به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

**هوش مصنوعی پاییز 98**

**پروژه دو**

**بازی**

**نام و نام خانوادگی**

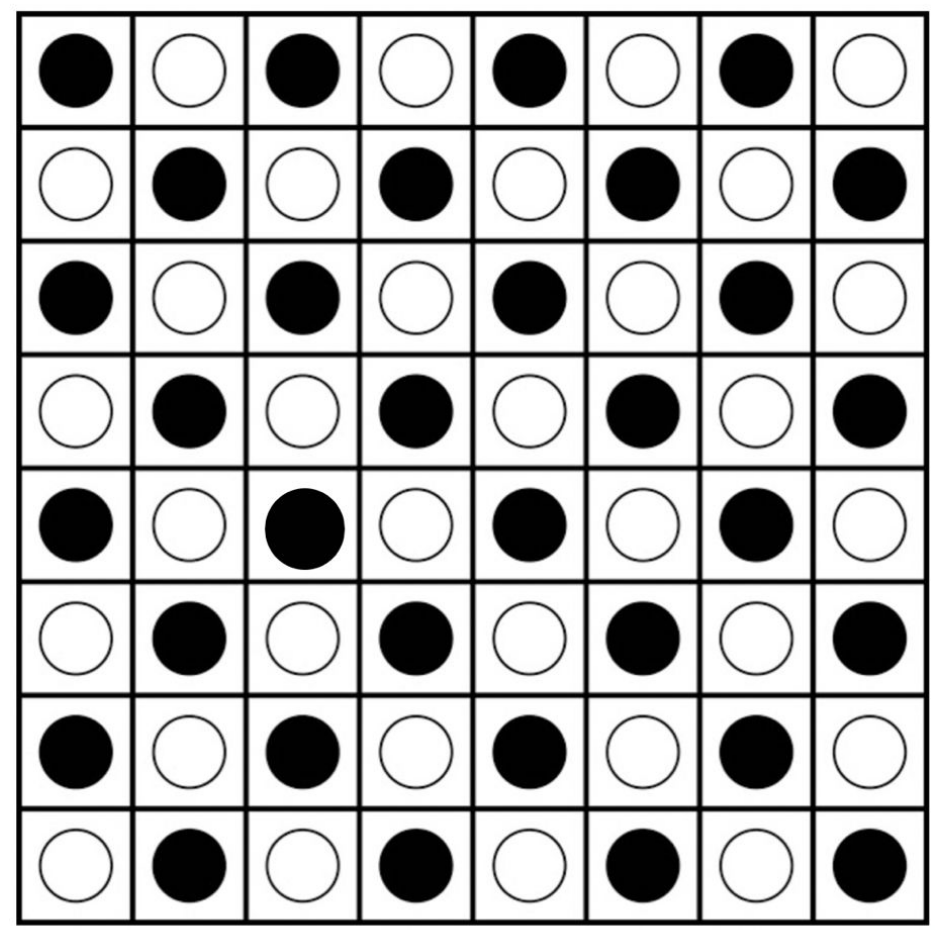
**علیرضا زارع نژاد اشکذری**

**شماره دانشجویی**

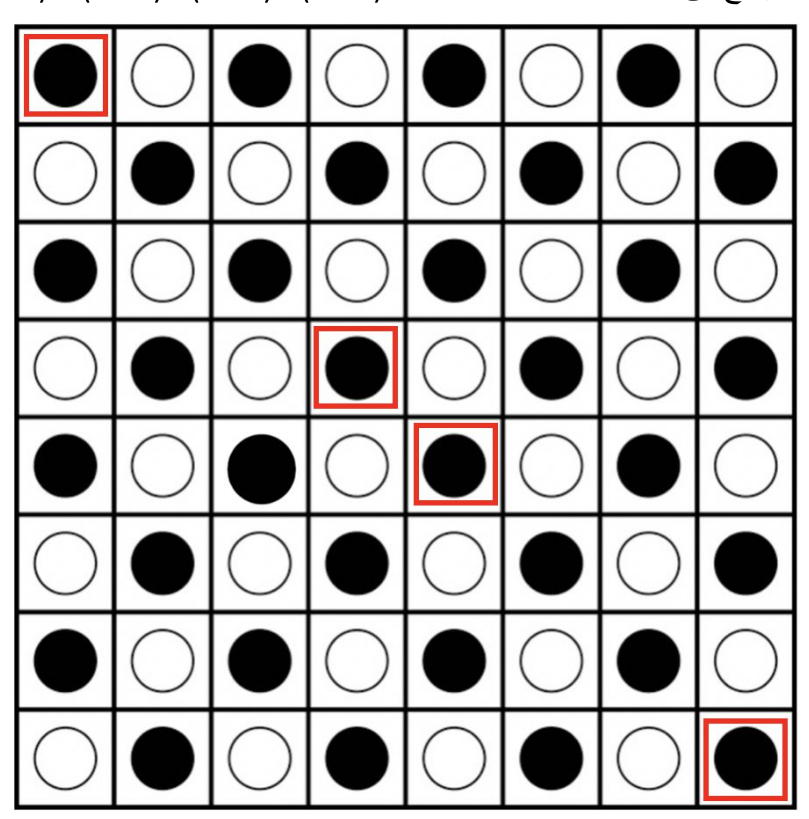
**810196474**

# شرح مختصر پروژه

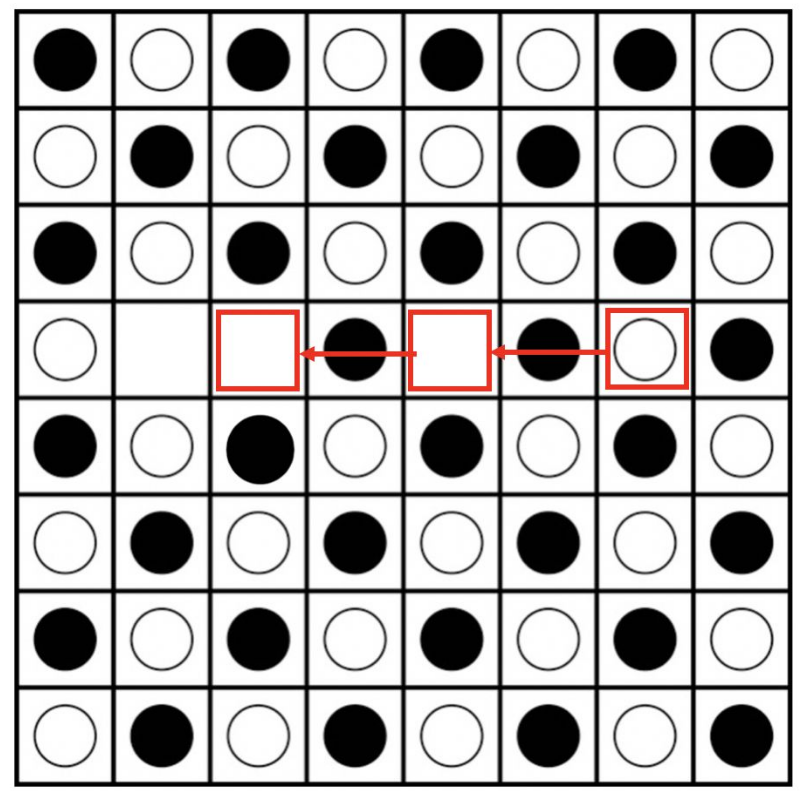
در این بازی یک صفحه ۸\*۸ شامل مهره‌های سفید و سیاه به شکل زیر داده شده است.

‬

در ابتدا بازیکن اول که پیش‌فرض بازیکن سیاه مد نظر است یکی از مهره‌های خانه‌های مشخص شده زیر را حذف می کند.



در حرکت بعدی بازیکن مقابل می بایست یکی از خانه‌های مجاور خانه ی حذف شده در مرحله قبل را حذف کند. سپس بازی آغاز می‌شود. در هر مرحله بازیکن می‌تواند مهره خود را از روی مهره حریف عبور دهد و مهره ی حریف را حذف نماید به شرط آنکه فضای خالی موجود باشد. همچنین به شرط آنکه مسیر حرکت عوض نشود بازیکن می‌تواند چندین بار حرکت کند.به عنوان مثال شکل زیر را در نظر می گریم.

