به نام خدا

هوش مصنوعی

تمرین کامپیوتری شماره 2 ، بخش : Game

امیرحسین کهربائیان – 810199478

** **استفاده** grace **در این پروژه**

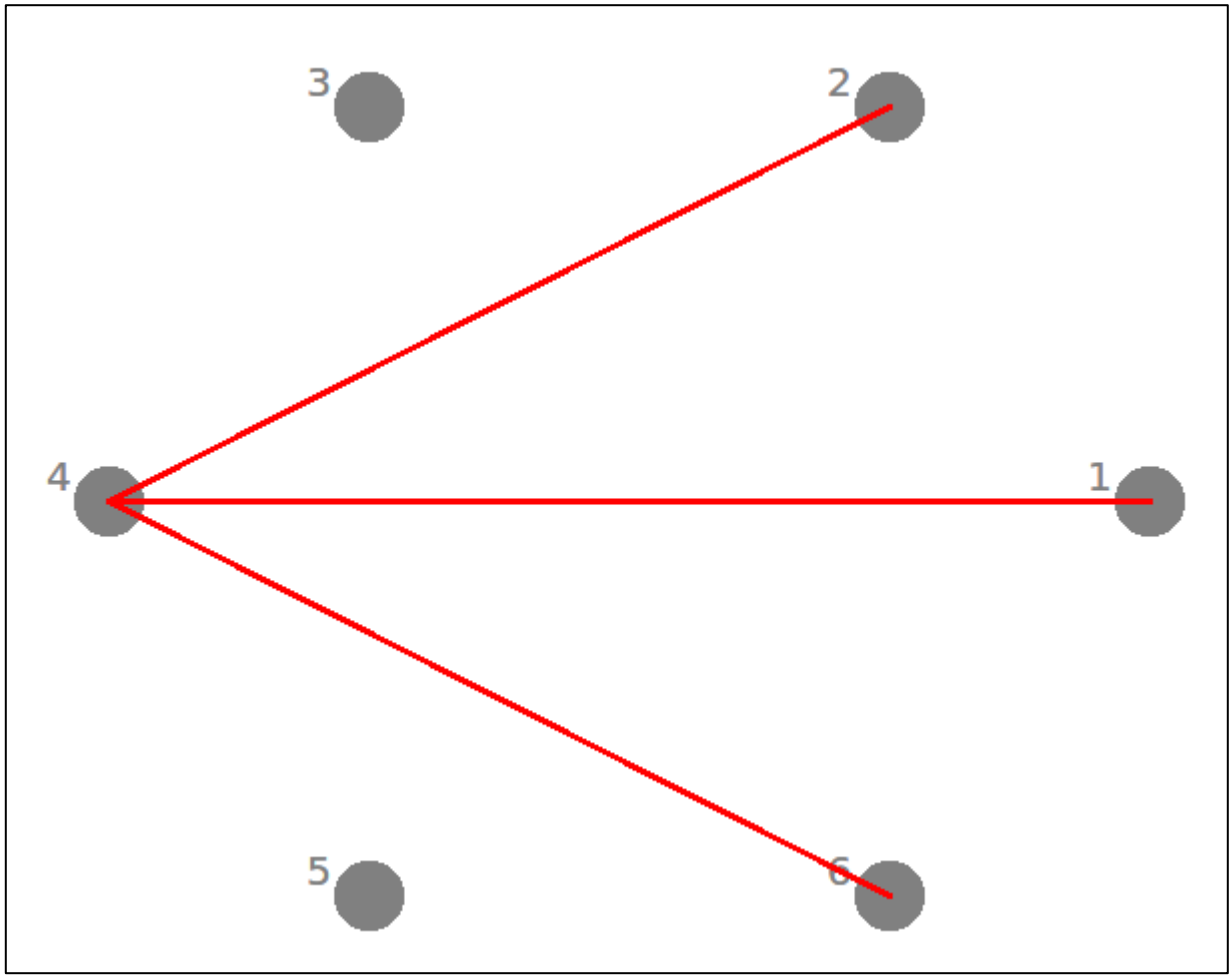
شرح پروژه :

در این قسمت باید با کمک الگوریتم min-max بازی Sim را طوری پیاده سازی کنیم که شانس پیروزی بالایی داشته باشیم.

