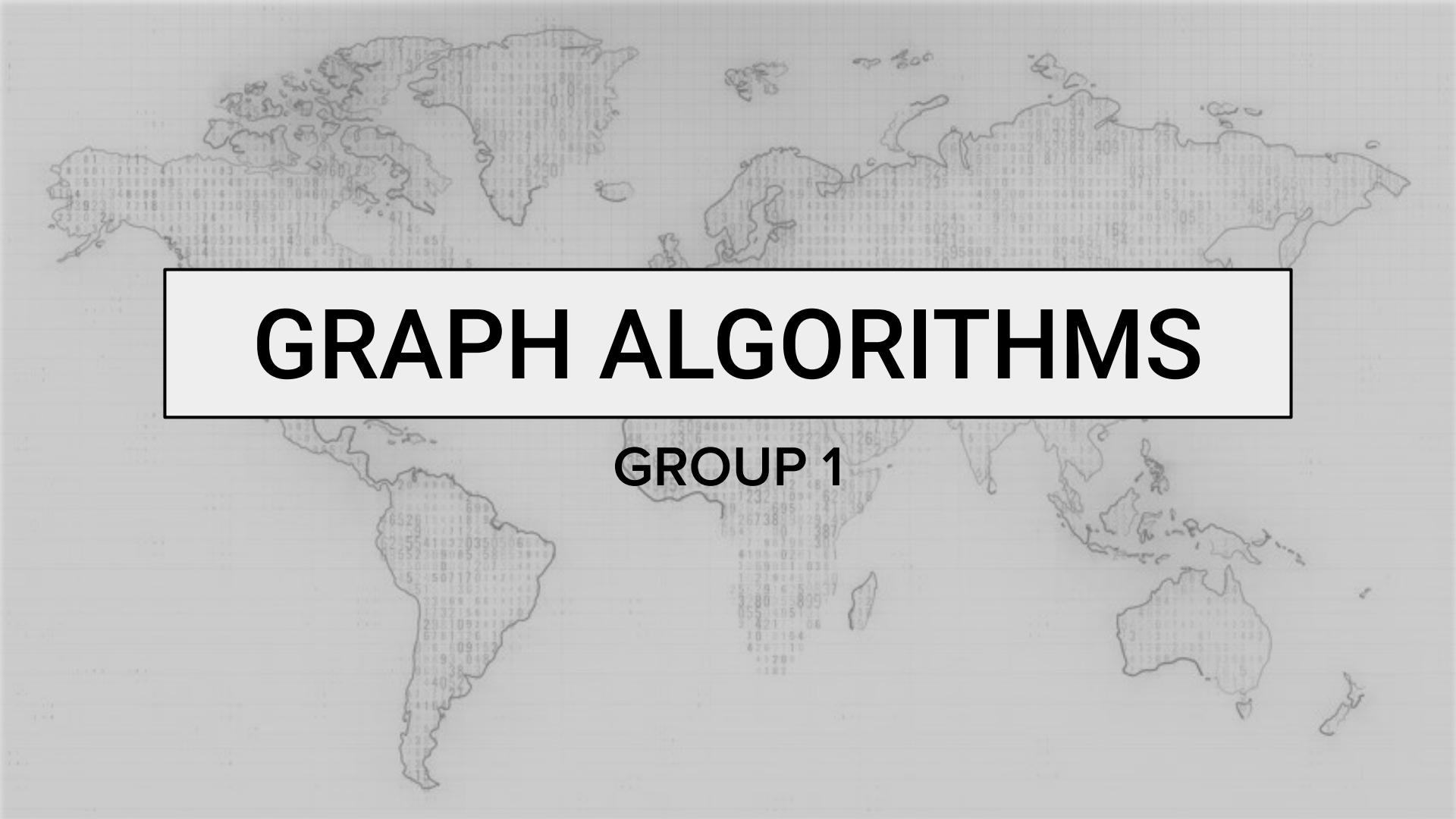


GRAPH ALGORITHMS

GROUP 1

INTRODUCTION

Road networks are:

- Vast and complex
- Often represented graphically
- Nodes being intersections/endpoints
- Edges being the roads that connect them together

Graphs are then:

- Parsed into applications such as google maps
- Traversing algorithms would then be used
- To find the best and most effective path to a particular destination

PROJECT BRIEF

We are given:

- Undirected unweighted graph
- Containing n number of nodes and m number of edges
- With h number of target nodes (hospitals)
- Where we are interested in finding k number of paths to

Where our aim is to:

- Explore different graph traversing algorithms
- Modify them so as to be able to meet our needs
- Perform theoretical and empirical study
- Focusing on the effects of values h and k on our algorithms

PLAN OF ACTION - BREADTH FIRST SEARCH

Single Source Single Path BFS	Single Source Multi Path BFS
Multi Source Single Path BFS	Multi Source Multi Path BFS

OUR CODE'S SIGNIFICANCE

- Road network graphs given are actually not fully connected

California road network

Dataset information

A road network of California. Intersections and endpoints are represented by nodes and the roads connecting these intersections or road endpoints are represented by undirected edges.

Dataset statistics	
Nodes	1965206
Edges	2766607
Nodes in largest WCC	1957027 (0.996)
Edges in largest WCC	2760388 (0.998)
Nodes in largest SCC	1957027 (0.996)
Edges in largest SCC	2760388 (0.998)
Average clustering coefficient	0.0464
Number of triangles	120676
Fraction of closed triangles	0.02097
Diameter (longest shortest path)	849

OUR CODE'S SIGNIFICANCE

- Our team strived to fix this issue as we would like to ensure that all towns can access hospitals
- Thus for an unconnected graph, we will add edges
- To ensure that all nodes can get have a path and access to the target nodes
- It would then simulate us adding roads
- So as to ensure that all towns can access the hospitals

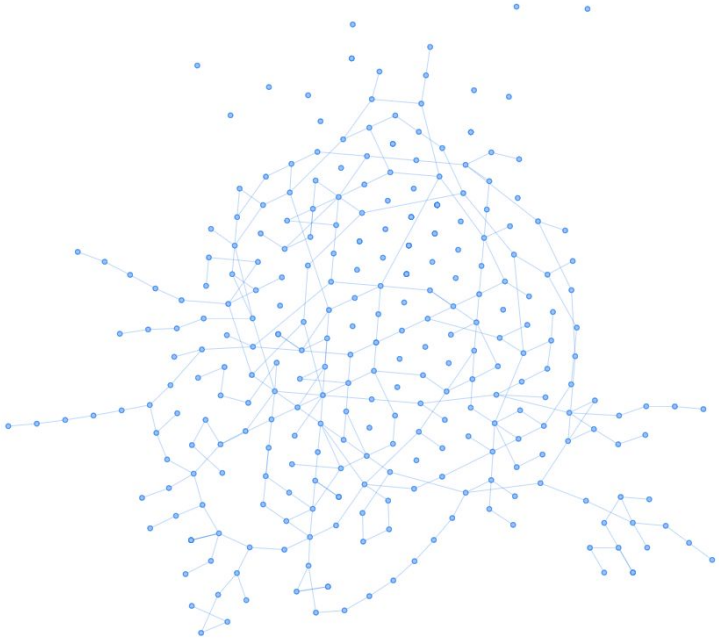
OUR CODE'S SIGNIFICANCE

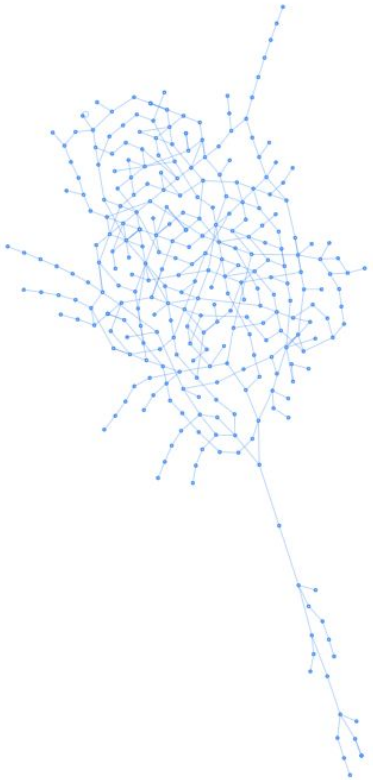
- If after multi sourced multi path bfs is done, we check for unconnected nodes.
- In the connected cluster, we look for nodes with only one edge to label as a suitable node.
- Then for all unconnected nodes, we add an edge connecting to one of the suitable nodes.

ONLY IMPLEMENTED FOR MULTI SOURCED MULTI PATH BFS

ABLE TO DISABLE THE FEATURE, THROUGH COMMENTING OUT SEVERAL LINES

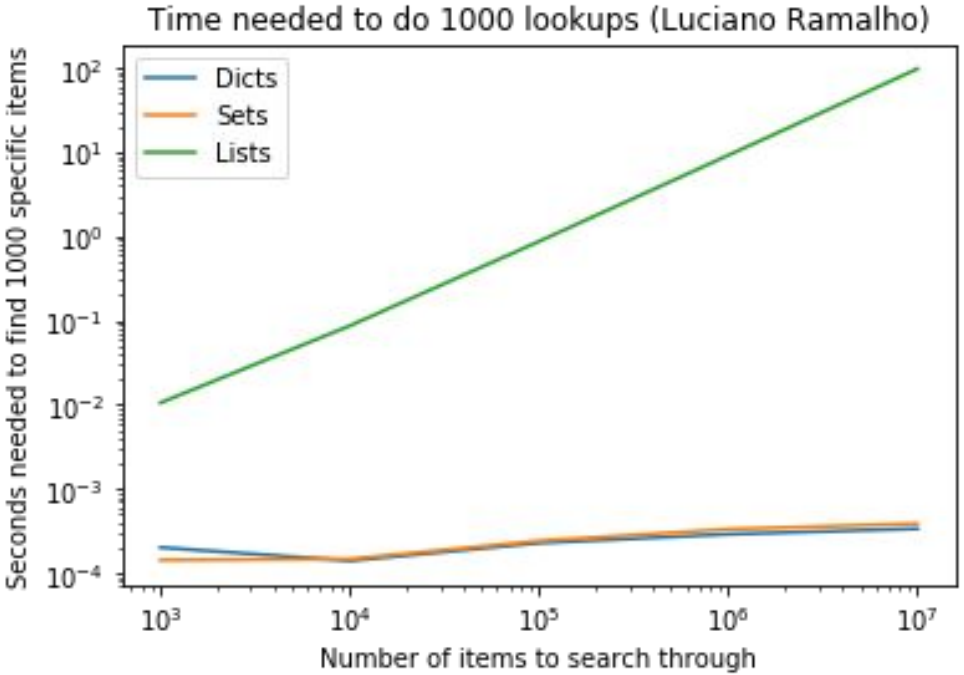
CODE DEMO





THE USE OF PYTHON DICTIONARIES AND SETS

https://www.jessicayung.com/python-lists-vs-dictionaries-the-space-time-tradeoff/



