به نام خدا

داک آموزشی مرحله سوم رویداد گلابی

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال اول ۰۱ ـ ۰۰

دبیر رویداد:

محمدطه جهانی‌نژاد

مسئول داک:

حسین گلی

نویسنده داک:

محمدطه جهانی‌نژاد

ویراستاران داک:

علی‌پاشا منتصری

عرفان صدرائیه

مسئول لتک داک:

حسین علی‌حسینی

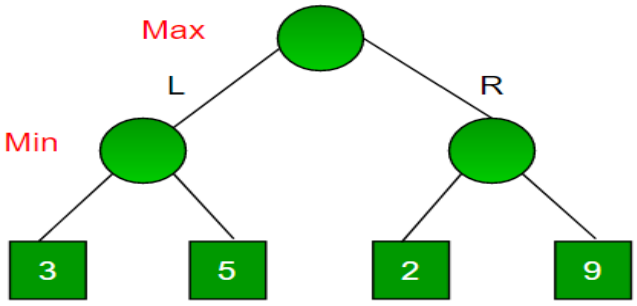
فهرست مطالب

بخش ۱. الگوریتم minimax

الگوریتم minimax یک نوع الگوریتم هوش مصنوعی ساده برای حل بازی‌های رقابتی مانند شطرنج، Tic-Tac-Toe و ... می‌باشد. الگوریتم minimax یک نوع خاصی از backtracking (توضیحات مربوط به این بخش در داک مرحله دوم ذکر شده است) می‌باشد. ایده‌ی کلی به این صورت است که فرض می‌کنیم رقیب بهینه‌ترین حرکت ممکن را انجام می‌دهد (امتیاز ما را مینیمم می‌کند) و در این حالت ما هم تلاش می‌کنیم امتیاز خود را ماکسیمم کنیم (نام این الگوریتم از همین الگو گرفته شده است). برای انجام این تصمیم گیری، از یک درخت استفاده می‌کنیم که تمامی حالات مختلف بازی در آن وجود داشته باشند.

تصمیم‌گیری با استفاده از درخت حالات

فرض کنید یک بازی کوچک داریم که دو مرحله دارد و در هر مرحله بازیکنی که نوبت اوست تنها دو انتخاب دارد. پس در کل برای پایان این بازی، ۴ حالت داریم که به شکل درخت زیر نمایش داده شده اند:

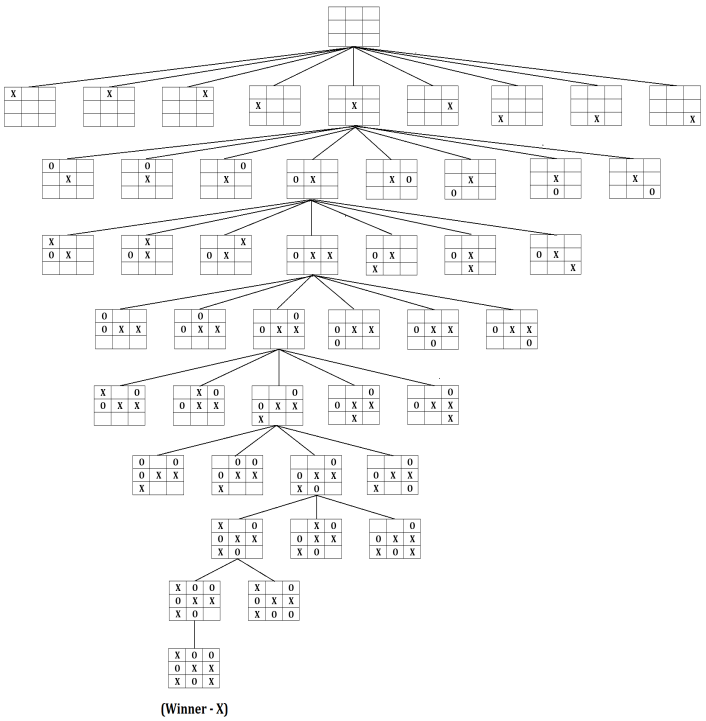


توجه کنید که بازیکن اول ما هستیم و در برگ‌ها هم امتیازی که در نهایت کسب می‌کنیم نوشته شده است. می‌خواهیم از روی این درخت، حرکت اول خود را (L یا R) انتخاب کنیم.
اگر به زیر درخت راست برویم، و فرض کنیم که رقیب هم قطعا بهینه بازی می‌کند و مسیری را انتخاب می‌کند که امتیاز ما مینیمم شود، حداکثر امتیازی که می‌توانیم کسب کنیم برابر

با ۲ خواهد بود.
اما اگر به زیر درخت چپ برویم، و باز هم همان فرض قبل را داشته باشیم، حداکثر امتیازی که می‌توانیم کسب کنیم برابر با ۳ خواهد بود. بنابراین زیر درخت سمت چپ را انتخاب می‌کنیم!
به بیان دیگر، برای تصمیم گیری، سطح به سطح در درخت پایین می‌رویم و در هر مرحله با توجه به اینکه نوبت کدام بازیکن است، امتیاز مینیمم یا ماکسیمم را انتخاب می‌کنیم.

