

به نام خدا



پروژه درس هوش مصنوعی – فاز اول

۱۴۰۱ پاییز

سارا شاه طوسی - امیرحسین عرب پور - علی ناظری

جناب دکتر حسین کارشناس

- بررسی فضای بازی
- بررسی بخش اول الگوریتم
- بررسی بخش دوم الگوریتم
- بررسی بخش سوم الگوریتم
- چگونگی کارکرد الگوریتم
- چگونگی جهت یابی
- بررسی مزایا و معایب این روش
- جمع بندی

قبل از هر چیز در ابتدا به سراغ توصیف محیط بازی رفته و پس از آن به توضیح الگوریتم به کار رفته میوردازیم

نوع عامل ما کاراکتر بازی است

Performance : محاسبه بهترین مسیر و خوردن الماس ها برای گرفتن بیشترین امتیاز با در نظر گرفتن زمان پردازشی یک ثانیه

Environment : محیط بازی که شامل دیوار الماس، سیم خاردار، در و کلید است و محدود می باشد

Actuators : حرکت به ۹ جهت در اطراف عامل و جمع آوری کلید و الماس

Sensors : ماتریسی از تمام محیط بازی که با هر بار حرکت بروز می شود

خواص محیط بازی

کاملا مشاهده پذیر است چون ما تمام نقشه بازی را در هر حرکت در اختیار داریم و مکان همه المان ها مشخص است.

محیط تک عاملی است چون فقط عامل ما بازی می کند.

محیط قطعی است چون حالت بعدی کاملاً توسط حالت فعلی و کنش انجام شده قابل تایین است.

مرحله ای است چون فعالیت انجام شده در هر مرحله به مرحله قبل وابسته نیست.

محیط ایستا است و در مروز زمان و در هر دست از بازی تغییر نمی کند.

گسسته است چون زمان هر حالت های محدودی در هر مرحله وجود دارد.

شناخته شده است به دلیل اینکه کاملاً عامل با محیط آشنا است و می‌داند در آن چگونه عمل کند.

حال به سراغ بررسی الگوریتم میرویم :

الگوریتم به کار رفته در طراحی آن را می‌توان در سه بخش خلاصه کرد:

بخش اول:

زمین بازی را که به صورت مشاهده پذیرکامل است خانه به خانه بررسی کرده و المان هایی که در ادامه با آن‌ها کار خواهیم داشت به شکل لیست هایی از مختصات ذخیره می‌کنیم.

المان‌هایی که نیاز به ذخیره کردن دارند:

الماس‌های زرد-سبز-آبی-قرمز

کلید‌های آبی-قرمز-سبز

تابع analysis این کار را برای ما انجام می‌دهد و در خروجی به ما یک دیکشنری است که در هر عضو آن یک لیست از مختصات قرار دارد.

بخش دوم:

در مرحله دوم از الگوریتم به سراغ ساخت حدس هایی مختلف برای برداستن المان هایی که اطراف ما هستند می رویم.

در این بخش با در نظر گرفتن موقعیت فعلی آخرین الماس برداشته شده به ترتیب امتیاز رنگ بعدی نزدیک ترین الماس از آن رنگ را پیدا می کنیم و موقعیت آن را به عنوان اولین الماس از حدس خود در نظر می گیریم.

به طور مثال اگر آخرین رنگ الماس که برداسته ایم زرد بوده باشد از آنجایی که بعد از خوردن الماس سبز با ۲۰۰ بیشترین امتیاز، قرمز با ۱۰۰ و زرد و آبی هر کدام ۵۰ امتیاز هستند.

پس بعد از زرد ابتدا به بررسی مسیر هایی با الماس اول سبز، قرمز، زرد و آبی می پردازیم. اما بعد از در نظر گرفتن اولین حدس بررسی می کنیم که بعد از برداشتن آن حدس چه کار هایی می توانیم انجام دهیم پس با فراض تغییر موقعیت فعلی به مختصات الماس اول و (درستش کن) تغییر حدس اولیه، حدس دوم را پیدا می کنیم و با تکرار این فرآیند حدس سوم را نیز پیدا می کنیم.

پس حدس کامل ما شامل سه مختصات الماس اول، دوم و الماس سوم است. الماس اول در واقع جایی است که باید به آن برویم و دو الماس دیگر برای آینده نگری بعد از آن پیدا شده است. ما وزنی را به این حدس اختصاص می دهیم. به صورتی که حدس ما برابر حاصل جمع تقسیم امتیاز بدست امده از خوردن الماس با توجه به الماس قبلی برا میزان دور بودن آن الماس از موقعیت قبل به صورت همیلتونی است. (در این میزان دور بودن ما صفحه خالی از هر گونه دیوار و سیم خاردار (نمیتونم بخونم)

$$W = \sum_{i=1}^3 (\text{فاصله همیلتونی}/\text{امتیاز خاص})$$

بعد از تهیه لیستی از حدس ها که در ان حد اکثر 64 حدس می تواند قرار دارد. آن ها را با توجه به وزن شان مرتب می کنیم.