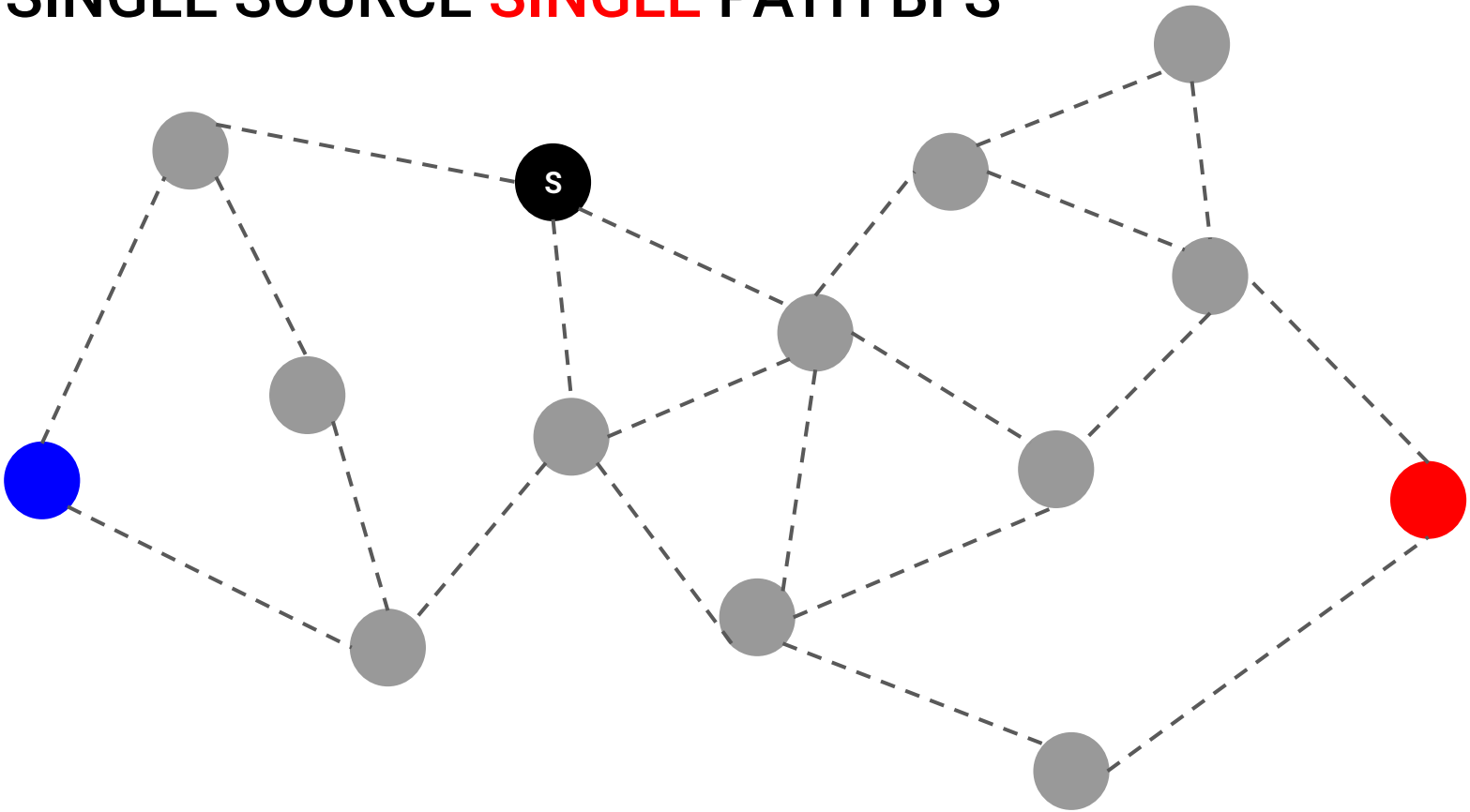

SINGLE SOURCE SINGLE PATH

SINGLE SOURCE SINGLE PATH BFS

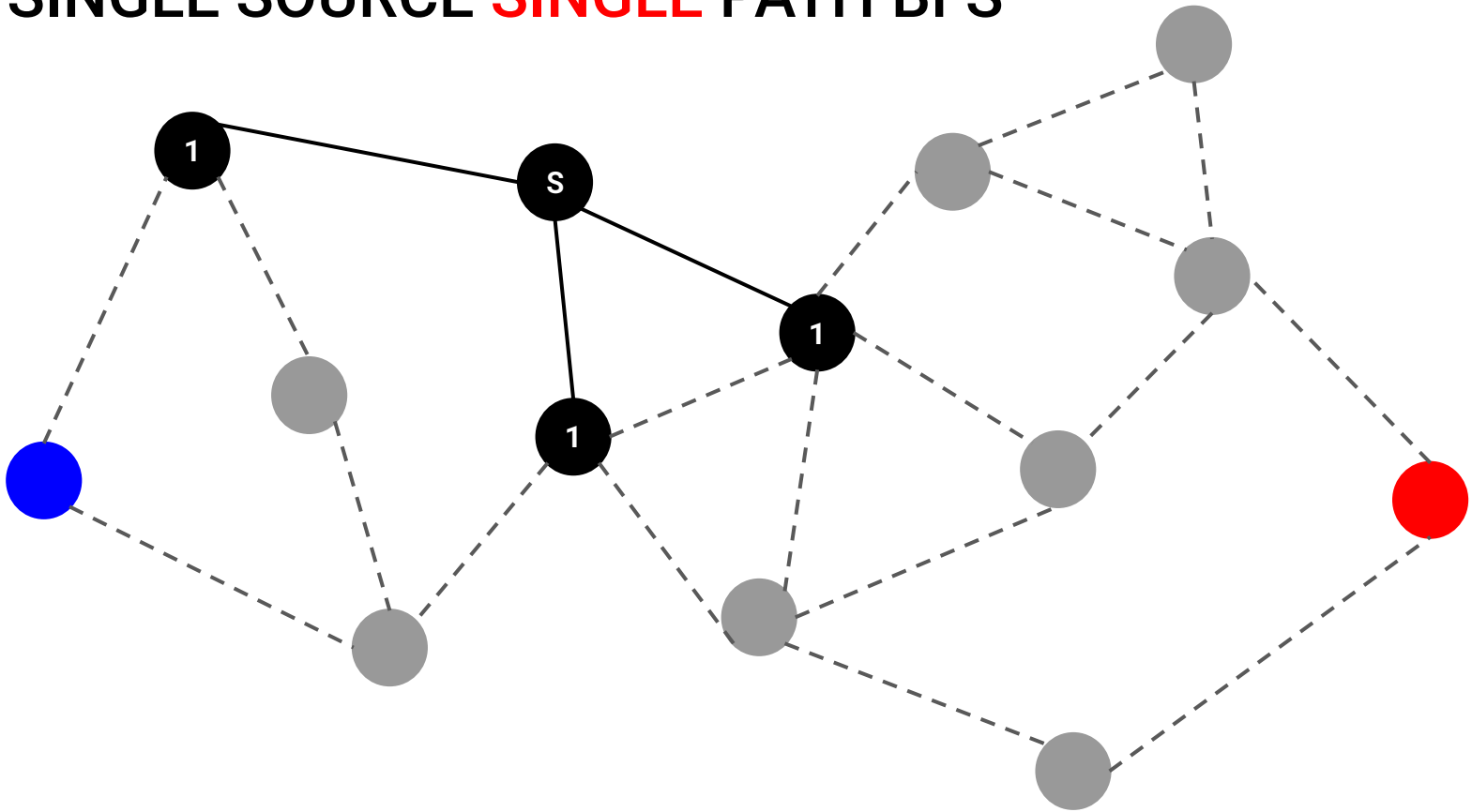
Implemented with 2 lists in python:

1) List with all the visited nodes
2) List with all the nodes to visit next

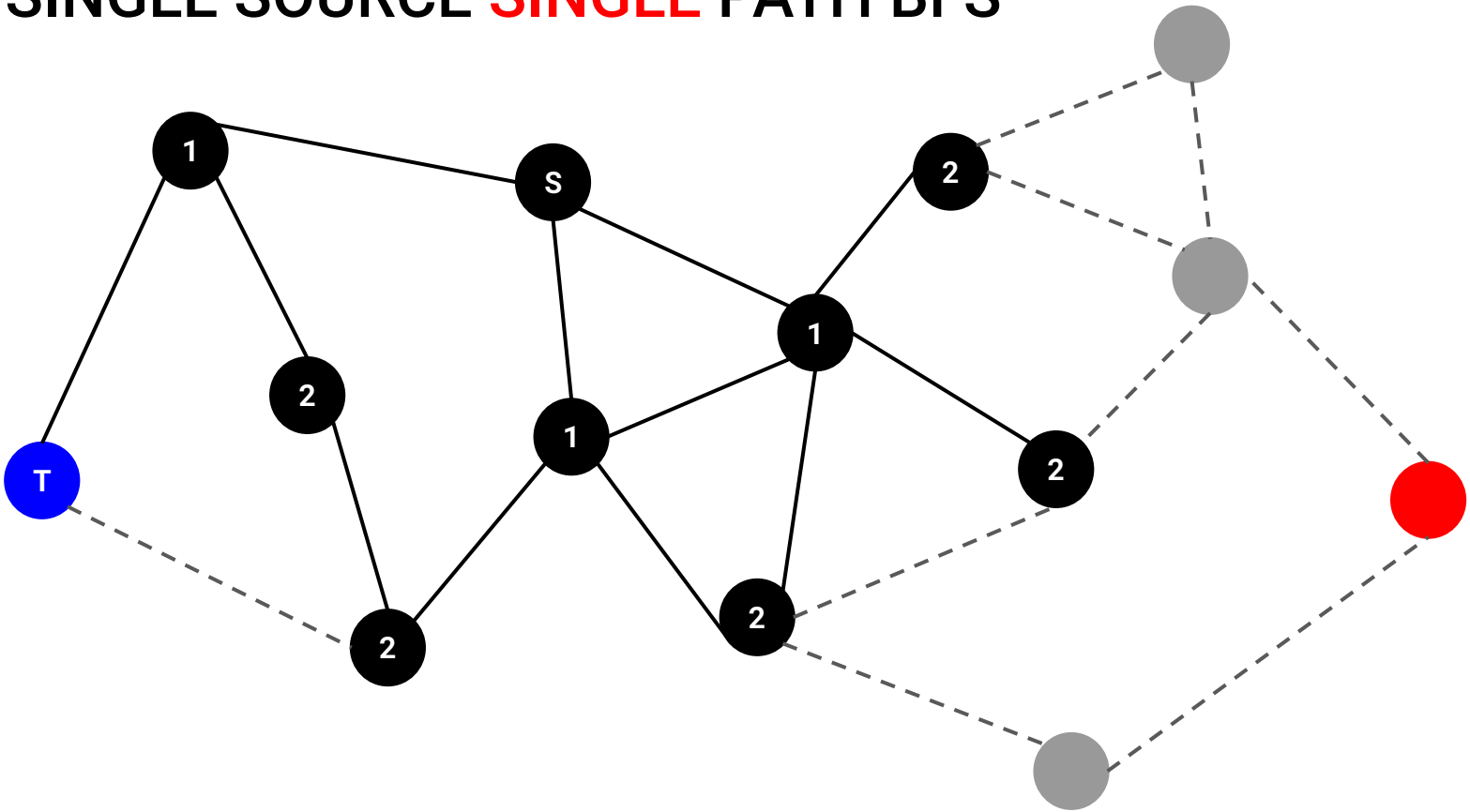
SINGLE SOURCE SINGLE PATH BFS



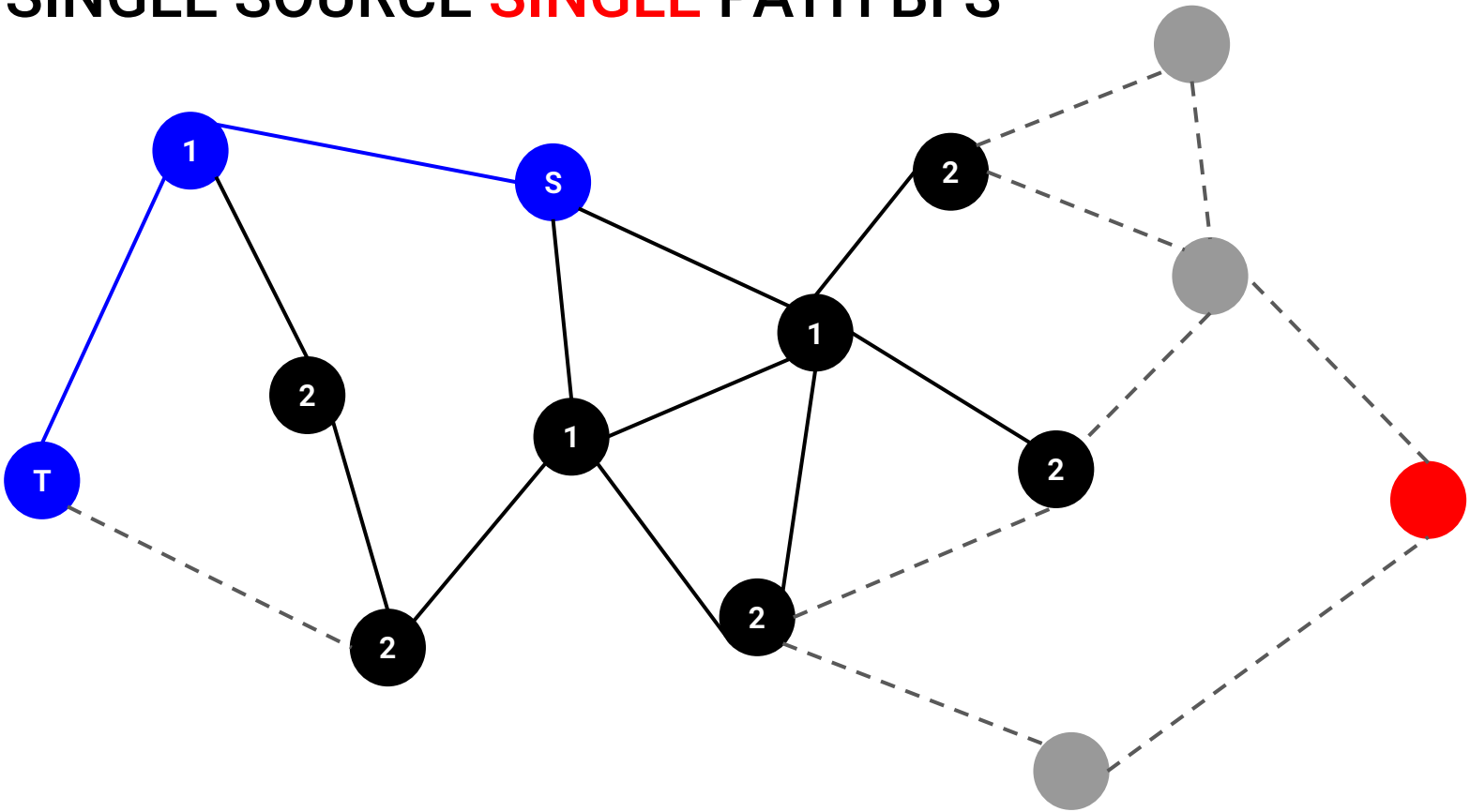
SINGLE SOURCE SINGLE PATH BFS



SINGLE SOURCE SINGLE PATH BFS



SINGLE SOURCE SINGLE PATH BFS



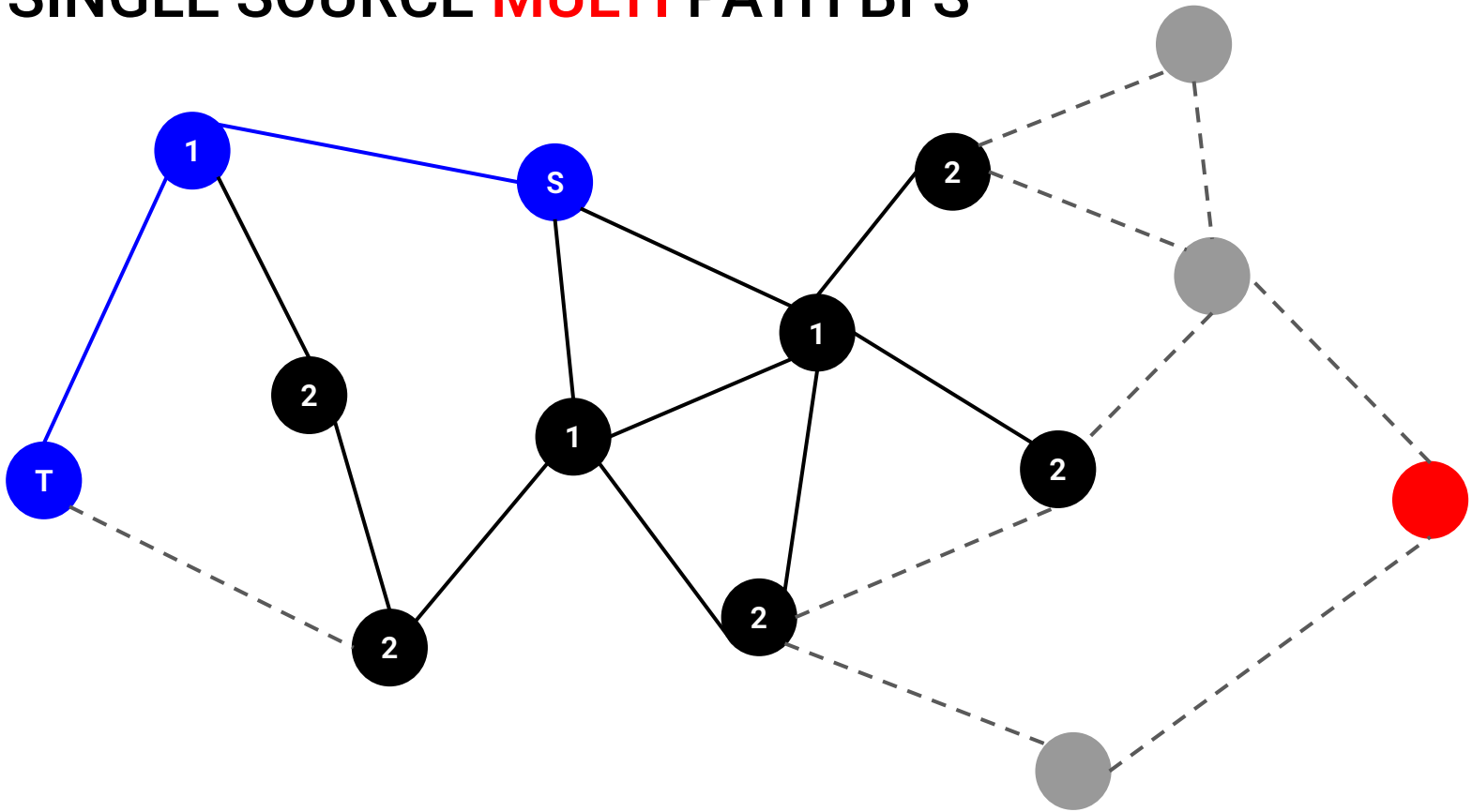

SINGLE SOURCE MULTI PATH

SINGLE SOURCE MULTI PATH BFS

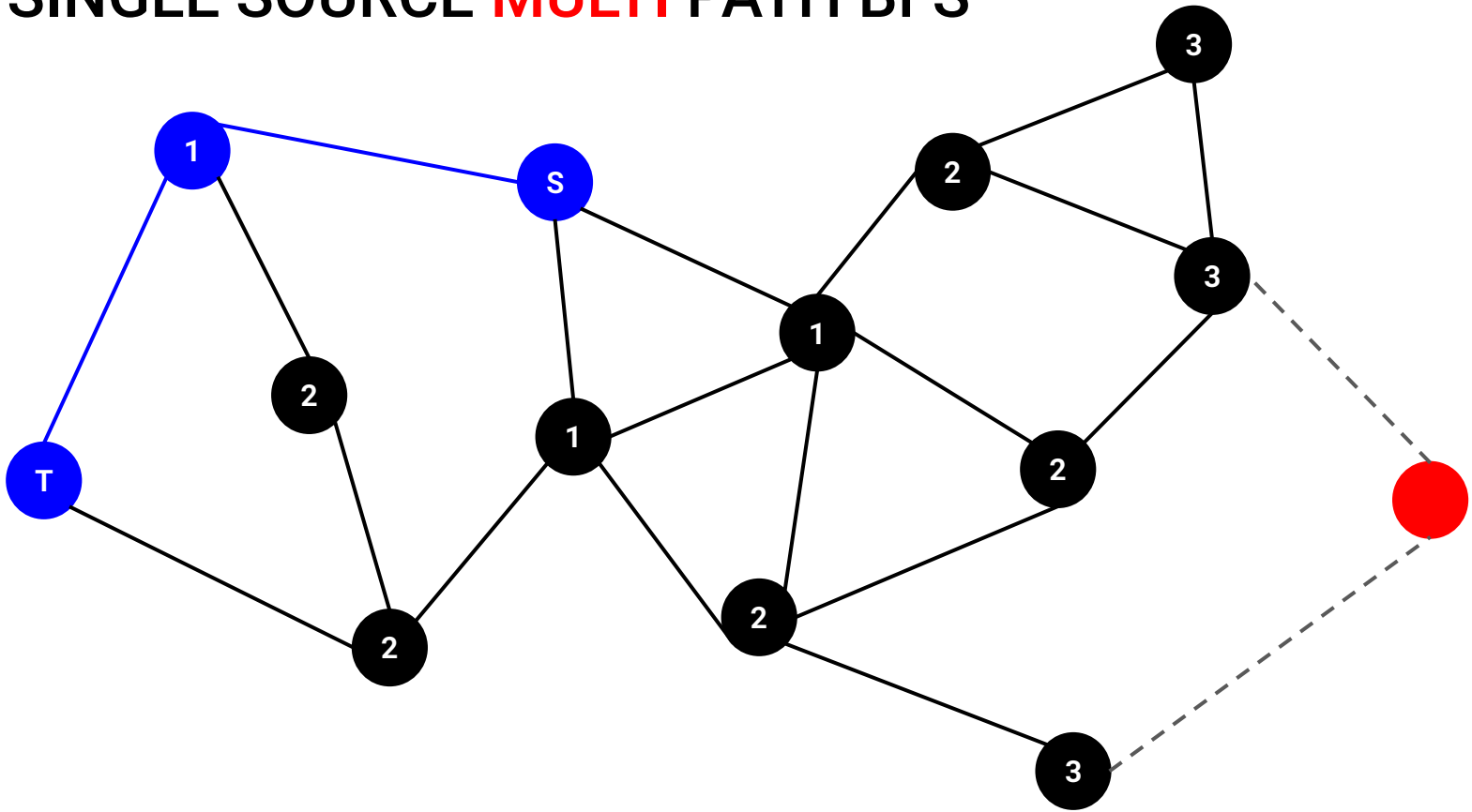
Implemented with 3 lists in python:

1) List with all the visited nodes
2) List with all the nodes to visit next
3) A temp list for every node with all the paths that has been found

