دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر

«مبانی و کاربردهای هوش مصنوعی ترم پائیز ۱۴۰۴»

پروژه دوم

در انجام پروژه‌ها به نکات زیر توجه فرمائید:

۱- پیاده‌سازی پروژه‌ها را به زبان برنامه‌نویسی پایتون انجام دهید.

۲- مطابق قوانین دانشگاه هر نوع کپی‌برداری و اشتراک کار دانشجویان غیرمجاز است و پاسخ به پروژه‌ها باید به صورت انفرادی و بدون استفاده از ابزارهای هوش مصنوعی انجام شود، در صورت مشاهده چنین مواردی با طرفین شدیدا برخورد خواهد شد.

۳- فایل پروژه را با فرمت StudentID_AI_P02.zip تا ساعت ۲۳:۵۹ روز ۱۴۰۴/۰۹/۰۲ فقط در بخش مربوطه در سایت درس آپلود نمائید.

۴- توجه نمائید پاسخ پروژه‌ها تنها در صورت آپلود در سامانه کورسز پذیرفته خواهد شد و ارسال پاسخ از طریق ایمیل یا تلگرام بررسی نخواهد شد.

۵- در مجموع برای پروژه‌ها ۱۰ روز تاخیر مجاز دارید که می‌توانید در طول ترم بسته به شرایط از آن استفاده نمائید، در صورت اتمام تاخیر مجاز، هر روز تاخیر منجر به کسر ۲۰٪ نمره از پروژه مربوطه خواهد شد.

فهرست مطالب

بخش اول: حل جدول کلمات متقاطع با استفاده از مسئله ارضای قیود

مقدمه

در این بخش، هدف آشنایی با مفاهیم پایه‌ای مسئله ارضای قیود (CSP) و پیاده‌سازی آن در قالب حل یک جدول کلمات متقاطع است.

در این پروژه، جدول به‌صورت یک شبکه از خانه‌های خالی و پر تعریف می‌شود و دانشجو باید با استفاده از الگوریتم‌های CSP، مانند سازگاری گره‌ای (Node Consistency)، سازگاری یالی (Arc Consistency) و جست‌وجوی بازگشتی (Backtracking Search) همراه با هیوریستیک‌های MRV و LCV، جدول را به‌صورت خودکار حل کند.

در این مسئله:

- هر کلمه در جدول به‌عنوان یک متغیر (Variable) در نظر گرفته می‌شود.
- مجموعه کلمات ممکن برای هر متغیر، دامنه (Domain) آن متغیر است.
- قیود (Constraints) تعیین می‌کنند که دو کلمه در محل‌های تلاقی خود باید حرف مشترک سازگار داشته باشند.

ساختار کلی پروژه

پروژه از چند بخش اصلی تشکیل شده است که هر بخش وظیفه خاصی بر عهده دارد:

- **crossword.py**

این فایل ساختار کلی جدول را تعریف می‌کند و وظیفه تشخیص کلمات افقی و عمودی، ایجاد متغیرها، و یافتن روابط تقاطع بین آن‌ها را بر عهده دارد. در این فایل کلاس‌های Variable و Crossword تعریف شده‌اند که هسته اصلی مدل CSP را می‌سازند.

- **creator.py**

مهم‌ترین فایل پروژه که باید توسط دانشجو تکمیل شود. در این فایل، منطق اصلی حل‌کننده CSP پیاده‌سازی می‌شود، شامل:

  - اعمال سازگاری گره‌ای (Node Consistency)
  - پیاده‌سازی الگوریتم AC-3
  - طراحی جست‌وجوی بازگشتی (Backtracking Search)
  - انتخاب متغیر با ترکیب MRV + Degree Heuristic
  - مرتب‌سازی مقادیر دامنه با استفاده از LCV (Least Constraining Value)

- **main.py**

فایل اجرایی پروژه است که ورودی جدول و فایل کلمات را از Command Line دریافت کرده و روند حل جدول را آغاز می‌کند. پس از اجرا، در صورت موفقیت، جدول نهایی در خروجی ترمینال نمایش داده می‌شود.

