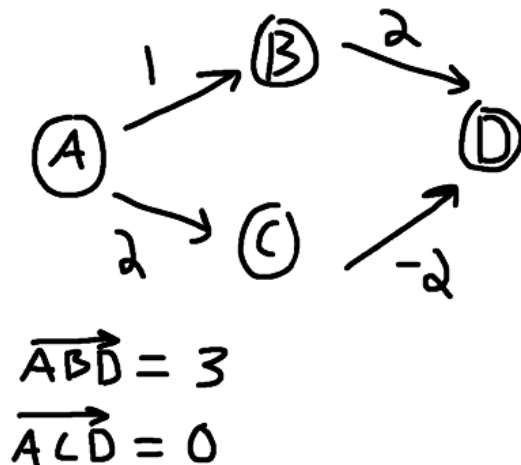


1. BFS and DFS for a graph starting from node A
 - a. BFS: { A, B, D, F, G, C, E }
 - b. DFS: { A, B, F, C, D, E, G }
2.
 - a. MinPuzzle.py
 - b. Let M be the rows of the matrix and N be the columns of the matrix. At the worst case, we'll have to traverse all cells of a matrix which gives us a time complexity of $O(M*N)$. We also have to consider the minheap which would in the worst case be containing $M*N$ cells, so updating the minheap would be an $O(\log(M*N))$ operation. Thus, the time complexity is $O((M*N) * \log(M*N))$.
- 3.

Dijkstra's algorithm will not work on a graph with negative weights due to how the algorithm operates. It assumes the graph edges are weighted and are non-negative, because the algorithm operates greedily, choosing the local optimal edge in order to obtain the globally optimal solution.

If we had negative weights in a graph, they might be hidden by a positive weight that isn't the local optimal edge, so the algorithm will make the wrong choice in terms of obtaining the shortest weighted path, as shown below.



4.
 - a. BFS: { A, B, C, D, E, G, F, I, H, J }
 - b. DFS: { A, B, C, D, G, F, H, I, J, E }