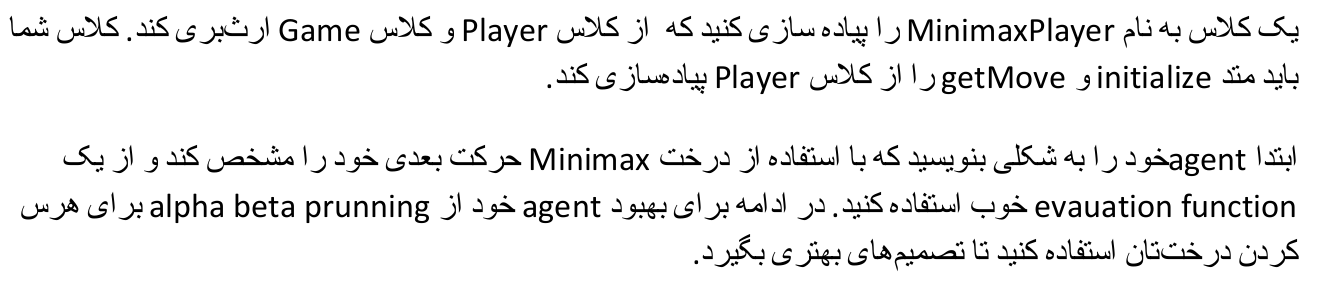


بازی زمانی تمام می‌شود که بازیکن حرکتی برای انجام نداشته باشد.

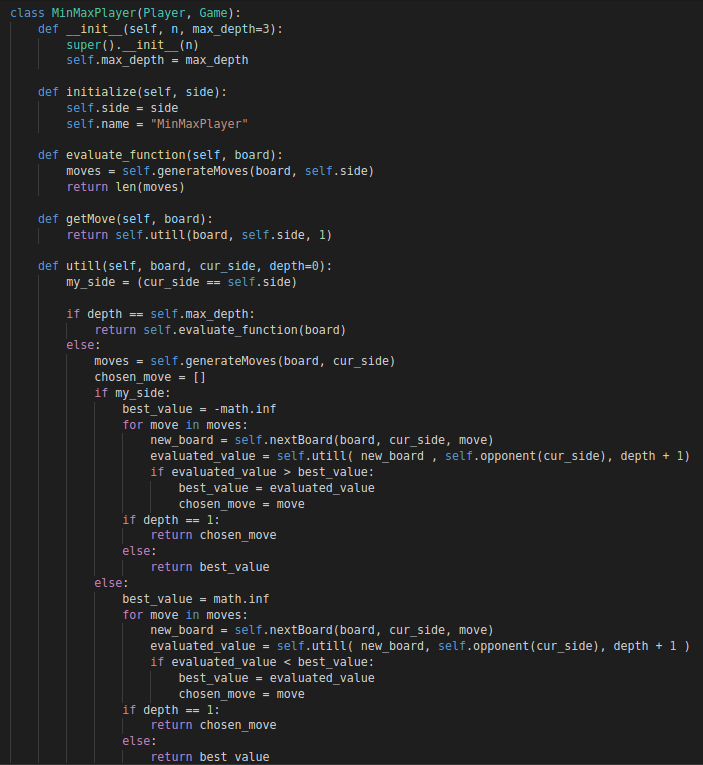
# بررسی کد ها و توابع آماده در پروژه



# گام پروژه

لذا تنها کافی است دو کلاس گفته شده را طراحی کنیم و روند آن در زیر توضیح داده شده است.

