مدرس: **دکتر فدایی و دکتر یعقوب‌زاده** طراحان:
بازی: امیرحسین عباسکوهی، علیرضا ابراهیمی
ژنتیک: حمید خدادادی، حسین باقرزاده یزدی

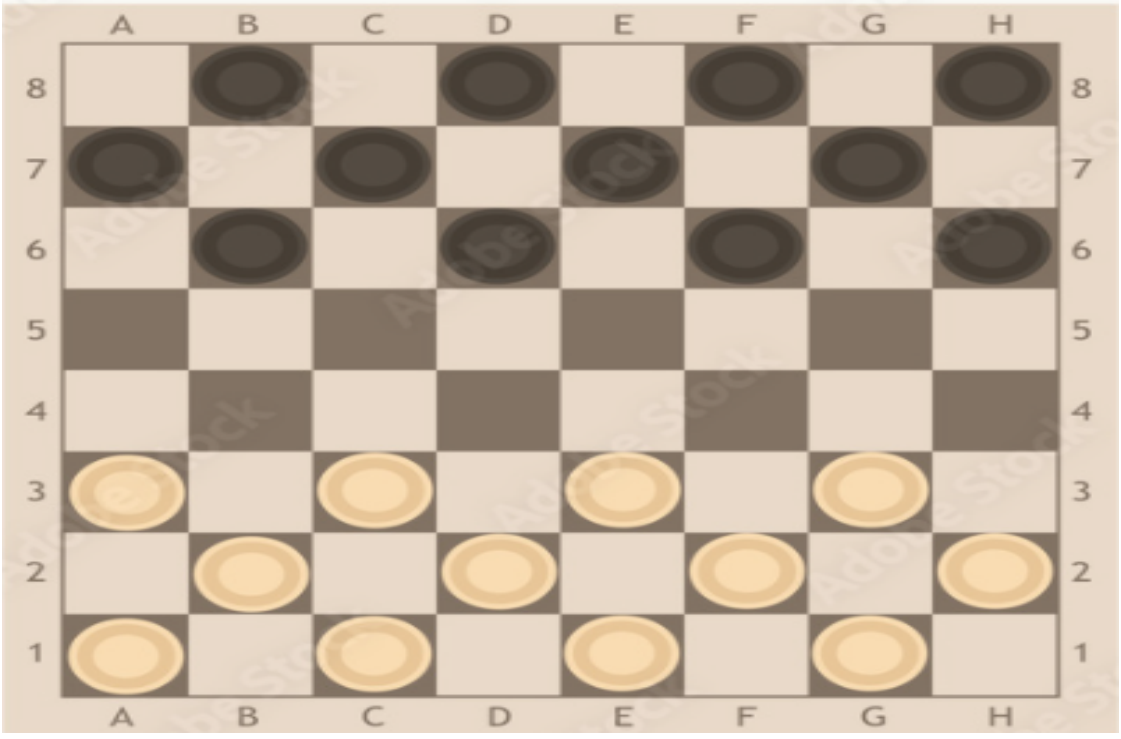
مهلت تحویل: دوشنبه 17 آبان 1400، ساعت 23:55

مقدمه:

در این هندز آن از شما خواسته شده است که دو سوال را با الگوریتم های min-max و ژنتیک پیاده سازی و حل کنید.

بازی چکرز:

در این سوال با الگوریتم min-max که در بسیاری از بازی ها استفاده می شود آشنا می شوید، و ایده هایی را برای ارزیابی و تعریف مقدار کمی مناسب برای بازی چکرز را استفاده می کنیم.
بازی چکرز یک بازی تخته ای دو نفره با ابعاد 8 در 8 است که تمامی خانه های جدول به دو رنگ، رنگ آمیزی شده اند، این بازی دارای 24 مهره در دو رنگ است که هر فرد در ابتدا 12 مهره خواهد داشت که در خانه های سیاه جدول در 3 ردیف ابتدایی قرار می گیرند.



