# Link State Algorithm

- Niranjan Gopal
(IMT2022543)

## Working

Using dijkstra's algorithm we calculate shortest path of given weighted edge list and using priority queue In the provided example, the network topology is read from a file called "test_linkstate.txt". The shortest path from node "A" to node "I" is calculated,
and the output on the terminal shows that the shortest path is "A -> B -> D -> F -> G -> I" with a total cost of 9.

The algorithm iterates while the priority queue is not empty. In each iteration, it extracts the node with the minimum distance from the priority queue.

If the current node is the destination node, the algorithm breaks the loop since the shortest path has been found.

The algorithm checks if the current distance is greater than the distance stored in the dist dictionary for the current node.
If it is, the node has already been visited with a shorter distance, so the algorithm continues to the next iteration.

For each neighbor of the current node, the algorithm calculates a new distance by adding the cost of the link from the current node to the neighbor. If this new distance is shorter than the distance stored in the dist dictionary for the neighbor, the dist and prev dictionaries are updated, and the neighbor is added to the priority queue.

After the algorithm finishes, it reconstructs the shortest path from the prev dictionary and prints the path and the total cost if a path was found.

## Shortcomings and Improvement

The algorithm fails to account for the situation where multiple shortest paths are possible to reach from src to dest. In the given example it was noted that even though a path through the node H was equally valid the algorithm did not notice this path. This implementation can only give one of the paths that is guaranteed to be one of the valid shortest paths.

## Output seen

```
Shortest path: A -> B -> D -> F -> G -> I
Total cost: 9
```