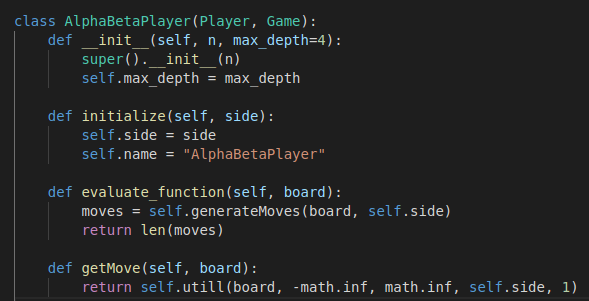
# کلاس **MinmaxPlayer**

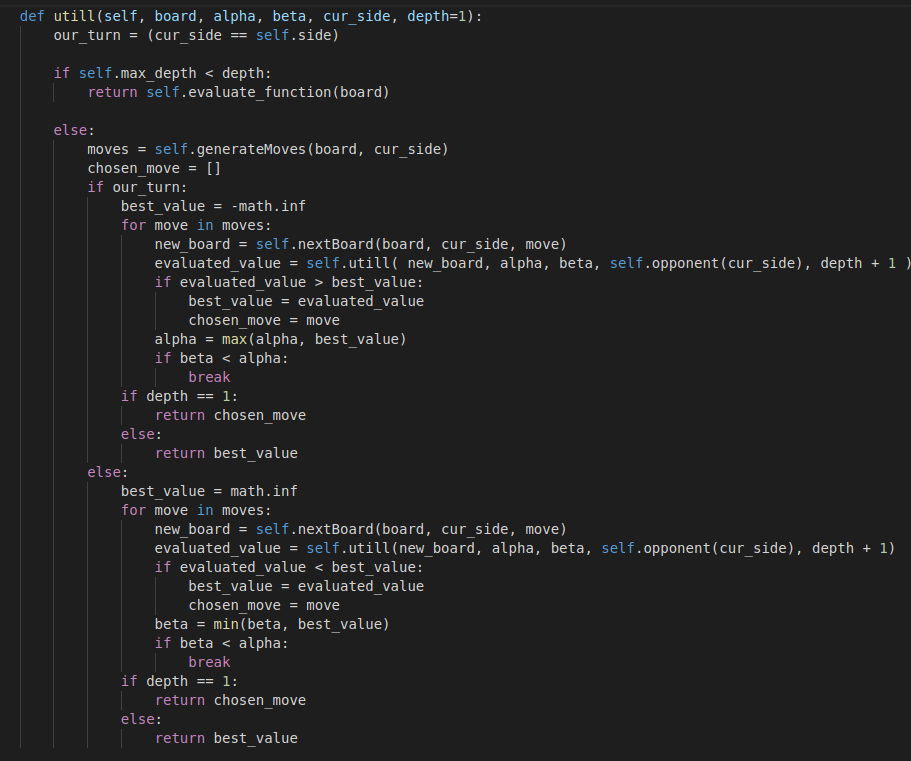


با توجه به قطعه کد بال برای agent مان constructor‌ای بمی نویسیم که حداکثر depth را در زمان ساختن آن به constructor بدهیم. زمان ساختن درخت‌، آن را تا این ارتفاع ساخته و زمانی که به این عمق رسیدیم دیگر درخت را نمی‌سازیم و از evaluation function مناسب استفاده می‌کنیم.این عمق در متغیر self.depth‌ ذخیره شده است.در مورد evalution function نیز تعداد حرکات ممکن مرحله مورد نظر نعریف می کنیم.

در تابع utill نیز که مشخص می‌کند با متغیر cur\_side که الان در لول max هستیم یا در لول min و حال اگر به عمق مورد نظر تعریف شده رسیده باشیم یعنی به برگ‌ها کافی است که مقدار eval function برگردانده شود در غیر این صورت باید اگر در لول max هستیم ماکسیمم مقدار node های بچه را برگردانده و آن حرکت را انجام دهیم و اگر در لول min هستیم مینیمم مقدار node های بچه یا به عبارتی حرکت‌های ممکن را برگردانیم.که برای ای کار هر بار new\_booard جدید را می‌سازیم و نوع بازیکن را عوض می‌کنیم و depth را یکی زیاد می کنیم. پس از ماکسیمم و مینیمم گیری مناسب حرکت مناسب برگردانده می شود.

# **کلاس AlphaBetaPlayer**





برای این بخش دو پارامتر alpha و bata را به تابع مورد نظر که مشابه minmax هست می‌دهیم. آلفا نشان دهنده ی بیشترین مقدار utility پیدا شده برای بازیکن سیاه و بتا کمترین مقدار utility پیدا شده برای بازیکن سفید است.حال اگر در پیمایش یک نود‌ ( برای مثال برای آلفا‌) مقدار utility یکی از برگ‌های آن نود از الفا کمتر باشد می‌توانیم از پیماش سایر برگ‌های آن نود صرف نر کنیم چرا که در این حالت نود مینیمم مقدار برگ‌های خود را به عنوان یوتیلیتی انتخاب می‌کند و این مقدار عددی کمتر از بهترین یوتیلیتی پیدا شده برای آن نود (الفا) می‌باشد پس بنابراین پیمایش سایر برگ‌های آن نود تأثیری مثبتی نخواهد داشت.

**آیا حرکات انجام این دو الگوریتم با یکدیگر متفاوت است‌؟**

خیر – اگر عمق آن‌ها با هم یکی باشد حرکات آن‌ها با هم فرقی ندارد. زیرا تنها کاری که در الگوریتم alpah-beta انجام می‌دهیم این است که حرکاتی که به اندازه حرکات اصلی خوب نیستند را حذف می‌کنیم نه اینکه حرکات درست تا آن عمق را حذف کنیم.