- **puzzles/**

شامل فایل‌های جدول‌های مختلف (به‌صورت متن) است. هر جدول با کاراکتر "_" برای خانه‌های خالی و "#" برای خانه‌های پر مشخص می‌شود.

- **words/**

شامل فایل words.txt است که فهرستی از کلمات ممکن را برای استفاده در جدول نگه‌می‌دارد.

نحوه اجرای پروژه

برای اجرای پروژه، کافی است در ترمینال دستور زیر را وارد کنید:

```
python main.py puzzles/p1.txt
```

می‌توانید به جای جدول بالا، از جداول دیگر در پوشه puzzles استفاده کنید.

گزارش کار و تحلیل نتایج

- دانشجو باید گزارشی تحلیلی از عملکرد پروژه خود تهیه و ارسال کند. این گزارش باید شامل موارد زیر باشد:
- شرح پیاده‌سازی الگوریتم‌ها:

  توضیح منطق هر الگوریتم و چگونگی تعامل آن‌ها در حل مسئله.
- تحلیل زمان اجرا:

  بررسی مدت زمان حل جدول و عوامل مؤثر بر کاهش یا افزایش آن.
- مقایسه هیوریستیک‌ها:

  بررسی تفاوت عملکرد هنگام استفاده از MRV تنها در مقابل MRV+LCV
- تحلیل خطاها و محدودیت‌ها:

  بیان شرایطی که جدول حل نمی‌شود و علت آن (مثلاً دامنه خالی یا قیود ناسازگار)
- نمونه خروجی:

  ارائه یک یا چند نمونه جدول حل‌شده و توضیح روند منطقی حل آن‌ها.

بخش دوم: پیاده‌سازی بازی Isolation با استفاده از جست‌وجوی خصمانه

مقدمه

در این بخش، هدف آشنایی با مفاهیم پایه‌ای جست‌وجوی خصمانه و طراحی عامل‌های هوشمند در قالب یک بازی دونفره به نام Isolation است.

هر بازیکن در این بازی به‌صورت نوبتی حرکت کرده و تلاش می‌کند با انجام حرکات بهینه، رقیب خود را در موقعیتی قرار دهد که دیگر نتواند حرکتی انجام دهد. بازیکنی که هیچ حرکت قانونی نداشته باشد، بازنده‌ی بازی خواهد بود.

در این نسخه از پروژه، حرکت‌ها به‌صورت حرکت اسب در شطرنج تعریف شده‌اند، یعنی هر بازیکن در نوبت خود می‌تواند به یکی از خانه‌هایی که در فاصله‌ی دو خانه در یک جهت و یک خانه در جهت عمود قرار دارد (الگوی L شکل) حرکت کند. این موضوع باعث می‌شود که رفتار بازی شباهت زیادی به حرکت‌های واقعی در صفحه‌ی شطرنج داشته باشد.

ساختار کلی پروژه

پروژه از چند بخش اصلی تشکیل شده است که هر بخش وظیفه‌ی خاصی بر عهده دارد:

- **isolation/**

  این پوشه شامل منطق اصلی بازی است و فایل‌های زیر را دارد:

  - فایل isolation.py: این فایل شامل تعریف کامل کلاس بازی، صفحه‌ی بازی، قوانین حرکت، و اعمال محدودیت‌ها است. این بخش به‌عنوان هسته‌ی اصلی سیستم عمل می‌کند.
  - فایل __init__.py : فایل پیکربندی ماژول پایتون برای پوشه‌ی isolation

- **isoviz/**

این پوشه برای نمایش گرافیکی بازی در مرورگر طراحی شده است.

