

DATA STRUCTURES PROJECT

NAME: RAHUL SINGHAL

BATCH: B10

ENROLLMENT NO:18103316

Synopsis

* This project aims to explore different applications of graphs and incorporate it to study different pathfinding algorithms.
* It is basically a C++ implementation of a Maze data structure using graphs with four pathfinding algorithms.
* The implementation of the data structure aims to be memory efficient.
* The code is designed to be easily adaptable to read and solve mazes of any format.
* Different files will be made for building the maze, then finding path using different algorithms and finally printing it which will all be later imported into the driver program.
* A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph.
* Four pathfinding algorithms are implemented using this project:   
  1)Depth First Search:Depth First Traversal. Depth FirstSearch (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

2)Breadth First Search: Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

3)Dijkstra :Dijkstra's algorithm can be used to determine the shortest path from one node in a graph to every other node within the same graph data structure, provided that the nodes are reachable from the starting node.

4)A\*: It is used in [pathfinding](https://en.wikipedia.org/wiki/Pathfinding) and [graph traversal](https://en.wikipedia.org/wiki/Graph_traversal) and enjoys widespread use due to its [performance](https://en.wikipedia.org/wiki/Computer_performance) and accuracy.

* It will include the usage of data structures like graphs, lists, ,stacks, queues, arrays and others if required.
* Templates will be used for performing functions on generic path rather than carrying it individually for each algorithm.
* Extensive use of effective data structures and multiple optimization routines would be done to provide a rich experience.