به نام خدا

نرگس غلامی

810198447

پروژه اول هوش مصنوعی

با موضوع سرچ

هدف انجام پروژه: در این پروژه با انواع سرچ آگاهانه و ناآگاهانه آشنا می‌شویم و معایب و مزایای این روش ها را از لحاظ سرعت، حافظه و... با یکدیگر مقایسه می‌نماییم.

توضیح کلی پروژه و تعریف صورت مسئله: در این پروژه قرار است که دکتر با روش های مناسب تمام معجون ها را جمع نماید. ممکن است در طول راه به بعضی داروهای دوبرابر کننده برسد که باعث این می‌شود که یک دکتر دیگر ایجاد شود و ما باید تمام دکترهای حاضر در معبد را به مقصد برسانیم. برای این کار از انواع سرچ کمک می‌گیریم. در قسمت اول مسئله باید با سرچ BFS پیاده سازی شود، در قسمت دوم با IDS و در قسمت بعد با روش سرچ ناآگاهانه *A معمول و *A وزن دار باید مسئله را حل کرد.

شرح نحوه مدل کردن مسئله :

ما یک نقشه داریم که n سطر و m ستون دارد. موقعیت ها در این صفحه به این صورت توصیف می‌شود (x, y)

Initial state : موقعیت اولیه دکتر خانه (0,0) در جدول می‌باشد.

Goal state : ما به موقعیتی هدف می‌گوییم که همه دکتران به خانه (m-1, n-1) رسیده باشند و همه معجون ها نیز برداشته شده باشند.

Actions : عملیات هایی که دکتر ما می‌تواند انجام دهد رفتن به راست، چپ، بالا، پایین است به شرطی که از نقشه بیرون نزند و روی سنگ ها نیز نرود.

Transition model : هنگامی که در یک موقعیت دکتر به یک سمت حرکت کند اگر آن خانه مجاز باشد در آن خانه قرار خواهد گرفت.

Path cost : هر قدمی که دکتر برای رسیدن به استیت هدف انجام می‌دهد یکی به هزینه ها اضافه می‌کند. پس مسیری که تعداد قدم های کمتری داشته باشد بهینه تر می‌باشد.

نحوه مدل کردن مسئله در کد :

نقشه به صورت یک آرایه n*m پیاده سازی شده که خانه های خالی با صفر، معجون با یک، داروی دوبرابر کننده با دو و سنگ با سه نشان داده شده است. هر استیت در این مسئله با یک استرینگ پیاده سازی شده است که ساختار کلی آن به شکل زیر می باشد:

position doctor[1] position doctor[2] ...| position potion 1 position potion 2 ... | position drug 1 position drug 2

یعنی ابتدا مکان دکترهایی که در صفحه وجود دارند نمایش داده خواهد شد سپس با یک | مکان معجون ها نوشته خواهد شد سپس با یک | فاصله مکان داروهای دوبرابر کننده موجود در صفحه نوشته خواهد شد.

Initial state : با این حساب استیت شروع میشود ←

"0 0 | | "

Goal state : با یک تابع بررسی میشود که استیتی که در آن هستیم استیت هدف هست یا نه. به این صورت که با توجه به تعداد داروهای خورده شده تعداد دکترها را بدست می اوریم سپس بررسی میکنیم که آیا همه ی دکترها به هدف رسیده اند یا نه. سپس بررسی میکنیم همه ی داروهای موجود در صفحه خورده شده یا نه. اگر یک استیت همه ی این ها را داشته باشد استیت هدف محسوب میشود.

Actions : با یک حلقه داخل هر تابع دکتر را در چهار جهت پیش میبریم.

Transition model : با استفاده از تابع makeState ، transition جدید دکتر ساخته میشود.

