ARK_TASK_03

PROBLEM STATEMENT:

Your first encounter is with an image called Level1.png. As you might know in a normal RGB scale the colour is represented by a number from 0 to 255 inclusive for each channel. We also know that an ASCII character can be represented as a binary number from 0 to 255 inclusive. So could the thin initial greyscale pixels of the image contain some useful information? Could it help us on what to do with the rest of the pixels in the image? There are a total of 4 files you would need for this task and are provided here. If builtin functions are used, you must be able to tell how they work, its limitations and the different parameters it takes up and how it affects its working.

RELATED WORK:

The first part was a direct approach to simply extract the gray values of the pixels of image.

The second task was to find the image of Zuckerberg in Level1.png(which was distorted).The third part was to match the image of zuckerberg to the original image of 'zucky_elon.png' and get the x-coordinate of the top left of coordinate.Next we had to decode a colour image and find the maze.Naturally as human instinct will do I solved the maze by hand at first to get APPLE.Then i opened the zip file and found a grey image which i had to converted to music.This was the most tricky part and i regret wasting a lot of time on how to get the music 'RICK ROLL'.While i was not able to decode the music for several days I implemented BFS,DFS(not using recursion but stack as recursion caused error "recursion value exceeded").

I implemented the Astar,Djikstra and also tried to code RRT(which was not satisfactory for mazes)

INITIAL ATTEMPTS

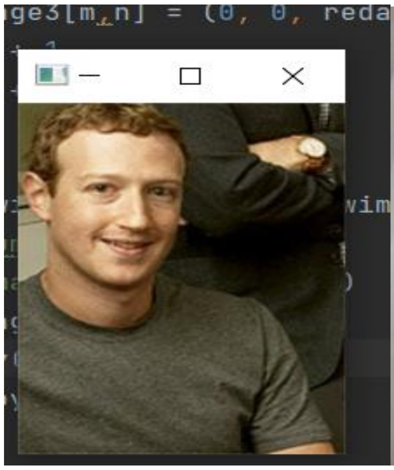
There was such no initial attempt as this was a treasure hunt and each treasure had an initial attempt.Nonetheless at first i was having absolutely no clue about what to do.Then i noticed in Level1.img I noticed some sort of gray patch in the top.Then the idea dawned upon me that the gray pixels have colour in range[0,255].(Actually this seems silly now but I was learning openCV from scratch so it took me time to realise).After getting the message things were easy for the next steps.The only error i was getting while compiling was the exchange of rows and columns,trying to design proper limits for while loops to extract elements and other normal errors.Next i got the maze and was naturally unable to solve it at first as I had no idea of algorithms like DFS,BFS.So i solved the maze using hand and got APPLE and also got the last gray image from which music was to be extracted.This part was easy but very very slick.I spent a lot of time,tried to learn scipy and stuff and after 4-5 days could find a easy to extract the music(like a serendipity😆).Then i learnt DFS,BFS,A* and RRT (to an extent).

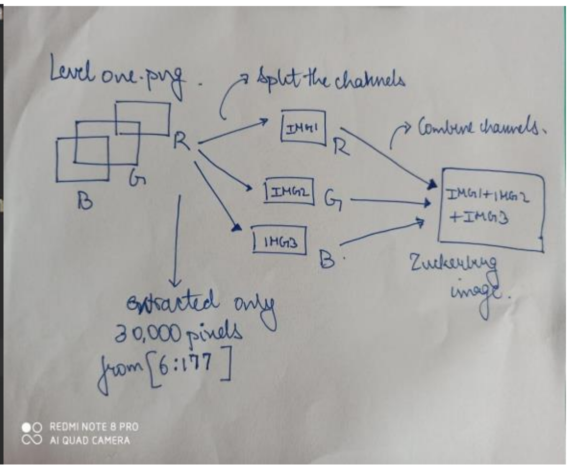
FINAL APPROACH:

STEP1: In the first step (**T3L0.py**)I imported the image in grayscale and they printed the pixel values,and got the message.The message was like”**Congratulations on solving ………...starts after the colon….”**.

STEP2: In this i properly calculated the number of pixels (200*150) i carefully printed out the image.First I extracted the pixel values from (Level1.png) and stored the b,g,r values in an array.

