بسمه تعالی

Artificial Intelligence
Project ۲ Report

Mohammad Mahdi Islami
۸۱۰۱۹۵۵٤۸

شرح کلیات

در وهلـه ی اول بایـد درخـت بـازی را تشـکیل دهـیم. بـه همـین منظـور بـه دنبـال سـاختاری مناسـب بـرای نگهداری درخت بازی میگردیم.

سـاختاری کـه در نظـر گرفتـه شـد بـدین صـورت اسـت کـه در یـک list کـه در بـه صـورت پـیش فـرض در python قرار دارد استفاده میکنیم و درایه های این لیست را به این صورت پر میکینم

GameTree = [[NodeIndex , Level , Chidren [] , Score] …]

سـپس در یـک فایـل حالـت بـرد بـازی را کـه متنـاظر بـا یـک Node در GameTree اسـت ماننـد مثـال زیر ذخیر میکنیم:

[W , B , . , W , B , B , W , B …]

و بـا اضـافه شـدن هـر گـره جدیـد ایـن درخـت را آپـدیت میکنـیم. در واقـع از الگـوریتم DFS بـرای تشـکیل درخت بازی استفاده میکنیم.

مرحلـه دوم مرحلـه امتیـازدهی اسـت کـه بایـد بـه گـره هـای درخـت امتیـاز اختصـاص دهـیم. بـرای ایـن منظـور بـه بـرگ هـای درخـت بـازی بـا اسـتفاده از Evaluation Function یـک امتیـاز اختصـاص میدهیم.

دو الگوریتم Minimax و Alpha-Beta Pruning در امتیاز دهی به ما کمک میکنند.

الگوریتم Minimax

نـود هـای بـرگ را کـه فرزنـدی ندارنـد امتیازشـان برابـر بـا تـابع Evaluation Function در آن نـود اسـت و نودهای پدر این برگ ها به صورت بازگشتی از روی امتیاز فرزندانشان با کمک دو الگوریتم

در الگـــوریتم Minimax هـــر نـــود پـــدر بســـته بـــه level ای کـــه در آن قـــرار دارد (minimizer maximizer, مقدار max یـا min فرزنـدانش را بـه عنـوان محتـوای خـود بـر میگزنیـد و ایـن مرحلـه تـا گره ریشه ادامه پیدا میکند.

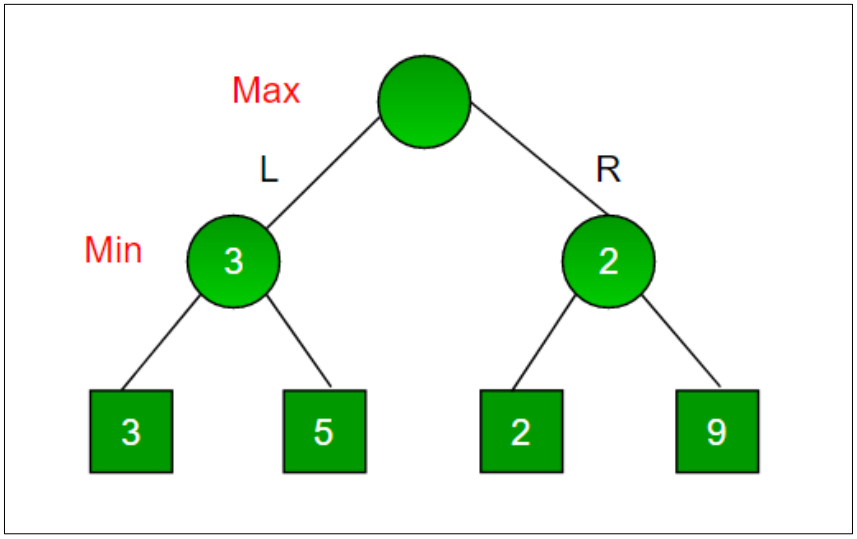

Fig ۱ – Minimax Algorithm

الگوریتم Alpha-Beta

در الگـوریتم Alpha-Beta Pruning هماننـد الگـوریتم Minimax هـر نـود پـدر بسـته بـه level ای کـه در آن قـرار دارد (maximizer ,minimizer) مقـدار max یـا min فرزنـدانش را بـه عنـوان محتوای خود بر میگزنید و این مرحله تـا گـره ریشـه ادامـه پیـدا میکنـد بـا ایـن تفـاوت کـه در اینجـا درخـت لازم نیست میان فرزندان یک پـدر بـه طـور کامـل پیمـایش کنـیم تـا بزرگتـرین یـا کـوچکترین آن هـا را بـه عنوان پاسخ برگردانیم. بلکـه کافیسـت اطمینـان حاصـل کنـیم کـه بزرگتـرین یـا کـوچکترین آن هـا انتخـاب شده اسـت. بـدین منظـور از دو متغیـر a و b اسـتفاده میکنـیم. بـدین منظـور در هـر مرحلـه مقـدار a برابـر inf+ و مقدار b برابر با inf– قرار میگیـرد و بـا مقایسـه ی فرزنـدان بـا ایـن متغیـر هـا مقـادیر آن هـا بـه روز میشـود و سـپس بـا مقایسـه ی ایـن دو مقـدار و بررسـی شـرط **beta <= alpha** بـرای ادامـه ی رونـد پیمـایش تصـمیم گیـری میکنـیم. چنانچـه ایـن شـرط برقـرار بـود از ادامـه دادن خـود داری میکنیم در غیر این صورت آنقد ادامه میدهیم تا این شرط ارضا شود.