**زمان اجرایی دو الگوریتم را با یکدیگر مقایسه کنید‌؟**

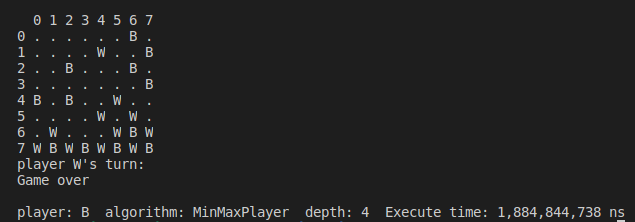
زمان اجرایی الگوریتم minmax به eval function وابسته است همچنین پس از هرس شدن شاخه‌های اضافی در alpha beta این مقدار زمانی در این الگوریتم کاسته می شود.

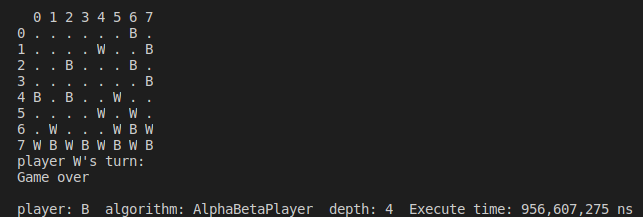
در اینجا حالت‌های مختلف بازی را تست کرده و زمان آن را بررسی می کنیم.

Eval را نهایتاً تعداد حرکت‌های خود منهای تعداد حرکت‌های حریف کردم.

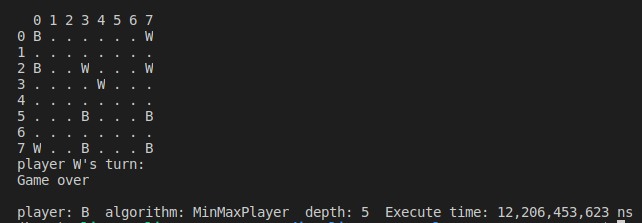
در ابتدا خروجی به ازای عمقهای مختلف در دو حالت minmax و alpha beta با بازیکن simple player و زمان آن آورده می شود.

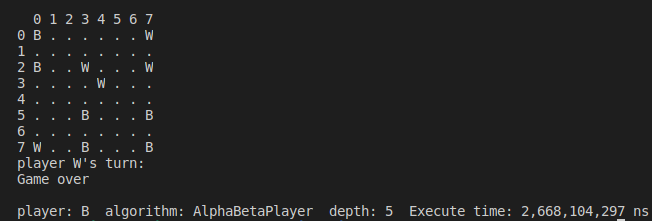
**Depth = 4 , player1 : minmax , player2 : simple:**

****

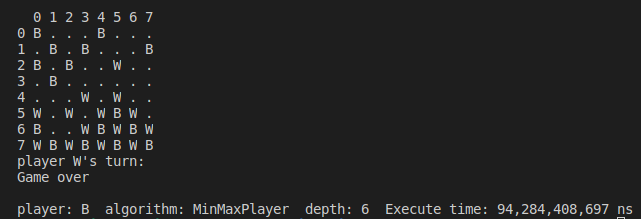
****

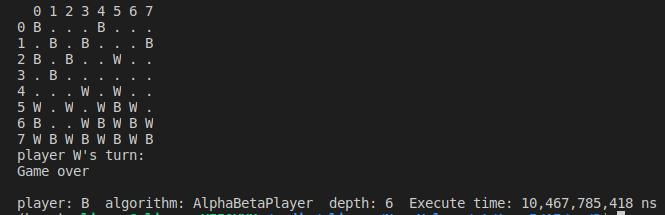
**depth = 5 , player1 : minmax , player2 : simple :**





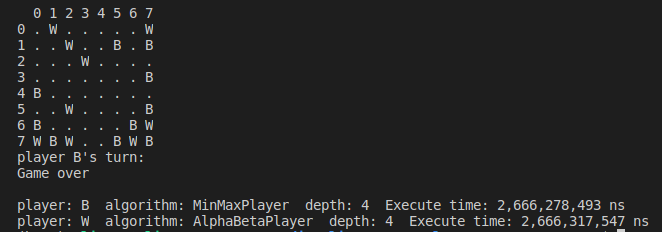
**depth = 6 , player1 = minmax , player2 = simple**