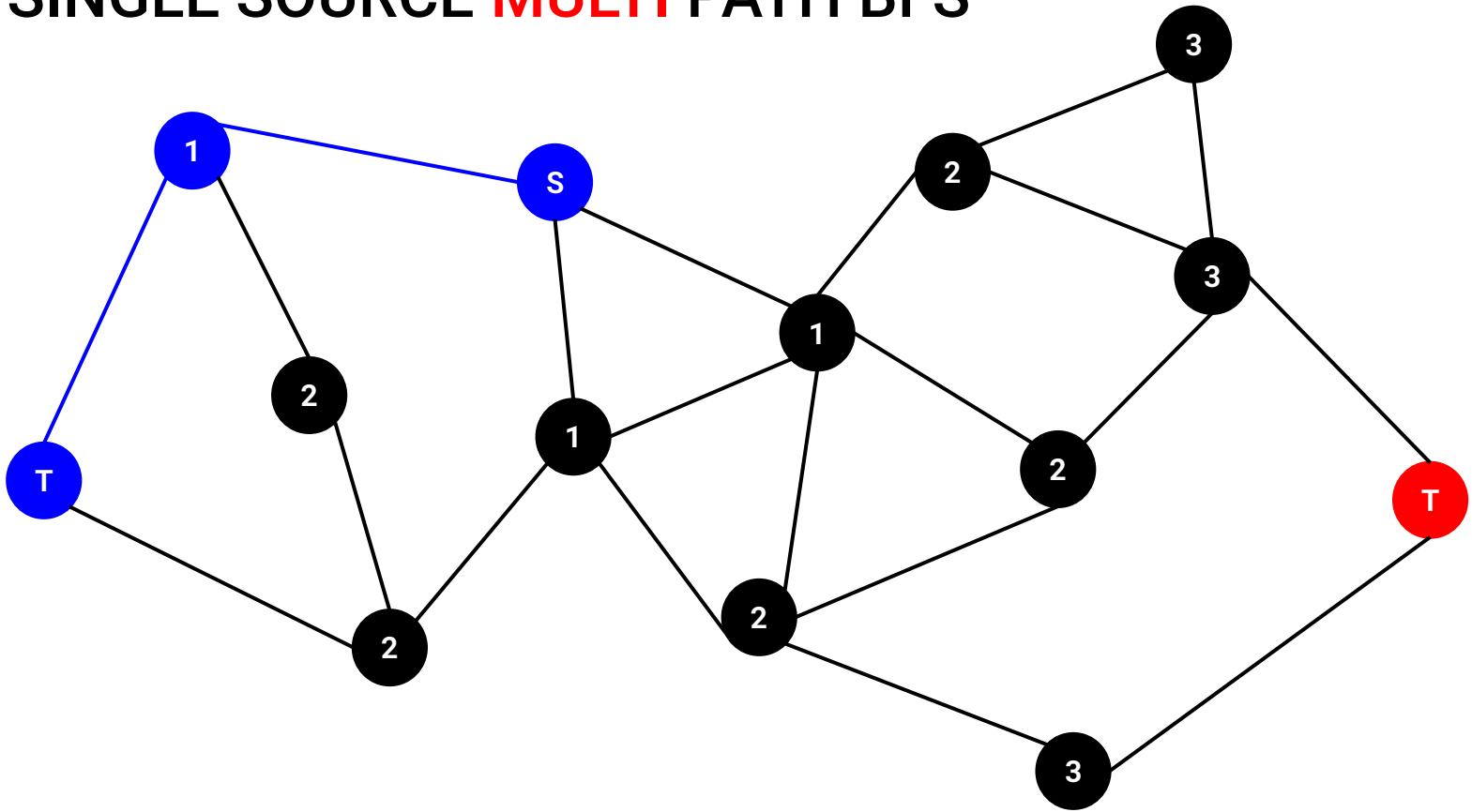
SINGLE SOURCE MULTI PATH BFS



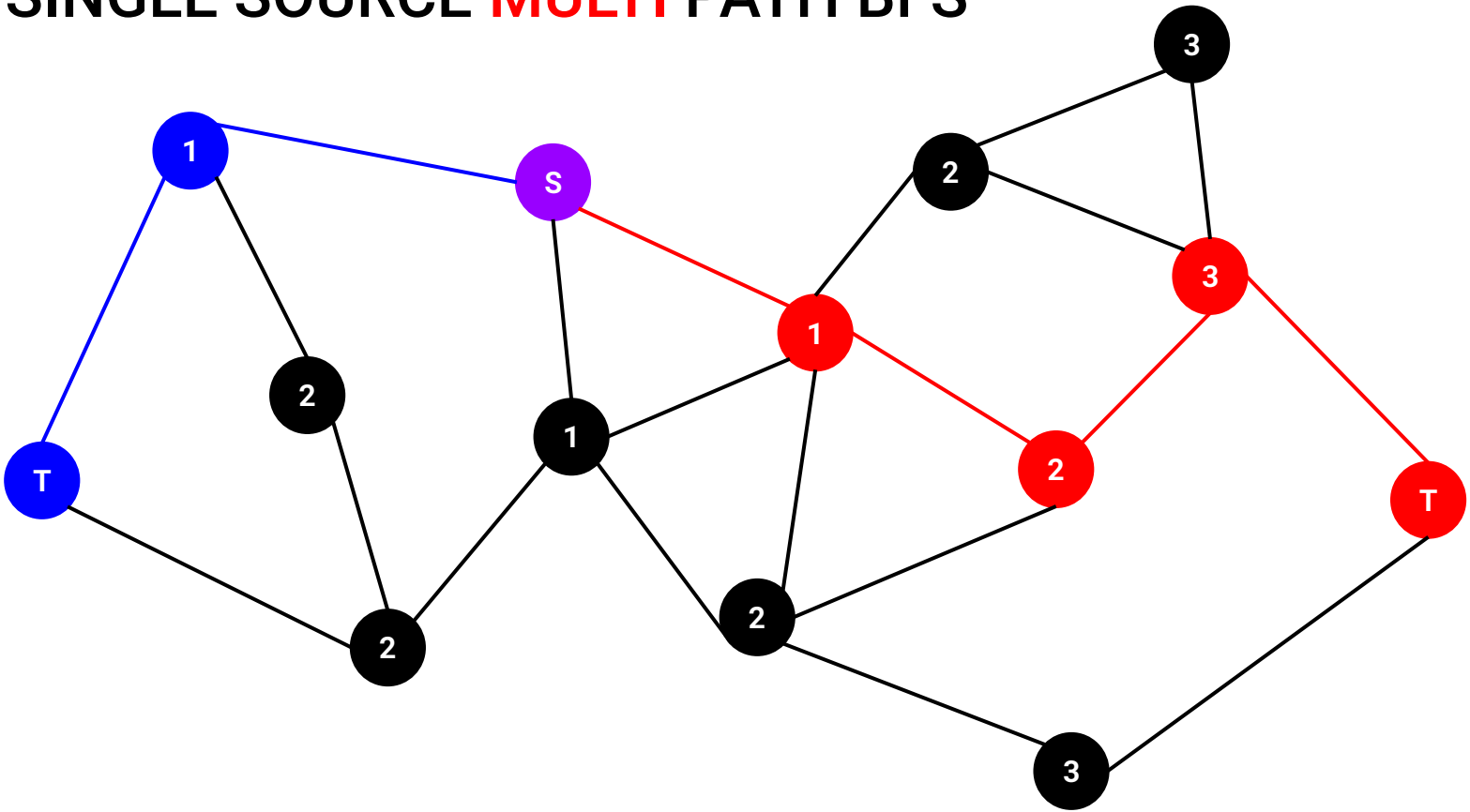
SINGLE SOURCE MULTI PATH BFS



SINGLE SOURCE MULTI PATH BFS



SINGLE SOURCE MULTI PATH BFS



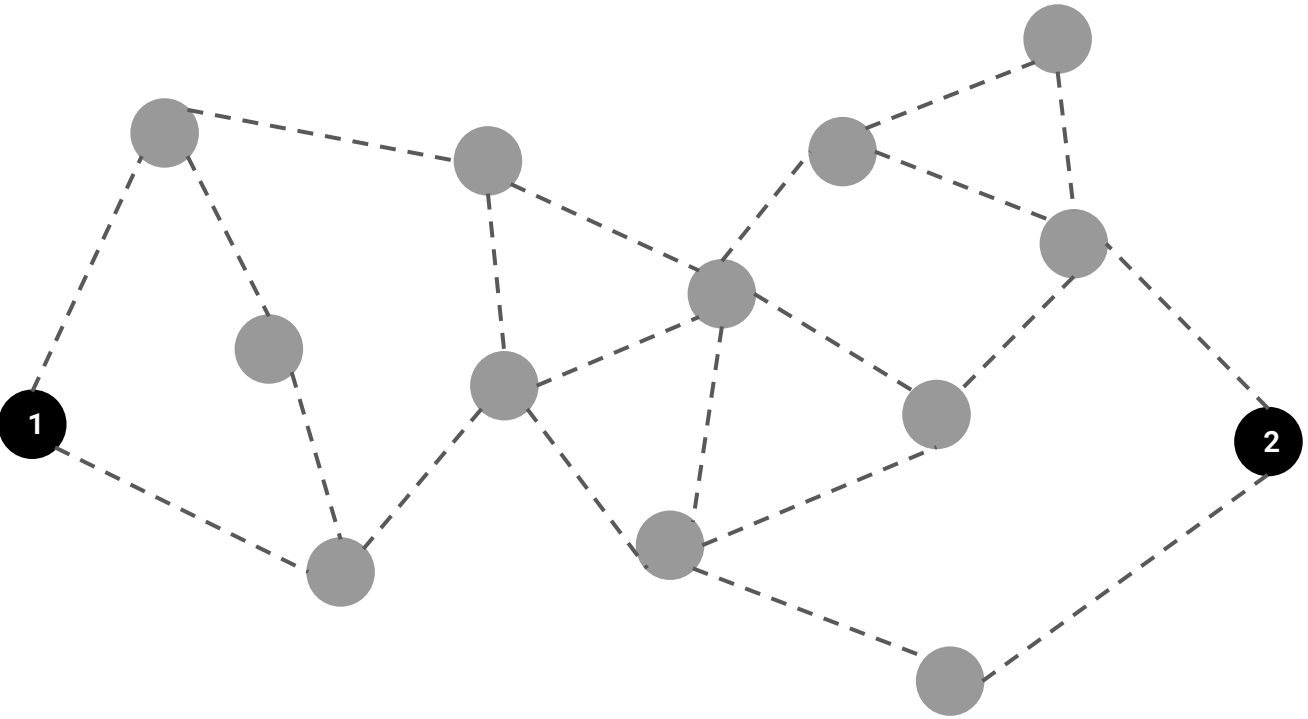

MULTI SOURCE SINGLE PATH

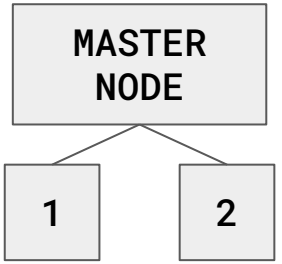
MULTI SOURCE SINGLE PATH BFS

Implemented with:

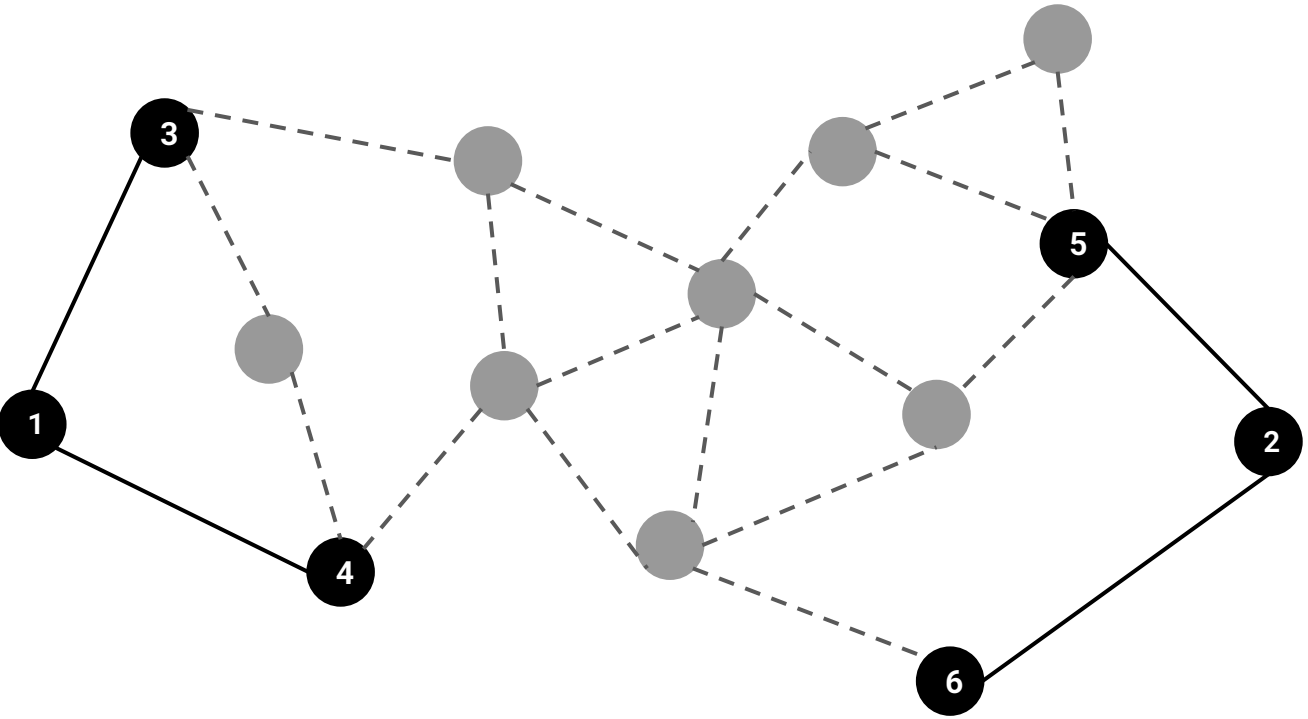
1) A single tree with a master node to keep track of all paths
   - Made possible with the treelib library
2) A dictionary to keep track of nodes that have been visited before