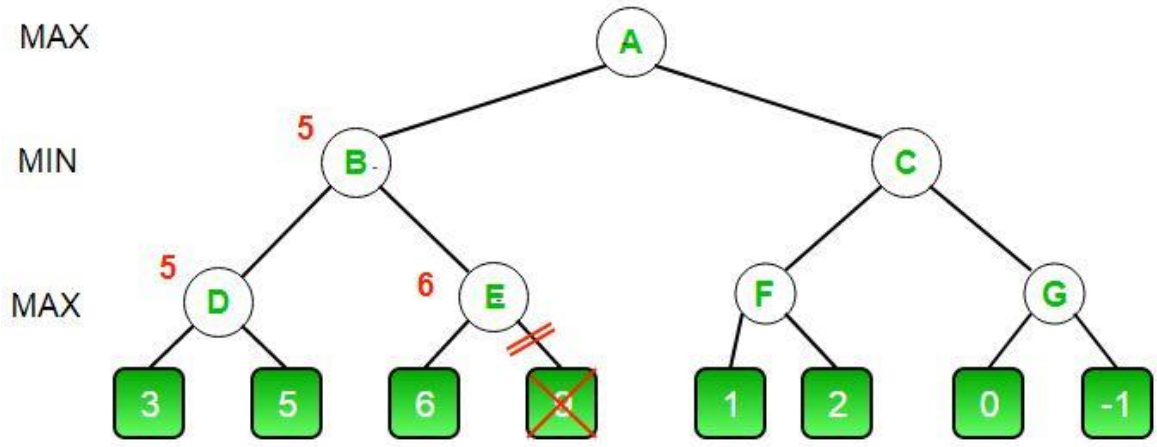

Fig۲ – Alpha Beta Pruning Algorithm

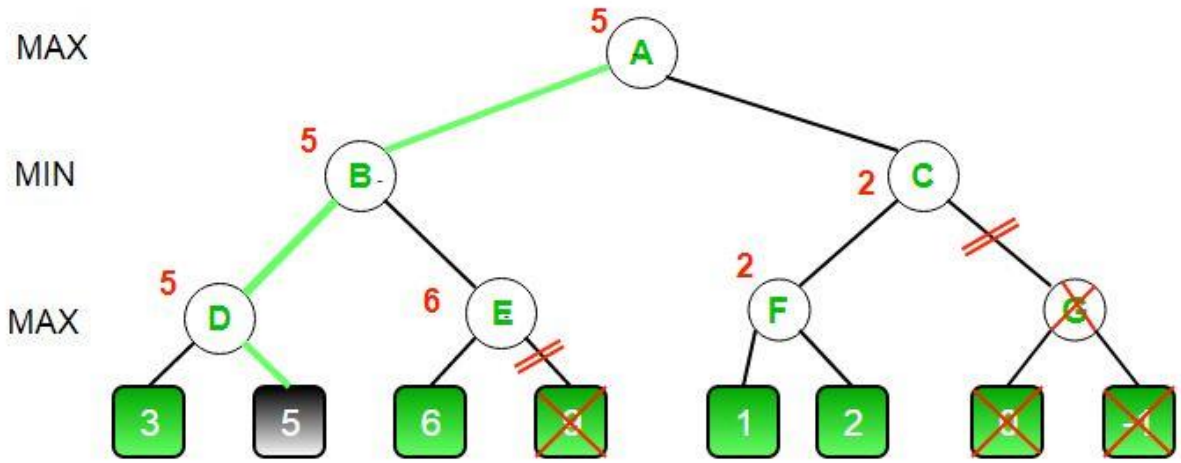

Fig۳ – Alpha Beta Pruning Algorithm

Evaluation Function

این مقدار را برابر با تفاضل تعداد حرکت های ممکن برای بازیکنی که نوبتش است و تعداد حرکت های حریف در نظر گرفتیم. چرا که اگر بازیکن بتواند تعداد حرکت های خود را افزایش داده و حرکات حریف را محدود کند میتواند شانس بیشتری برای به دام نیوفتادن داشته باشد.

پس داریم:

```
Evaluation Function = len(self.generateMoves(board, 'B')) - len(self.generateMoves(board, 'W'))
```

مقایسه زمان اجرای الگوریتم ها

Alpha Beta Pruning Duration Time For ٦ Level	۰,۰۰۱۳۴۳۹۶۵۵۳۰۳۹۵۵۰۷۸ s
Minimax Duration Time For ٦ Level	۰,۰۰۹۱۹۳۸۹۷۲۴۷۳۱۴۴۵۳ s

به وضوح میزان زمانی که طول میکشد الگوریتم minimax اجرا شود بیشتر است چرا که گره هایی که بررسی آن ها کمکی به ما نمیکند در الگوریتم در نظر گرفته میشود.