به نام خدا

گزارش پروژه مبانی هوش مصنوعی

سید پوریا احمدی ۹۷۲۳۰۰۲

امیر اله ورن ۹۷۲۳۰۱۰

-۱

مدل سازی مسئله:

هر کدام از خانه های میز یک node گراف در نظر گرفته شده است و همچنین هرکدام از حرکت های بالا، پایین، چپ و راست در صورت مانع نبودن، فرزند خانه (node) فعلی محسوب می شوند و روی این node ها جستجو را طبق الگوریتم های خواسته شده انجام می دهیم.

همچنین ربات agent است و خانه هدف شخص است

-۲

تابع شهودی:

فاصله اقلیدسی به عنوان تابع تابع شهودی انتخاب شده است.

ربات نمیتواند حرکت اریب انجام دهد ، اما فاصله اقلیدسی ، فاصله اریب را محاسبه میکند که طبق قانون مثلث ، مجموع دو ضلع از ضلع سوم بزرگتر است، پس تابع شهودی انتخاب شده قابل قبول خواهد بود.

-۳

IDS :

برای این الگوریتم:

ابتدا یک تابع read_file داریم که فایل را میخواند و مقادیر تعداد ستون ها ، تعداد سطر ها و اینکه در خانه های میز ما چه موجودیتی(کره ،ربات، شخص ،مانع یا خانه خالی) را پیدا می کند و بر میگرداند.

Butter_map :

در این تابع موارد لازم مانند اینکه آیا هدف ما در رنج مورد قبول ما است چک می شود و تابع IDS که جستجوی ما را انجام می دهد را با پاس دادن آرگومان های لازم مانند تعداد سطر و ستون و خانه های جدول و cutoff تابع فرا خوانی میکند .

در نهایت اگر به جواب برسد true و اگر به جواب نرسد false را برمیگرداند.

تابع action :

در این تابع که در هر مرحله از جستجوی ما فراخوانی می شود چک می شود که حرکتی که قرار است انجام شود آیا مجاز است یا خیر برای مثال ممکن است حرکت ما بخواند به خانه ای برود که مانع در آن وجود دارد یا اینکه از محیط ما خارج شود .

تابع IDS :

در این تابه الگوریتم IDS ما اجرا می شود .

در این تابع که به صورت بازگشتی تعریف شده است ابتدا برای هر جهت جستجو انجام شده و اگر به cutoff مورد نظر نرسیده باشیم به مرحله بعد می رویم و اگر به limit رسیده باشیم با توجه به اینکه مکان مورد نظر را پیدا کرده ایم یا نه true یا false بر میگرداند.

تابع robot_move :

در این تابع جهت حرکت ربات تعیین شده و بعد از آن جدول را update می کنیم .

تابع robot_cost :

با توجه به حرکت ربات که ادامه دهنده مستقیم راه خود است یا باید تغییر مسیر بدهد هزینه حرکت هر حانه که یک فرض شده است را تغییر می دهد(هزینه حرکت ربات تا مقصد بعدی را حساب میکند)

تابع main :

با استفاده از توابع تعریف شده راه ربات مشخص کرده و در انتها راه طی شده توسط ربات ، هزینه مصرف شده و عمق سرچ را چاپ می کند.

: BFS

ابتدا یک تابع read_file داریم که فایل را میخواند و مقادیر تعداد ستون ها ، تعداد سطر ها و اینکه در خانه های میز ما چه موجودیتی(کره ،ربات، شخص ،مانع یا خانه خالی) را پیدا می کند و بر میگرداند.

: Butter_map

در این تابع موارد لازم مانند اینکه آیا هدف ما در رنج مورد قبول ما است چک می شود و تابع bidirectional bfs که جستجوی ما را انجام می دهد را با پاس دادن آرگومان های لازم مانند تعداد سطر و ستون و خانه های جدول cutoff تابع فرا خوانی میکند .

: bidirectional bfs

در این تابع که مختصات مبدا و مقصد را می گیرد، و تابع bfs فراخوانی می شود .

در این تابع صف هایی برای حرکت از مبدا به مقصد و مقصد به مبدا تشکیل شده و در جهت های مجاز سرچ ما انجام شده و با رسید گره های سرچ به هم جواب برگردانده می شود.