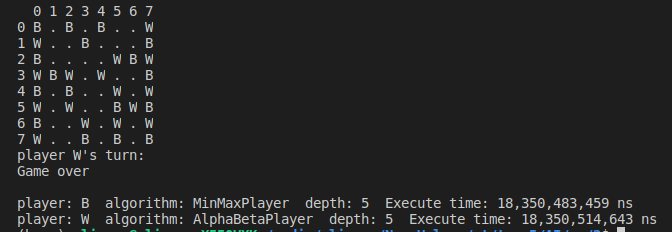


حال حالتی را در نظر بگیریم که بازیکن اول mimmax player باشد و بازیکن دوم alpha beta و به ازای عمق های مختلف خروجی به صورت زیر است

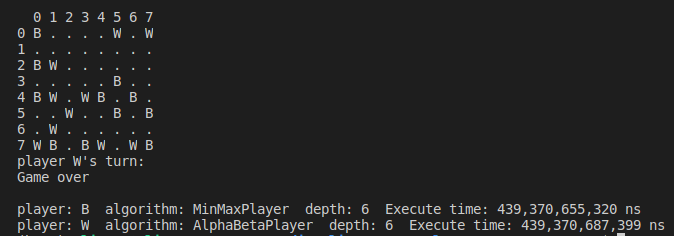
, **player1 : minmax , player2 : alphabeta , depth = 4 :**



**depth = 5 , player1 :minmax , player2 : alphabeta :**

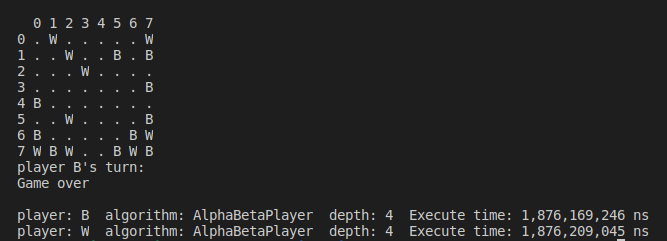


**depth = 6 , player1 : minmax , player2 : alphabeta :**

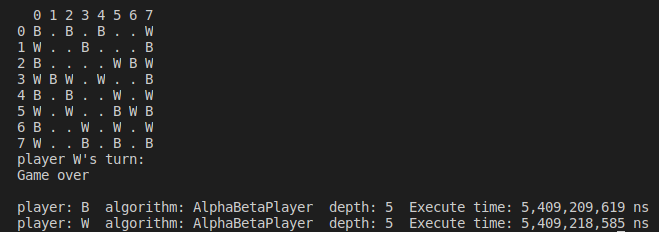
****

حال حالتی در نظر بگیریم که جفت بازیکن ها از alpha beta استفاده کنند.

**Depth = 4 :**

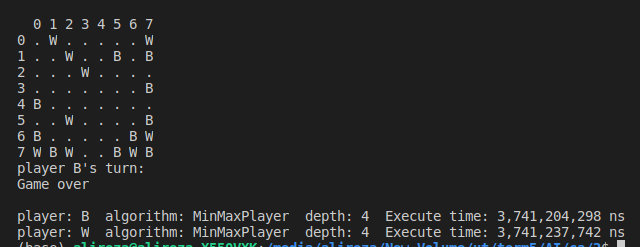
****

**depth = 5** :



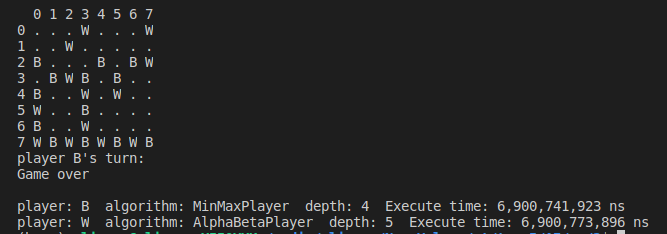
و در صورتی که جفت بازیکن ها از minmax استفاده کنند:

**depth = 4**



و نهایتاً حالتی که عمق جست‌و‌جو فرق کند:

**player1 : minmax , depth1 = 4 , player2 : alphabeta , depth2 = 5:**

****

واضح است هر کدام که عمق بیشتری داشته باشند برنده می باشد.

**پایان**