  - فایل display.html: فایل اصلی رابط کاربری است که امکان مشاهده‌ی حرکات بازی را در مرورگر فراهم می‌کند.
  - پوشه‌های js/ , css/ و img/: شامل فایل‌های لازم برای ظاهر گرافیکی، کدهای جاوااسکریپت و تصاویر مورد استفاده در صفحه‌ی نمایش هستند.

- **game_agent.py**

این فایل مهم‌ترین بخش پروژه است که باید توسط دانشجو تکمیل شود. در این فایل، الگوریتم‌های جست‌وجوی Minimax، Alpha-Beta و Expectimax پیاده‌سازی می‌شوند و سه تابع heuristic نیز تعریف می‌شوند تا کیفیت تصمیم‌گیری عامل بهبود یابد.

- **sample_players.py**

این فایل شامل چند بازیکن ساده‌ی از پیش‌تعریف‌شده است، مانند:

  - RandomPlayer: بازیکنی که به‌صورت تصادفی حرکت می‌کند.
  - GreedyPlayer: بازیکنی که صرفاً بر اساس بیشترین حرکت ممکن تصمیم می‌گیرد.

این بازیکن‌ها برای تست و مقایسه با عامل هوشمند اصلی استفاده می‌شوند.

- **visualize.py**

این فایل وظیفه دارد بازی را بین دو بازیکن (عامل‌ها) اجرا کند، حرکات انجام‌شده را ذخیره نماید و نتیجه‌ی بازی را به‌صورت فایل match.json در پوشه‌ی isoviz قرار دهد. سپس مرورگر را باز کرده و فایل display.html را اجرا می‌کند. دانشجو باید محتوای match.json را در قسمت "Move History" داخل صفحه‌ی مرورگر کپی کند تا بتواند بازی را مرحله‌به‌مرحله مشاهده کند.

قوانین بازی Isolation با حرکت اسب

۱. صفحه‌ی بازی به‌صورت پیش‌فرض یک ماتریس ۷×۷ است.

۲. هر بازیکن با استفاده از حرکت اسب در شطرنج (الگوی L شکل) در صفحه حرکت می‌کند.

۳. پس از هر حرکت، خانه‌ی قبلی بازیکن مسدود شده و هیچ بازیکنی نمی‌تواند به آن خانه بازگردد.

٤. اگر بازیکنی هیچ حرکت قانونی نداشته باشد، بازنده است.

وظایف دانشجو

در فایل game_agent.py باید کلاس‌ها و توابع زیر تکمیل شوند:

۱. **تابع‌های ارزیابی (Heuristic Functions)**

- custom_score
- custom_score_2
- custom_score_3

این توابع وضعیت فعلی بازی را ارزیابی کرده و باید با در نظر گرفتن تعداد حرکت‌های ممکن برای هر بازیکن، میزان کنترل مرکز، و شرایط پیروزی، عددی بازگردانند که بیانگر مطلوبیت وضعیت برای بازیکن باشد. پس از پیاده‌سازی این توابع عملکرد هرکدام را مقایسه کنید و بهترین تابع ارزیابی را تحلیل کنید.

۲. **کلاس MinimaxPlayer**

- پیاده‌سازی الگوریتم Minimax با عمق مشخص
- استفاده از self.score() برای ارزیابی وضعیت‌ها
- بررسی زمان باقی‌مانده‌ی اجرای تابع با متغیر self.TIMER_THRESHOLD

۳. **کلاس AlphaBetaPlayer**

- پیاده‌سازی جست‌وجوی عمق‌افزایشی (Iterative Deepening)
- به‌کارگیری Alpha-Beta Pruning برای بهینه‌سازی Minimax

- بازگرداندن بهترین حرکت قبل از اتمام زمان

٤. کلاس ExpectimaxPlayer

- پیاده‌سازی الگوریتم Expectimax با عمق مشخص
- استفاده از self.score() برای ارزیابی وضعیت‌ها
- بازگرداندن بهترین حرکت قبل از اتمام زمان

اجرای بازی و مشاهده‌ی نتایج

برای اجرای بازی، از دستور زیر استفاده کنید:

python visualize.py

پس از اجرا، بازی میان دو بازیکن آغاز شده و فایل match.json در مسیر isoviz ذخیره می‌شود. سپس مرورگر به‌صورت خودکار باز شده و فایل display.html نمایش داده می‌شود.

