تمرین کامپیوتری ۱ : الگوریتم های جستجو

در این تمرین الگوریتم های جستجویی که در درس یاد گرفته اید را برای حل کردن یک ماز (Maze) پیاده سازی کرده و نتیجه عملکرد هر یک از این الگوریتم ها روی یک ماز را مشاهده خواهید کرد.

قسمت های جانبی برای اجرای این تمرین برای شما پیاده سازی شده و شما کافیست توابع مربوط به پیدا کردن راه حل را کامل کنید.همچینین برای اجرای قسمت گرافیکی تمرین نیاز به نصب کتابخانه pyGame دارید ک نکاتی در رابطه با نصب آن در انتهای صورت پروژه امده است.

فایل های همراه پروژه :

Maze.py
کلاس اصلی پروژه که اطلاعات مربوط به ماز را نگهداری می کند و توابع جستجو کننده رافراخوانی و بررسی میکند.

Displayer.py

شامل پیاده سازی توابع لازم برای نمایش ماز در محیط گرافیکی . نیازی به بررسی و یا تغییر کد مربوط به این کلاس ندارید.

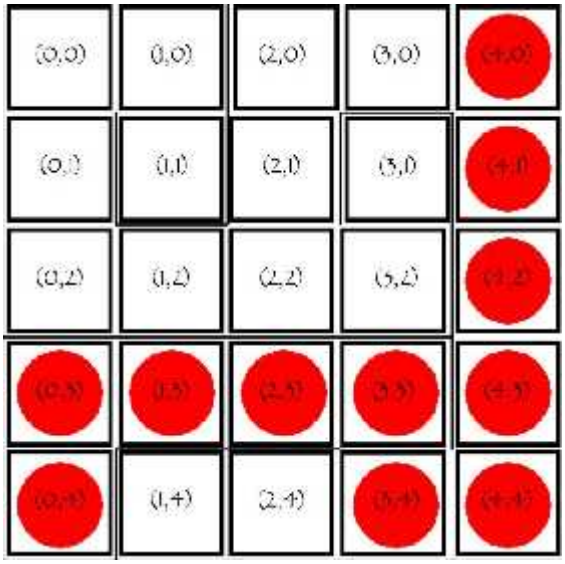
Solver.py

فایل اصلی که شامل تعدادی تابع بدون بدنه است که باید متناسب با نام هر تابع الگوریتم جستجوی مناسب آن را پیاده سازی کنید.

در ادامه تعدادی از توابع مهم پیاده شده درmaze.py و توابعی که باید پیاده کنید به صورت مختصر شرح داده شده اند:

Validate_answer(maze,path)

این تابع پس از صدا زدن الگوریتم جستجوی مورد نظر برای بررسی قابل قبول بودن جواب خروجی تابع مربوط استفاده میشود . فرمت خروجی مورد نظر به صورت یک لیست از تعدادی tuple دوتایی (x,y) است که هر tuple موقعیت خانه بازدید شده را نشان میدهد. در زیر به عنوان نمونه یک مسیر روی یک ماز ۵ در ۵ به همراه نحوه نامگداری خانه ها نمایش داده شده است.



مسیر متناظر :

[(0, 4), (0, 3), (1, 3), (2, 3), (3, 3), (3, 4), (4, 4), (4, 3), (4, 2), (4, 1), (4, 0)]

یک مسیر در صورتی‌که خانه شروع (گوشه پایین چپ) و خانه ی هدف(گوشه بالا راست) را داشته باشد و پیوسته باشد قبول میباشد.

setup_maze()

این تابع در ابتدای اجرای برنامه صدا زده میشود و در صورت درخواست معتبرکاربر یک مازپیش ساخته شده را از یک فایل میخواند و در غیراین صورت یک سایز در اندازه دلخواه کاربر به صورت رندوم تولید می کند .باتوجه به این که سایز صفحه نمایش ۵۰۰ در ۵۰۰ است از مقسوم علیه های این عدد مثل ۲۵،۱۰ و ۱۰۰ استفاده کنید .

get_neighbors(cell)

این تابع به عنوان خروجی لیستی از همسایه های خانه cell را میدهد که به صورت مستقیم به آن دسترسی دارد و دیواری بین ان دو نیست. همان طور که گفته شد cell ها به صورت tuple نمایش داده میشوند و ورودی این تابع هم یک tuple است.

روند کلی اجرای برنامه به این صورت است که پس از لود /تولید ماز هر یک از الگوریتم ها اجرا می شوند و در صورت معتبر بودن خروجی،تعداد خانه های بررسی شده برای رسیدن به حواب نمایش داده میشوند و خود جواب هم به صورت گرافیکی قابل نمایش است. فراخوانی هر تابع به صورت بلاک هایی از هم جدا شده است که با comment کردن هر بلاک میتوانید جلوی اجرای ان بلاک را بگیرید.

```
# """ UNCOMMENT TO DISABLE A*
print( ">> Testing A* solver...")
astar_path = astar_solver(m)
try:
validate_answer(m, astar_path)
print "A* solved maze. cost: ", len(astar_path), "cells visited"
print "Display A* solution? [y/n]"
display_command = raw_input()
if "y" in display_command:
d.draw_path(astar_path)
except AssertionError as e:
print "A* answer is invalid: " + e.message
# UNCOMMENT TO DISABLE A* """
```

توابعی که باید پیاده سازی کنید:

- dfs_solver(maze)
- dls_solver(maze, limit)
- iterative_dfs_solver(maze)
- bfs_solver(maze)

در هر یک از این توابع الگوریتم مورد نظر را پیاده سازی کنید تنها نکته قابل ذکر این است که با توجه به اینکه هزینه همه عملیات در این مساله برابر است تابع bfs_solver عملا کار الگوریتم Uniform Cost Search را هم انجام میدهد و نیازی به پیاده سازی UCS نیست.همچنین با توجه به اینکه عملکرد وپیاده سازی Iterative_deepening_search شبیه به DFS است و ممکن است در سایز بالا وقت بیشتری بگیرد،بیشتر پیاده سازی ان مد نظر است و بیشتر تست روی الگوریتم DFS انجام میشود .برای پیاده سازی iterative_dfs_solver باید از dls_solver استفاده کنید .

- astar_heuristic(maze, cell)
- astar_solver(maze)

این دوتابع هم برای پیاده سازی الگوریتم A* استفاده می شوند. astar_heuristic به ازای هر خانه ورودی مقداری عددی متناسب با آن خانه به عنوان تخمین هزینه آن خانه تا مقصد به عنوان خروجی بر می گرداند . برای تعیین‌تابع heuristic میتوانید از توابع معروف استفاده کنید یا ایده های خود را امتحان کنید .لزومی ندارد که نتیجه ی پیاده سازی این‌الگوریتم بهینه باشد و اینکه عملکرد درست داشته باشد و heuristic منطقی برای آن استفاده کرده باشید کافیست.

برای تکمیل تابع astar_solver باید از astar_heuristic استفاده کنید.

نکات :

- برای نصب pyGame به لینک زیر مراجعه کنید

http://www.pygame.org/download.shtml

- برای اطمینان از درست نصب شدن pyGame پس از نصب یک ترمینال پایتون باز کنید و دستور زیر را وارد کنید :

```
>>> import pygame
```

در صورتی که اتفاقی نیافتاد و خطایی دریافت نکردید، pyGame به درستی نصب شده است.

- این تمرین را در قالب گروه های ۲ نفره انجام دهید.
- در صورتی که مشکل و باگی در کد پیاده شده پیدا کردید در فروم مطرح کنید.