به نام خدا

الگوریتم MiniMax و a-b purning

در این الگوریتم دو موجود وجود دارند، ماکسیمایزر و مینیمایزر.ماکسیمایزر موجودی است که به دنبال بالاترین امتیاز و درواقع برد بازی است و مینیمایزر حریف اوست و به دنبال کم کردن امتیاز حریف است. از دید هرکدام از آن ها میتوان الگوریتم را نوشت.این الگوریتم، یک الگوریتم هماوردی است که برای بازی های نوبتی طراحی شده و با استفاده از اعمال هرس a b میتوانیم آن را بهینه تر کنیم، اگر بخواهیم این الگوریتم را از دید ماکسیمایزر بنویسیم و هرس a-b نیز انجام دهیم به اینصورت عمل میکنیم.
وقتی بازی در حالت ابتدایی است و صفحه بازی در اختیار ماست، ما به ازای تمام حرکات ممکن امتیازات را حساب میکنیم و بین آن ها ماکسیمم میگیریم، و حالتی را که بیشترین امتیاز را به ما میدهد انتخاب میکنیم و صفحه را در اختیار حریف قرار میدهیم، حریف ما مینیمایزر است و به دنبال کم کردن امتیازات است، بنابراین در حالت فعلی بازی بین تمام حرکاتی که میتواند انجام دهد آن حرکتی که کمترین امتیاز را دارد انتخاب میکند و صفحه بازی را در اختیار ما قرار میدهد.

ممکن است برای بدست اوردن امتیازات حالت های فرزند حالت فعلی، به لایه های پایین تر برویم و در اینصورت یعنی ما باید باید

حرکات حریف خود را نیز پیش بینی کنیم، که در اینصورت استفاده از هرس a-b برای ساده تر کردن محاسبات و درنظر نگرفتن حالات اضافی مفید است. این روش به اینصورت است که به طور مثال وقتی ماکسیمایز بیشترین امتیاز را تا اینجای محاسبات برابر 4 تشخیص داده باشد و در حال چک کردن امتیاز حالت دیگری باشد که مینیمایزر است و به فرض 5 حالت فرزند داشته باشد و تا الان 1 فرزند را چک کرده و امتیاز 3 را برای خود در نظر گرفته باشد، در اینصورت چون 3 از 4 کوچکتر است دیگر چک کردن بقیه حالات را ادامه نمیدهیم زیرا مینیمایزر ممکن نیست که امتیازی بیشتر از 3 به ما بدهد درحالی که امتیازی که ما الان درنظر داریم تا الان 4 است پس ادامه محاسبات این شاخه به نفع ما نیست پس اصطلاحا a-cut اتفاق میفتد. دوگان این اتفاق را b-cut میگوییم.

شیوه پیاده سازی

برای پیاده سازی این پروژه کلاس ها متد هایی به شرح زیر نوشته شده اند

1. State (class)
    a. changeBoard (method)
    b. evaluate (method)
2. GameWindow (class)
    a. startGame (method)

b. userAct (method)
c. botAct (method)
d. resetGame (method)
e. UiComponent (method)
f. FirstState (method)

3. MiniMaxBot (class)

a. Minimax(method)
b. MaxValue (method)
c. minValue (method)

مورد اول برای ساختن حالات مختلف بازی است، که شامل یک ارایه سه در سه به عنوان صفحه بازی و مقدار امتیاز این صفحه(از دید کامپیوتر) و یک مقدار بولین برای اینکه ایا این حالت حالت پایانی هست یا نه؟ و متد هایی برای محاسبه امتیاز، تغییر حالت بازی و حالت پایانی بودن یا نبودن هست

مورد دوم برای مدیریت بازی است که درواقع صورت ظاهری بازی را میسازد و حرکات و نوبت دهی بین کاربر و کامپیوتر را مدیریت میکند و حاوی متد هایی است که برای تحلیل حرکات انجام شده توسط بات و کاربر به کار میرود، و همچنین متدی برای ساختن ظاهر بازی، شروع بازی و تعیین حالت اولیه بازی است.

مورد سوم برای انجام حرکت توسط بات است، که الگوریتم مینیماکس و هرس الفا بتا در اون پیاده سازی شده و یک حالت جدید را پس از انجام حرکت بات باز میگرداند.

فرایند کار به این شکل است که ابتدا ظاهر بازی ساخته میشود و با شروع بازی توسط متد startGame اجازه انجام حرکت به بازیکن داده میشود و تا موقع انجام حرکت توسط بازیکن اجرای ادامه کد متوقف است. پس از انجام حرکت توسط بازیکن حالت و همچنین ظاهر بازی در متد userAct بروزرسانی میشود و نوبت انجام حرکت توسط بات میشود که با استفاده از الگوریتم هرس الفا بتا و مینیماکس صفحه بازی و آن حالتی که درآن هست را تحلیل میکند و حالت جدیدی از بازی را که انتخاب کرده بازمیگرداند (متد minimax) و متد botAct این بروز رسانی را در وضعیت بازی انجام میدهد و دوباره نوبت انجام حرکت توسط بازیکن میشود. و این روند تا جایی ادامه پیدا میکند که بازی به حالت نهایی خود برسد و نتیجه مشخص شود. در این حالت در صورت تمایل بازیکن میتواند با زدن دکمه restart بازی را دوباره شروع کند که باعث فراخوانی متد resetGame میشود که وضعیت بازی را برای انجام دوباره بازی اماده میکند. در این بازی برای حالتی که کامپیوتر در آن پیروز است امتیاز 1 و برای مساوی 0 و برای باخت کامپیوتر امتیاز -1 درنظر گرفته شده است که در زمان تحلیل بازی توسط بات و اجرای

الگوریتم هرس الفا بتا این امتیازات روی حالات بالایی بک اپ شده و بات بر اساس آن ها انتخاب خود را انجام میدهد.