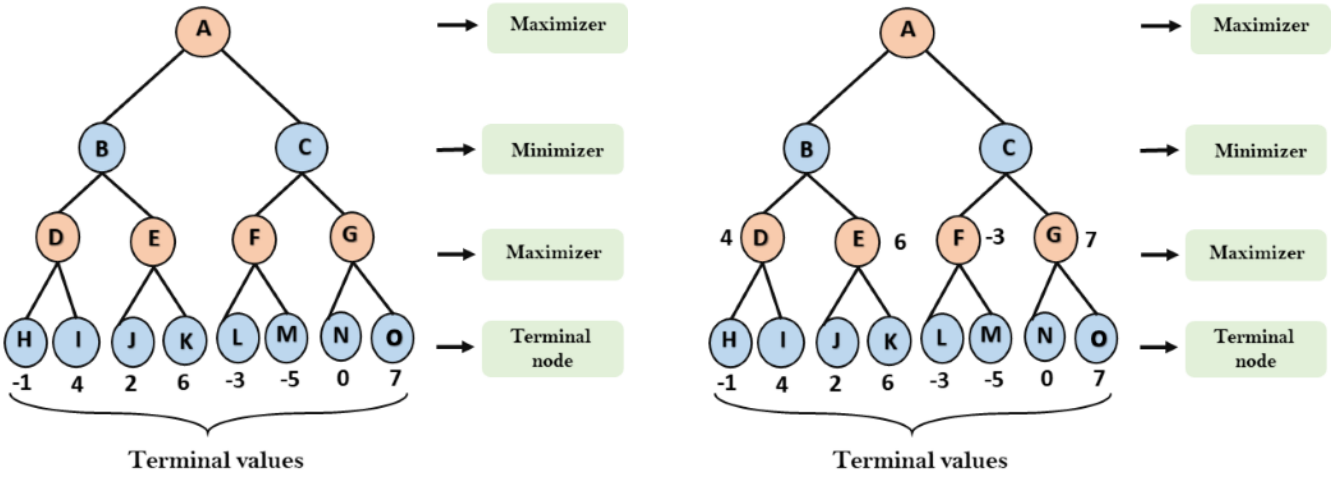
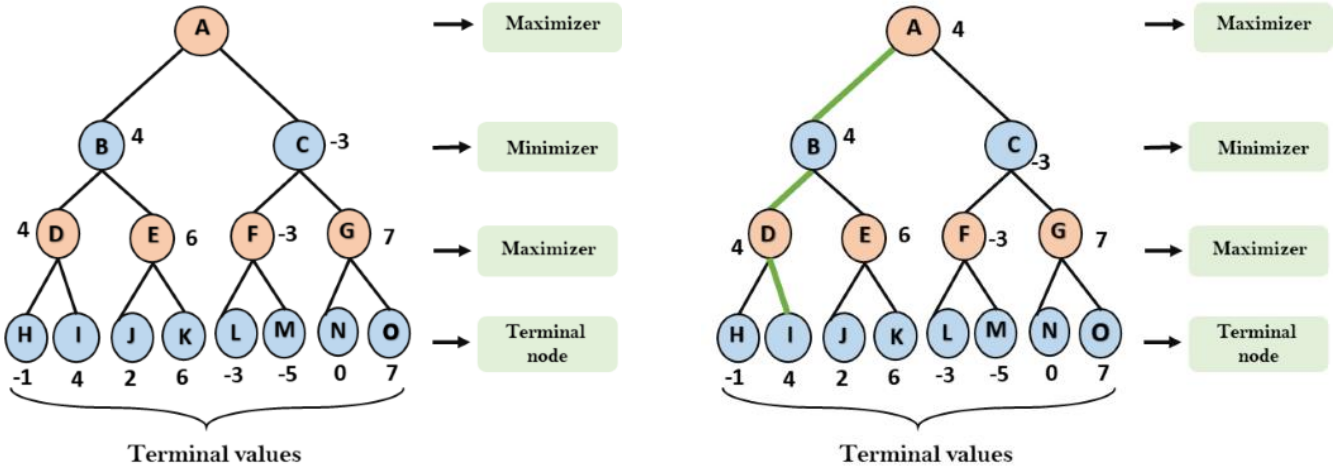
# گزارش پروژه میان فصل هوش مصنوعی