در هر حرکت مهره ها نمی توانند از این خانه های سیاه خارج شوند. بازی به صورت نوبتی بوده و در هر نوبت یک بازیکن می تواند یک مهره خود را جا به جا کند. در هر حرکت، مهره می تواند یک واحد (به سمت بالا سمت چپ یا بالا سمت راست) برود و در صورتی که یک مهره حریف در آن نقطه وجود داشته باشد که در امتداد آن در خانه (2 واحد راست/چپ و 2 واحد بالا) مهره ای قرار نداشته باشد فرد می تواند آن مهره را بزند. همچنین اگر چند مهره حریف به گونه ای قرار داشته باشند که بازیکن بتواند پس از زدن یکی بقیه را نیز به ترتیب بزند باید این کار را انجام دهد. اگر مهره ای به انتها رسید، ارتقا یافته و تبدیل به مهره King می شود و می تواند به صورت اریب در هر دو سمت جلو و عقب در هر نوبت حرکت کند (اندازه حرکت ثابت بوده و برابر با یک است بدین معنا که در هر حرکت می توان به اندازه یک خانه جابجا شد مگر اینکه از روی مهره حریف بپرد.).

توضیح تمرین الگوریتم min-max:

کد داده شده در این قسمت، با استفاده از کتابخانه pygame امکان پیاده سازی گرافیکی بازی را به شما می دهد. شما با کامل کردن قسمت هایی از کد که کامل نیستند و به صورت "Your Code Goes here" مشخص شده اند، الگوریتم min-max را پیاده سازی کرده به بررسی تاثیر عمق درخت در تصمیم گیری ها می پردازید. بازی به صورت کامل پیاده سازی شده است و شما تنها نیاز دارید که قسمت min-max را پیاده سازی کنید لذا اکثر پیاده سازی شما در فایل minimax.py خواهد بود. البته لازم به ذکر است که تغییرات عمق بازی در main.py و همچنین تابع getValidMoves در فایل board.py قرار دارند.

بخش اول: نصب نیازمندی ها

کتابخانه pygame را نصب کنید. (pip install pygame)

بخش دوم: آشنایی با کد و برد بازی

تابع getAllMoves را به صورتی کامل کنید که با گرفتن board، رنگ بازی کنی که نوبتش است و همچنین وضعیت کل بازی بتواند تمامی حرکات مجاز را بازگرداند. تابع getValidMoves را نیز کامل کنید تا حرکات مجازی را که هرکسی می تواند انجام دهد را برگرداند.

بخش سوم: یافتن معیار مناسب به منظور ارزیابی در الگوریتم مین-مکس

تعریفی مناسب از evaluation بیان کنید و کد تابع minimax را کامل کنید. این تابع board بازی، وضعیت کل بازی، عمق باقی مانده و بولین اینکه آیا باید مینیمم کند یا ماکزیمم را می گیرد و مقدار evaluation به همراه بهترین حرکت را باز می گرداند.

بخش چهارم: تست الگوریتم با عمق کم

مقدار عمق را برای هر دو بازیکن 1 در نظر گرفته و بازی را اجرا کنید و نتیجه را تحلیل کنید.

بخش پنجم: تست الگوریتم با عمق بیشتر

مقدار عمق را برای یک بازیکن 7 و برای دیگری 3 در نظر بگرید، برعکس این عمق را نیز در نظر بگیرید، بازی را اجرا کرده و نتیجه را تحلیل کنید.

بخش ششم: تست عمق برابر در الگوریتم

مقدار عمق را برای هر دو بازیکن 5 در نظر گرفته و بازی را اجرا کنید و نتیجه را تحلیل کنید.

توضیح تمرین الگوریتم ژنتیک:

در این سوال، با روش هایی که برگرفته از طبیعت و انتخاب طبیعی هستند، آشنا می شویم. در این روش ها که به طور کلی الگوریتم های ژنتیک نامیده می شوند؛ ایده هایی برای مدل سازی جفت گیری، جهش و انتخاب طبیعی به کار می گیریم.
همچنین یاد می گیریم که بعضی اوقات با انتخاب معیار های ساده انتخاب طبیعی، این الگوریتم ها ضعیف عمل می کنند و باید معیار انتخابی ای در نظر بگیریم که علاوه بر عملکرد فردی، به گوناگونی جمعیت نیز اهمیت دهد.
الگوریتم های ژنتیک به طور کلی در مسئله هایی با فضای حالت بزرگ کاربرد دارند؛ این الگوریتم ها این کار را با نمونه گرفتن از جمعیت و ترکیب و تغییر افراد و ارزیابی آنها انجام می دهند و سعی می کنند که نسل به نسل جواب ها را بهبود دهند تا به جواب مورد نظر برسند.
جدول سودوکو، بازی جدول اعداد ژاپنی است که یکی از بهترین تمرین‌های مغز و تقویت IQ به شمار می رود.
مسئله سودوکو در حالت کلی یعنی وقتی جدولی به اندازهٔ داریم، np complete است. در واقع سودوکو معادل مسئله رنگ‌آمیزی گراف است که تعداد رنگ‌ها برابر است.
سودوکو پازلی به شکل یک جدول ۹×۹ است که تعدادی از خانه‌های آن به طور پیش فرض با اعداد ۱ تا ۹ پر شده‌اند.
هدف، پر کردن خانه‌های خالی جدول است به طوری که، در نهایت هر یک از اعداد ۱ تا ۹ دقیقاً یک بار در هر سطر، هر ستون، و هر مربع مشخص شدهٔ ۳×۳ ظاهر شوند. سودوکوهای مناسب راه‌حلی یکتا دارند. دانشمندی ایرلندی در اثباتی ریاضی نشان داد که کمترین تعداد راهنمایی‌ها (یا ارقام ابتدای بازی) که برای تکمیل یک بازی لازم است، 17 خانه است و جدول‌هایی با 16 راهنمایی یا کمتر، پاسخ یکتایی ندارند.
در نتیجه بار دیگر مسئله ای که باید با الگوریتم ژنتیک حل کنیم را مرور می کنیم:
نوع متداول سودوکو یک جدول ۹x۹ است که کل جدول هم به ۹ جدول کوچک‌تر ۳x۳ تقسیم شده‌است. در این جدول چند عدد به طور پیش فرض قرار داده شده که باید باقی اعداد را با رعایت سه قانون زیر پیدا کرد:

- قانون اول: در هر سطر جدول، اعداد ۱ الی ۹ بدون تکرار قرار گیرد.
- قانون دوم: در هر ستون جدول، اعداد ۱ الی ۹ بدون تکرار قرار گیرد.
- قانون سوم: در هر ناحیه ۳x۳ جدول، اعداد ۱ الی ۹ بدون تکرار قرار گیرد.

پیاده سازی مسئله:

در این قسمت از شما خواسته شده تا راه حلی با استفاده از الگوریتم ژنتیک برای حل جدول سودوکو پیدا کنید. برای پیاده سازی آن، شما باید مراحل زیر را همانطور که در درس نیز آموخته اید پیاده سازی کنید و سپس با تجمیع تمام این مراحل یک الگوریتم کلی برای حل مسئله پیاده سازی کنید.

بخش اول: مشخص کردن مفاهیم ژن و کروموزوم

در الگوریتم های ژنتیک ابتدا باید یک تعریف برای ژن ارائه دهید و سپس با استفاده از آن یک کروموزوم را بسازید. هر کروموزوم مجموعه ای از ژن ها است و این مجموعه یا همان کروموزوم، یک راه پیشنهادی برای حل مسئله مورد نظر می باشد. توجه داشته باشید که در الگوریتم های ژنتیک باید اکثر کار ها را باید به صورت رندوم انجام دهید چرا که اگر فضای حالت بزرگ باشد پیدا کردن شرطی که همه ی محدودیت ها را برقرار سازد بسیار دشوار است. تعریف کروموزوم اهمیت ویژه ای دارد و باید به گونه ای باشد که امکان اعمال تابع تناسب و توابع دیگر بر روی آن فراهم باشد.

بخش دوم: تولید جمعیت اولیه

پس از تعریف و پیاده سازی ساخت یک کروموزوم، باید جمعیت اولیه ای از کروموزوم ها به صورت کاملا رندوم و بدون هیچ جهت گیری خاصی بسازید. تعداد این جمعیت می تواند به عنوان یک پارامتر حل مسئله باشد و به انتخاب های شما بستگی دارد.

بخش سوم: پیاده سازی و مشخص کردن تابع معیار سازگاری (Fitness Function)

بعد از تولید جمعیت اولیه، نیاز به تعریف تابع معیاری برای تشخیص افراد یا کروموزم های بهتر از این نظر که شرایط و محدودیت های مسئله را فراهم کنند، داریم. پس معیار ارزیابی ای نسبت به مسئله طرح شده برای مقایسه کروموزوم ها پیاده سازی کنید.