توضیح تفاوت‌ها و مزیت‌های الگوریتم‌ها:

اولین الگوریتم پیاده‌سازی شده BFS بود. BFS دارای مزیت‌هایی می‌باشد من جمله این که هیچوقت در یک مسیر بی‌فایده گیر نمی‌کند، اگر جوابی وجود داشته باشد حتما آن را پیدا خواهد کرد و اگر چند راه حل وجود داشته باشد این الگوریتم بهینه‌ترین آن‌ها را خواهد بافت. ا

از بدی‌های این الگوریتم می‌توان به این موضوع اشاره کرد که تک تک استیت‌ها را ذخیره می‌کند که این ذخیره کردن حافظه بسیار زیادی می‌برد و همچنین اگر جواب از استیت شروع خیلی دور باشد زمان زیادی طول می‌کشد تا جواب پیدا شود.

دومین الگوریتم پیاده‌سازی شده IDS می‌باشد که مانند BFS یک الگوریتم ناآگاهانه می‌باشد. IDS نسبت به BFS کندتر می‌باشد. از مزیت‌های این الگوریتم می‌توان به کامل بودن و اپتیمال بودن این الگوریتم اشاره کرد که یک برتری نسبت به الگوریتم DFS معمولی به حساب می‌آید. برتری این الگوریتم نسبت به الگوریتم BFS آن است که حافظه کمتری اشغال می‌کند.

سومین الگوریتم، آ استار می‌باشد. این الگوریتم یک الگوریتم آگاهانه است و نسبت به دو الگوریتم قبلی برتری دارد. این الگوریتم با تابع heuristicای که پیدا می‌کند می‌تواند در زمان مناسبی جواب مسئله را بیابد. این الگوریتم نیز اپتیمال و کامل می‌باشد. همچنین آ استار نسبت به بقیه‌ی الگوریتم‌ها تعداد Nodeهای کمتری را گسترش می‌دهد که یکی دیگر از مزایای این الگوریتم محسوب می‌شود.

خوب بودن این الگوریتم خیلی به خوب بودن تابع heuristic آن بستگی دارد که گاهی پیدا کردن این تابع راحت نمی‌باشد که در نتیجه حل مسائل سخت و مدل‌سازی ان با این روش سخت می‌وشود. ولی یافت جواب در این مسئله نیز زمان اکسپوننشال دارد.

در آخر نیز الگوریتم Weighted A* را داریم که یک تابع هیرویستیک دارد که کمی دقتش کمتر است در نتیجه در زمان کمتر، جوابی را می‌دهد که تقریبا به جواب بهینه نزدیک است. در حقیقت در Weighted A* ما یک مصالحه بین زمان و بهینه بودن الگوریتم داریم.

با توجه به توضیحاتی که داده شد تمام الگوریتم‌ها جواب بهینه را تولید می‌کنند به جز الگوریتم Weighted A* که به جای تولید جواب بهینه سریع‌تر می‌باشد.

توضیح کوتاهی راجع به چگونگی الگوریتم‌ها:

BFS: بدین صورت عمل می‌کند که ابتدا یک صف که استیت ابتدایی ما در آن است داریم سپس وارد یک حلقه می‌شویم و تا وقتی صف خالی نشده حلقه را ادامه می‌دهیم. در هر دور حلقه اولین عضو صف را برداشته و به همسایه‌های آن می‌رویم و چک می‌کنیم آیا این استیت، استیت هدف است یا نه. اگر نبود به استیت‌های دیده شده اضافه می‌کنیم(اگر قبل از آن وجود نداشت) و همچنین اگر استیت، استیت هدف بود مسیر را برمی‌گردانیم.

IDS : در حقیقت این الگوریتم همان دی اف اس است منتها با این فرق که برای عمق‌ها حد تعیین می‌کنیم. به این صورت که از عمق صفر شروع می‌کنیم و تا عمق بی‌نهایت پیش می‌رویم و اگر به استیت هدف رسیدیم آن را برمی‌گردانیم. نحوه نوشتن آن به صورت بازگشتی است ولی کلیت چک کردن استیت‌ها مانند بی اف اس می‌باشد.