## محمدعندلیبی – محمدمهدی محمدی

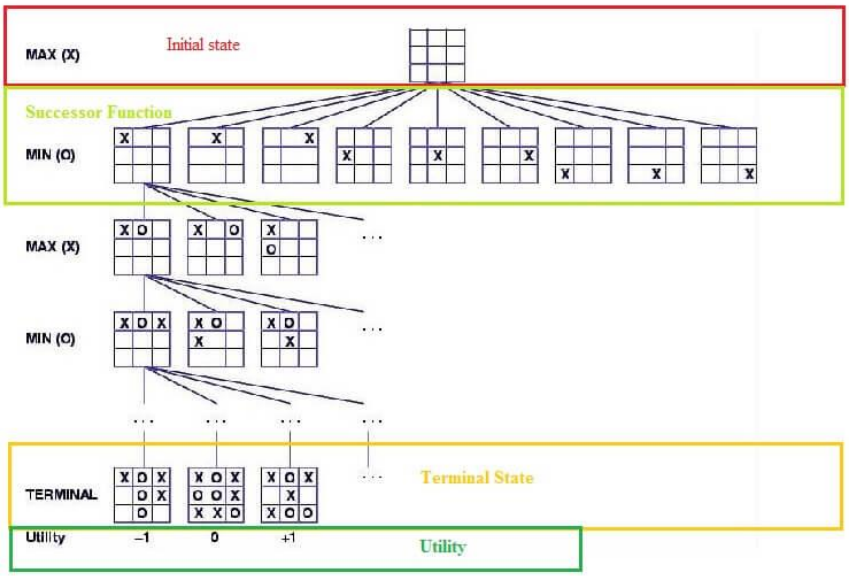

**الگوریتم Mini-Max :** در مسئله ی جستجوی عادی، جواب بهینه، دنباله ای از فعالیت هاست که منجر به حالت هدف میشوند. در جست وجوی خصمانه، MIN چیزی برای گفتن درباره حالت هدف دارد. بنابراین، MAX باید یک استراتژی اقتضایی را پیدا کند که حرکت MAX را در حالت شروع مشخص می کند، سپس حرکت های MAX در این حالت ها با توجه به پاسخ های MIN مشخص میشود، و سپس MAX به حالتهایی میرود که با پاسخ MIN به آن حرکتها بستگی دارد و این روند ادامه می یابد بطوریکه MAX در نقش OR و MIN در نقش AND بازی میکند. به عبارت دقیقتر استراتژی بهینه نتایجی تولید میکند که حداقل به خوبی استراتژی های دیگر در زمانی است که با یک حریف بدون اشتباه بازی میکند.تعریف بازی بهینه برای MAX فرض میکند که MIN نیز بطور بهینه بازی میکند، یعنی نتیجه بدترین حالت برای MAX را ماکزیمم میکند. اگر MIN بطور بهینه عمل نکند، به آسانی میتوان نشان داد که MAX هنوز بهتر بازی خواهد کرد.

الگوریتم **minimax** تصمیم **minimax** را از حالت فعلی محاسبه میکند. این الگوریتم مقادیر minimax مربوط به هر حالت پسین را بطور بازگشتی با پیاده سازی مستقیم معادلات تعریف شده، محاسبه میکند. این محاسبات بازگشتی به طرف پایین تا برگهای درخت پیش میرود و سپس در برگشت از حالت بازگشتی، مقادیر minimax هر گره را اختصاص میدهد.
الگوریتم minimax اکتشاف عمقی کاملی را روی درخت بازی انجام میدهد. اگر حداکثر عمق درخت m باشد و در هر نقطه b حرکت معتبر وجود داشته باشد، آنگاه پیچیدگی زمانی الگوریتم برابر است با **O(bm)** . برای الگوریتمی که تمام فعالیت ها را همزمان تولید میکند، پیچیدگی فضا (حافظه) برابر با **O(bm)**، یا برای الگوریتمی که هر بار یک فعالیت را تولید میکند، برابر با **O(m)** است. البته برای بازی های واقعی هزینه زمان کاملا غیرعملی است.