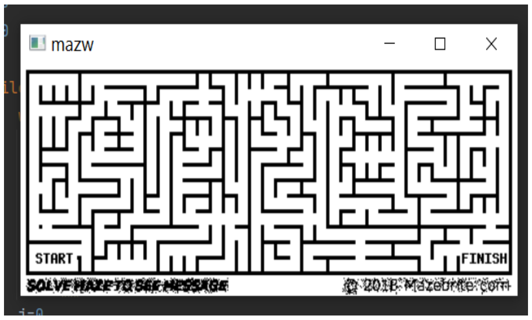
Then I created three images image1,image2,image3 consisting of only g,g,r channels and then I added the three images together to get the image of zuckerberg which looked like this.





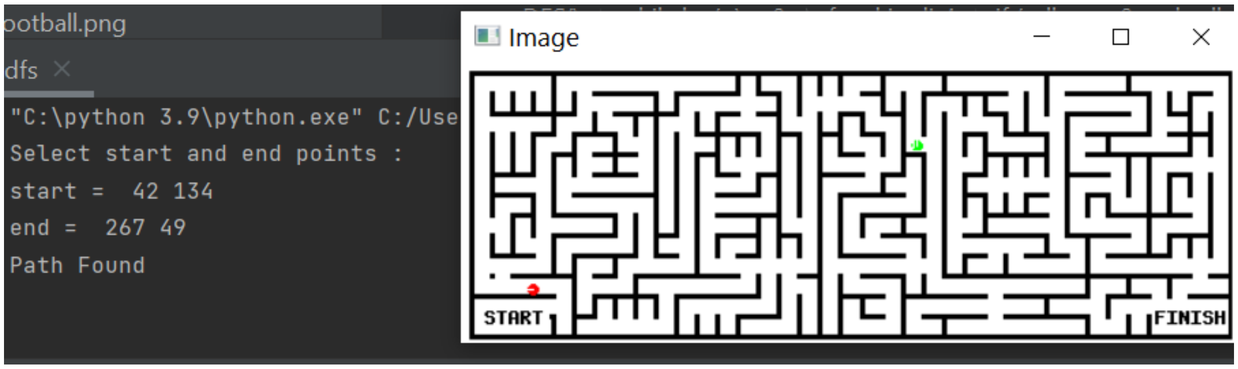
STEP 3: Here we had to find the x coordinate of the image.So i printed all the coordinates of the image of mark zuckerberg obtained from step 2.

STEP 4: Now I had to find out the maze in an image with colored noise.For that i needed to find the x coordinate in the previous step.There were many values of x coordinate but one thing was evident that x coordinate cannot be greater than 255 and by speculating the zucky_elon.png ,certainly the x coordinate was greater than 180-190.Therefore in a matter of 10-12 trials(where i had to change only a single value) and got the maze when the value it 230.To find out the maze in(**T3L3.py)** I ran through all the pixels of the image and printed only those pixels whose channels had value 230.



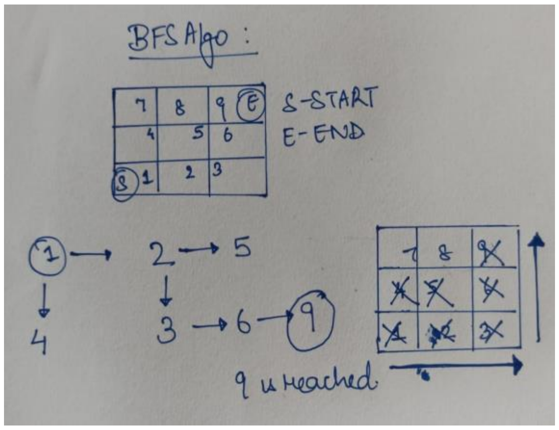
STEP 5: Thanks to ARK for giving us the names of the algorithms to solve a maze.I worked on BFS,DFS,Astar,Djikshtra and to some extent of RRT.