مثال

برای درک بیشتر مسئله، بازی Tic-Tac-Toe را در نظر بگیرید. (اگر با این بازی آشنا نیستید به این لینک مراجعه کنید) می‌خواهیم روند کلی این بازی را در قالب یک درخت نشان دهیم. ریشه‌ی این درخت حالتی می‌باشد که صفحه بازی خالی است. بازیکن اول ۹ انتخاب برای جایگذاری اولین مهره‌اش دارد. بنابراین با در نظر گرفتن اولین حرکت، درخت بازی به صورت زیر در می‌آید:



این مسیر را آن‌قدر ادامه می‌دهیم تا به برگ‌ها برسیم. در برگ‌های این درخت، حالاتی را داریم که یا صفحه بازی پر شده (تساوی) و یا یک برنده داریم. اگر به برگ‌های برنده امتیاز ۱، به برگ‌های بازنده امتیاز ۱ـ و به برگ‌های تساوی امتیاز ۰ نسبت دهیم، می‌توانیم الگوریتم {minimax} را روی این درخت اجرا کنیم. برای اجرای الگوریتم minimax، کافیست با استفاده از DFS (برای اطلاعات بیشتر به داک مرحله دو رجوع شود) روی سطح‌های مختلف این درخت حرکت کنیم. در هر سطح، با توجه به اینکه نوبت کدام بازیکن می‌باشد، امتیاز ۱ـ یا ۱ را انتخاب می‌کنیم. در نهایت با استفاده از این امتیازها می‌توانیم تصمیم بگیریم که مهره خود را در کدام خانه بگذاریم.

کد مخصوص بازی minimax در این لینک وجود دارد. اکنون به تحلیل این کد می‌پردازیم.

در تابع main این کد ابتدا یک صفحه بازی با ۹ خانه تشکیل می‌شود. سپس تا زمانی که برنده‌ای مشخص نشده و همچنین زمین خالی است، درحلقه‌ی اصلی بازی، برحسب این که نوبت کنونی (متغیر turn) چه مقداری می‌باشد، یا کامپیوتر و یا بازیکن بازی می‌کنند.

در اینجا تابع computerMove و تابع minimax مورد بررسی قرار می‌گیرند.

در تابع computerMove، به ازای هر خانه از صفحه که خالی می‌باشد، یک مهره به طور فرضی قرار می‌گیرد و با استفاده از تابع minimax، ماکسیمم امتیاز احتمالی که کامپیوتر می‌تواند با این حرکت تا انتهای بازی کسب کند (با فرض بهینه بازی کردن کاربر) محاسبه می‌شود. در خط بعدی چون این مهره به طور فرضی قرار داده شده بود، برداشته می‌شود تا بازی تغییری نکند. از بین تمامی حالات، حداکثر امتیازی که کامپیوتر می‌تواند کسب کند (یعنی حداقل امتیاز بازیکن حریف) انتخاب می‌شود و کامپیوتر در آن خانه مهره خود را می‌گذارد.

اما در تابع minimax چه اتفاقی رخ می‌دهد؟ در ورودی این تابع زمین بازی و بازیکنی که اکنون نوبت اوست ورودی داده می‌شود.

در این تابع نیز اتفاقی مشابه تابع computerMove رخ می‌دهد؛ اما با این تفاوت که این تابع به جای هر دو بازیکن، بازی فرضی را ادامه می‌دهد و امتیازات را محاسبه می‌کند. در بدنه این تابع، به ازای هر خانه خالی در صفحه بازی، ابتدا مهره بازیکن (player) قرار داده می‌شود سپس حداکثر امتیاز قابل کسب توسط این بازیکن با بازفراخوانی تابع minimax با بازیکن دیگر محاسبه می‌شود. در نهایت خانه‌ای انتخاب می‌شود که امتیاز آن بازیکن را حداکثر کند.

دقت کنید که این تابع یک تابع بازگشتی است و در خطوط ابتدایی آن، شروط پایه تعریف شده‌اند. در ابتدا بیان شده اگر بازی به اتمام رسیده و بازیکن برنده‌ای داریم (یعنی در برگ

درخت حالات قرار داریم) مقدار امتیاز با توجه به بازیکن برنده برگردانده شود.
در انتها، برای درک کامل این الگوریتم، پیشنهاد می‌شود آن‌را کامپایل و مرحله به مرحله اجرا کنید تا متوجه نحوه کارکرد آن شوید.