**اما این الگوریتم به عنوان مبنایی برای تحلیل ریاضی بازی ها و برای بسیاری از الگوریتم های عملی است.درچهار شکل بالا گذرکلی این الگوریتم با حالت نود هایی که هریک دارای فرزندانی هستند نمایش داده شده است اما مساله اصلی این است که در این پروژه هریک از نودهای ما یه حالت از مساله میباشد و بدین ترتیب به صورت دقیق تر داریم که :**




**بحث کامل بودن:** کامل است، اگر درخت متناهی باشد

**بحث بهینگی:**

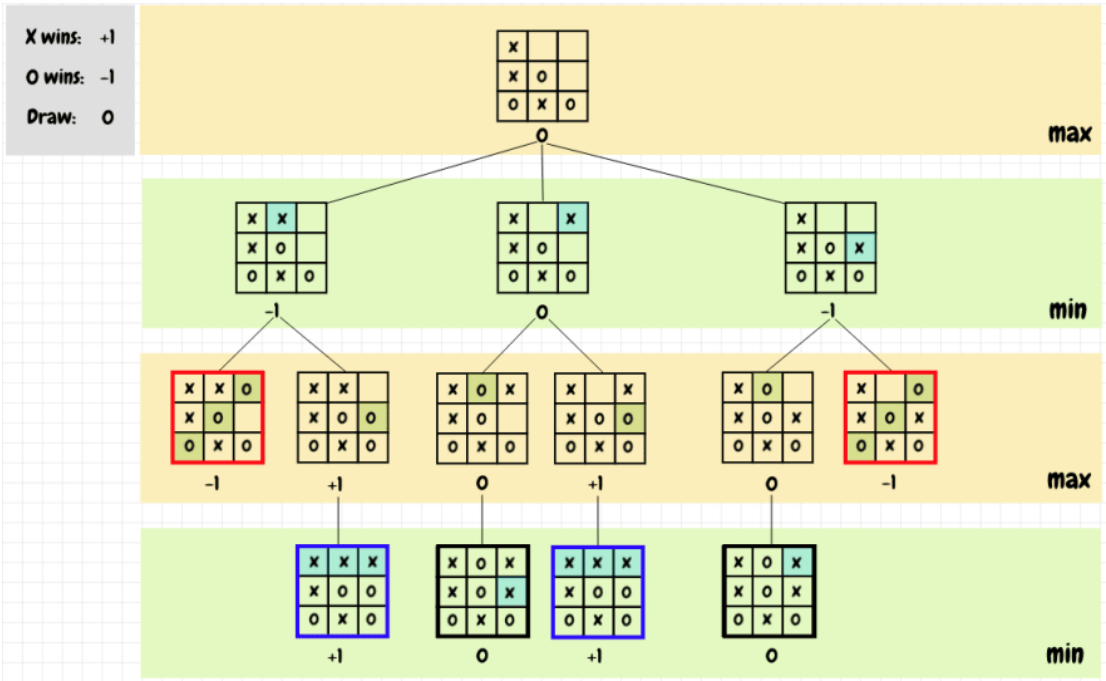
- در مواجهه با رقیب بهینه(هوشمند) بهینه است
- در مواجهه با رقیب غیرهوشمند، بهینه نیست، اما شکست هم نمی خورد.

