# گزارش پروژه

**علی عبداللهی – 9731116**

**کیوان ایپچی حق - 9831073**

1.نحوه مدلسازی مساله : در این جا ابتدا یک ماتریس تشکیل میدهیم که شامل کره و موانع و مقاصد است و ابتدا با الگوریتم های سرچ جستجو میکنیم که کدام کره اولویت دارد و به کدام مقصد و مساله ی سرچ به این نحو است که برای هر خانه ی ماتریس تعدادی همسایه وجود دارد که به عنوان فرزندان آن در نظر میگیریم هر یک از مراحل جا به جایی کره که میخواهیم به صورت فرضی آن را حرکت دهیم و پیمایش کنیم و آن را از نود خود به نود فرزند که یک همسایه اش ببریم باید ربات به پشت کره منتقل شو تا بتواند آن را حرکت دهد بنابر این ربات از مکان قبلی خود باید به پشت کره برود و برای هر حرکت کره این ماجرا تکرار میشود بنابر این یک الگوریتم سرچ کلی برای کره انجام میدهیم تا مقصد و برای هر حرکت کره یک الگوریتم سرچ را انجام میدهیم تا ربات را از مکان قدیمش به مکان جدیدش بیاوریم.

2)تابع شهودی انتخاب شده همان فاصله ی منهتن است برای کره تا مقصد و همین طور در هر بار حرکت ربات بین مبدا و مقصدش و واضح است که فاصله ی منهتن از هزینه ی تحمیلی کمتر است در اینجا

3)کلاس نود : در واقع مختصات و جزییات درگر یک نود را نگهداری میکند

مختصات آن نود در صفحه را بر میگرداند.Get_Coor و تابع

کلاس گراف :

:Abundantبرای این است که کره های دیگر به غیر از کره ی فعلی را به

حالت مانع تبدیل کنیم در ماتریس

این تابع برای این است که همسایه های آزاد Get_neighbors

هر نود را برگرداند

این تابع برای بازگرداندن پرنت های هر نود به حالت اولیه است برای reset شروه مجدد سرچ

این کلاس توابع مربوط به سرچ مورد نظر را دارد a_star: کلاس

این تابع مقدار هیوریستیک بر حسب فاصله ی منهتن را محاسبهheuristic: میکند

این تابع برای تغییر مکان ربات برای یک حرکت کره استpositioning:

این تابع مقصد مد نظر و نقطه ی شروع را میگیرد و بر حسب Search: سرچ را انجامم میدهد.A_star الگوریتم

bi_bfs: کلاس

است bfs این کلاس برای سرچ دو طرفه ی

همان کاری را انجام میدهند که در کلاس search , positioningتابع انجام میدادندA_star

را انجامbfs این تابع هر بار که فراخوانی میشود یک گام از الگوریتم: Bfs میدهد

تعیین میکند که از دوطرف به همbfs بعد از یک گام انجام Is_intersecting: رسیده اند یا نه و در چه نودی

همسایه ها را به عنوان یک نود bfs در هر مرحله سرچAdd_edge: فرزند به درخت ما اضافه میکند.

پیاده سازی شده ids: در این کلاس سرچ ids کلاس

همان کاری را انجام میدهند که در کلاس search , positioningتابع انجام میدادندA_star

Dls:

: این کلاس وظیفه ی خواندن فایل ورودی و ساختن main کلاس

آبجکت های الگوریتم های سرچ و ساختن ماتریس مورد نظر و نود ها و ...

و نمایش خروجی را دارد

: برای نمایش حرکت ربات بر روی صفحه ساخته شده است.Gui کلاس

برای مرتب سازی فایل اصلی برنامه، توابع به این کلاس منتقل شدهmapper:کلاس

مقایسه های انجام شده:

- زمان صرف شده: بهترین زمان برای*A است به دلیل تابع Heuristic استفاده شده بهترین انتحاب و درنتیجه کمترین زمان سرچ را دارد. در رتبه دوم BI-BFS و در آخرIDS قرار دارد. (تست زمان اجرا)

- پیچیدگی زمانی: به ترتیب زمان صرف شده است.

- تعداد گره های تولید شده: در IDS به طور کلی تعداد گره بیشتری تولید میشود به دلیل مرتب صدا زدن DFS. در مرحله بعد BI-BFS و در آخر *A که بهترین عملکرد را دارد.

- تعداد گره های گسترش داده شده: بیشترین گره گسترش داده شده را الگوریتم IDS دارد. در مرحله دوم Bi_BFS و در آخر *A

عمق راه حل: بیشترین عمق برای IDS است به دلیل اینکه DFS انجام شده لزوما بهترین راه حل را بر نمیگرداند. در مرحله دوم *A و بهترین عملکرد برای BI-BFS است.