عملیات شرح داده شده را تابع making_list_of_guess_ways انجام می‌دهد و در پایان لیست مرتب شده در اختیار ما قرار می‌دهد. ما برای بهتر و مرتب شدن کلاس تحت عنوان ways تعریف کرده ایم که اطلاعات هر یک از دسته‌های سه تایی الماس را ذخیره می‌کند.

بخش سوم:

در بخش سوم به سراغ لیست حدس هایی که در بخش دوم بدست آمده بود رفته و ۲۰ تا ۵۰ حدس ان را که بر اساس وزنی که به آنها داده ایم و بیشترین وزن را داشته اند رفته و وزن دومی دقیق تر بر اساس **COST** رسیدن به آن خانه اختصاص میدهیم.
وزن دوم اختصاص یافته به آن حدس با فرمول زیر محاسبه می شود:

هزینه واقعی رسیدن به آن خانه را با مسیریابی که با تابع **Find_way** انجام داده ایم، می یابیم. در مورد نحوه ی مسیریابی و حل مشکل در ها و کلید ها در ادامه صحبت میکنیم.

نحوه‌ی مسیریابی (حل مشکل درها و کلیدها):

برای مسیریابی به کمک تابع `Find_way`، دو مختصات را گرفته و از نقطه اول به نقطه دوم `BFS` میزنیم. سپس تک تک خانه‌های این مسیر را بررسی میکنیم. اگر در این مسیر هیچ دری وجود نداشته باشد، که این مسیر بهترین مسیر است و لیست خانه‌های این مسیر به همراه `COST` آن را به عنوان جواب بر میگرداند.

اگر در این مسیر در وجود داشت، دو مسیر جدید را پیدا خواهیم کرد:
مسیر اول: با تبدیل درها به دیوار و بستن آنها و با `BFS` جدید بهترین مسیری که در آن هیچ دری وجود نداشته باشد را پیدا میکنیم و فرض میکنیم هزینه‌ی آن `c1` باشد.

مسیر دوم: از خانه‌ای که هستیم به کمک `BFS` به نزدیکترین کلید هم رنگ در رفته و به کمک `BFS` به در بازگشته و ادامه مسیر را طی میکنیم. سپس مجموعه مسیر هارا به عنوان مسیر دوم با هزینه‌ی `c2` در نظر میگیریم.

(برای پیدا کردن مسیرها به صورت بازگشتی عمل میکنیم، یعنی برای پیدا کردن مسیری در دل مسیر دوم دوباره `Find_way` را صدا میزنیم که اگر دوباره در طول آن در وجود داشت.)

بین `c1` و `c2`، هر کدام که کوچکتر بود به عنوان بهترین مسیر از تابع بازگردانی خواهد شد.

چگونگی کارکرد کلی الگوریتم:

بعد از خوردن هر الماس برای پیدا کردن الماس جدید ، کل الگوریتم اجرا می شود و الماس بعدی و مسیر آن مشخص می شود و در **round** های بعد تا رسیدن به الماس بعد از لیست مسیر حرکت بعدی تعیین می شود.

در اجرای الگوریتم ابتدا مرحله اول اجرا و زمین بررسی می شود. سپس مرحله دوم اجرا شده و لیستی از حدس ها به دست می آید .

با انتخاب تعدادی از حدس ها ، هزینه آنها در مرحله سوم بدست می آید و بهترین آن به عنوان مسیر برتر انتخاب شده و به سمت اولین الماس آن حرکت میکنیم .

باید به این نکته اشاره کنیم که برای انتخاب مسیر برتر این موضوع که تعداد حرکت لازم برای رسیدن به آن و نیز تعداد **round** های باقی لحاظ می شوند.

و همچنین اگر بهترین حدس دارای وزن منفی باشد **agent** بدون حرکت در جای خود باقی خواهد ماند.

مزایا و معایب و جمع بندی

:

مزایا: به دلیل متغیر بودن امتیاز هر رنگ الماس با توجه به رنگ الماس قبل، آینده نگرانه بودن حدس ها در این روش عامل بسیار مهم و مؤثر در بهینه بودن انتخاب های agent است.

مزیت دوم آن این است که تعداد راه بررسی شده در آن کم بوده (۶۴ حدس و ۴۰ حدس منتخب برای محاسبه هزینه واقعی) که زمان اجرای الگوریتم را نسبتاً مناسب و پایین ایجاد میکته و میتوان گفت در بسیاری از موارد و نقشه ها بهینه و سریع عمل میکند.

معایب:

این الگوریتم در نقشه هایی که بهترین الماس برای انتخاب الماس به جز نزدیکترین الماس در آن رنگ است، عملکرد بدی خواهد داشت به دلیل عدم بررسی الماس های بعدی و فقط بررسی نزدیک ترین در هر رنگ و نیز به دلیل عدم تغییر روش با توجه به زمان های مختلف در مواردی که زمان بین هر بسیار کم است (و یا کمتر از زمان الگوریتم است) عملکرد بد خواهد داشت.