| | Random | Minimax |
|---|---|---|
| **Random** | X win : 59.6%<br>O win : 26.7%<br>Draw : 13.7% | X win : 0%<br>O win : 79.7%<br>Draw : 20.3% |
| **Minimax** | X win : 99.4%<br>O win : 0 %<br>Draw : 0.6% | X win : 0%<br>O win : 0%<br>Draw : 100% |

**مشکل اصلی پیاده سازی این الگوریتم زمان است. درخت گسترش داده شده این بازی بسیار بزرگ است و این مساله ای بود که آقای راسل نیز آن را پیش بینی کرده بودند و در متن کتاب آمده است که اگر تعداد حالت های بازی برابر ۹ باشد و اگر کامپیوتر شروع کننده بازی باشد برابرست با ۳۶۲۸۸۰ = !۹.**

| node | branch |
|---|---|
| 3,274,668 | 4,010,368 |

**فارغ از بحث احتمال پیروزی که در جدول بالا مطرح شد ، داده های حاصل از این پروژه برای قسمت مینی مکس مطابق جدول روبروست.**



## الگوریتم هرس کردن آلفا – بتا

مشکل جست وجوی **minimax** این است که تعداد حالتهای بازی که باید بررسی شوند، برحسب تعداد حرکتها، یک رابطه ی نمایی است. متاسفانه این رابطه نمایی را نمیتوان حذف کرد ولی میتوان آنرا به نصف تقلیل داد. علتش این است که محاسبه تصمیم minimax صحیح بدون دیدن همه گره های درخت بازی امکانپذیر است. یعنی با استفاده از مفهوم هرس کردن میتوان بخش های بزرگی از درخت را از بین برد. با استفاده از تکنیکی به نام هرس کردن آلفا – بتا میتوان این کار را انجام داد. وقتی این تکنیک به درخت minimax اعمال میشود، همان حرکتی را برمیگرداند که minimax برخواهد گرداند، اما انشعاب هایی که در تصمیم نهایی تاثیر ندارند، حذف خواهند شد.

هرس کردن آلفا – بتا که یکی از استراتژی های جستوجوی خصمانه است، میتواند به هر درختی با هر عمق اعمال شود. بطور کلی میتوان کل زیردرخت را به جای برگ ها هرس کرد. اصل کلی این است : گره n را در جایی از درخت در نظر بگیرید بطوریکه بازیکن بتواند به آن گره برود. اگر بازیکن، در گره والد n یا هر نقطه ای در سطح بالاتر، انتخاب بهتری مثل m دارد، آنگاه در بازی واقعی هرگز به n نخواهد رسید. لذا وقتی اطلاعات کافی در مورد n کسب کردیم (با بررسی بعضی از فرزندان آن) تا به این نتیجه برسیم، میتوانیم آنرا هرس کنیم.

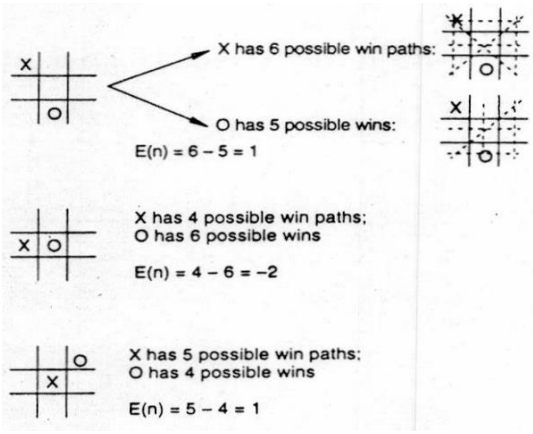
به یاد داشته باشید که جست وجوی minimax عمقی است، لذا در هر زمان فقط باید گره های موجود در یک مسیر درخت را در نظر بگیریم. نام هرس کردن آلفا – بتا از دو پارامتر زیر به دست آمده که کران مقادیر ذخیره شده در تمام طول مسیر هستند:

**α** = مقدار بهترین (یعنی بالاترین) انتخابی است که تاکنون در هر نقطه انتخاب در مسیر مربوط به **MAX** پیدا شده است.

**β** = مقدار بهترین (یعنی پایین ترین) انتخابی است که تاکنون در هر نقطه انتخاب در مسیر مربوط به **MIN** پیدا شده است.

جست وجوی آلفا – بتا ، مقادیر α و β را با هرس کردن انشعاب های باقیمانده در یک گره، به هنگام سازی میکند. این کار به محض اینکه مشخص شد مقدار گره فعلی بدتر از مقدار α یا β مربوط به MAX یا MIN است، انجام میگیرد.

