AI Assignment Report

Tabish Khalid Halim ,
200020049
Anand Hegde ,
200020007

Department of Computer Science, IIT Dharwad

December 25, 2021

Contents

1 Approach

The basic approach we used for the A.I assignment is to use the concept of graphs & vertices and applied BFS{Breadth First Search} , DFS {Depth First Search} and DFID {Depth First Iterative Deepening Search}.

Along with applying the above algorithms for the PacMan , we have also set the preference order for adding the neighbor nodes which are:

$$DOWN > UP > RIGHT > LEFT$$

If the input number $\in \{0, 1, 2\}$, then the program executes the algorithms BFS , DFS and DFID respectively .

After visiting the neighbors in the maze graph, We can easily find out the length of the path and the number of states for the maze.

2 Variables used in Python Program

- **dfs_stop** : tells when dfs to stop.
- **goaldfs** : target state to achieve for DFS.
- **goaldfid** : target state to achieve for DFID.
- **statesdfs** : No. of states explored during DFS traversal .
- **statesdfid** : No. of states explored during DFID traversal .
- **DFIDstop** : to break out of recursion .
- **visited** : Variable created to store the set of visited vertices .
- **parent** : Tuple to store the parent of each node . It is used for finding path .
- **graph_input** : This stores the input given in as a list of lists .
- **m** : No. of rows in the Maze .
- **n** : No. of columns in the Maze .
- **states** : Variable to store no. of states explored .
- **pathlength** : Variable to store length of the path .

3 Functions created in Python Program

- **goal_state(i,j,graph_input)** : This function determines whether the coordinate (i,j) is the end goal for the PacMan or not .
- **move_gen(i,j,graph_input)** : This function's task is returning all possible moves available to the pacman , if the adjecent block has a space(' ') or astrik('*') , funtion returns its coordinates .
- **DFSUtil(v, visited,parent,graph_input,open_list)** : This is the recursive DFS Utility function .
- **DFS(graph_input,v=(0,0))** : The function to do DFS traversal. It uses recursive DFSUtil()- dfs utility function .
- **DFID(graph_input, depth,v=(0,0))** : The function to do DFID traversal. It uses recursive DFSUtil()- DFS Utility Function.
- **DFIDUtil(v, visited,parent,graph_input, depth)** : This is recursive DFID- utility function .
- **dfid(graph_input,v=(0,0))** : This is the Main DFID function- which calls DFID- which is dfs version for DFID . The extra thing is the depth here.
- **bfs(graph_input,s=(0,0))** : This function is used to perform BFS .
- **searchmethod(bdd,graph_input)** : Simple function to deal with the case wise operation to perform BFS, DFS or DFID as per the requirement

4 Pseudo Code

The main logic for the code is to find the value of k while determining the type of distribution for W_i.

def move_gen(i,j,graph_input):

global open_list

templist=[]

if(i<n-1):

if((graph_input[i+1][j]==' ' or graph_input[i+1][j]=='*') and ((i+1,j) not in open_list)):

templist.append((i+1,j)) if(i¿0): if(graph_input[i-1][j]==' 'or graph_input[i-1][j]=='*') and ((i-1,j) not in open_list): templist.append((i-1,j)) if(j¡n-1): if(graph_input[i][j+1]==' 'or graph_input[i][j+1]=='*')and ((i,j+1) not in open_list): templist.append((i,j+1)) if(j¿0): if(graph_input[i][j-1]==' 'or graph_input[i][j-1]=='*') and ((i+1,j) not in open_list): templist.append((i,j-1)) return templist

Logic :

Now , we have gone through each case of probable probability distribution functions for W_i ,

Next , we must first round of the value of k obtained and check whether the rounded off integer value is $\in \{2, 3, 4\}$.

If the value of $k \in \{2, 3, 4\}$ then , the test case is labelled successful but if not the it is labelled unsuccessful.

If multiple test cases are found successful , then the test case which has the least deviation of the value of k from the nearest integer is taken as the true test case and is considered for the final answer .

5 Graphical Analysis

I have made the P.D.F(Probability Distribution Function) of Y from the final P.D.F type obtained for W_i .

Based on my dataset , I have gotten my P.D.F type for W_i and Y as **half-normal distribution function** .

6 Results

The following conclusions have been made after evaluation of my program :

1. The P.D.F of W_i is **half-normal distribution function** .
2. The value of k is 2.
3. The value of σ is 2

Conclusion

Thus , the distribution function is of Half-Normal type and the value of $k = 2, \sigma = 2$.

7 References

- http://geeksforgeeks.com
- https://wikipedia.org
- https://stackoverflow.com