همانطور که گفته شد این مسئله را باید به کمک الگوریتم min-max حل کنیم. چند نکته مهم :

1- به طور کلی اکثر قسمت های آماده کد تغییر خاصی نکرده اند. فقط در بدنه اصلی اجرای کد بخش هایی برای پرینت نتایج و ثبت نتایج اضافه شده است و همچنین در قسمت تشخیص برنده و بازنده اصلاح کوچکی شکل گرفته است (در حالت اولیه در صورتی که میزان خطوط قرمز کمتر از سه بود، خطوط آبی بررسی نمیشد...)

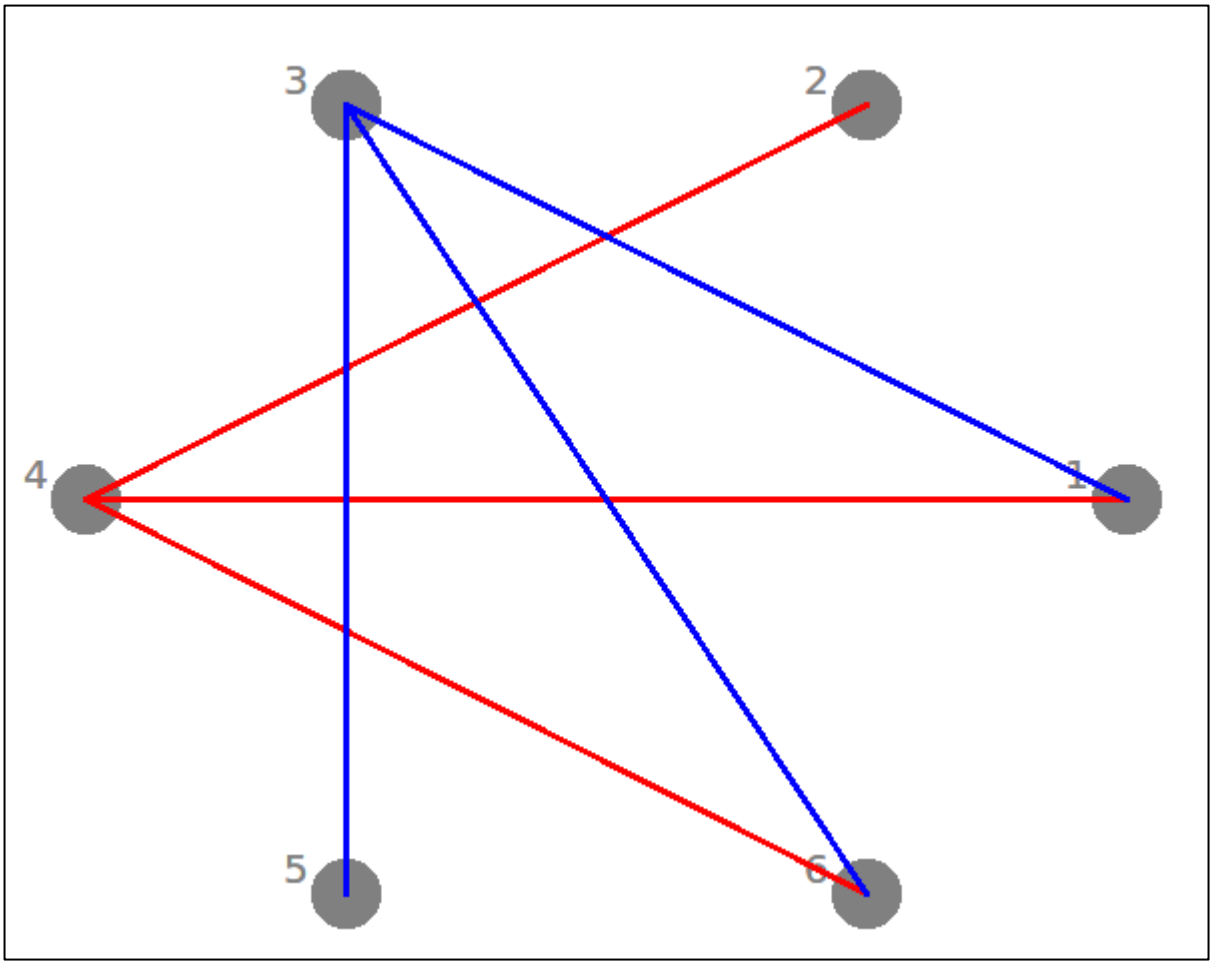
2- برای آشنایی با تابع heuristic پیاده سازی شده از چند مثال شروع می کنیم. حالت زیر را در نظر بگیرید:



در این حالت در صورت اضافه شدن خطوط (1, 2) یا (1, 6) یا (2, 6) یک مثلث قرمز ساخته می شود و بازی کن قرمز بازنده می شود (با فرض این که ما بازیکن قرمز باشیم بازی را می بازیم)

در واقع در این حالت با اضافه کردن یک خط می توانیم 3 تا مثلث بسازیم.

حال شکل زیر را در نظر بگیرید :



در این شکل نیز از راس 4 سه یال خارج شده است و طبق محاسبات گفته شده در قسمت قبل سه حالت وجود دارد که ببازیم. حال با نگاه کردن به راس 3 و دیدن یال های آبی می توان متوجه شد که با اضافه کردن هر یک از یال های $(5, 6)$ یا $(6, 1)$ یا $(1, 5)$ مثلث آبی تشکیل شده و بازیکن قرمز برنده می شود.

در واقع ایده اصلی استفاده شده در تابع heuristic تمرکز بر روی تعداد مثلث هایی که می توان با اضافه کردن یک یال ساخت است.

با فرض این که ما بازیکن قرمز باشیم، بدیهیست که اگر بتوان با اضافه کردن یک یال تعداد بیشتری مثلث آبی و تعداد کمتری مثلث قرمز ساخت به نفع ماست. بنابراین می توان تابع هیوریستیک را به این شکل تعریف کرد :

$$h(n) = w_1 f_1 + w_2 f_2$$

f_1 : تعداد مثلث های آبی که می توان با اضافه کردن یک یال به وجود آورد.

f_2 : تعداد مثلث های قرمز که می توان با اضافه کردن یک یال به وجود آورد.

w_1 : وزن مربوط به f_1 . با توجه به این که زیاد بودن تعداد مثلث های آبی به نفع ماست، این وزن برابر با 1 است.

w_1 : وزن مربوط به f_2 . با توجه به این که زیاد بودن تعداد مثلث های قرمز به ضرر ماست، این وزن برابر با -1 است.

در نتیجه تابع heuristic به این شکل تعریف می شود :

$$h(n) = f_1 - f_2$$

در واقع تعداد مثلث های آبی که می توانیم با اضافه کردن یک یال بسازیم را منهای تعداد مثلث های قرمز می کنیم تا شانس برنده شدن خود را به دست آوریم. اگر این مقدار بزرگ تر از صفر باشد به این معناست که تعداد مثلث های آبی که می توان با اضافه کردن یک یال به دست آورد بیشتر از تعداد مثلث های قرمز است و این یعنی شانس مثبتی برای برنده شدن داریم و در صورتی که این مقدار منفی باشد، شانس کمی برای برنده شدن داریم چون تعداد مثلث های قرمزی که می توانیم با اضافه کردن یک یال بکشیم از تعداد مثلث های آبی بیشتر شده است.

