گزارش تمرین اول

این پروژه با هدف پیاده‌سازی الگوریتم‌های جستجوی مسیر در محیط‌های دو بعدی (مثل نقشه‌های شبیه‌سازی شده یک ربات) طراحی شده است. هدف اصلی یافتن مسیر بهینه از یک نقطه شروع (start) به یک نقطه هدف (goal) با استفاده از الگوریتم‌های مختلف جستجو می‌باشد. در این پروژه علاوه بر پیاده‌سازی الگوریتم‌های *BFS*، *Bidirectional Iterative Deepening Depth-First Search (BiIDDFS)*، *A** و *Uniform Cost Search (UCS)*، بخش‌های دیگری مانند *ایجاد نقشه(Generator)* ، *محیط (Environment)* و *نمایش (Plotting)* نیز پیاده‌سازی شده اند.

اجزای اصلی پروژه

1. **ماژول محیط:(Environment - Env)**
   این بخش شامل تنظیمات محیط، خواندن نقشه از فایل (مانند موانع و تله‌پورت‌ها) و یا تولید تله‌پورت‌های تصادفی در صورت نیاز می‌باشد.

2. **ماژول عامل:(Agent)**
   عامل‌ها در این پروژه مسئول جستجو در محیط هستند. عامل‌ها از کلاس پایه‌ی AbstractSearchAgent ارث‌بری می‌کنند و سپس الگوریتم‌های مختلفی مانند *BFS*, *BiIDDFS*, *A** و *UCS*را پیاده‌سازی می‌کنند.

3. **ماژول نمایش:(Plotting)**
   بخش نمایش، وظیفه رسم نقشه، موانع، تله‌پورت‌ها، و نمایش گام به گام مسیر و نقاط بازدیدشده را بر عهده دارد. این بخش انیمیشنی از فرایند جستجو تولید می‌کند.

4. **ماژول تولید نقشه:(Generator)**
   به کمک این بخش، کاربر می‌تواند نقشه و موانع را به صورت دستی اصلاح کند و در نهایت نقشه را به صورت فایل JSON ذخیره کند.

پیاده‌سازی

پیاده‌سازی الگوریتم BFS

در کلاس BFSAgentالگوریتم BFS پیاده‌سازی شده است. این الگوریتم به شیوه‌ی سطح به سطح (level by level) به جستجو می‌پردازد

توضیحات:

- از صف (queue) برای ذخیره‌ی گره‌های بعدی استفاده می‌شود.
- ابتدا نقطه شروع در صف قرار می‌گیرد.
- در هر تکرار گره از صف خارج و همسایگان آن بررسی می‌شوند.

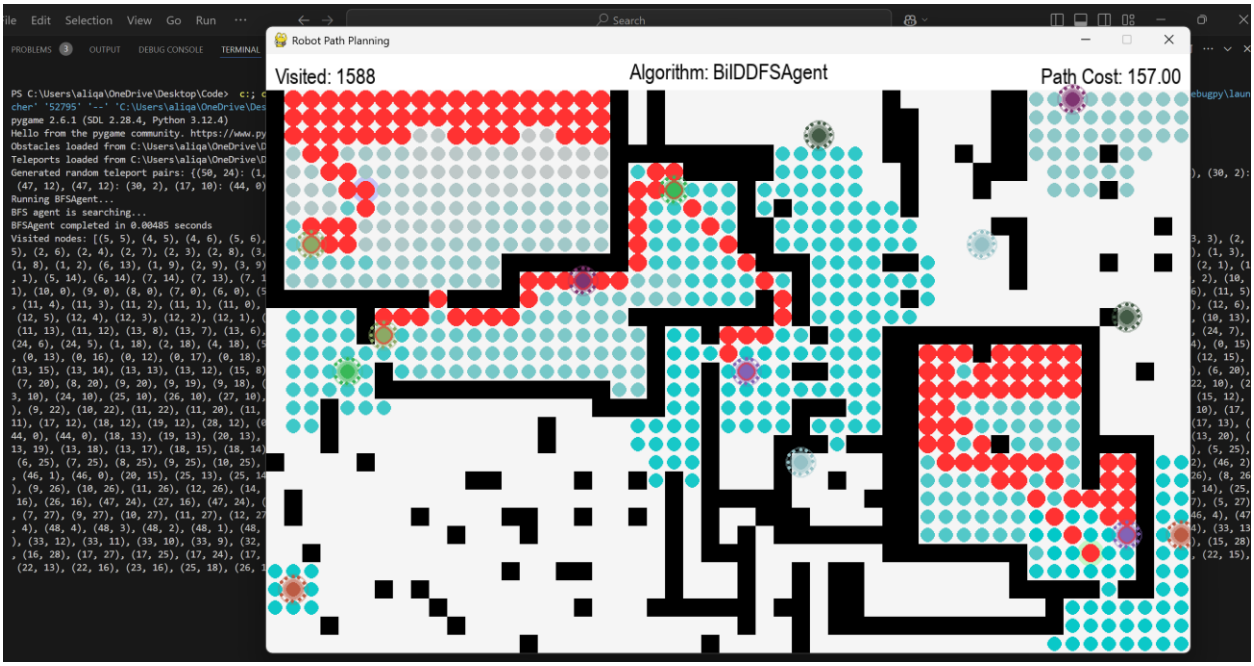
- اگر گره هدف پیدا شود، مسیر استخراج و لیست گره‌های بازدیدشده برگردانده می‌شود.