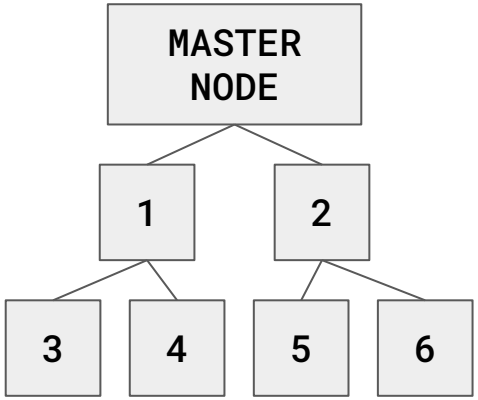
MULTI SOURCE SINGLE PATH BFS



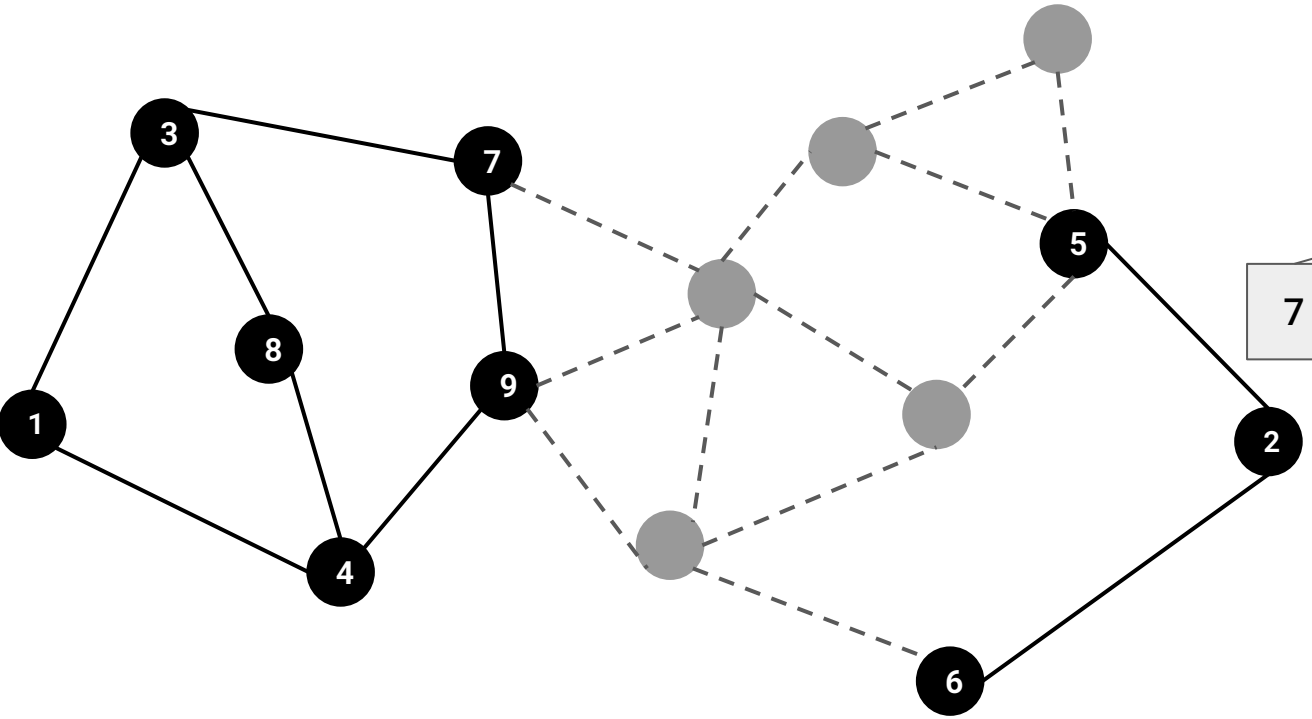


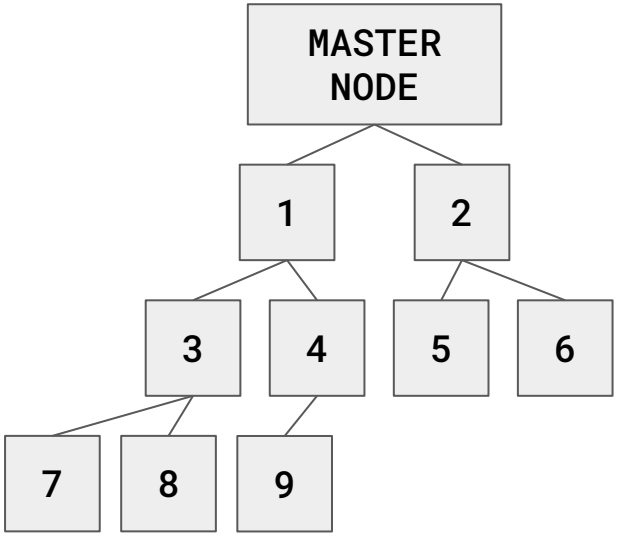
MULTI SOURCE SINGLE PATH BFS



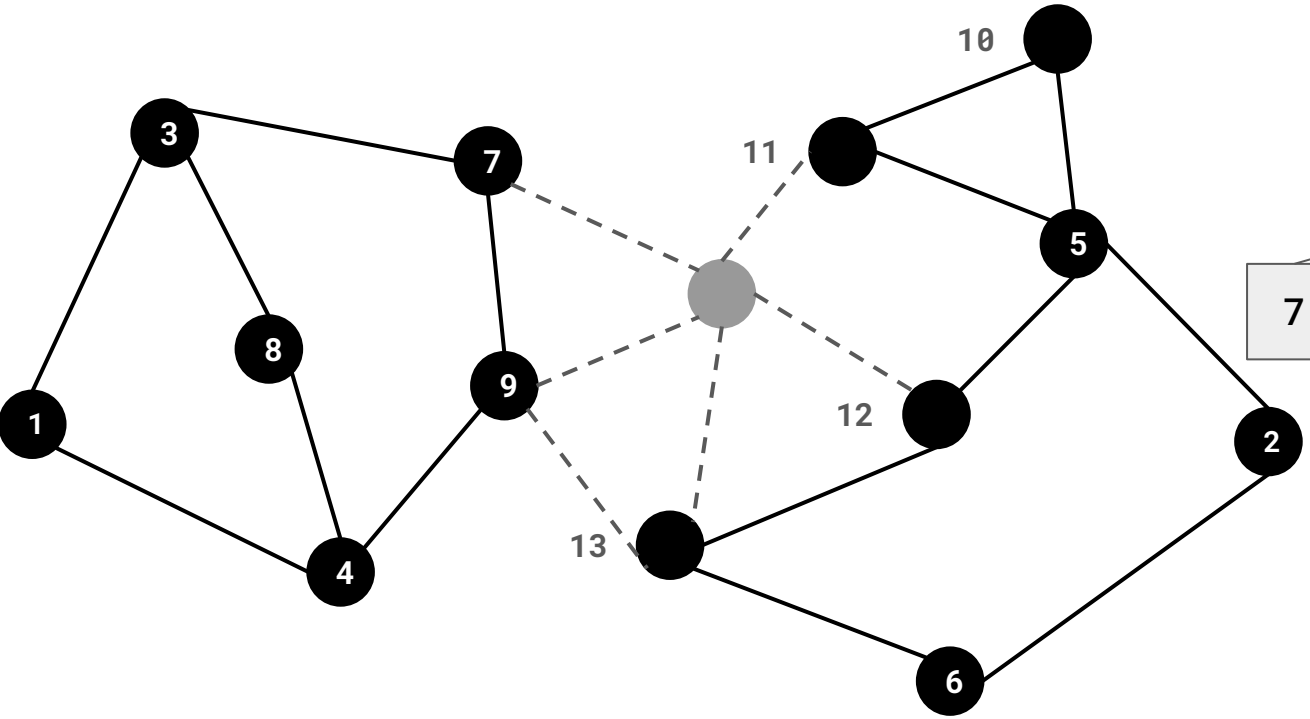


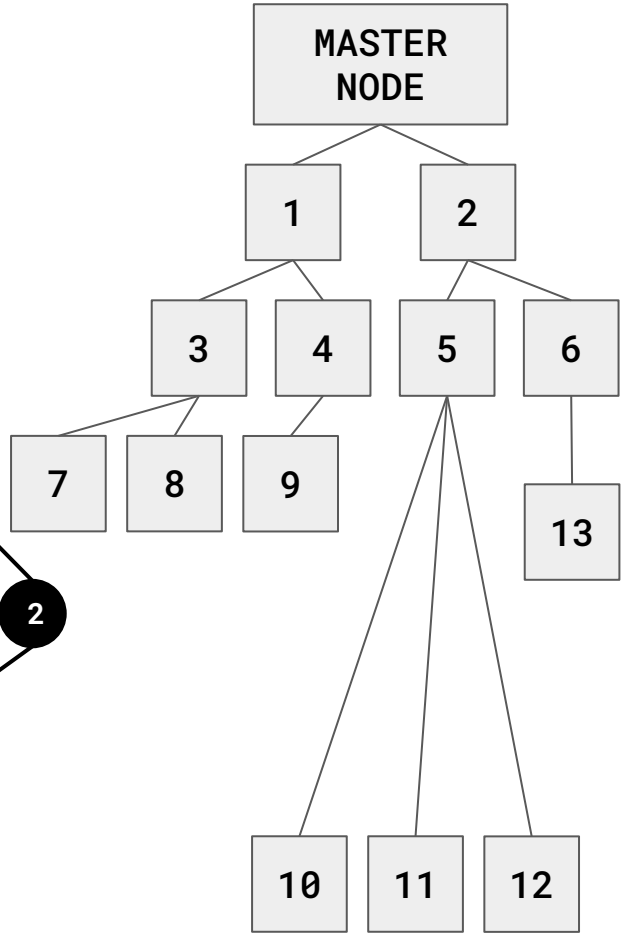
MULTI SOURCE SINGLE PATH BFS



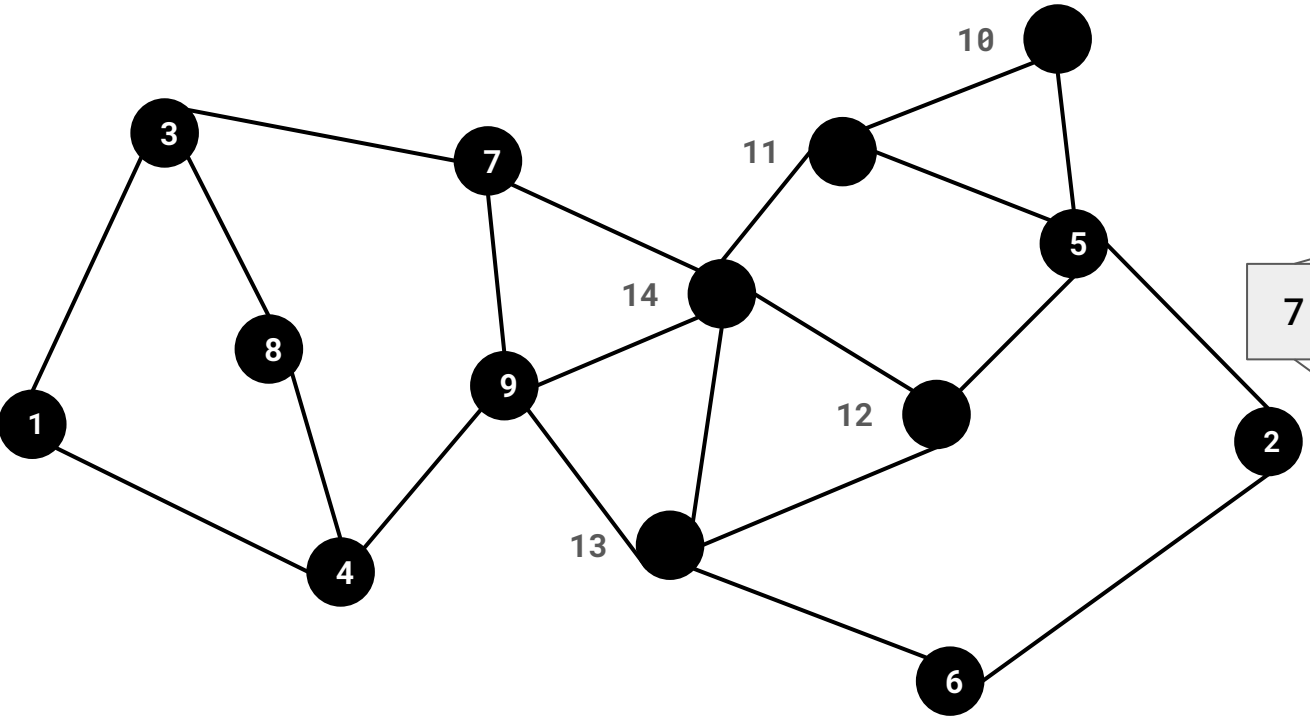


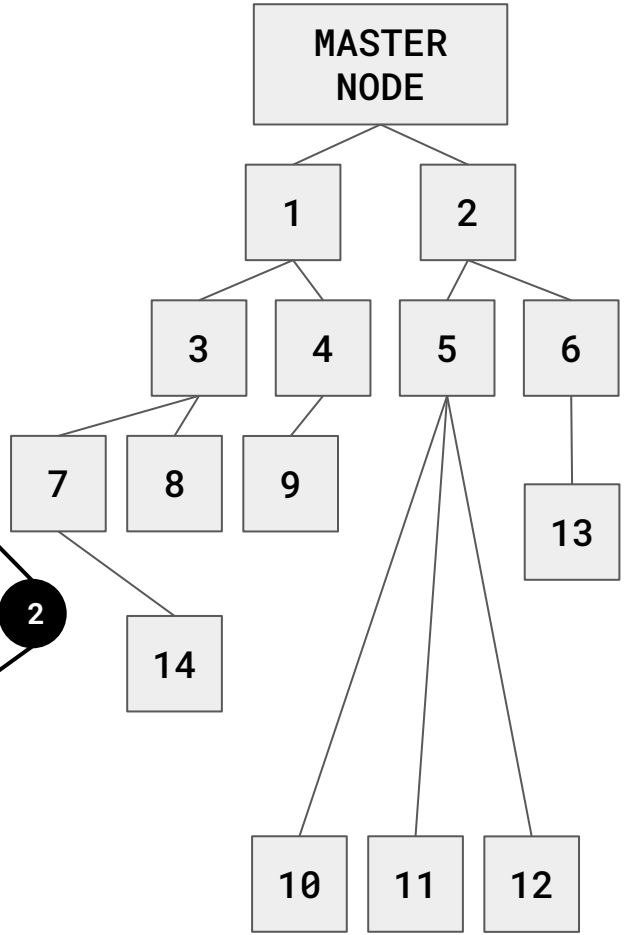
MULTI SOURCE SINGLE PATH BFS





MULTI SOURCE SINGLE PATH BFS





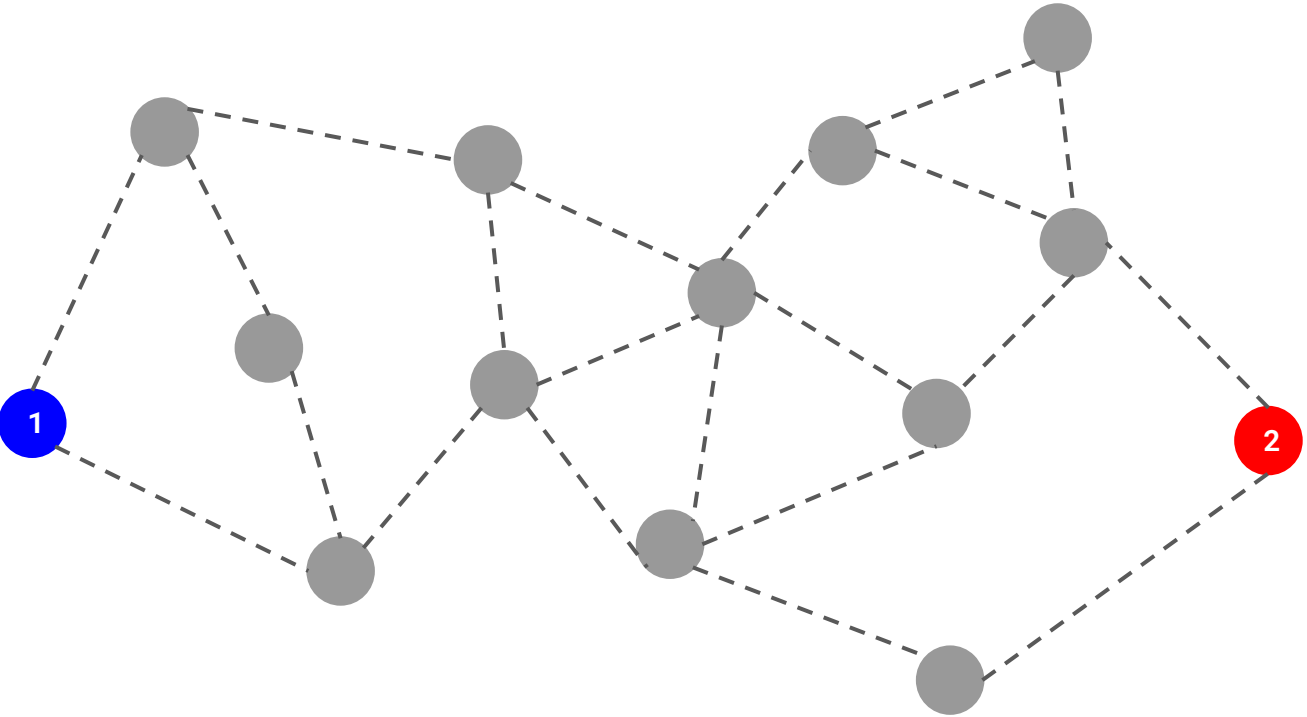

MULTI SOURCE MULTI PATH

MULTI SOURCE MULTI PATH BFS

Implemented with:

1) Dictionary of trees with the target nodes as the master node
    - Made possible with the treelib library
2) A dictionary to keep track of the hospitals that have been visited before
    - Key → Node
    - Value → Set of hospitals that have been visited before

