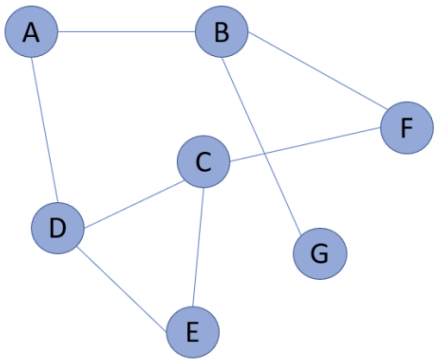
Homework 7: Graph Algorithms - 1

1. **Write BFS and DFS for a graph:** What would be BFS and DFS traversal for the below graphs. Write the nodes for BFS and DFS. Start at node A.



BFS: A, B, D, F, G, C, E

DFS: A, D, C, E, B, F, G

2. **Apply BFS/DFS to solve a problem**

You are given a 3-D puzzle. The length and breadth of the puzzle is given by a 2D matrix puzzle[m][n]. The height of each cell is given by the value of each cell, the value of puzzle[row][column] give the height of the cell [row][column]. You are at [0][0] cell and you want to reach to the bottom right cell [m-1][n-1], the destination cell. You can move either up, down, left, or right. Write an algorithm to reach the cell with minimal effort. How effort is defined: The effort of route is the maximum absolute difference between two consecutive cells.

If a route requires us to cross heights: 1, 3, 4, 6, 3, 1

The absolute differences between consecutive cells is: |1-3| = 2, |3-4|=1, |4-6|=2, |6-3|=3, |3-1|=2; this gives us the values: {2, 1, 2, 3, 2}. The maximum value of these absolute differences is 3. Hence the effort required on this path will be: 3.

Example:

Input: puzzle[][] = [[1, 3, 5], [2, 8, 3], [3, 4, 5]]

Output: 1

Explanation: The minimal effort route would be [1, 2, 3, 4, 5] which has an effort of value 1. This is better than other routes for instance, route [1, 3, 5, 3, 5] which has an effort of 2.

1	3	5
2	8	3
3	4	5

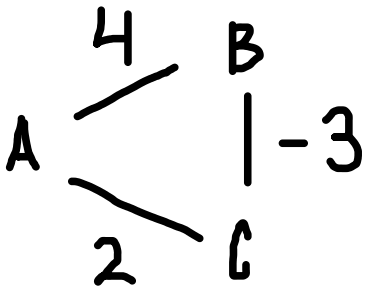
a. Implement the algorithm. Name your function **minEffort(puzzle)**; puzzle will be in the form of an 2D matrix as shown in the above example. Name your file **MinPuzzle.py**

b. What is the time complexity of your implementation?

BFS approach (commented out): O(mn)

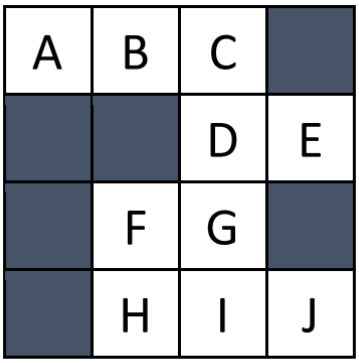
DFS approach: O (4^{m*n})

3. **Analyze Dijkstra with negative edges:** Analyze with a sample graph and show why Dijkstra does not work with negative edges. Give the sample graph and write your explanation why Dijkstra would not work in this case.



Let's examine the shortest path from A to C. The Dijkstra algorithm chooses the unvisited node with the smallest distance, leading the algorithm to think that the best way to proceed is to go from A to C because that distance is shorter than the distance from A to B. The algorithm will go from A to C and stop. Dijkstra will not explore the path from A to B, and thus it will not discover that the path from B to C is –3 and that the A-B-C route has a net distance of 1, which is smaller than the A-C route chosen.

4. **What would be BFS and DFS traversal in below puzzle.** Start at node A.



BFS: A, B, C, D, E, G, F, I, H, J

DFS: A, B, C, D, G, F, H, I, J, E