پیاده‌سازی الگوریتم BiIDDFS

کلاس BiIDDFSAgent الگوریتم **Bidirectional Iterative Deepening DFS** را پیاده‌سازی می‌کند. ایده این الگوریتم این است که از هر دو سمت (از مبدا و مقصد) جستجو انجام شود و در نهایت نقاط تلاقی پیدا شده و مسیر بازسازی شود.

توضیحات:

- ابتدا از مبدا و مقصد به صورت مجزا به کمک DFS با محدودیت عمق جستجو انجام می‌شود.
- در هر عمق یک مجموعه از گره‌های بازدید شده استخراج شده و در نهایت نقاط تلاقی بررسی می‌شوند.
- پس از یافتن تلاقی، مسیر از مبدا تا تلاقی و از تلاقی تا مقصد ترکیب می‌شود.
- هزینه هر گام به صورت ثابت (۱) محاسبه می‌شود.



پیاده‌سازی الگوریتم A*

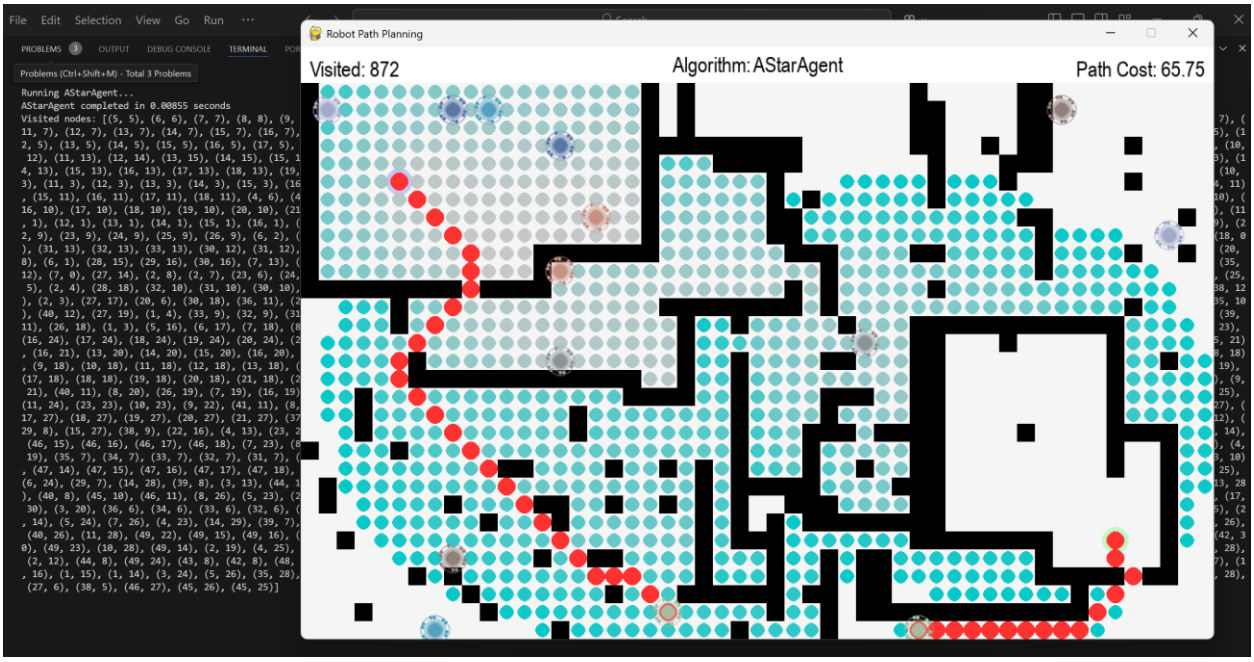
کلاس AStarAgentاز الگوریتم A* استفاده می‌کند. این الگوریتم با ترکیب هزینه فعلی مسیر (g-cost) و تابع تخمین (h-cost)که در اینجا از فاصله‌ی اکتایل استفاده می‌شود، به جستجو می‌پردازد.