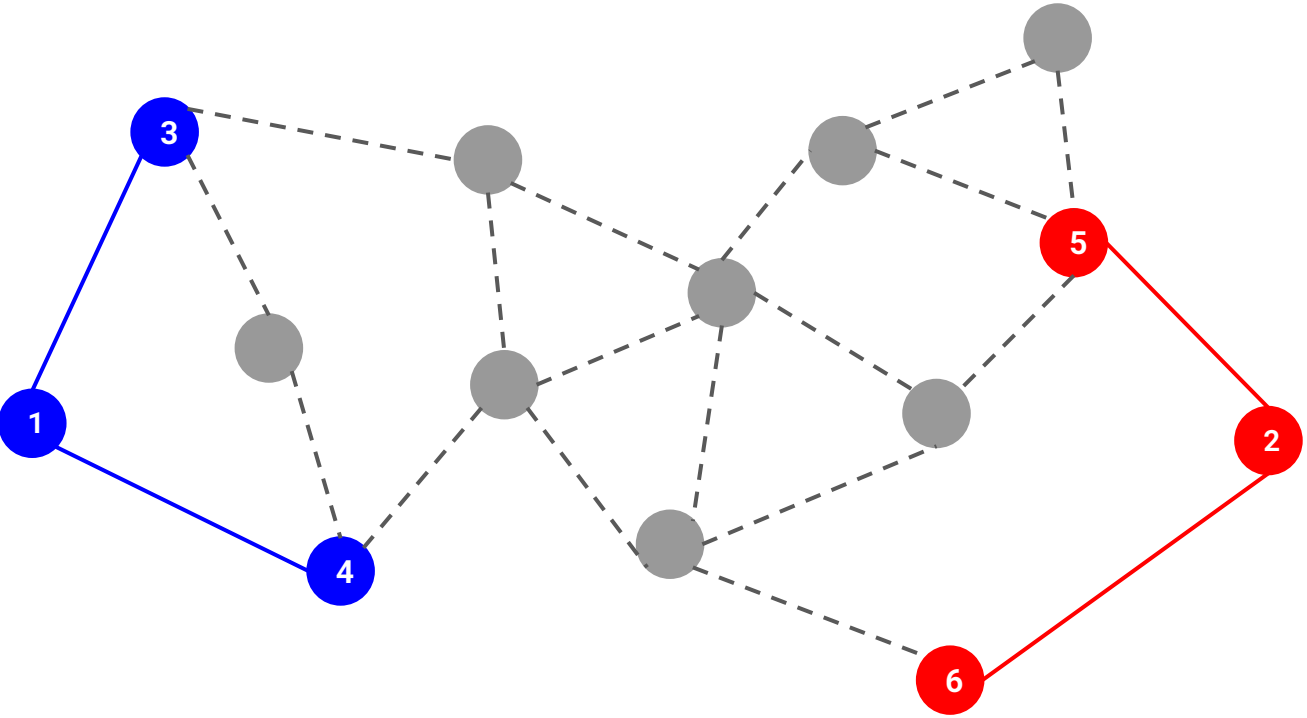
MULTI SOURCE MULTI PATH BFS

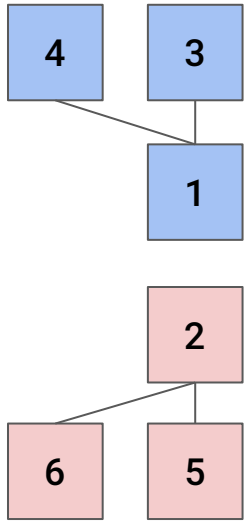


1

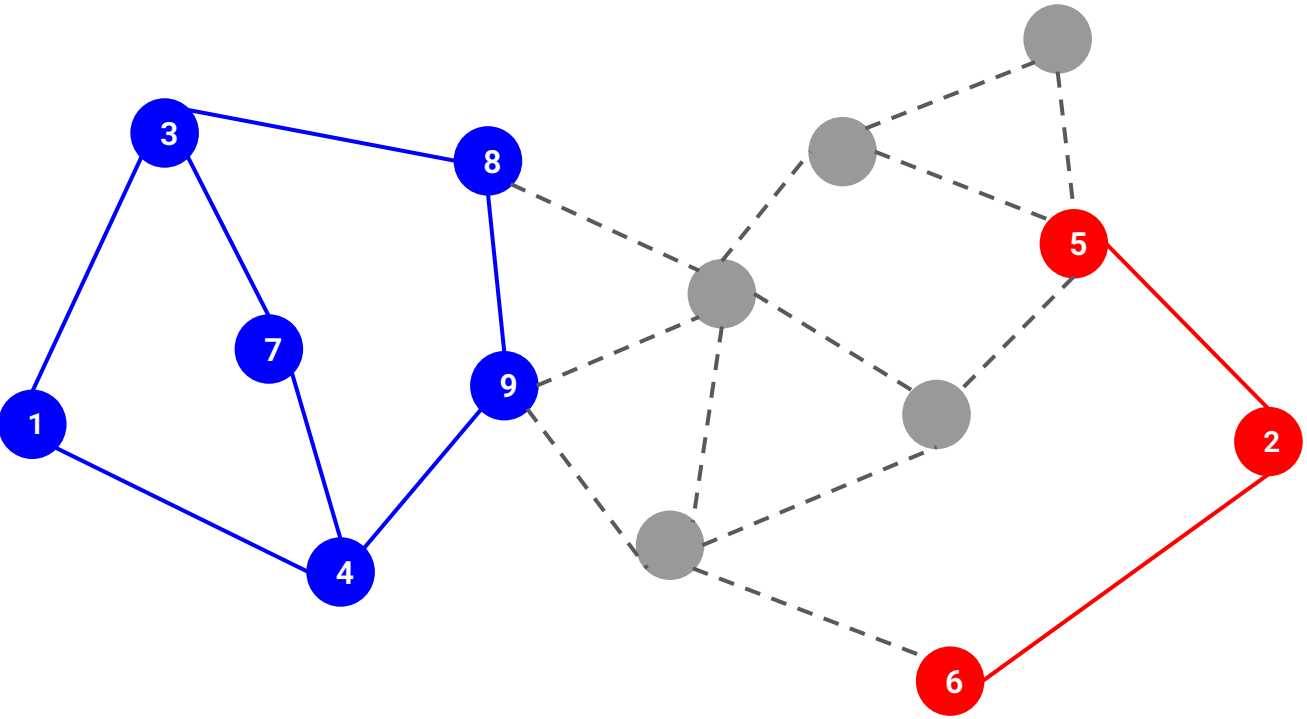
2

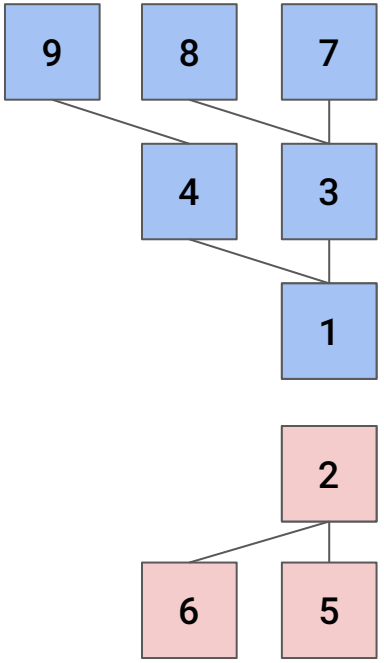
MULTI SOURCE MULTI PATH BFS



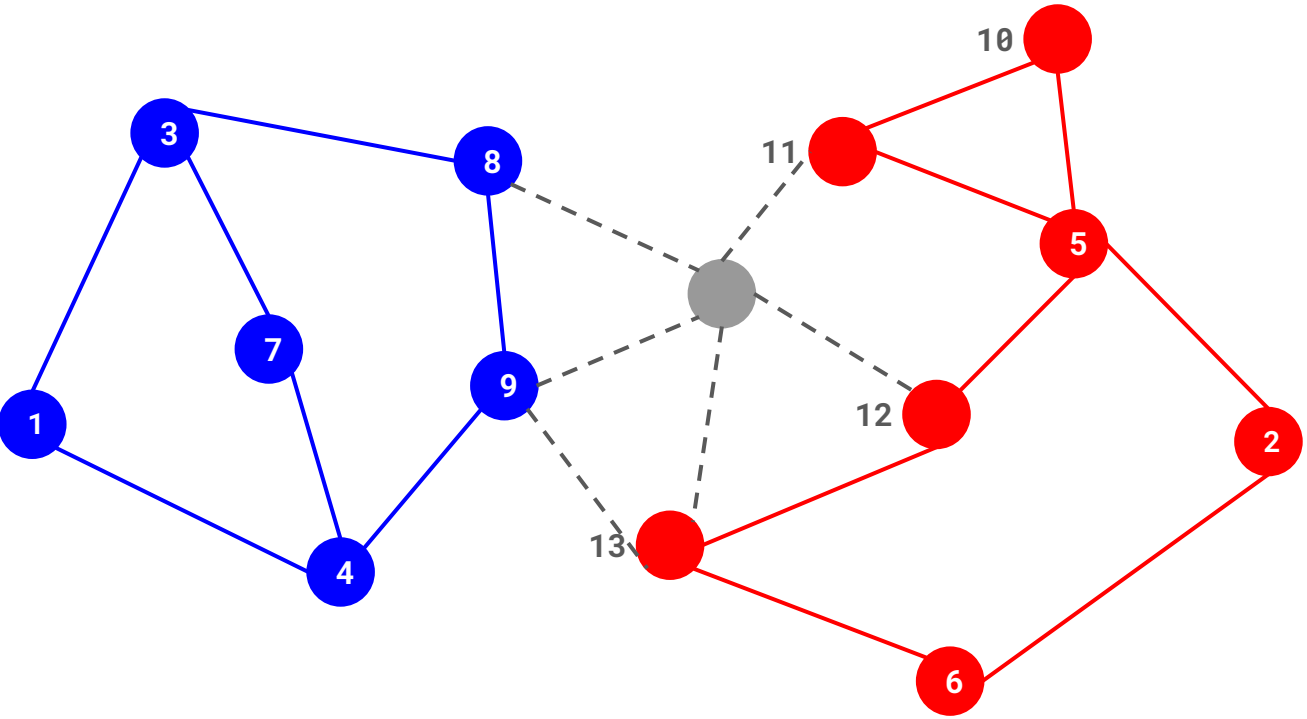