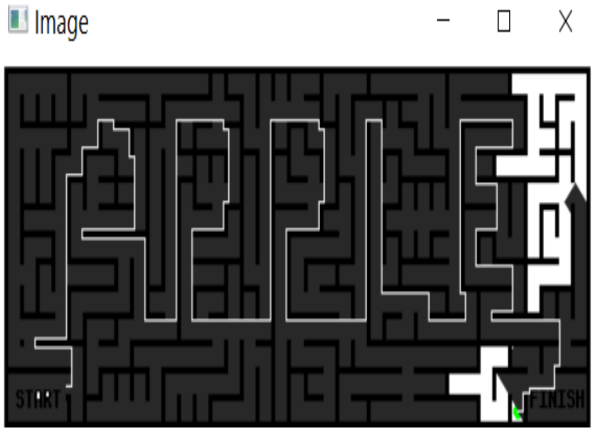
##To solve a maze i.e to reach an end point from starting point first you have to select a start point in red and an end point in green.And you get an output on screen.The console also prints the coordinates you selected and whether path found or not found.



```
"C:\python 3.9\python.exe" C:/Use
Select start and end points :
start =  42 134
end =  267 49
Path Found
```

A.)BFS: This is a very smart way to traverse a graph and gives us the shortest path between the start and end nodes.It uses simple queues which can be implemented using the concept of array(works on FIFO-First In First Out).Here we visit all the adjacent nodes of a node and also dont come back to a point already visited.Now the challenge was to implement in mazes.So in mazes each pixel can be made into a node.So i started writing the code using pixels as nodes and each time i visited the pixel i marked it 1 which helped me maintain a record of the visited and unvisited cells.Also in this program i hv used the concept of operator overloading(which is an amazing python not even in Java n CPP).The following image shows a 3*3 matrix of pixels and how we cover each pixel using BFS.





In this image BFS visits each of the neighbours of

each cell and while tracing back the path it traces 1,2,3,6,9.

Drawbacks are discussed in the end section

of this document.

B.)DFS: This is also a way although i don't prefer it personally and also it is less efficient.

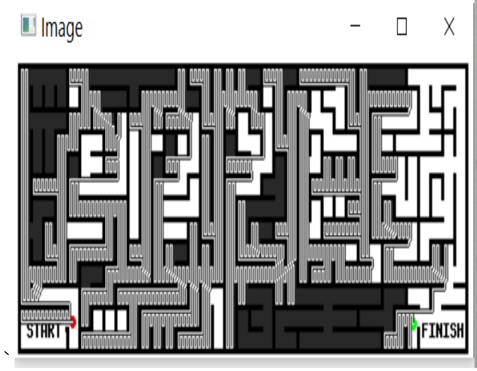
Now DFS can done in two ways.One is using recursion and other is using stacks(using arrays).

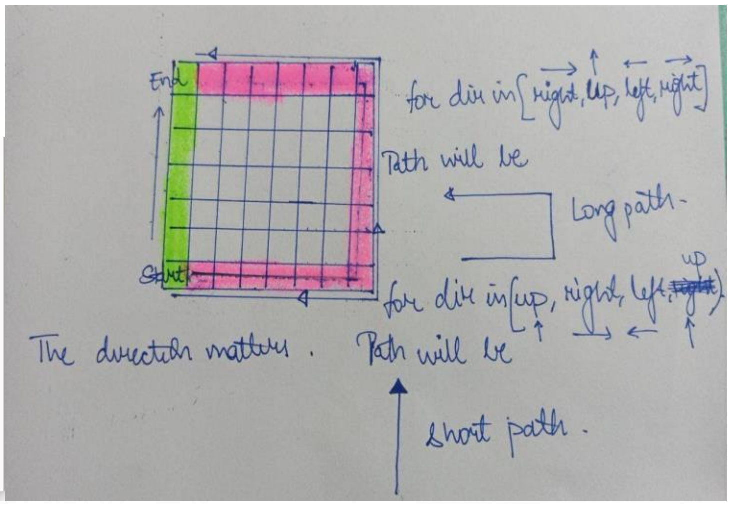
But the recursion once fails in mazes as the depth of mazes is greatly large and I got an error”**RECURSION LIMIT EXCEEDED”.**I searched about it in internet and got a solution to increase the recursion limit.But then I also found somewhere that changing recursion limit is not advised so i didn’t change it.Second is using stacks(LIFO).Here we go the end of path unless we encounter an obstacle and after we encounter an obstacle we reverse and take another path and the path goes on.The image will clearly depict.

