Task 1A - Finding a Path

Note: Go through this file only after you have gone through the tutorials provided in *resources.pdf* document in the *1.Theory* folder.

Please find the ***task_1a.py*** file in the ***codes*** folder.

Modify the ***task_1a.py*** to accomplish the following:

Given:

A set of five images, each contain a maze image.

- These images are provided in ***task_1a_images*** folder. An example image of maze **maze00.jpg** is shown in Figure 1.

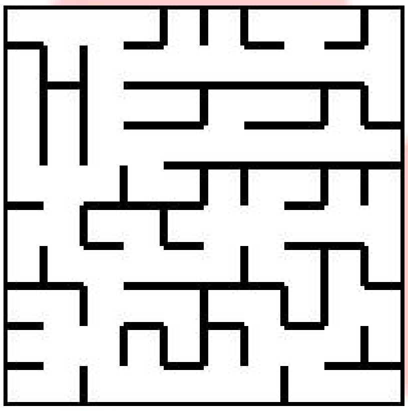


Figure 1: Example maze

- Cells are numbered in a maze using a coordinate system as given below:

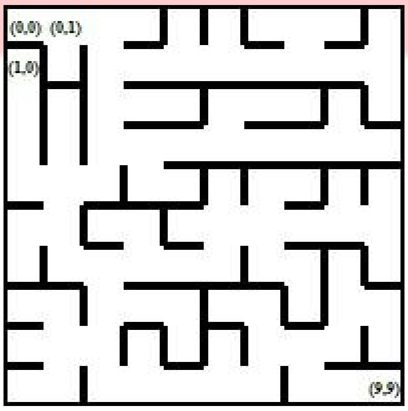


Figure 2a: Example maze (10X10) with START and END coordinates

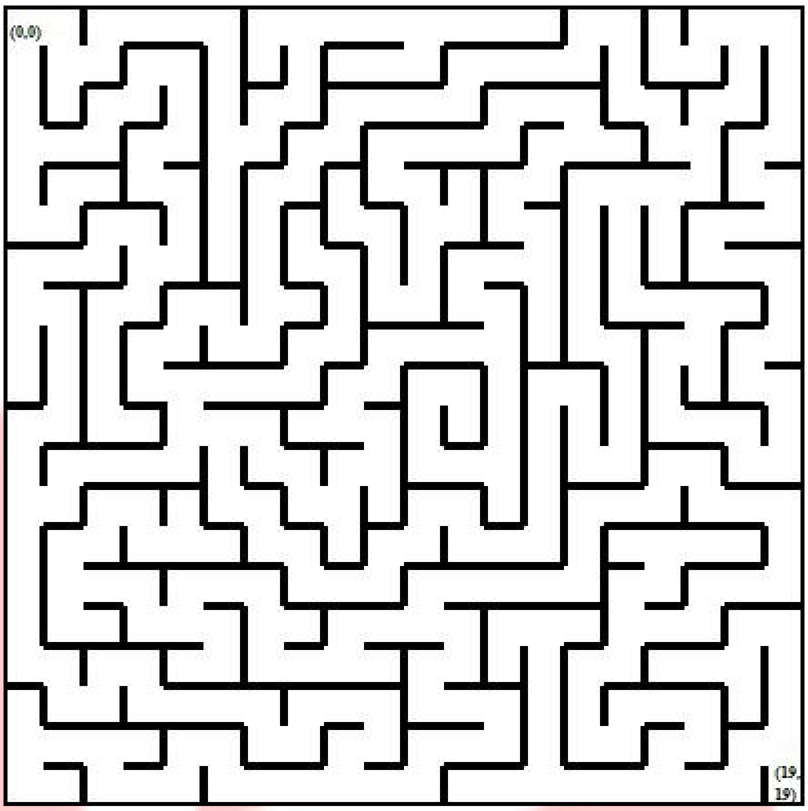


Figure 2b: Example maze (20X20) with START and END coordinates

Problem Statement:

Given such a maze, the task is to navigate from a START location in the maze to an END location, where the coordinates of START are (0,0) and END are (9,9).

