دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

تمرین پیاده‌سازی ۱

جستجوی A*

محمدرضا دولتی

۹۷۱۱۰۴۱۱

استاد:

دکتر آرش عبدی هجراندوست

آذر ۱۴۰۰

گزارش کد:

در قسمت اول کد چهار جهت بالا،پایین،چپ و راست را برای ربات در نظر گرفته‌ایم و سپس یک کلاس نود تعریف کرده‌ایم تا در آن بتوانیم فقط با یک تابع مقایسه اُردری از تمام آنها بدست آوریم.

بعد از آن برای کلاس یک Init تعریف می‌کنیم تا صفات کلاس را در آن بیاوریم و مقداردهی کنیم.

سپس یک طول رشته برای یکی از صفات کلاس که سلف تعریف می‌کنیم تا طول رشته در جهت x,y را داشته باشیم و در تابع equal ای که تعریف کرده‌ایم مقدار سلف و آبجکت را مقایسه می‌کنیم تا اگر در یک نود برای مثال self.x == o.x بود برگرداننده شود.(به طور مشابه برای y,distance نیز هم همین عملکرد را داریم.)

در قسمت بعدی تابع gt ای تعریف کرده‌ایم تا ببینیم بقیه مقادیر از سلف بزگتر است یا نه که در این قسمت از Manhattan Distance برای دو بردار x,y استفاده کرده‌ایم.

در چند خط آخر این قسمت نیز شرایط حرکت کردن ربات نوشته شده است.

در قسمت بعدی is_movable ای تعریف کرده‌ایم تا ببینیم ربات کاوشگر می‌تواند حرکت کند یا خیر که قسمت مانع و یا دیواری که جلوی حرکت ربات را نیز می‌گیرد در همین قسمت تعریف شده‌است.

برای قسمت بعدی یک find_valid_adjacent تعریف کرده‌ایم تا سلول های قابل حرکت برای ربات را پیدا کند و آنها را در یک آرایه برگرداند.

برای قسمت بعدی هم یک initialize تعریف کرده‌ایم تا مقدار اولیه و تمامی input ها را در این قسمت تعریف کنیم با توجه به متن سوال و گفته های استاد که فرض خودمان را بگوییم، من به این شکل فرض کردم که برای ورودی‌ها در خط اول ورودی دو عدد داده می‌شود که بیانگر مختصات ربات کاوشگر ما هستند.

در خط بعد موقعیت باتری ربات در قالب دو عدد داده می‌شود که عدد اول سطر و دومی ستون را مشخص می‌کند.

در خط بعد، دو عدد m,n داده می‌شود که ابعاد صفحه هستند. عدد اول تعداد سطرها و عدد دوم تعداد ستون‌ها را نشان می‌دهد.

در n خط بعدی در هر خطر m کاراکتر ظاهر می‌شود که بیانگر نقشه است و طبق داده‌های سوال و جدول SampleRoom.xml داده شده کاراکتر obstacle به معنای دیوار غیرقابل عبور، کاراکتر empty نقاط قابل عبور و کاراکتر robot نقطه شروع ربات کاوشگر و در نهایت کاراکتر Battery محل قرارگیری باتری را نشان می‌دهد.

در قسمت بعدی بعد از گرفتن ورودی‌ها مهمترین قسمت کد که همان الگوریتم A* می‌باشد را نوشته‌ایم که نحوه‌ی عملکرد این الگوریتم و کارکرد این الگوریتم برای ربات و کمک گرفتن از آن برای پیدا کردن باتری را نیز نشان می‌دهد.

در آخرین تعریف هم چاپ راه ربات و آمدن آن دو خروجی آمده که نشان می‌دهد ربات برای رسیدن به باتری چه مسیری را باید طی کند که بدین معنی است که در خروجی در هر خط مسیر رسیدن ربات کاوشگر به باتری با کمک الگوریتم A* چاپ می‌شود و این موضوع به این صورت است که برای خروجی هر خط شامل دو عدد مختصات است که نشانگر مسیر ربات به باتری است و همچنین نقاط ابتدا و انتهای مسیر هم نوشته می‌شود.

در پیاده سازی، در صورتی که چند خانه اولویت برابر برای بررسی شدن و اضافه شدن به Frontier دارند، به ترتیب ابتدا خانه بالا، سمت خانه چپ، سپس راست و بعد خانه پایین (در صورت امان پذیر بودن حرکت) اضافه می‌شوند.

در محاسبه هزینه مسیر به روش A*، هزینه حرکت همه نقاط برابر ۱ است به جز نقطه‌ای که باتری در آن قرار دارد و هزینه صفر را دارد.

به عنوان مثال برای مثالی که در SampleRoom.xml نیز آمده است به این صورت داده را به کد می‌دهیم که :

Inputs:
1 4
4 4
6 6
emptyemptyemptyrobotemptyobstacle
obstacleemptyobstacleemptyemptyempty
emptyemptyemptyobstacleemptyempty
emptyobstacleobstacleBatteryobstacleempty
emptyobstacleemptyemptyobstacleempty
emptyemptyemptyemptyemptyempty

Outputs:
1 4
2 4
3 4
4 4

که این به این معنی است که در ابتدا طبق همان ورودی‌های که گفته شد، ورودی داده می‌شود و سپس نقشه که به صورت بالا و پشت هم نوشته می‌شود به برنامه داده می‌شود ر در نهایت مسیر نهایی ربات به باتری با روش A* در خروجی چاپ می‌شود.

نکته: کد مربوطه همراه این گزارش آپلود شده است.