**Project proposal**

**Project Title:**

***“Maze Solver Using BFS”***

**Group members:**

**Shahir Anzal (sa-06081)**

**Zaeem Baig (mb-05962)**

**Project description:**

**Aim:**

The Aim of this is Project is to implement BFS algorithm learned in our course “Data Structure and Algorithm”

We are implementing BFS algorithm to solve a maze using shortest path using python Data structures

**Introduction:**

**What is BFS algorithm?**

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level

A standard BFS implementation puts each vertex of the graph into one of two categories:

1. Visited

2. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The algorithm works as follows:

1. Start by putting any one of the graph's vertices at the back of a queue.

2. Take the front item of the queue and add it to the visited list.

3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.

4. Keep repeating steps 2 and 3 until the queue is empty.

The graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the BFS algorithm on every node

Pseudocode:

**Pseudocode:**

create a queue Q

mark v as visited and put v into Q

while Q is non-empty

remove the head u of Q

mark and enqueue all (unvisited) neighbours of u

**Project Outcome:**

We learned how to solve maze using shortest path using algorithms learned in DSA like BFS algorithms and also learned some pros and cons of using BFS algorithm to solve maze like time complexity and real world challenges faced while implementing Data structure and algorithms which seems quite simple while learning

**Libraries:**

Time

Queue

Turtle