سپس جواب برگردانده شده به صورت درست(از ابتدا به انتها راه طی شده را پشت هم چاپ میکند) بر می گرداند.

تابع action :

در این تابع که در هر مرحله از جستجوی ما فراخوانی می شود چک می شود که حرکتی که قرار است انجام شود آیا مجاز است یا خیر برای مثال ممکن است حرکت ما بخواند به خانه ای برود که مانع در آن وجود دارد یا اینکه از محیط ما خارج شود

تابعroad :

هر بار مسیر ربات تا مکان درستی که برای قرار گیری پشت کره است را بر میگرداند.

: BFS

یک سرچ برای رسید ربات به پشت کره انجام میدهد.

: Robot_move

در این تابع مسیر کلی ربات تا رساندن کره به مقصد تشکیل شده و برگردانده می شود (در این تابع دول ما به روز رسانی می شود).

تابع main :

با استفاده از توابع تعریف شده راه ربات مشخص کرده و در انتها راه طی شده توسط ربات ، هزینه مصرف شده و عمق سرچ را چاپ می کند

: A*

کلاس Node:

یک model class است که برای هر کدام از خانه های جدول والد، موقعیت مکانی، و مقدار توابع f,g,h را نگهداری میکند.

تابع read_input:

نقشه داده شده برای میز را میخواند و ابعاد میز، نقشه هزینه ها و نقشه شامل ربات و کره و مانع ها را از آن استخراج میکند.

تابع convert_path_to_action:

کوتاه ترین مسیر که براساس موقعیت مکانی خانه ها هست را میگیرد و توالی حرکت های انجام شده برای تولید این مسیر توسط ربات را بر میگرداند.

تابع search:

تابع A* را اجرا میکند و کوتاه مسیر پیدا شده برای اجرای وظیفه ربات و هزینه این مسیر و عمق گراف را پیدا میکند.

تابع robot_goal:

در هر بار اجرا با توجه به move ورودی، هدف فعلی ربات را تعیین میکند.

تابع get_position:

این تابع با گرفتن هر کدام از موجودیت ها مانند کره یا ربات، موقعیت مکانی آنها را پیدا میکند.

تابع get_career_num:

این تابع تعداد موجودیت هایی مانند کره یا مانع های موجود داخل نقشه را پیدا میکند.

تابع main:

ای تابع با استفاده از سایر توابع، به طور حرکت های ربات انجام میدهد و در انتها حرکت های ربات و هزینه و عمق را چاپ میکند.

۴-

الف)

به صورت جدول زیر است:

	IDS	Bidirectional BFS	A*
Test 1	۰,۰۰۱	0.001	۰,۰۰۰۹
Test 2	۰,۰۰۲	0.005	۰,۰۰۱
Test 3	۰,۰۰۱	0.002	۰,۰۰۱
Test 4	Can not pass	Can not pass	Can not pass
Test 5	۰,۰۰۱	0.001	۰,۰۰۴

ب)

الگوریتم IDS -> $0(b^d)$

الگوریتم bidirectional bfs -> $O\left(b^{d/2}\right)$

الگوریتم A* -> به طور کلی نمایی است

که d در آنها depth of shallowest node است

ج) الگوریتم IDS پیچیدگی فضایی $O(d)$ دارد.

الگوریتم bidirectional bfs پیچیدگی فضایی $O\left(b^{\frac{d}{2}}\right)$ دارد.

الگوریتم A* پیچیدگی فضایی $O(b^d)$ است.

د) node های گسترش داده شده توسط IDS بیشتر از bidirectional bfs و هر دو آنها بیشتر از A* هستند.

ه) عمق طی شده توسط IDS اندازه cutoff است که آن هم برابر عمق هدف خواهد بود. عمق طی شده توسط bidirectional bfs برابر نصف عمق هدف است زیرا هر کدام از الگوریتم های bfs که از هدف و ریشه شروع میشوند تا تقریبا نصف عمق را طی میکنند. عمق طی شده توسط الگوریتم A* بزرگتر مساوی عمق هدف خواهد بود زیرا امکان این وجود دارد که به علت هزینه یک مسیر اشتباه را طی کند و عمقی که طی میکند از عمق هدف بیشتر شود.