

# *MAZE SOLVER USING BFS*


(Breadth First Search)

**Rishit (191IT141)**

**Yash gupta (191IT158)**

**Sarthak Jain (191IT145)**

# INTRODUCTION

- In this project we are trying to implement a maze solving tool which identifies shortest path between the source and the destination.
  - The algorithm we used to solve our problem is Breadth First Search.
  - For generation of GUI we have used python turtle module.
  - Maze of any dimension irrespective of no. of rows and columns can be solved using this algorithm.
- 

# WORK DONE

For developing GUI :

- we have used python turtle module
- The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. Because it uses tkinter for the underlying graphics.
- The TurtleScreen class defines graphics windows as a playground for the drawing turtles. Its constructor needs a tkinter.Canvas or a ScrolledCanvas as argument. It should be used when turtle is used as part of some application.

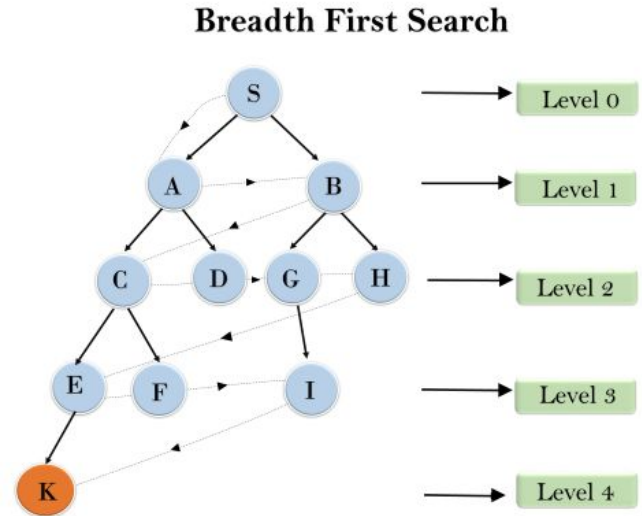
For finding Shortest path:

- Breadth first search
  - Backward movement from destination to source to determine shortest path
- 

# ALGORITHM

## 1)BFS: pseudo code

```
Set all nodes to "not visited";  
q = new Queue();  
q.enqueue(initial node);  
while ( q ≠ empty ) do  
{  
    x = q.dequeue();  
    if ( x has not been visited )  
    {  
        visited[x] = true;           // Visit node x !  
        for ( every edge (x, y) /* we are using all edges ! */  
            if ( y has not been visited )  
                q.enqueue(y);         // Use the edge (x,y) !!!  
    }  
}
```



BFS will search for all possible paths in order to find the shortest path we will be saving parent node of each vertex while doing BFS traversal of graph.

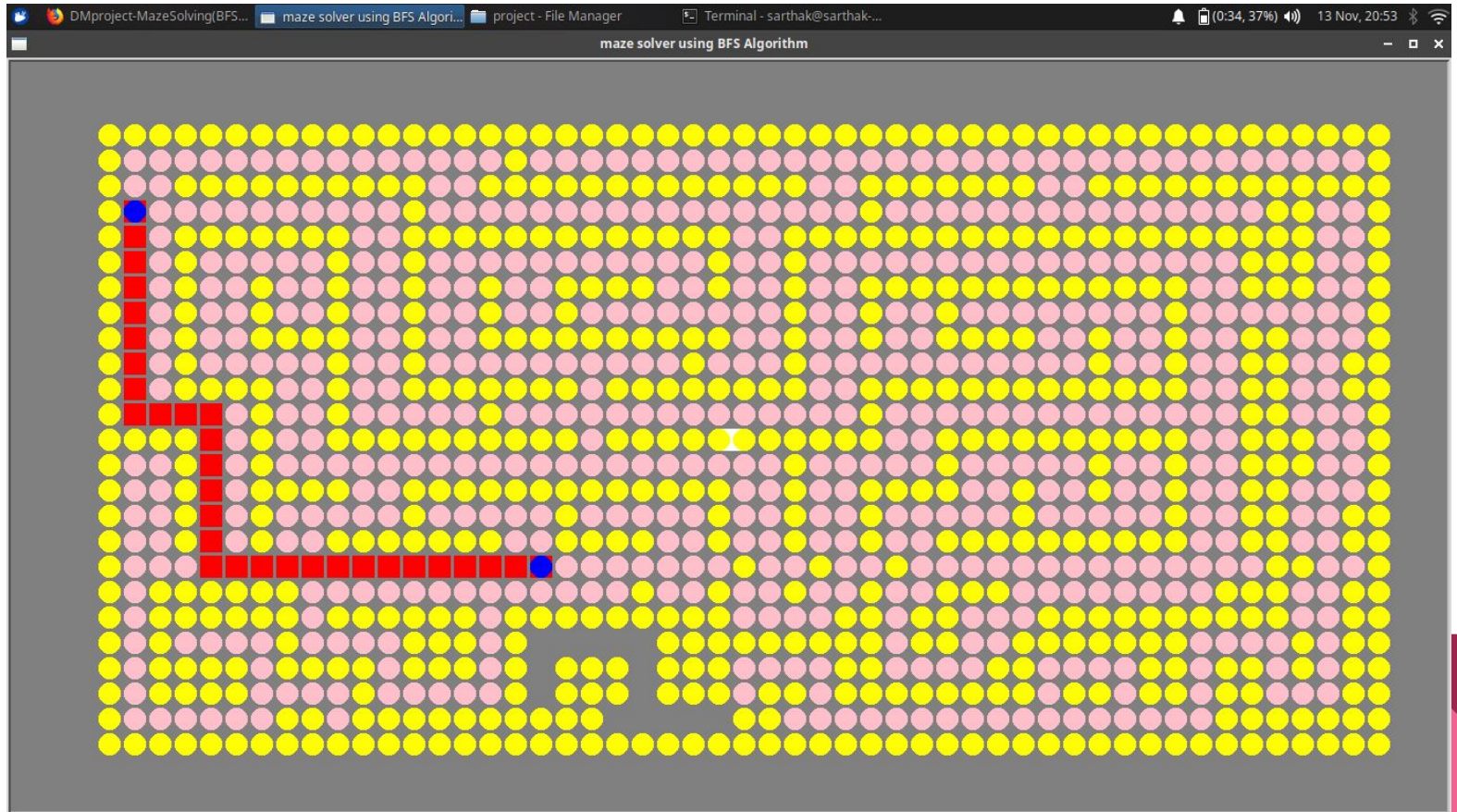
From destination we will be going in the backward direction till we encounter the source