بخش چهارم: پیاده سازی mutation و crossover و تولید جمعیت بعدی

حال برای اینکه به جدول هدف مسئله نزدیک شویم، در این قسمت نیاز به ایجاد جمعیت جدید از جمعیت های نسل قبل خود داریم. این کار را باید با روش های معروفی که در الگوریتم ژنتیک وجود دارد و در درس نیز با آن ها آشنا شده اید، یعنی mutation و crossover انجام دهید.

تابع crossover بر روی دو کروموزوم اعمال می شود و آن ها را ترکیب می کند تا به کروموزوم های بهتر برسد. روش نحوه این ترکیب و نرخ ایجاد آن می تواند به عنوان پارامترهای مسئله باشد.

تابع mutation بر روی یک کروموزم اعمال می شود و ژن های آن را جهش داده و تغییر می دهد، به این هدف که بتواند به کروموزوم بهتری دست یابد.
همچنین می توانید درصد معقولی از ژن های برتر را به نسل آینده منتقل کنید.

بخش پنجم: ایجاد الگوریتم ژنتیک روی مسئله

در آخر باید این توابع پیاده سازی شده را در یک الگوریتم استفاده کنید. توجه کنید که می توانید پارامتر هایی برای راه خود داشته باشید که با تغییر آن به جواب بهتری برسید.

فاز ششم: سوالات

1. روش انتخاب کروموزم های برتر برای تولید جمعیت بعدی و دلیل انتخاب روش به کار برده شده را توضیح دهید.
2. دلیل انتخاب معیار تناسب خود را ذکر کنید.
3. تاثیر تابع های crossover و mutation و احتمال هر یک از آن ها و دلیل انتخاب مقدار احتمال را ذکر کنید.
4. با وجود استفاده از این روش ها، ممکن است که کروموزوم ها پس از چند مرحله دیگر تغییر نکنند. دلیل این اتفاق و مشکلاتی که به وجود می آورد را شرح دهید. برای حل آن چه راهکاری پیشنهادی می دهید؟

نکات پایانی:

• نتایج و گزارش خود را در یک فایل فشرده بدین صورت تحویل دهید.

1. برای سوال گیم: AI_min_max_<#SID>.zip
2. برای سوال ژنتیک: AI_Genetic_<#SID>.zip

این پوشه باید حتما شامل موارد زیر باشد:

1. یک فایل code.py یا یک پوشه codes که شامل تمامی کد های شماست.
2. گزارش پروژه با فرمت PDF و شامل شرح تمامی کارهای انجام شده، نتایج به دست آمده و تحلیل ها و بررسی های خواسته شده در صورت پروژه.
3. می توانید از jupyter-notebook برای نوشتن گزارش استفاده کنید در این صورت فایل Notebook به همراه خروجی html آن را ارائه دهید توضیح و نمایش خروجی های خواسته شده بخشی از نمره این تمرین را تشکیل می‌دهد. از نمایش درست خروجی های مورد نیاز در فایل html مطمئن شوید. در غیر این صورت، گزارش را در فایلی با فرمت PDF ارائه دهید.

• در سوال ژنتیک، تمام مراحل کار خود شامل تعریف کروموزوم، تولید جمعیت اولیه، تابع تناسب، عملیات های mutation و crossover، جلوگیری از سوگیری کروموزوم ها و دلیل تمام تصمیماتی که می گیرید را در گزارش کار خود بیاورید.

• در سوال ژنتیک، تعیین hyper parameter ها در این مسئله اهمیت زیادی دارند و همچنین باید به دلیل تعیین و اثرات آن ها نیز توجه داشته باشید.

• دقت کنید که شیوه پیاده سازی موارد خواسته شده در این پروژه اهمیت زیادی دارد. بدست آوردن نتیجه درست بدون روش های خواسته شده در اینجا مدنظر نمی باشد.

• در صورتی که سوالی در مورد پروژه داشتید بهتر است در فروم درس مطرح کنید تا بقیه از آن استفاده کنند؛ در غیر این صورت توسط ایمیل با طراحان در ارتباط باشید.

• هدف از تمرین، یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در غیر این صورت مطابق قوانین درس با افراد خاطی برخورد خواهد شد.

در صورتی که سوالی داشتید می توانید با ما در ارتباط باشید.

سوال گیم:

- amirhossein.abaskohi@gmail.com
- a.r.ebrahimi79@gmail.com

سوال ژنتیک:

- hamidkhodadadi@ut.ac.ir
- hbyazdi99@gmail.com

موفق باشید.

پاییز 1400