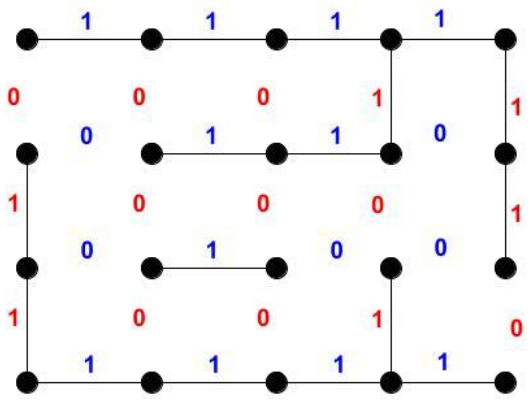
Lab9: Solving a Maze using Graphs

We are going to represent a maze as follows. There is a rectangular array/grid of points with N rows and M columns. Adjacent points in the grid are connected to represent a wall in the maze, and if they are disconnected then there is no wall there (can pass through).

Part-1:
Represent the grid using a graph. You can take as input from STDIN (or a file) a rectangular array of 1s and 0s, first to represent walls “across/horizontal” followed by “down/vertical”. For instance, the following maze and its input are shown below.



You can assume that the entrance is from the top left and exit of the maze is from the bottom right as shown. Your input can take the blue (horizontal lines) input as a grid followed by the red (vertical lines) input as a grid.

Part-2:
Now you must form a graph of “cells” in the grid. Adjacency in this graph will have the obvious meaning: is there an opening between the two cells? Note that the number of cells will be (N-1) x (M-1). You can “label” each vertex in the graph with (i,j) representing the cell’s 2D index (starting from (0,0) for the top left).

Part-3:
Find the shortest path between the top left cell and the bottom right cell, using a BFS on the above graph. You can output the cell labels in the shortest path to show a demo. The output for the above maze would be: (0,0) (1,0) (1,1) (1,2) (1,3) (2,3)

Part-4 (optional):
Show an animation of the maze being solved, in simplecpp.