Here the direction array greatly matter

```
     dir4 = [Point(1,0 ), Point(0, 1), Point(0,-1), Point(-1, 0)];

p = q.pop()

for d in dir4:

   cell = p + d

   if (cell.x >= 0 and cell.x < w and cell.y >= 0 and cell.y < h and 
v[cell.y][cell.x] == 0 and

           (img[cell.y][cell.x][0] != 0 or img[cell.y][cell.x][1] != 0 or 
img[cell.y][cell.x][2] != 0)):

       q.append(cell)

       v[cell.y][cell.x] = v[p.y][p.x] + 1
```

This is a part of my code here the dir4 array matters.The kind of direction chosen will matter a lot.




End
Start
for dir in [right, up, left, right]
Path will be
Long path.
for dir in [up, right, left, ~~right~~ up]
The direction matters.
Path will be
Short path.

DFS PATH

C.)A STAR: This is also an efficient algorithm.where the concept of heuristics comes in (i.e intelligent thinking).

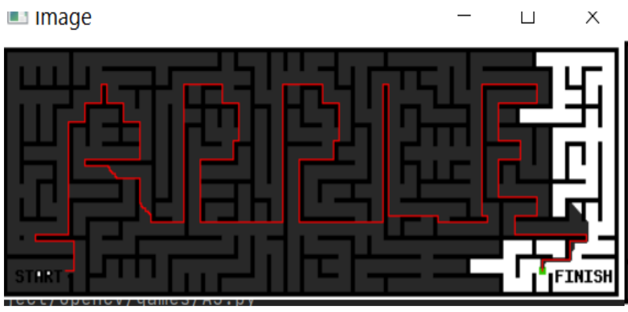
f(n)=g(n)+h(n) f(n)=g(n)+h(n)

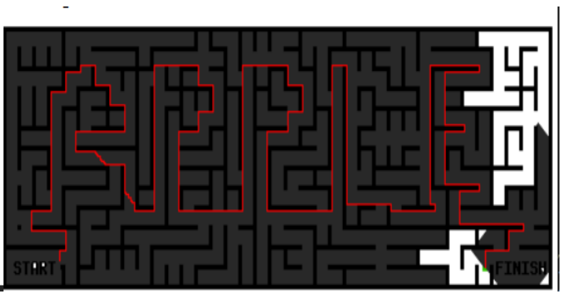
g(n) = shows the shortest path's value from the starting node to node n

h(n) = The heuristic approximation of the value of the node

D.)DJIKSTRA: This is similar to A star but A star is more efficient and A star is an informed search(we know where we are heading) but in Djikstra we don't know where we are heading.

The following two images show the output of two algos.





DJIKSTRA ASTAR

I TRIED THE RRT ALGO BUT WAS NOT ABLE TO DO IT AND IT SEEMS RRT IS NOT FIT FOR MAZES.

https://www.researchgate.net/publication/273391036_Intelligent_bidirectional_rapidly-exploring_random_trees_for_optimal_motion_planning_in_complex_cluttered_environments/link/58d9f8a392851ce5e92badf5/download

I tried to understand this but the maths was beyond my comprehension and I had no time .

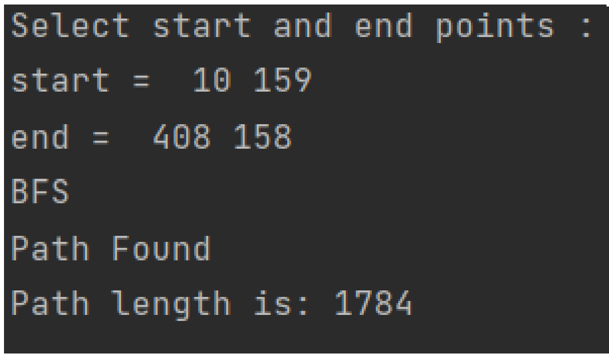
STEP 6: After solving the word APPLE I downloaded the zip file and extracted the image.It was a black and white image(had to be and music was stored in it).I took a lot of time tried learning scipy and all but then thought to convert all to binary,as in TEAMS channel i got the hint to think of bytes.

After converting to binary I saved it in a bin file.Then changed the extension of .bin file to .mp3 and that worked!

