. نقاط رنگی همان جفت تله‌پورت‌ها هستند.



سپس با اجرای الگوریتم جستجو مربوطه (برای مثال الگوریتم BFS در شکل‌های روبرو) خانه‌ها به ترتیب دیده‌شدن نمایش داده می‌شوند.



با پایان روند جستجو و پیدا شدن کوتاه‌ترین مسیر، این مسیر به رنگ قرمز درآمده و هزینه مسیر پیدا شده با توجه به اقلیدسی بودن یا نبودن هزینه‌ها (تعیین‌شده توسط متغیر euclidean_cost در فایل main.py) در بالا سمت راست نشان داده می‌شود.

تحویل تمرین

پس از کامل‌کردن کد ایجنت‌ها، فایل implemented_agents را به همراه گزارشی از خروجی‌های مختلف و مقایسه آن‌ها زیپ کرده و با نامگذاری به فرمت CHW1-[StudentID]-[StudentName] در سایت کوئرا آپلود کنید. نمره تمرین شما وابسته به تحویل تمرین بوده که شیت هماهنگی برای شما قرار داده خواهد شد. نوشتن گزارش از توضیحات کد برای این تمرین اجباری نیست اما می‌تواند در روند تحویل به شما کمک کند.

معیارهای ارزیابی

علاوه بر صحت الگوریتم‌های پیاده‌سازی شده، موارد زیر بررسی خواهند شد:

- تعداد خانه‌های بازدید شده توسط الگوریتم‌ها
- زمان اجرای الگوریتم‌ها
- چک‌کردن خروجی الگوریتم A* با UCS، جهت بررسی بهینگی عملکرد A*

تسلط به کد و اجرا در زمان تحویل نمره پایانی شما را تعیین خواهد کرد. کشف هرگونه تقلب به منزله نمره صفر خواهد بود.

پاسخ به سوالات

در صورت بروز سوال یا اشکال، آن را در گروه تلگرامی درس و یا در چت خصوصی با (Farzinium@) یا (hdf25@) در میان بگذارید.