مجید ادیبیان 9631003 پروژه امتیازی درس طراحی الگوریتم ها (MAZE) زمستان 98

در این پروژه باید با استفاده از الگوریتم های جستوجو مسئله گفته شده را حل کنیم.

مسئله به این صورت است که در یک گراف جهت دار و دارای رنگ از دو نقطه شروع می کنیم و هر دفعه از یک گره می توان از یالی استفاده کرد که همرنگ گره دیگر است و به دنبال مجموعه حرکات از این دو گره هستیم تا به گره هدف برسیم.

پیاده سازی برنامه با استفاده از زبان پایتون صورت گرفته است. در برنامه نوشته شده از الگوریتم DFS برای حل مسئله استفاده شده است.

در این برنامه در ابتدا ورودی ها از کاربر گرفته شده و به ازای هر گره یک آبجکت از کلاس Node ساخته شده است و اطلاعات آن گره در آن ذخیره گردیده است. برای ذخیره سازی همسایه های یک گره باید متفاوت عمل کنیم و چون در گراف گره ها و یال ها رنگی هستند در روش پیاده سازی شده به ازای هر رنگ از یال های آن تمام گره های همسایه که با یالی با آن رنگ هستن را ذخیره می کنیم و ساختمان داده استفاده شده یک دیکشنری می باشد که کلید در آن نام رنگ یال و مقدار در آن مجموعه تمام همسایه های با آن رنگ می باشد.

توجه: در مثال تعریف پروژه تمامی گره ها به ازای هر رنگ تنها یک یال دارند ولی در پیاده سازی انجام شده حالت کلی تر آن در نظر گرفته شده است.

برای پیاده سازی الگوریتم به صورت بازگشتی عمل شده است. ابتدا آبجکتی از کلاس DFS ساخته می شود و دو گره ابتدایی به عنوان ورودی به آن داده می شود و سپس تابع run از این کلاس صدا زده میشود که آغاز کننده تابع بازگشتی برای حل مسئله است. در این تابع بازگشتی هر دفعه دو گره ابتدایی و عددی که تعیین کننده نوبت برای بسط کدام یک از گره های ابتدایی را می گیریم و اگر این عدد صفر است گره ورودی اول را بسط می دهیم و اگر یک است گره ورودی دوم را بسط می دهیم. در هنگام بسط یک گره فرزندان آن گره را با توجه به محدودیت این مسئله باید تعیین کنیم و فرزندان آن تمام همسایه های آن است که هم رنگ با گره دیگر باشند که با توضیح گفته شده در قسمت قبل تمامی این همسایه های خاص در O(1) قابل دسترس خواهند بود. با پیمایش این فرزندان به ازای هر کدام باید همین تابع را صدا بزنیم و به جای گره ورودی ابتدایی فرزند ساخته شده را قرار می دهیم و نوبت را به گره دیگر تغییر می دهیم. (با استفاده از 1-turn می توان مقدار یک در turn را به صفر و مقدار صفر را به یک تغییر داد)

با گرفتن خروجی تابع در صورتی که جوابی بدست نیامده باشد (failed برگشت داده شده باشد) باید تابع را به ازای همان ورودی گره ها وولی با نوبت دهی بسط متفاوت نیز امتحان کرد و در صورتی که آن هم خروجی failed برگرداند به فرزند دیگر آن گره می رویم و اگر هر یک جوابی را برگرداندند باید مقدار این تابع را به آن اضافه کنیم و آن را برگردانیم تا در نهایت این تابع بازگشتی تمام اقدامات برای رسیدن به هدف را برگرداند.

برای جلوگیری از گیر کردن برنامه در حلقه بینهایت و همچنین برای جلوگیری از بسط گره های غیر لازم یک set تعریف شده است و هر دفعه که می خواهیم تابع را فراخوانی کنیم بررسی می کنیم که اگر مجموعه این سه ورودی (گره اول و گره دوم و نوبت) در مجموعه set حضور داشت یعنی در مراحل قبلی آن را طی کرده ایم و در نتیجه به ازای آن مقادیر تابع صدا زنده نمی شود ولی اگر در مجموعه وجود نداشت باید آن را به آن اضافه کنیم تا در دفعات بعدی به آن نرویم. همچنین هنگام برخورد با failed و بازگشت به مراحل قبلی باید این مقدار را از آن مجموعه حذف کنیم تا از راه های دیگر امکان بررسی آن ها وجود داشته باشد.

این مقادیر visited شده در set نگهداری می شوند و در نتیجه در O(1) امکان بررسی حضور یا عدم حضور یک عضو در آن وجود دارد.

در پایان برنامه نیز خروجی گرفته شده از الگوریتم را به روش گفته شده در تعریف پروژه نمایش می دهیم.

خروجی این برنامه برای ورودی مثال پروژه به شکل زیر است:

2)

R 21

L 7

R 20

L 13

R 19

R 13

L 15

R 14

L 16

R 9

L 11

R 0

L 2

L 1

R 3

L 5

R 7

L 10

R 13

L 6

R 14

L 1

R 9

1)

R 27

L 19

L 13

R 24

L 14

R 25

L 9

R 26

L 0

R 22

L 5

R 0

L 10

R 3

L 6

R 7

L 1

R 13

L 5

R 14

L 10

R 9

L 6

L 3