اثر بخشی هرس کردن آلفا – بتا به ترتیب بررسی حالت ها بستگی دارد پس بهتر است ابتدا پسین هایی بررسی شوند که ممکن است بهترین باشند. اگر فرض کنیم این کار امکانپذیر باشد، نتیجه میگیریم که در آلفا – بتا فقط $O(b^{m/2})$ گره باید بررسی شوند تا بهترین انتخاب صورت گیرد در حالیکه در minimax، این تعداد $O(b^m)$ بوده است.

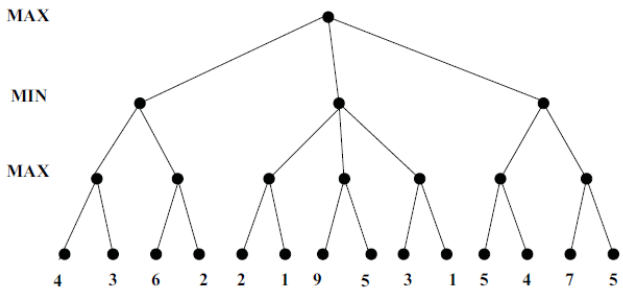



با اضافه کردن طرح های تعیین حرکتها بصورت پویا، مثل طرح هایی که سعی میکنند اول حرکتهایی را انجام دهند که در گذشته بهترین بودند، ضریب انشعاب به مقداری که در محاسبات نظری به دست می آید، نزدیک میشود. حرکت گذشته، میتوانست حرکت قبلی باشد یا میتوانست از اکتشاف قبلی مربوط به حرکت فعلی تعیین شود.

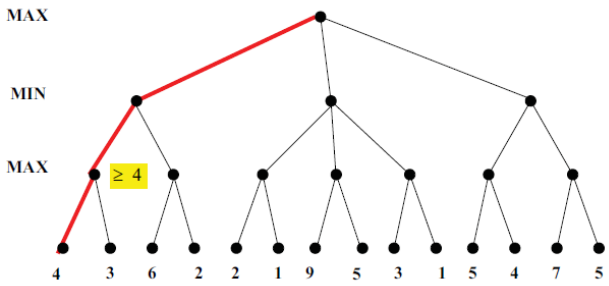
تابع ارزیابی :

