دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

تمرین سری سوم

هوش مصنوعی

پاییز ۱۴۰۳

استاد درس: دکتر حسین فلسفین

دستیار آموزشی: پوریا دیانتی

لطفا پیش از حل سوالات به نکات زیر توجه فرمایید:

- پاسخ‌ها را به صورت یک فایل rar یا zip با نام HW3_StudentNumber آماده و سپس در سامانه و یکتا و در بخش مشخص‌شده آپلود نمایید. برای تمامی کدهای بخش عملی، توضیحاتی آماده کنید و در کنار فایل‌های پاسخ خود قرار دهید.
- از ارسال پاسخ‌ها از طریق ایمیل یا تلگرام خودداری نمایید. فقط پاسخ‌هایی که از طریق سامانه یکتا و **در مهلت مشخص‌شده** آپلود شوند، بررسی خواهند شد.
- لطفا در صورت استفاده از ابزارهای هوش‌مصنوعی مانند ChatGPT و مشابهات آن، به قوانین استفاده از آن‌ها دقت نمایید.
- در صورت وجود هرگونه ابهام یا سوال، می‌توانید از طریق تلگرام با دستیار آموزشی درس در ارتباط باشید: diap_2003

بخش اول: تئوری

سوال اول:

در یک گراف وزن‌دار، یک عامل می‌خواهد مسیر بهینه از نود شروع (S) به نود هدف (G) را پیدا کند. الگوریتم‌های جستجوی مختلف برای این منظور استفاده می‌شوند. با توجه به مفاهیم جستجو در گراف‌ها، به سوالات زیر پاسخ دهید:

- مقایسه الگوریتم‌ها:

الف) تفاوت اصلی بین الگوریتم جستجوی با هزینه یکنواخت (Uniform-Cost Search) و جستجوی (A*) چیست؟

ب) آیا همیشه (A*) بهتر (از نظر سرعت پیدا کردن جواب و یافتن جواب بهینه) از جستجوی با هزینه یکنواخت است؟ دلیل بیاورید.

- Heuristic Admissibility:

الف) توضیح دهید که h(n) باید چه شرایطی داشته باشد تا الگوریتم (A*) یک جواب بهینه پیدا کند.

ب) مثال بزنید که اگر تابع h(n)، شرط admissibility را نداشته باشد، چگونه ممکن است جواب بهینه را پیدا نکند.

- الگوریتم‌های حریصانه و (A*):

الف) چرا جستجوی حریصانه (Greedy Best-First Search) گاهی اوقات جواب بهینه(بهترین جواب) را پیدا نمی‌کند؟ برای این موضوع مثالی بزنید.

ب) چگونه (A*) با ترکیب هزینه مسیر و heuristic این مشکل را برطرف می‌کند؟

- تحلیل پیچیدگی:

الگوریتم (A*) در بدترین حالت چه زمانی پیچیدگی زمانی و حافظه‌ای بسیار بالا پیدا می‌کند؟ آیا راهی برای کاهش این پیچیدگی وجود دارد؟ توضیح دهید.(می‌توانید در مورد نسخه‌های بهبود یافته این جستجو مانند (IDA*)، (MA*) و (SMA*) تحقیق کنید)

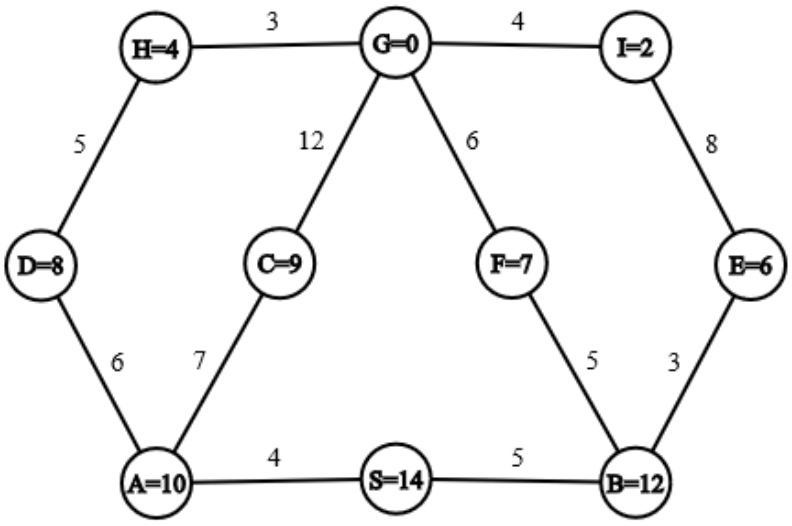
سوال دوم:

در مورد درستی یا نادرستی موارد زیر، با ذکر دلیل اظهار نظر کنید.