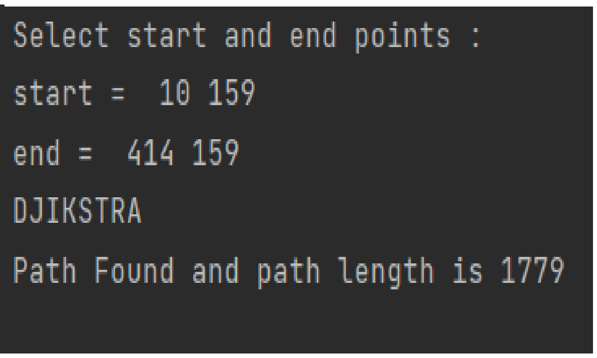
Finally it was solved!!!!.

RESULTS:

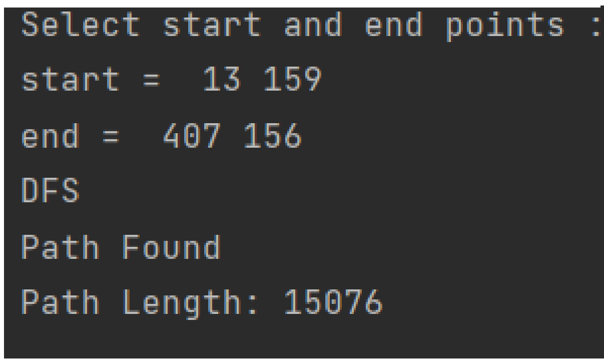
All algos gave the path but let us try to compare the result.



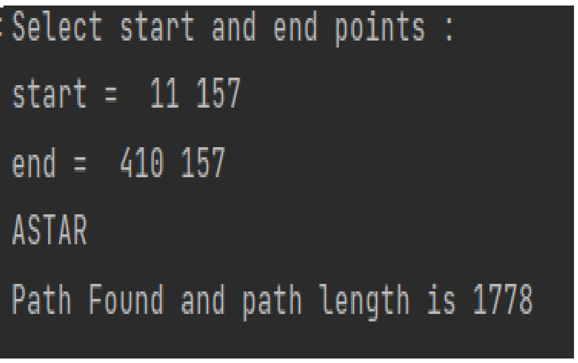
```
Select start and end points :
start =  10 159
end =  408 158
BFS
Path Found
Path length is: 1784
```



```
Select start and end points :
start =  10 159
end =  414 159
DJIKSTRA
Path Found and path length is 1779
```



```
Select start and end points :
start =  13 159
end =  407 156
DFS
Path Found
Path Length: 15076
```



```
Select start and end points :
start =  11 157
end =  410 157
ASTAR
Path Found and path length is 1778
```

THE ABOVE PICTURES PROVIDE A QUALITATIVE IDEA ABOUT PERFORMANCE.

FUTURE WORK:

In the algos there can exist more than one path but my algo gives only one.So I will ponder upon how I can get all the paths of equal length.(This can be used in navigation where we can get two paths whose length is same between start and destination).So the rider can chose any path of this choice seeing traffic and all as the distances are equal.

I will read on RRT,S*(mainly the heurestics and how can I improve that).

I will try to use Pygame to make it more interactive.

Right now I have added only two points start and stop.I would like to add more pitstops(or more points).Let us say I add a fuel station between start and end.So the path will be evaluated from start to fuel station and from fuel station to end.I will compare the results of the algos in this case.

CONCLUSION:

This problem was initially relatively easier but the algos and the music part made it tough.It took me time in reading about the algos.And I liked the form of treasure hunt.It was exciting and gave sparks of joy after clearing each level.

In ARK I feel Astar can be used for drone navigation and path finding.One of the examples which comes first to my mind is the parking lot!.**Self driving cars can use BFS easily to manouever their path through the complicated maze of walls in a parking lot.**These algorithms are designed just like how a human would navigate.I would also think in the same way as BFS or DFS works. So I feel this is one step closer to AI too.

REFERENCES:

Internet(many many websites)

GeeksforGeeks

Tutorialspoint

Used the book:

Competitive Programming in Python 128 Algorithms to Develop your Coding Skills by Christoph Dürr, Jill-Jênn Vie