

Amin Amiri Torshizi – 9812399206

Iraj Gharaee – 9912223225

Paria Razavi - 9812223261

**داکیومنتیشن**

* **فاز اول :**

پیاده سازی تابع و توابع و کلاس هایی که برای استفاده در الگوریتم های جستجو نیاز است .

**مدلسازی مسئله**

هر محل محیط ( میز ) میتواند یک یا چند حالت داشته باشد . حالت ها به صورت زیر دسته بندی میشود

* در محل کره وجود دارد ( )
* محل جایگاهیست که کره درخواست داده شده است ( )
* در محل مانع وجود دارد ( )
* در محل ربات وجود دارد ( )
* محل خالی است

اشیا کره و ربات ، تنها اشیا متحرک محیط است و بقیه حالت ها ( هدف ها ، موانع ) ثابت هستند .

فرض کنیم محیط باشد ، برای هدف ها و موانع دو ماتریس زیر را درنظر میگیریم

ربات هنگاه عبور از محل محیط با توجه به نوع محل باتری آن مصرف میشود ، باتری مصرف شده را هزینه عبور از هر محل در نظر میگریم و برای هزینه ماتریس زیر را درنظر میگیریم ( اگر در محل مانع وجود داشت را بیشترین مقدار ممکن در نظر میگیریم )

جایگاه کره هایی که در هدف اند و کره هایی که در هدف نیستند و ربات را در نظر میگیریم و هر شی یا حالت را در نظر میگیریم .

* + **کلاس**
* **ویژگی ها**
* محل قرار گیری حالت یا شی ( دو متغیر از نوع ) که در چه سطر و ستونی قرار دارد
* نوع حالت یا شی ( )
* **متدها**
* بازنویسی متد های مقایسه ، جمع ، هش و کپی
* متد که سطر و ستون را به خروجی میبرد

مقایسه دو شی در مکانی که قرار دارند فقط انجام میشود و شامل نوع شی نیست . جمع شی سطر و ستون آن ها باهم جمع میشود .

* + **کلاس**
* **ویژگی ها**
* ربات ( متغیری از نوع )
* لیستی از کره هایی که در هدف قرار ندارند ( متغیر های لیست از نوع )
* لیستی از کره هایی که در هدف قرار دارند ( متغیر های لیست از نوع )
* عمق ( متغیر از نوع )
* هزینه ( متغیر از نوع )
* **متدها**
* باز نویسی متد های مقایسه ، بزرگتر ، کوچکتر ، کپی، پیدا کردن هیورستیک و هش

مقایسه دو به صورت مقایسه جایگاه ربات در دو و مقایسه تعداد کره هایی که در مکان مشابیه قرار دارند در دو است . مقایسه بزرگتری و کوچکتری بر اساس هزینه انجام میشود .

با گرفتن

* + تابع
* ورودی : و مجموعه ای از های دیده شده
* خروجی : لیستی از های تولیده شده توسط ورودی
* نحوه کارکرد

یک لیست از ها در نظر میگیریم ، در هر ربات میتواند به 4 جهت حرکت کند ( بالا ، پایین ، چپ ، راست ) بردار های حرکت این 4 جهت را درنظر میگیریم

و ابتدا با یک تابع بررسی میکنیم که ربات میتواند با این بردار حرکت کند یا خیر و اگر میتواند حرکت کند برگرداده شود و اگر حرکت ربات موجب شده کره ای حرکت کند ، کره را هم به خروجی ببرد .

اگر کره میتواند حرکت کند از نوع است و در غیر این صورت است

اگر ربات میتواند حرکت کند و اگر نتواند حرکت کند است

حال اگر ربات توانست حرکت کند یک جدید با جابجایی ربات ساخته میشود ( اگر کره هم جابجا شده بود آن را هم در نظر میگیری برای جدید ) و اگر این در مجموعه وجود نداشت به لیست ها افزوده میشود و در پایان ورودی به مجموعه ها افزوده میشود و لیست ها به خروجی برده میشود .

* مرتبه زمانی

اگر تعداد کره ها را در نظر بگیریم مرتبه زمانی تابع از میباشد .

* **فاز دوم :**

پیاده سازی الگوریتم های

1. BFS ( Breadth First Search )
2. DFS ( Depth First Search )
3. IDS ( Iterative Deepening Search )
4. UCS ( Uniform Cost Search )

* پیاده سازی

در نظر میگیریم اصلی ( زمانی که هنوز ربات حرکتی نکرده ) را .

با استفاده از صف میخواهیم را پیاده سازی کنیم ، فرض کنیم صف باشد . را به اضافه میکنیم و این الگوریتم را روی اجرا میکنیم .

را از صف خارج میکنیم بررسی میکنیم که تمام کره ها در سرجایشان است اگر نبود های گسترش یافته را به صف اضافه کرده و همین روند را تکرار میکنیم تا به هدف برسیم .

* پیاده سازی

در نظر میگیریم اصلی ( زمانی که هنوز ربات حرکتی نکرده ) را .

با استفاده از پشته میخواهیم را پیاده سازی کنیم ، فرض کنیم پشته باشد . را به اضافه میکنیم و این الگوریتم را روی اجرا میکنیم .

را از پشته خارج میکنیم بررسی میکنیم که تمام کره ها در سرجایشان است اگر نبود های گسترش یافته را به پشته اضافه کرده و همین روند را تکرار میکنیم تا به هدف برسیم .

* پیاده سازی

روند الگوریتم را با شرط توقف در عمق ادامه میدهیم و را در هر مرتبه تکرار تغییر میدهیم .

* پیاده سازی

در نظر میگیریم اصلی ( زمانی که هنوز ربات حرکتی نکرده ) را .

با استفاده از مجموعه میخواهیم را پیاده سازی کنیم ، فرض کنیم مجموعه باشد . را به اضافه میکنیم و این الگوریتم را روی اجرا میکنیم .

در ، که دارای کمترین هزینه است را خارج میکنیم بررسی میکنیم که تمام کره ها در سرجایشان است اگر نبود های گسترش یافته را به مجموعه اضافه کرده و همین روند را تکرار میکنیم تا به هدف برسیم .

* **فاز سوم :**
* پیاده سازی A\*:

ابتدا که هنوز ربات حرکتی انجام نداده state آن را root در نظر می‌گیریم.

مجموعه‌ی حاوی تمام state هایی که قابل دسترسی هستند را stateSet در نظر می‌گیریم و تنها state موجود ما که همان root است را به آن اضافه می‌کنیم.

در با بررسی تمام state های موجود در setState آن stateای که heuristic + cost کمتری دارد را خارج می‌کنیم. شرط اتمام الگوریتم که رسیدن همه کره ها به هدف هستند را بررسی می‌کنیم، اگر برقرار نبود نود های فرزند state انتخاب شده را به setState اضافه می‌کنیم.