توجه ۱: برای مشاهده‌ی بازی در مرورگر، باید فایل match.json را باز کرده و محتوای آن را در بخش Move History صفحه‌ی display.html قرار دهید. سپس با فشردن دکمه‌ی Run Game، می‌توانید بازی را به‌صورت گرافیکی مشاهده کنید.

توجه ۲: برای تست کردن عامل‌های پیاده سازی شده توسط خودتان باید بازیکن‌های داخل فایل visualize.py را تغییر دهید.

گزارش کار و تحلیل نتایج

در پایان پروژه، هر دانشجو باید گزارشی از کار خود تهیه و ارسال کند. هدف از این گزارش، توضیح روند پیاده‌سازی، تحلیل رفتار عامل هوشمند و بررسی نتایج حاصل از اجرای الگوریتم‌های مختلف بر روی مراحل گوناگون بازی است.

گزارش باید به‌صورت فنی، تحلیلی و با استدلال منطقی نوشته شود و شامل بخش‌های زیر باشد:

۱- شرح دهید که پیاده‌سازی الگوریتم‌ها به چه شکل است. علاوه بر این، باید بیان کنید که مدیریت زمان (SearchTimeout) و جلوگیری از اتمام زودهنگام چطور انجام شده است و چه تأثیری بر عملکرد الگوریتم شما دارد. هدف از این بخش نشان دادن درک عمیق شما از منطق تصمیم‌گیری و طراحی عامل هوشمند است.

۲- در ادامه، لازم است درباره‌ی تابع‌های ارزیابی (Heuristic) که در پروژه پیاده‌سازی کرده‌اید، توضیح دقیق و تحلیلی ارائه دهید. بیان کنید. عملکرد تابع‌ها را با هم مقایسه کنید و توضیح دهید کدام یک در بازی واقعی نتیجه‌ی بهتری داشته است و چرا.

۳- در بخش بعد، باید عامل خود را در برابر بازیکن‌های پایه مانند RandomPlayer و GreedyPlayer آزمایش کنید. چند بازی متوالی بین عامل شما و این بازیکن‌ها اجرا کنید و نتایج آماری حاصل (تعداد برد، باخت و میانگین طول بازی) را گزارش دهید. همچنین تحلیل کنید که چرا عامل شما در برخی حالت‌ها برنده یا بازنده می‌شود. برای مثال، توضیح دهید که در چه موقعیت‌هایی الگوریتم Minimax تصمیم بهینه می‌گیرد و در چه شرایطی ممکن است عملکرد ضعیف‌تری نسبت به Alpha-Beta داشته باشد.

٤- در پایان گزارش، خلاصه‌ای از یافته‌های خود را بنویسید. بیان کنید که پیاده‌سازی هر الگوریتم چه چالش‌هایی داشته و چه عواملی باعث بهبود یا افت عملکرد عامل شما شده‌اند. در صورت امکان، با استفاده از ابزار isoviz تصویری از یک یا دو بازی نمونه قرار دهید و به‌صورت کیفی رفتار عامل را تحلیل کنید (مثلاً نحوه‌ی حرکت اسب، کنترل مرکز صفحه و جلوگیری از گیر افتادن).

در مجموع، گزارش نهایی باید تصویری روشن از فرایند طراحی، پیاده‌سازی، آزمایش و تحلیل عملکرد عامل هوشمند شما ارائه دهد، صرفاً ذکر نتایج عددی کافی نیست.