امین امیری و علیرضا کمرزردی

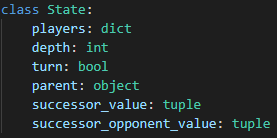
پروژه دوم هوش مصنویی

* فاز اول : مدلسازی

بازی دارای یک صفحه 8 در 8 میباشد و صفحه دارای چند دیسک سیاه یا سفید است .

مدلسازی به این صورت است که هر صفحه بازی را یک در نظر میگیریم و هر شامل دو مجموعه از های سفید و سیاه میباشد .

* معرفی پراپرتی های کلاس



نوع متغیر دیکشنری میباشد و شامل دو لیست است با ایندکس های و که به معنای

لیست شامل های بازیکن با مهره های سیاه و لیست شامل های بازیکن با مهره های سفید

میباشد .

نوع متغیر عدد صحیح که عمق را مشخص میکند .



نوع متغیر که نوبت بازیکنی که قرار است حرکت انجام دهد را مشخص میکند .



نوع متغیر از و پدر این را مشخص میکند .



نوع متغیر که دارای 2 عنصر میباشد عنصر اول لیستی از های شده و لیست دوم تعداد مهره های تغییر یافته ها به ترتیب لیست اول .



نوع متغیر که دارای 2 عنصر میباشد عنصر اول لیستی از های شده اگر نوبت حریف باشد و لیست دوم تعداد مهره های تغییر یافته ها به ترتیب لیست اول اگر نوبت حریف باشد .

* معرفی متد های

این تابع کنونی را اگر گسترش نداده بود گسترش می دهد .

روند گسترش

روی مهره های حریف پیمایش میکنیم ، فرض کنیم متغیر پیمایش مهره های حریف باشد، همسایه های را درنظر میگیریم که خالی باشد .

روی آن های پیمایش میکنیم فرض کنیم همسایه هایی باشد که روی آن پیمایش میکنیم .

عکس جهتی که به این همسایه بودیم را درنظر میگیریم به آن جهت پیمایش میکنیم تا زمانی که یا به خانه خالی برسیم یا به مهره خودمان برسیم یا از صفحه بیرون انداخته شویم اگر به مهره خودمان رسیدیم میتوانیم در این همسایه مهره بزاریم یعنی یک مکان انتخابی برای گزاشتن مهره میباشد .

حال مهره هایی که با گزاشتن تغییر میکند پیدا میکنیم ( با پیمایش 8 جهت بردار های یکه اگر به مهره های خودمان رسیدیم با پیمایش هر بردار تمام مهره ها تا مهره خودمان تغییر رنگ میدهد ) .

در نهایت با پیدا کردن تمام مهره هایی که باید تغییر کنند ، مهره ها را از لیست مهره های حریف به لیست مهره های خودمان اضافه میکنیم .

یک کران بالا برای تعداد عملیات های این تابع

این تابع یک و بردار میگیرد و در جهت بردار پیمایش میکند تا به بیرون از صفحه یا خانه خالی یا مهره خودمان برسد اگر خالی یا بیرون صفحه بود و در غیر این صورت خودمان را بر میگرداند

یک و لیستی از ها میگیرد را به مهره های خودمان اضافه میکند و لیست ها ورودی را از مهره های حریف حذف و به مهره های خودمان اضافه میکند .

**هیوریستیک**

تابع ارزیابی هیوریستیک اتلو علاوه شمارش تعداد مهره ها همچون یک بازیکن خوب اتلو پارامتر های موبیلیتی، استیبیلیتی و مهره های گوشه ای را در نظر میگیرد. که به هر مورد میپردازیم. خروجی تابع هیوریستیک به شکل زیر است.

return 2 \* disk\_parity + corner\_captured + mobility + stability

**موبیلیتی** :

بازیکنان خبره اتلو گاهی طوری بازی میکنند که شما در نوبت خود هیچ انتخابی برای گذاشتن مهره در صفحه نداشته باشید یا حداقل انتخاب های کمی داشته باشید. این پارامتر را موبیلتی مینامیم. خروجی تابع موبیلیتی به شکل زیر است.

100 \* (max\_player\_moves - min\_player\_moves) / (max\_player\_moves + min\_player\_moves)

**استیبیلتی** :

تعداد مهره هایی که در حرکت بعدی در معرض تغییر هستند را مهره های Not Stable مینامیم. در اتلو هر چه تعداد مهره های Not Stable ما نسبت به حریف کمتر باشد شانس موفقیت ما بیشتر خواهد بود. این پارامتر را stability مینامیم. خروجی تابع استیبیلیتی به شکل زیر است.

**کورنر** **کپچر** :

مهره هایی که در چهار گوشه صفحه قرار میگیرند دیگر تغییر پیدا نمیکنند. بنابراین اگر شما مهره ای را در گوشه قرار دهید علاوه بر اینکه حرف قادر به تغییر آن نیست، میتوانید

100 \* (max\_player\_corners - min\_player\_corners) / 4

**پریتی** :

این پارامتر نشان میدهد چند درصد از کل مهره های صفحه به مهره های ما اختصاص دارد.

100 \* max\_player\_disks / (max\_player\_disks + min\_player\_disks)