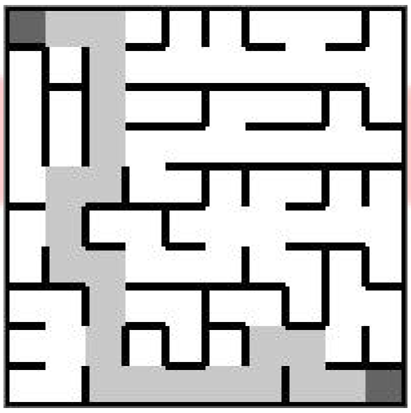


Figure 3: Example path of maze00.jpg with START (0,0) and END (9,9)

For the **task_1a.py** script, the output should be as following:

For maze00.jpg

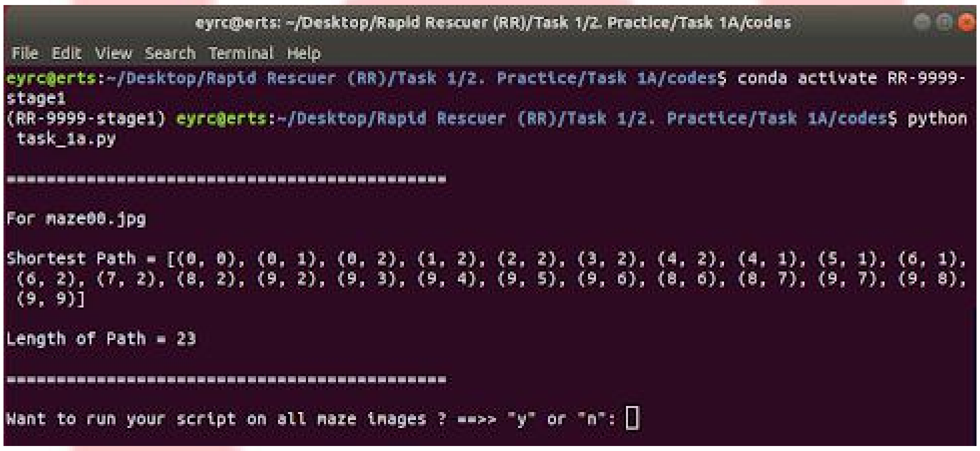
Shortest Path = [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (3, 2), (4, 2), (4, 1), (5, 1), (6, 1), (6, 2), (7, 2), (8, 2), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (8, 6), (8, 7), (9, 7), (9, 8), (9, 9)]

Length of Path = 23

Note: Output (Shortest Path) should be list of integer tuples as shown above.

A "snippet" of outline code is given in ***task_1a.py*** file.

- Teams modify the ***solveMaze()*** function to take a **test image** as **input** and return only **shortestPath and *readImage()*** function to take **image file path** as input and return only **binary image.**
- Make sure you run **task_1a.py** using the conda environment created in Task 0.
- For example, given the test image file in Figure 1 as input, the output will look as shown in Figure 4.