الگوریتم A* : در این الگوریتم باز هم یک صف و حلقه داریم که هر دفعه از بین آن صف، عنصری را اانتخاب می‌کنیم که کمترین مقدار هزینه از مبدا و تا مقصد را داشته باشد و همسایه‌های آن عنصر را در صورت مجاز بودن به صف اضافه می‌کنیم. قبل از اضافه کردن به صف هزینه همسایه‌ها را نیز محاسبه می‌کنیم و در محلی ذخیره می‌نماییم. آن قدر در این حلقه پیش می‌رویم تا به استیت هدف برسیم.

الگوریتم A* وزن دار: مانند همان A* می‌باشد منتها تابع محاسبه هزینه دقت کمتری دارد.

توضیح heuristic پیاده‌سازی شده در بخش جستوجوی آگاهانه:

heuristic را به این صورت پیاده سازی میکنیم که فاصله‌ی مکانی که دکتر ایستاده است را تا مقصد محاسبه می‌نماییم. یعنی ایکس نقطه مقصد را منهای ایکس نقطه کنونی به علاوه ایگرگ نقطه مقصد منهای ایگرگ نقطه کنونی.

در پایین تابع h همان تابع heuristic می‌باشد و تابع g هزینه‌ای است که نقطه‌ی کنونی تا اینجای مسیر طی کرده است می‌باشد.

```python
def calF(curr):
    h = 0
    g = 0
    numDoctor = countDoctor(curr)
    for i in range(numDoctor):
        x0, y0 = makePosition(curr, i)
        h += n - x0 + m - y0
        x1, y1 = makePosition(curr, i)
        g += x1 + y1
    return h + g
```

حال بررسی می‌نماییم که این تابع consistent هست یا خیر. می‌دانیم که هزینه یال بین دو نقطه یک می‌باشد.

دو نقطه که فاصله‌شان از یکدیگر برابر یک باشد در نظر بگیرید. تفاوت h این دو دقیقا برابر یک است چرا که فاصله‌شان یکی می‌باشد. حال هر دو نقطه‌ای را که در نظر بگیریم هزینه بینشان برابر فاصله بینشان می‌شود و فاصله بینشان اختلاف توابع heuristic این دو می‌باشد پس تابع heuristic ما consistent می‌باشد.

حال هر کدام از الگوریتم‌ها را اجرا می‌کنیم و میانگین زمانشان را ثبت می‌کنیم:

تست اول:

	فاصله جواب	تعداد استیت‌های دیده شده	میانگین زمان اجرا
BFS	13	4849	0.052 s
UCS	11		224 s (1 bar ejra)
A*	12	172	0.002
Weighted A* (به ازای هر α)	12	172	0.00366

تست دوم:

	فاصله جواب	تعداد استیت‌های دیده شده	میانگین زمان اجرا
BFS	21 :/	439562	4.09
UCS			
A*	11	213408	86s
Weighted A* (به ازای هر α)	11	213408	120 s

تست سوم:

	فاصله جواب	تعداد استیت‌های دیده شده	میانگین زمان اجرا
BFS	36	3885790	28.11280083656311
UCS			
A*	24	168152	104.56098341941833
Weighted A* (به ازای هر α)	24	168152	86.01984310150146

نتیجه گیری کلی: با توجه به این که چهار روش سرچ را با یکدیگر مقایسه کردیم در آخر به این نکته پی میبریم که باید دقت بکنیم بین مصالحه بین زمان، درستی جواب بهینه، حافظه و... کدام یک برنده می‌شوند. سپس با استفاده از نتیجه آن الگوریتم سرچ خود را انتخاب می‌نماییم.

ارائه راهکارهایی برای توسعه و بهبود پروژه:

راستش بنظرم برای پروژه اول سخت بود. برای بهبودش هم باید بگم اگه زمانای خواسته شده بیشتر می‌بود خوب میشد☺

در هر صورت خسته نباشید و ممنون از زحماتتون☺

منابع استفاده شده:

https://www.pythonpool.com/a-star-algorithm-python/#:~:text=A*%20Algorithm%20in%20Python%20or,a%20wide%20range%20of%20contexts.

https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/

https://medium.com/@vatsalunadkat/advantages-and-disadvantages-of-ai-algorithms-d8fb137f4df2

https://www.quora.com/What-are-the-advantages-of-breadth-first-search