توضیحات:

- از یک صف اولویت (heapq) برای مدیریت گره‌های بازدید نشده استفاده می‌شود.
- اولویت گره‌ها بر اساس مجموع هزینه طی شده و تخمین باقی‌مانده محاسبه می‌شود.

توضیحات هیوریستیک

هوریستیک استفاده‌شده در این الگوریتم بر اساس فاصله اکتایل طراحی شده است. ابتدا اختلاف مطلق مختصات افقی (dx) و عمودی (dy) محاسبه می‌شود. حرکت مورب (قطری) هزینه‌ای برابر با √2 و حرکت مستقیم (افقی یا عمودی) هزینه‌ای برابر با 1 دارد. برای تخمین هزینه، از فرمول max(dx, dy) + (√2 - 1) * min(dx, dy) استفاده شده است. این تخمین همواره کمتر یا مساوی هزینه واقعی است و موجب هدایت دقیق‌تر الگوریتم می‌شود.

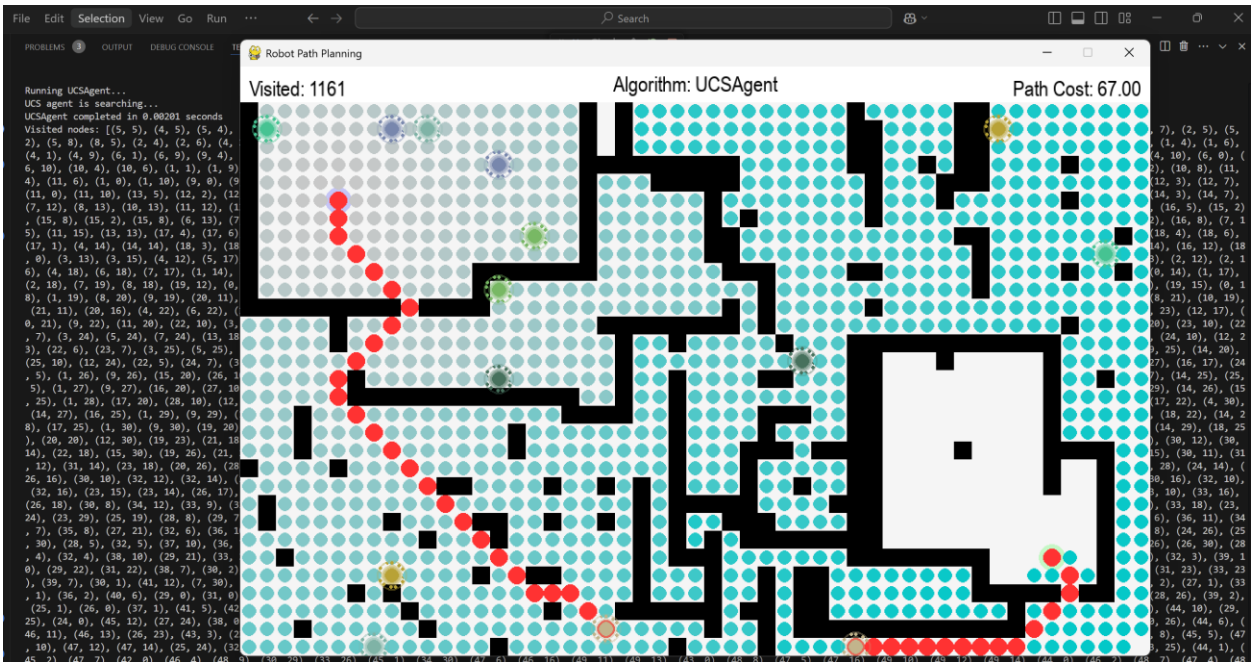


پیاده‌سازی الگوریتم UCS (Uniform Cost Search)

کلاس UCSAgentالگوریتم جستجوی هزینه یکنواخت را پیاده‌سازی می‌کند. در این الگوریتم، در هر لحظه گره‌ای که کمترین هزینه تا آن لحظه را داشته باشد انتخاب می‌شود.

توضیحات:

- درUCS ، تنها هزینه واقعی مسیر به عنوان معیار اولویت در نظر گرفته می‌شود.
- مشابه*A ، صف اولویت استفاده شده و گره با کمترین هزینه انتخاب می‌شود.



محیط (Env) و تولید نقشه

بخش Env محیط دو بعدی را تعریف می‌کند. این بخش شامل تنظیم ابعاد محیط، خواندن نقشه از فایل (که در آن موانع و تله‌پورت‌ها ذخیره شده‌اند) و یا ایجاد تله‌پورت‌های تصادفی (در صورتی که نیاز باشد) می‌باشد.
همچنین، در ماژولGenerator ، از کدهایی استفاده شده است که به کاربر اجازه می‌دهد از طریق تعامل (کلیک ماوس و کیبورد)، نقشه را ویرایش کرده و در پایان به صورت فایل JSON ذخیره کند.