Pseudo code:

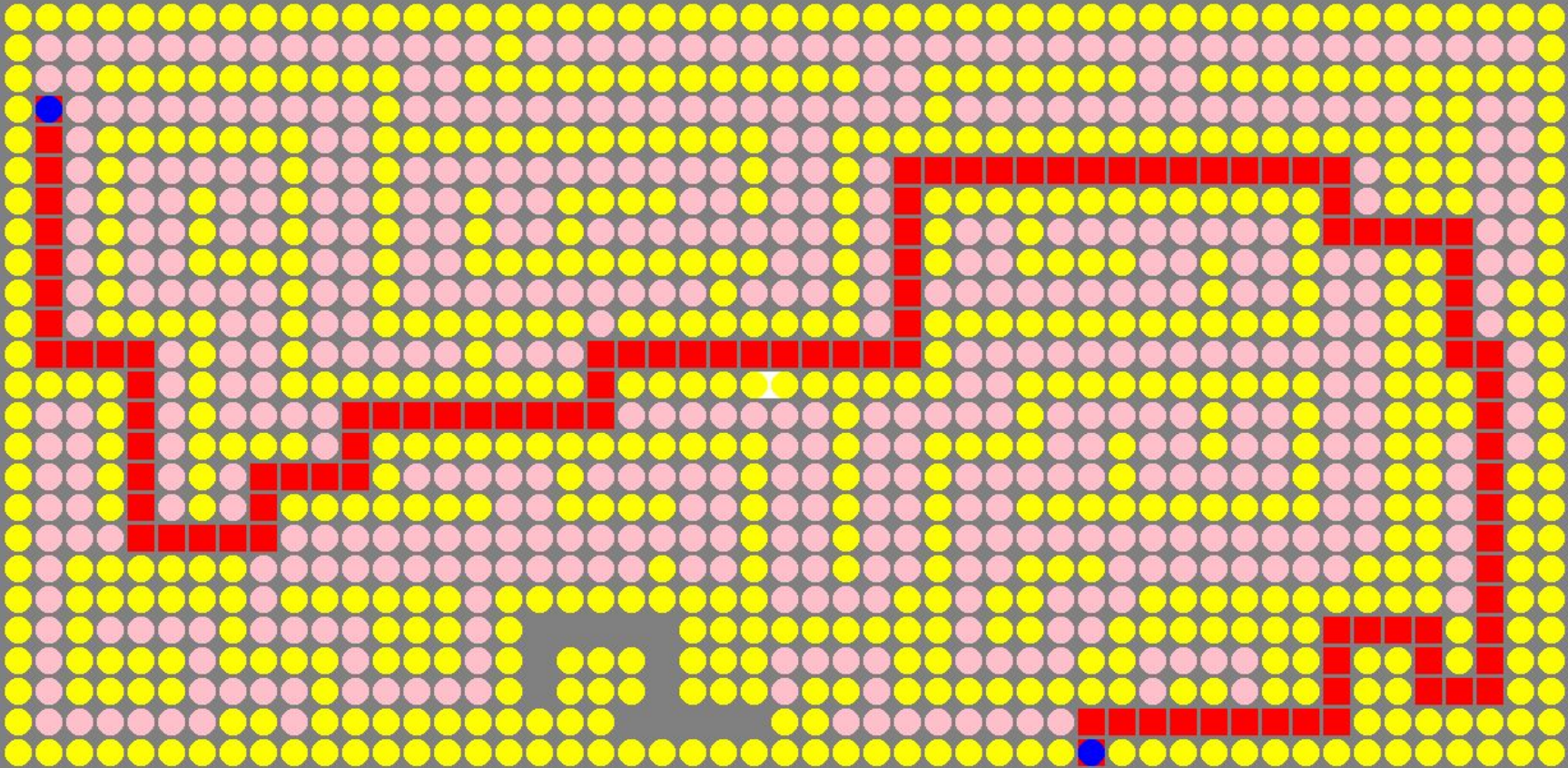
```
def back_route(end_x,end_y):  
    while (x, y) != (start_x, start_y):  
        x,y=parent[x,y]
```



# RESULTS







# CONCLUSION

- In the shown results yellow dots represent our maze.
- pink dots are formed during BFS traversal
- Blue dots represents source and destination
- Red dots represents shortest path

Turtle module of python, BFS and back movement are core of this project.

Project contains three files grid.py (for giving input) , setup.py  
(for maze and other stamps printing and main.py