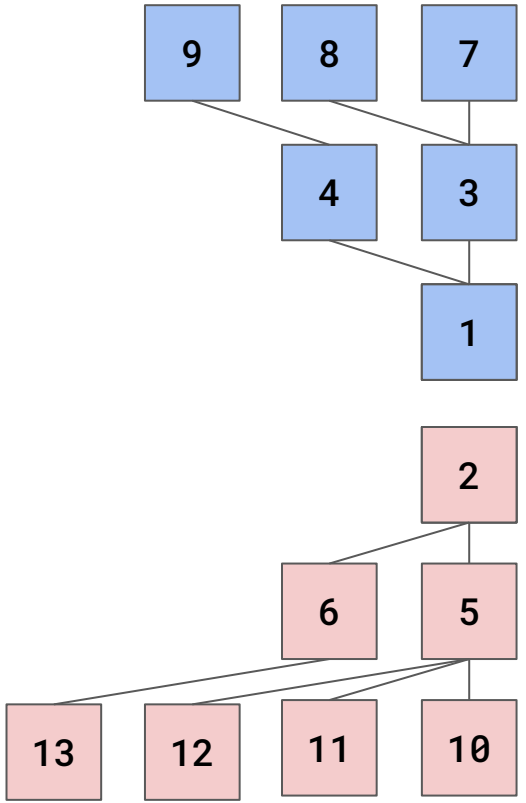
MULTI SOURCE MULTI PATH BFS



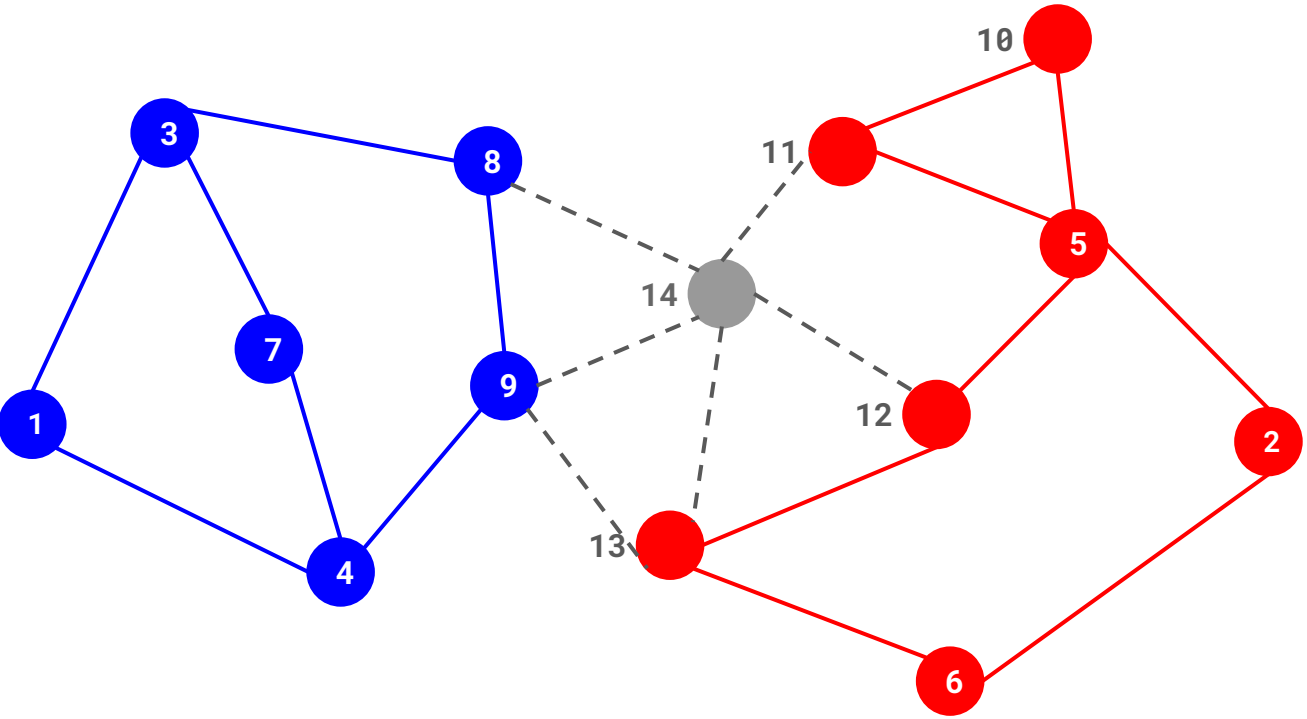


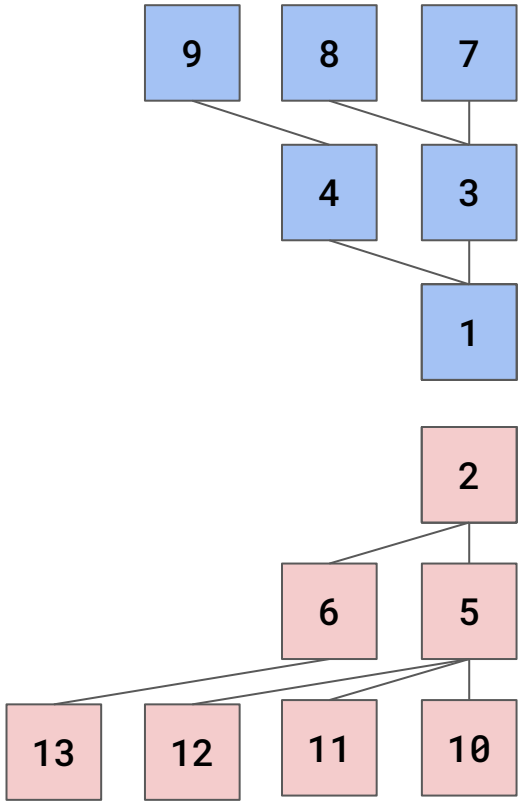
MULTI SOURCE MULTI PATH BFS



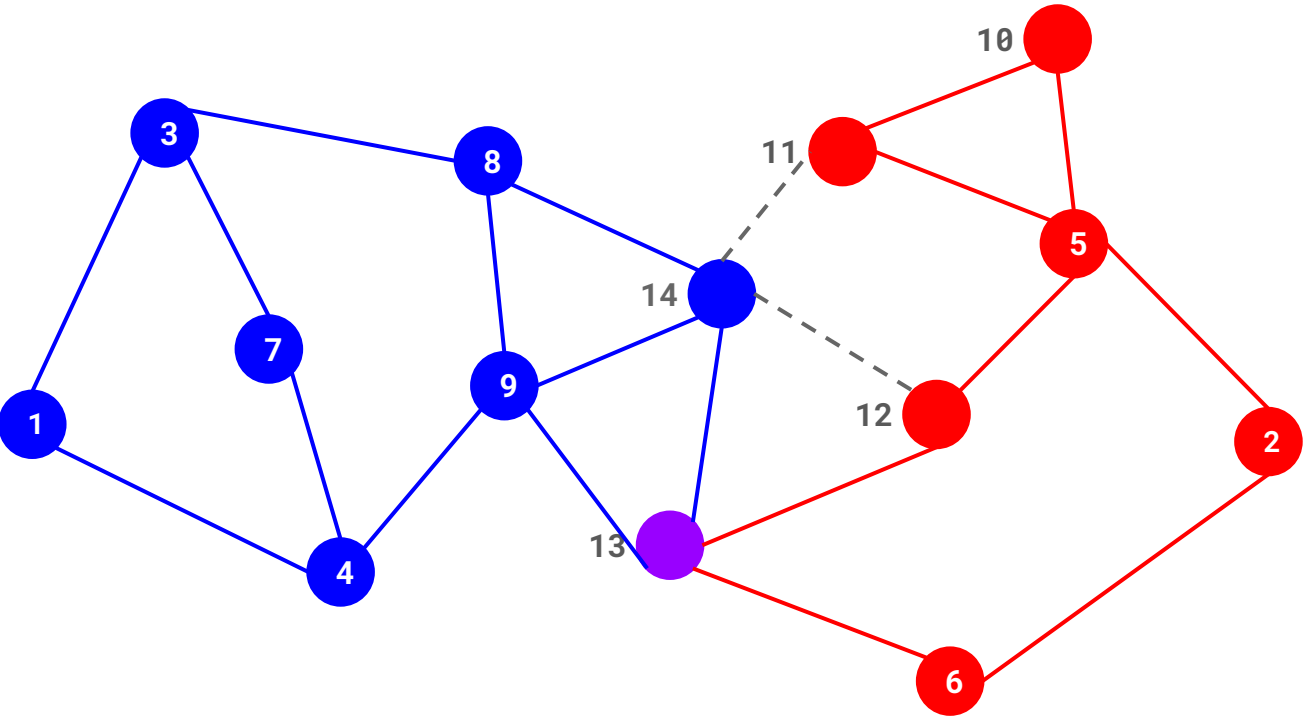


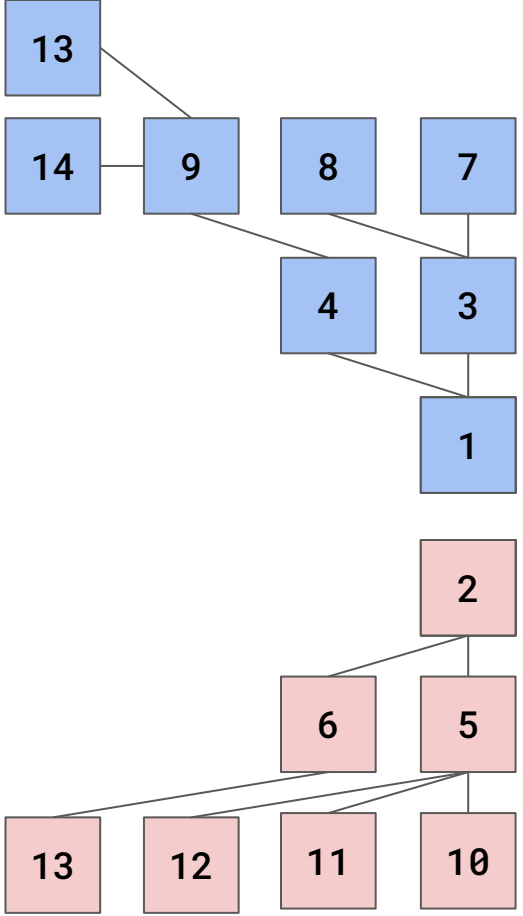
MULTI SOURCE MULTI PATH BFS