نمایش(Plotting)

بخش Plotting مسئول رسم گرافیکی محیط، موانع، تله‌پورت‌ها، مسیر پیدا شده و همچنین نمایش گام به گام (انیمیشن) می‌باشد.
این ماژول قابلیت‌هایی نظیر نمایش اطلاعات هزینه، تعداد گره‌های بازدیدشده و نام الگوریتم جاری را نیز فراهم کرده است.
در این قسمت از انیمیشن، علاوه بر رسم مسیر نهایی، گام به گام نقاط بازدید شده نمایش داده شده و یک بنر در بالای صفحه نیز وجود دارد که نام الگوریتم در آن نشان داده می‌شود.

مقایسه عملکرد (سرعت، حافظه و کوتاهی مسیر)

سرعت اجرا

- **BFS:** اگرچه ساختار آن ساده و سطح به سطح است، اما در گراف‌های بزرگ سرعت کاهش می‌یابد زیرا به تعداد زیادی از گره‌ها نگاه می‌کند.
- **BiIDDFS:** به علت انجام جستجو از دو جهت و افزایشی کردن عمق، ممکن است در شرایطی کندتر عمل کند؛ خصوصاً زمانی که عمق حد نهایی زیاد شود.
- **A*:** با استفاده از هیوریستیک مناسب، مسیرهای مناسبتر به سرعت انتخاب می‌شوند؛ بنابراین، به‌خصوص در محیط‌های شبکه‌ای، A* معمولا سریعتر است.
- **UCS:** بسته به توزیع هزینه‌ها، سرعت UCS می‌تواند نسبتاً پایین باشد زیرا به ترتیب کمترین هزینه را بررسی می‌کند.

مصرف حافظه

- **BFS:** نیاز به نگهداری تمامی گره‌های سطح فعلی دارد؛ بنابراین در گراف‌های وسیع، مصرف حافظه بسیار بالا می‌رود.
- **BiIDDFS:** از آنجا که از جستجوی عمیق (DFS) استفاده می‌کند، حافظه کمتری نسبت به BFS مصرف می‌شود؛ اما به دلیل تکرار برخی محاسبات، زمان بیشتری صرف می‌شود.
- **A*:** با استفاده از صف اولویت و ذخیره مقدار f برای هر گره، حافظه مصرفی آن بالا است.
- **UCS:** مشابه A* از نظر صف اولویت عمل می‌کند؛ لذا از فضای حافظه بیشتری استفاده می‌کند.

کوتاهی مسیر پیدا شده

- **BFS:** در مسائل بدون وزن، مسیر پیدا شده از نظر تعداد گام‌ها بهینه است؛ اما ممکن است اگر هزینه‌ها متفاوت باشند این شرط برقرار نباشد.
- **BiIDDFS:** به دلیل جستجوی دوطرفه، در صورتی که تقاطع به درستی شناسایی شود، مسیر بهینه پیدا می‌شود.
- **A*:** اگر هیوریستیک مورد استفاده admissible باشد، مسیر بهینه پیدا می‌شود. هیوریستیک اکتایل در این پیاده‌سازی بسیار مناسب برای محیط‌های شبکی است.
- **UCS:** همواره مسیر بهینه از لحاظ هزینه پیدا می‌کند، زیرا گره‌ها بر اساس هزینه تاکنون طی شده انتخاب می‌شوند.

- **BFS** مناسب مسائل بدون وزن یا زمانی که تعداد گام‌ها اهمیت بیشتری دارد.
- **BiIDDFS** در محیط‌هایی که حافظه محدود است و گراف بسیار بزرگ است، مناسب می‌باشد؛ گرچه ممکن است زمان اجرا افزایش یابد.
- **A*** به دلیل استفاده از هیوریستیک در بسیاری از مسائل شبکی عملکرد بسیار خوبی دارد؛ به شرط آنکه هیوریستیک انتخاب شده admissible و کارا باشد.

- **UCS** تضمین می‌کند که همیشه مسیر بهینه از لحاظ هزینه پیدا شود؛ اما ممکن است برای گراف‌های بزرگ زمان و حافظه زیادی مصرف کند.

این پروژه با هدف شبیه‌سازی مسائل مسیریابی و ارائه الگوریتم‌های مختلف جستجو) از جمله BFS ، BiIDDFS، *A و UCS طراحی شده است.
مزیت اصلی پروژه در مقایسه‌ی الگوریتم‌ها، نمایش تصویری مسیر، هزینه‌های طی شده و گام به گام بودن فرایند جستجوی هر کدام است.
همچنین، امکان ویرایش نقشه توسط کاربر و ذخیره آن در قالب فایل JSON ، انعطاف‌پذیری پروژه را افزایش داده است.