دقت کنیم این مقدار در صورتی محاسبه و برگردانده می شود که در بازی برنده مشخص نباشد، در صورتی که برنده مشخص باشد، تابع $h(n)$ این گونه تعریف می شود :

- اگر برنده بازیکن قرمز باشد (ما برنده شدیم)، $h(n) = +\infty$
- برنده بازیکن آبی باشد (ما بازنده شدیم)، $h(n) = -\infty$

برای محاسبه تعداد مثلث های آبی از ایده زیر استفاده می کنیم (برای مثلث های قرمز نیز به طریق مشابه عمل می کنیم) :

- ابتدا یال های آبی خروجی از هر راس را محاسبه می کنیم.
- فرض کنید از راس N_x ، تعداد M_x تا یال آبی خارج شده است.
- تعداد مثلث هایی که می توان با M_x یال ساخت عبارت است از :

$$\binom{M_x - 1}{2} = \frac{M_x(M_x - 1)}{2}$$

- این مقدار را برای تمامی رئوس محاسبه کرده و مجموع تمام آنها، تعداد مثلث های آبی را که با اضافه کردن یک یال می توانیم به دست آوریم محاسبه می کند.

نتایج بدون هرس آلفا و بتا :

```
amirhossein@amirhossein-u:~/Desktop/CA2$ python3 main.py 1 0
Depth : 1 & Number Of Play Game(n) :  200
Winning Chance :  0.905
Mean Execution Time :  0.0035987019538879393
Mean Seen States :  55
amirhossein@amirhossein-u:~/Desktop/CA2$ python3 main.py 3 0
Depth : 3 & Number Of Play Game(n) :  200
Winning Chance :  0.89
Mean Execution Time :  0.36434776067733765
Mean Seen States :  5317
amirhossein@amirhossein-u:~/Desktop/CA2$ python3 main.py 5 0
Depth : 5 & Number Of Play Game(n) :  10
Winning Chance :  1.0
Mean Execution Time :  33.8306407213211
Mean Seen States :  479318
amirhossein@amirhossein-u:~/Desktop/CA2$
```

نتایج با استفاده از هرس آلفا و بتا :

```
amirhossein@amirhossein-u:~/Desktop/CA2$ python3 main.py 1 0
Depth : 1 & Number Of Play Game(n) :  200
Winning Chance :  0.875
Mean Execution Time :  0.0036634492874145 51
Mean Seen States :  55
amirhossein@amirhossein-u:~/Desktop/CA2$ python3 main.py 3 0
Depth : 3 & Number Of Play Game(n) :  200
Winning Chance :  0.9
Mean Execution Time :  0.3487417423725128
Mean Seen States :  5149
amirhossein@amirhossein-u:~/Desktop/CA2$ python3 main.py 5 0
Depth : 5 & Number Of Play Game(n) :  10
Winning Chance :  1.0
Mean Execution Time :  33.26696894168854
Mean Seen States :  464940
amirhossein@amirhossein-u:~/Desktop/CA2$
```

سوالات :

1– تابع heuristic خوب باید به استیت هایی که شانس برنده شدن ما بیشتر است امتیاز بیشتر و نزدیک تر به واقعیت را بدهد. همچنین باید محاسبات لازم برای پیش بینی امتیاز هر استیت را در زمان معقول و مناسبی انجام دهد. علت انتخاب این تابع در این مسئله این است که پیش بینی نسبتا خوبی از برنده شدن ما دارد و ارتباط مستقیمی با تعداد مثلث هایی که می توان در حرکت بعدی ساخت دارد. همچنین به وضعیت های مختلف، امتیاز های مختلفی می دهد و تمامی خطوط را در نظر می گیرد. همچنین در حالاتی که برنده مشخص است مقادیر مناسب بی نهایت را بر می گرداند.

2- تاثیر دارد. با افزایش عمق جست و جو شانس ما برای برد بیشتر شده ولی تعداد استیت های دیده شده و زمان اجرا به طور نمایی افزایش می یابد.

3- بله، با توجه به این که با دیدن یک راس ممکن است زیر درخت کناری حذف شود، ترتیب اهمیت دارد. به طور رندوم انتخاب می کنیم تا بازی خیلی قابل پیش بینی نشود و خطوط در اطراف چند راس متمرکز نشوند.