**PROBLEM**

A picture containing text, indoor, white, tiled

Description automatically generated Design a program that will find the shortest path through a maze

Gray shaded cells are walls, you cannot go through them.

White, black, dark grey ones are passage.

We can represent the maze as a graph, with a node at each decision point

A picture containing diagram

Description automatically generated

With the maze represented as a graph, we need to find the shortest path from the start point (0) to the end point (12)

The bfs method will return an array of parents which we can use to find the shortest path to the end point:

* the smallest number of vertices
* but not necessarily the smallest number of cells

The program needs the following data structures:

* an external representation of the maze, consisting of the number of vertices and the edges
* an object of a class that implements the Graph interface
* an array to hold the predecessors returned from the breadthFirstSearch method
* a stack to reverse the path

Algorithm for Shortest Path

Text, letter

Description automatically generated



First 2 lines creates the graph.



k : length of the shortest path

We can also say O(n)

Text

Description automatically generated with medium confidence

Testing

Test the program with a variety of mazes

Diagram

Description automatically generated

**Topological Sort of a Graph**

Diagram

Description automatically generated

This is an example of a directed acyclic graph (DAG)

It is a directed graph which contains no cycles

Once you pass through a vertex, there is no path back to the vertex

A picture containing scissors, necklet, accessory, enamel

Description automatically generatedAnother DAG:

A topological sort of the vertices of a DAG is an ordering of the vertices such that is (u, v) is an edge, the u appears before v

This must be true for all edges

There may be many valid paths through a DAG and many valid topographical sorts of a DAG

0 1 2 4 5 7 3 6 8 🡪 a topological order for the above DAG

0 3 1 4 6 2 5 7 8 🡪 another topological order for the above DAG

0 1 2 3 4 5 6 7 8 🡪 another topological order for the above DAG

Pick any edge, for example pick the edge goes from 3 to 6. 3 comes before 6 in the topological order and it must come before.

You have to put 0 at the beginning because it has no predecessor.

You cannot put 4 at the second order because it has 1 as predecessor.

If there is an edge from u to v in a DAG,

* then if we perform a depth-first search of the graph the finish time of u must be after the finish time of v (v is finished before u 🡪 v comes early in finish order array)
  + you cannot finish u before v since we can reach v from u

Reverse of the finish order should be the topological order since it satisfies above condition

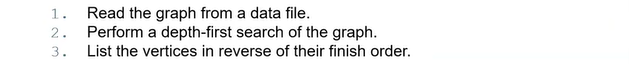
* In topological order we have to have u before v for an edge (u, v)

When we return to u, either v has not been visited or it has finished

It is not possible that v would be visited but not finished (a loop or cycle would exist)

*Having a back edge means there is a cycle, you have to make sure there is no back edge*

Algorithm for Topological Sort



Text, letter

Description automatically generated Text

Description automatically generated

Testing

Test the program on several different graphs

Use sparse graphs and dense graphs

Make sure each graph you test has no loops or cycles