$$E(n) = M(n) - O(n)$$

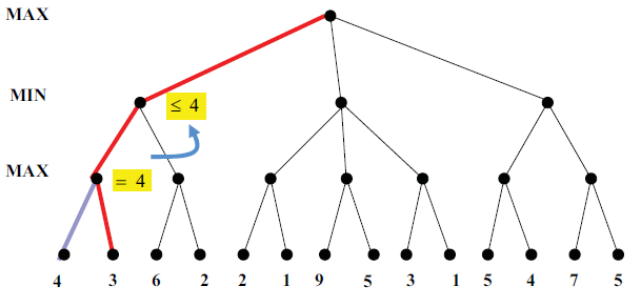
## Alpha beta pruning. Example



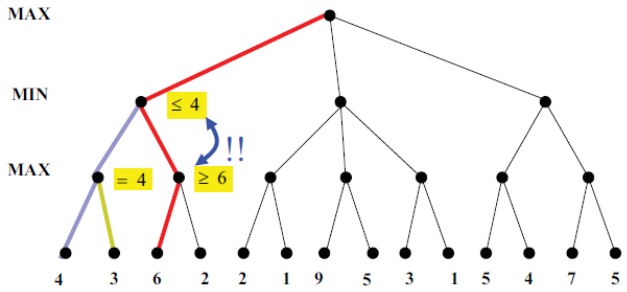

## Alpha beta pruning. Example



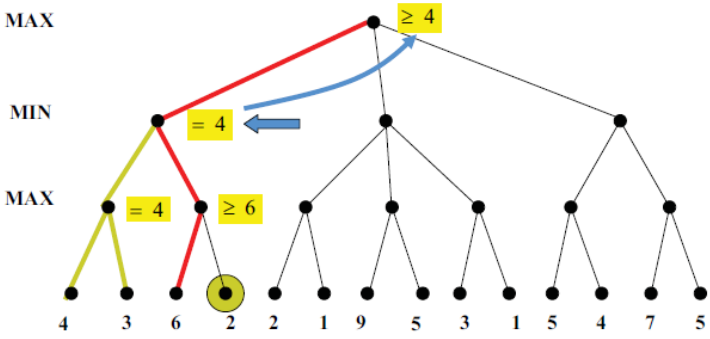

## Alpha beta pruning. Example



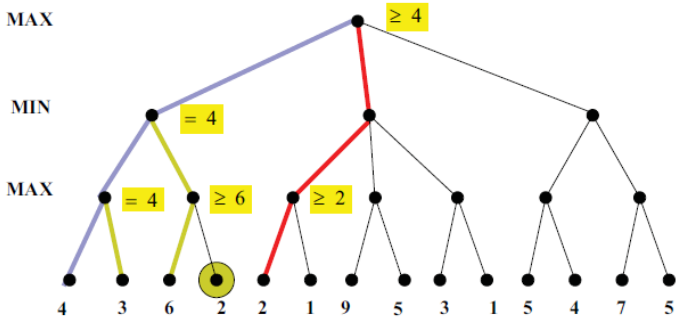

## Alpha beta pruning. Example



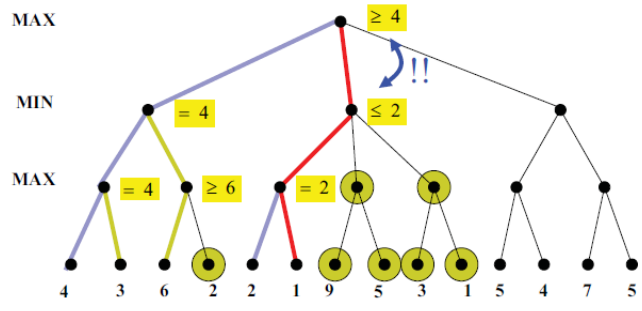

## Alpha beta pruning. Example



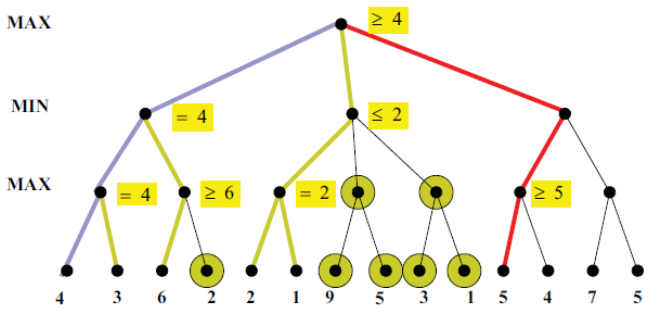

## Alpha beta pruning. Example



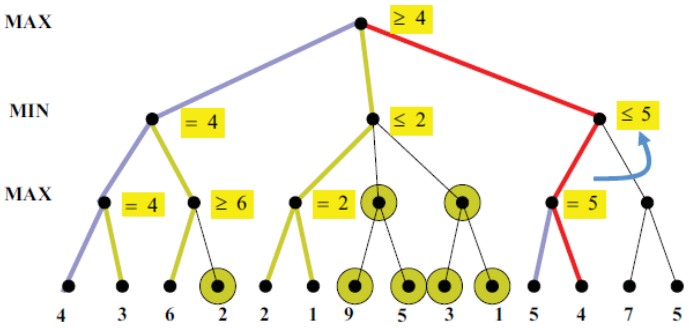
## Alpha beta pruning. Example



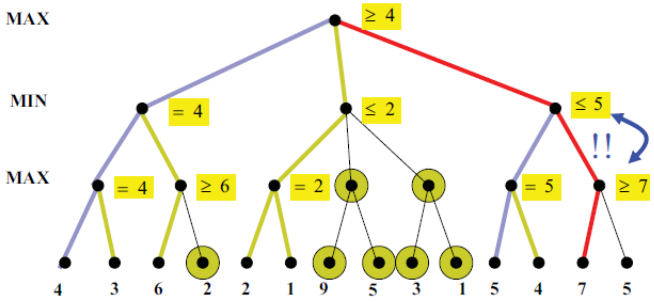

## Alpha beta pruning. Example



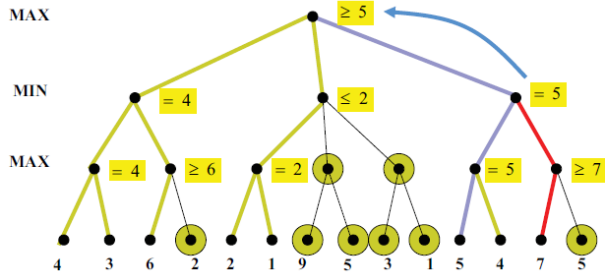

## Alpha beta pruning. Example



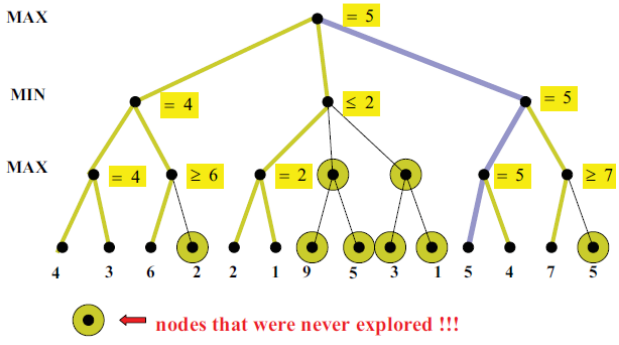

## Alpha beta pruning. Example




## Alpha beta pruning. Example




## Alpha beta pruning. Example



در این پروژه ما با بررسی هایی که انجام شد عمق محدود شده را با توجه به نود هایی که گسترش یافته اند