- اگر h_1 و h_2 توابعی admissible باشند، آنگاه $|h_1 - h_2|$ هم admissible است.
- اگر h1 و h2 توابعی consistent باشند، آنگاه $\max(h_1, h_2) + \min(h_1, h_2)$ هیچوقت نمی‌تواند consistent باشد.
- استفاده از توابع هیوریستیکی که consistent نیستند، در روش (A*) برای گراف ممکن است باعث گسترش مقدار کمتری گره شود.
- تابع ترکیبی $\max(h_1, h_2)$ زمانی می‌تواند تابع هیوریستیک مناسب‌تری نسبت به خود h_1 و h_2 باشد که حداقل در بعضی گره‌ها و نه در تمام گره‌ها مقدار h_1 بیشتر از h_2 باشد.

سوال سوم:

یک عامل می‌خواهد از یک نقطه شروع(S) به نقطه هدف(G) برسد. در گراف زیر، یال‌ها نمایانگر هزینه حرکت بین نودها هستند و اعداد نوشته‌شده در داخل هر نود مقدار تخمینی(heuristic) هزینه رسیدن از آن نود به هدف هستند.



برای هر یک از الگوریتم‌های جستجوی زیر، مراحل اجرای الگوریتم ترتیب بازدید نودها را بنویسید:

- جستجوی (DFS)
- جستجوی (BFS)
- جستجوی (Uniform-Cost Search)
- جستجوی (Greedy Best-First Search)
- جستجوی (A*)

بررسی کنید که آیا مقادیر Heuristic داده‌شده در این مسئله admissible هستند یا خیر. پاسخ خود را توضیح دهید.

سوال چهارم:

جستجوی دوطرفه (bidirectional search):

- نحوه کار این جستجو را توضیح دهید و ثابت کنید پیچیدگی زمانی این جستجو از $O(b^d)$ به $O(b^{d/2})$ کاهش می‌یابد.
- چالش‌های پیاده‌سازی جستجوی دوطرفه را به ویژه در مسائلی که شامل اعمال غیرقابل بازگشت (non-invertible actions) هستند، مورد بحث قرار دهید.

بخش دوم: عملی

سوال اول:

در این سوال باید مسئله maze را به وسیله الگوریتم (A*) پیاده‌سازی کنید.

ورودی‌ها:

- ابعاد ماتریس مربعی که با حرف N نمایش داده می‌شود.
- N عدد برای مقادیر هر کدام از N سطر(مقدار ۰ نشان‌دهنده خانه آزاد و ۱، نشان‌دهنده خانه مسدود می‌باشد).
- مختصات خانه‌های شروع و پایان.

خروجی:

- دنباله مختصات خانه‌های مسیر بهینه از نقطه شروع تا پایان.

سوال دوم:

هدف این سوال، حل مسئله N-puzzle یا Puzzle Sliding می‌باشد. برنامه‌ای پیاده‌سازی کنید که با گرفتن دو استیت ورودی و خروجی، مسیر رسیدن از ورودی به خروجی را نشان دهد. در پیاده‌سازی، به نکات زیر توجه کنید:

- برنامه شما باید از پازل‌های ۳*۳، ۴*۴ و ۵*۵ پشتیبانی کند.
- برای نوشتن برنامه از الگوریتم (A*) استفاده کنید.
- الگوریتم دارای دو تابع هیوریستیک زیر باشد که بتوان در ابتدای برنامه یکی از آنها را مشخص کرد:
  1. Euclidean Distance
  2. Manhattan Distance
- پیاده‌سازی این سوال با الگوریتم (IDA*) نمره امتیازی دارد.(می‌توانید در مورد الگوریتم از اینجا بخوانید)

سوال سوم:

در ادامه سوال قبل، برای این سوال نیاز است تابع هیوریستیک جدیدی با توجه به مطالب گفته‌شده در درس بسازید(روش pattern database برای هیوریستیک) که بهتر از دو هیوریستیک اقلیدسی و منهتن باشد. برای تولید هیوریستیک خود، از کد دیگری استفاده کنید و نتایج را در کد سوال دوم استفاده کنید و آن‌ها را از نظر زمان و حافظه مقایسه کنید.

سوال چهارم(امتیازی):

در این سوال باید کدی بنویسید که Sokoban را حل کند. کد خود را بر روی چند مثال اجرا کرده و آن را از نظر صحت جواب و بهینگی(پیدا کردن کوتاه‌ترین جواب) بررسی نمایید و نتایج خود را گزارش کنید. می‌توانید برای آشنایی بیشتر، بازی را از این لینک، انجام دهید.