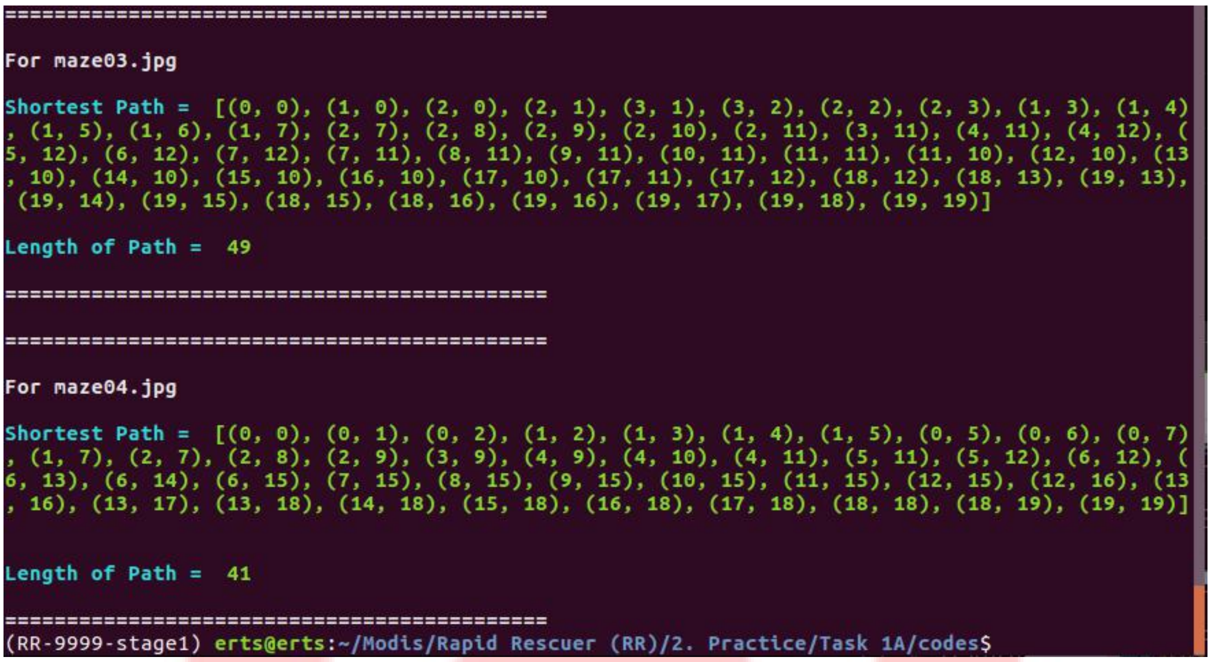
eyrc@erts: ~/Desktop/Rapid Rescuer (RR)/Task 1/2. Practice/Task 1A/codes
File Edit View Search Terminal Help

```
eyrc@erts:~/Desktop/Rapid Rescuer (RR)/Task 1/2. Practice/Task 1A/codes$ conda activate RR-9999-
stage1
(RR-9999-stage1) eyrc@erts:~/Desktop/Rapid Rescuer (RR)/Task 1/2. Practice/Task 1A/codes$ python
 task_1a.py

============================================

For maze00.jpg

Shortest Path = [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (3, 2), (4, 2), (4, 1), (5, 1), (6, 1),
 (6, 2), (7, 2), (8, 2), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (8, 6), (8, 7), (9, 7), (9, 8),
 (9, 9)]

Length of Path = 23

============================================

Want to run your script on all maze images ? ==>> "y" or "n": 
```

Figure 4: Output of maze00.jpg

Instructions:

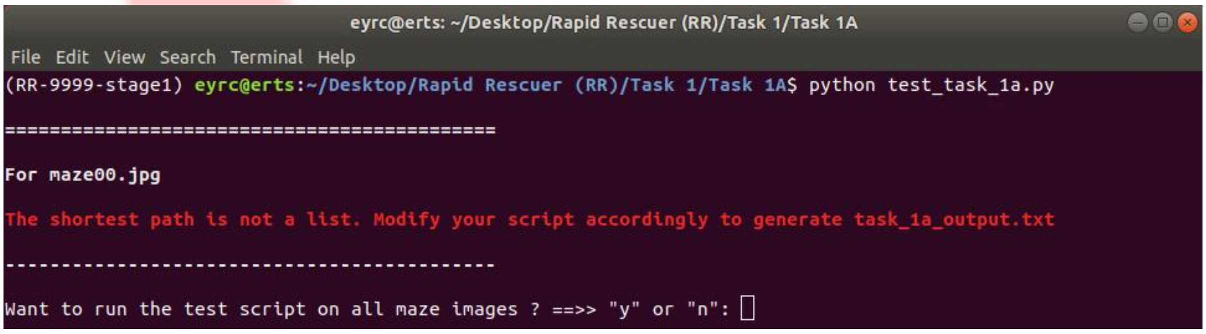
- Teams are **not allowed** to import any **library/module** related to **Image Processing** (apart from the ones installed in **Task 0**) in ***task_1a.py*** file. **If found so, it will lead to disqualification**.
- Do not edit the **main** function in ***task_1a.py*** file. Teams have to modify **only** the above mentioned functions.
- The **main** function, by default displays output for the one image, maze00.jpg and asks the user whether to check the output for rest images.
- To check the output for all the test images, type "**y**" and press Enter. Program shows the output of cell coordinates for all images as shown in Figure 5.



```
============================================

For maze03.jpg

Shortest Path =  [(0, 0), (1, 0), (2, 0), (2, 1), (3, 1), (3, 2), (2, 2), (2, 3), (1, 3), (1, 4)
, (1, 5), (1, 6), (1, 7), (2, 7), (2, 8), (2, 9), (2, 10), (2, 11), (3, 11), (4, 11), (4, 12), (
5, 12), (6, 12), (7, 12), (7, 11), (8, 11), (9, 11), (10, 11), (11, 11), (11, 10), (12, 10), (13
, 10), (14, 10), (15, 10), (16, 10), (17, 10), (17, 11), (17, 12), (18, 12), (18, 13), (19, 13),
 (19, 14), (19, 15), (18, 15), (18, 16), (19, 16), (19, 17), (19, 18), (19, 19)]

Length of Path =  49

============================================

============================================

For maze04.jpg

Shortest Path =  [(0, 0), (0, 1), (0, 2), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5), (0, 6), (0, 7)
, (1, 7), (2, 7), (2, 8), (2, 9), (3, 9), (4, 9), (4, 10), (4, 11), (5, 11), (5, 12), (6, 12), (
6, 13), (6, 14), (6, 15), (7, 15), (8, 15), (9, 15), (10, 15), (11, 15), (12, 15), (12, 16), (13
, 16), (13, 17), (13, 18), (14, 18), (15, 18), (16, 18), (17, 18), (18, 18), (18, 19), (19, 19)]


Length of Path =  41

============================================
(RR-9999-stage1) erts@erts:~/Modis/Rapid Rescuer (RR)/2. Practice/Task 1A/codes$
```

Figure 5: Overall Output

- Teams can verify the output with the help of **output_1a.pdf** file given.
- Once done with the Task, run **test_task_1a.py** provided in ***codes*** folder. It will show the output of your program on terminal and also generate **task_1a_output.txt** file in the same folder.
- If **shortestPath** is not returned in proper format, you may encounter an error (example is as shown in Figure 6)


eyrc@erts: ~/Desktop/Rapid Rescuer (RR)/Task 1/Task 1A
File Edit View Search Terminal Help

```
(RR-9999-stage1) eyrc@erts:~/Desktop/Rapid Rescuer (RR)/Task 1/Task 1A$ python test_task_1a.py

============================================

For maze00.jpg

The shortest path is not a list. Modify your script accordingly to generate task_1a_output.txt

--------------------------------------------

Want to run the test script on all maze images ? ==>> "y" or "n":
```

Figure 6: Error in Output

Note: If you are not getting any output, check the following:

1. You are running **task_1a.py** and **test_task_1a.py** using the conda environment created in Task 0.
2. **task_1a.py** file is present in the same folder (Do not change the name of .py file, it must be **task_1a.py)**

3. ***images*** folder location is unaltered (Do not change the name of folder or images provided in that folder)
4. Input and output arguments of ***solveMaze()*** function is as specified (Do not change the name of function, it must be ***solveMaze***)
5. You might have declared global variable/s outside **main** function.

Functions to edit:

readImage(img_file_path)

solveMaze(original_binary_img, initial_point, final_point, no_cells_height, no_cells_width)

Input and Output arguments of above two functions should not be changed at any cost.

Functions not to edit:

main()

NOTE:

If team wishes to create some helper functions to solve the task, they must define it above ***solveMaze()*** function and call in ***solveMaze()*** function.

image_enhancer module highlights the shortest path returned by the ***solveMaze() function.***

Points to remember:

No statements should be included for displaying maze images as they already have been included by us in the main function (statements such as cv2.imshow(), cv2.waitKey(0), cv2.destroyAllWindows()).

While completing the functions in **task_1a.py**, you might require to print some information on terminal for debugging purposes, but make sure to **comment each of those print statements** before submitting the scripts to us, since whatever information required for one to know whether the Task is successfully completed or not, we have already included print statements in the main function.