برابر ۵ درنظرگرفتیم و از طرفی دیگر برای تابع ارزیابی نیز به گونه ای عمل کردیم که در صورتی که با انجام آن حرکت پیروزی حاصل میشد مقدار یک ارایه شود و در صورتی که به انتها برسیم و مهره آخر را گذاشته و هنوز حالت خاصی اتفاق نیفتاده باشد مقدار صفر را برمیگردانیم و این به منزله حالت مساوی میباشد.

ضمنا اطلاعات اجرای برنامه برای پیاده سازی الگوریتم آلفا-بتا مطابق جدول زیر میباشد.

| | Computer vs random | Computer vs computer |
|---|---|---|
| node | 11784 | 1528320 |
| branch | 12324 | 1952916 |

نهایتا مساله ای که این پروژه برآن تاکید داشت بیان این موضوع است که باوجود کامل بودن و بهینگی الگوریتم مینی مکس ، این الگوریتم بدون محدود کردن عمق و بحث هرس کردن امکان پیاده سازی در دنیای واقعی را دارا نمیباشد.

## منابع :


https://www.javatpoint.com/mini-max-algorithm-in-ai

https://www.edureka.co/blog/alpha-beta-pruning-in-ai

https://www.cs.rochester.edu/u/brown/242/assts/studprojs/ttt10.pdf

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/