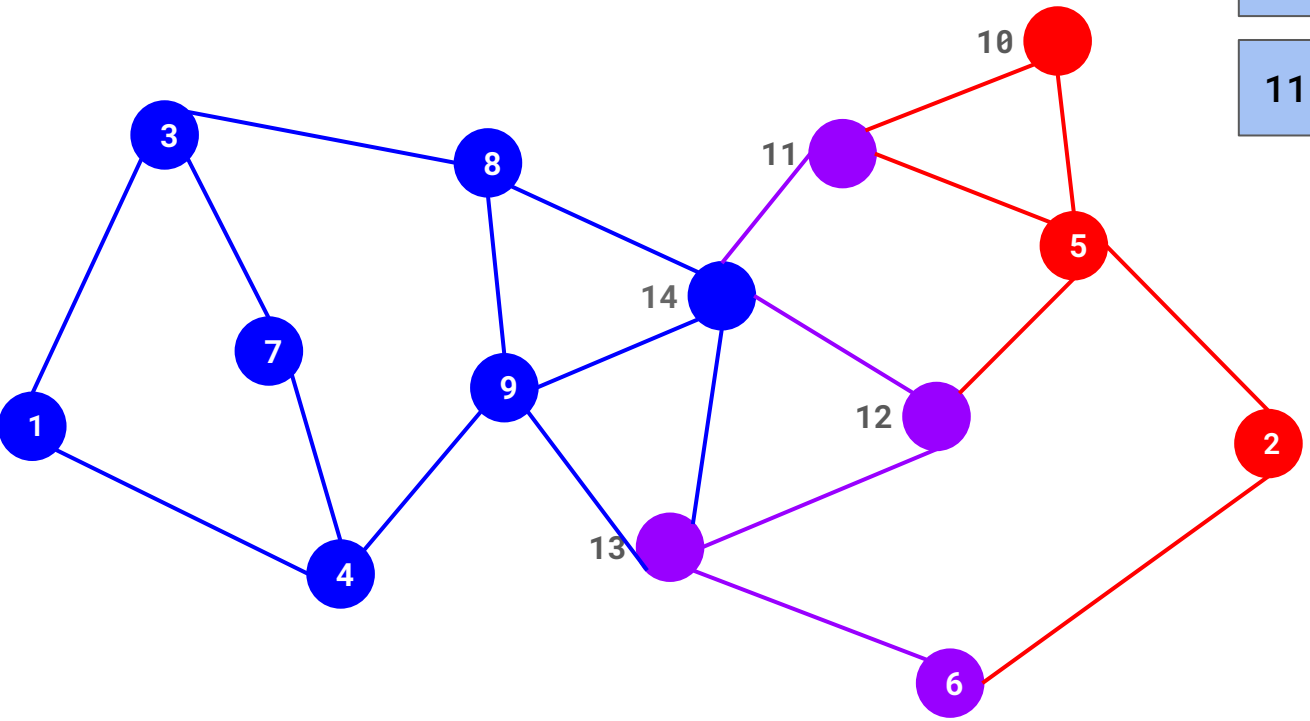


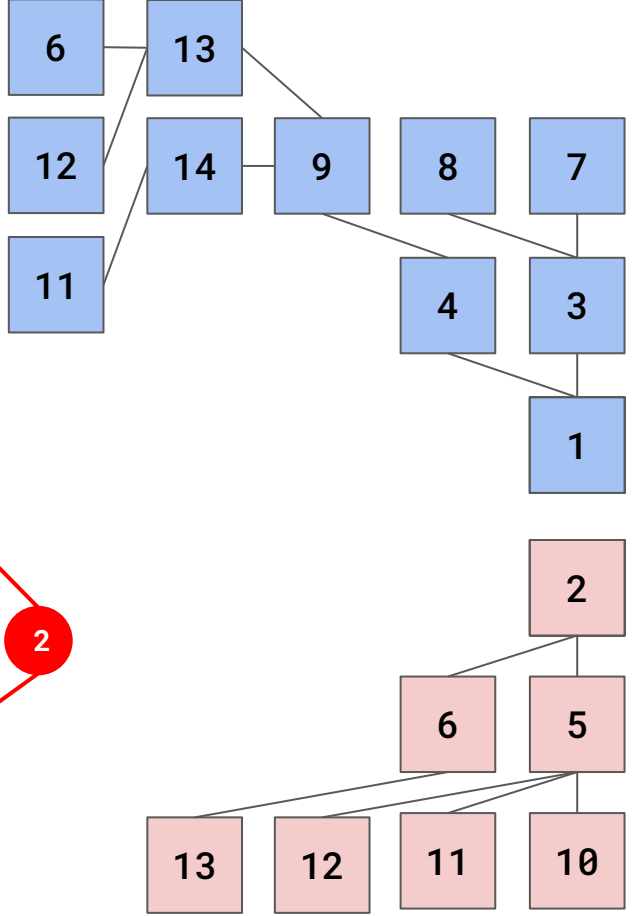
MULTI SOURCE MULTI PATH BFS



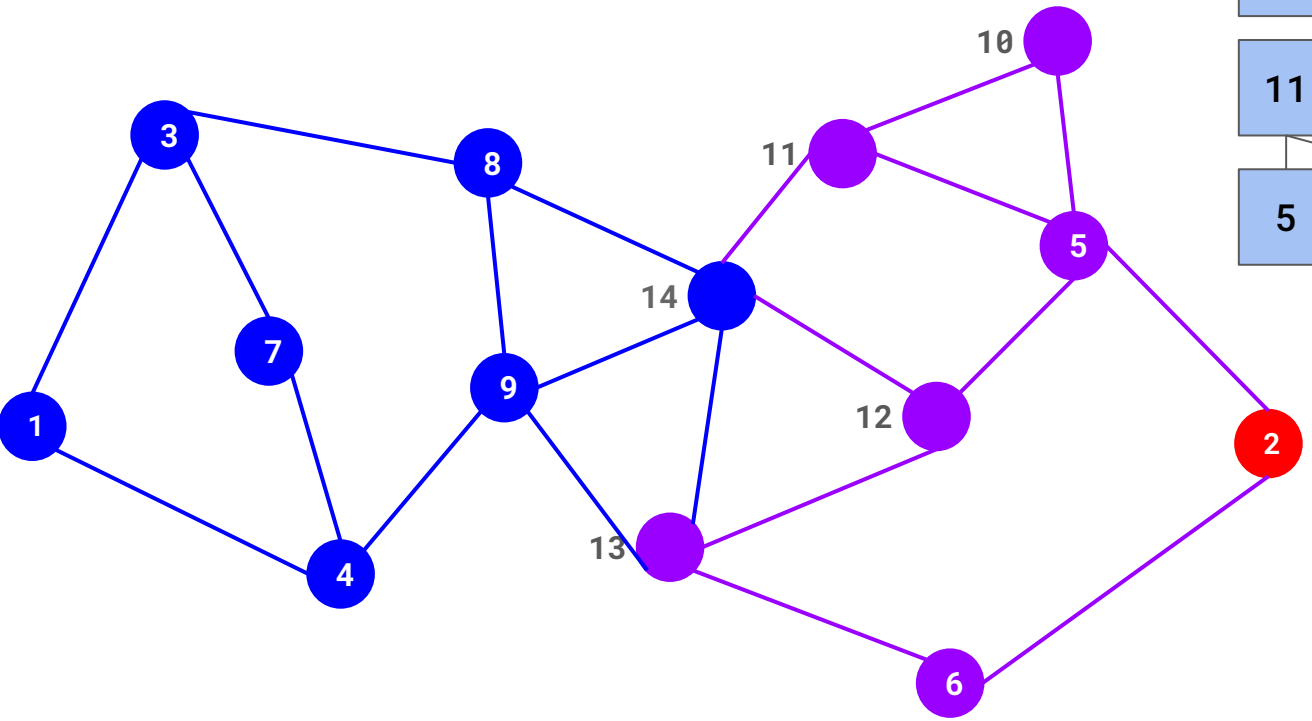


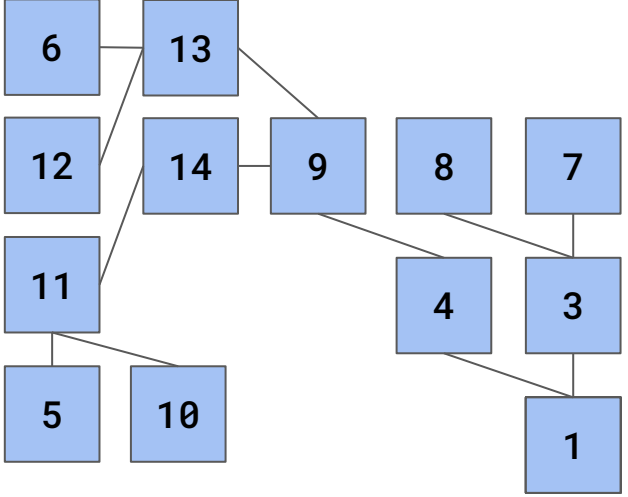
MULTI SOURCE MULTI PATH BFS

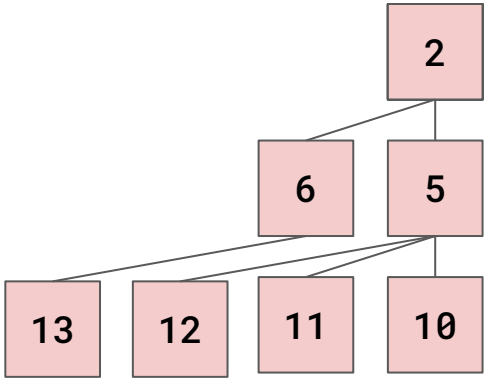




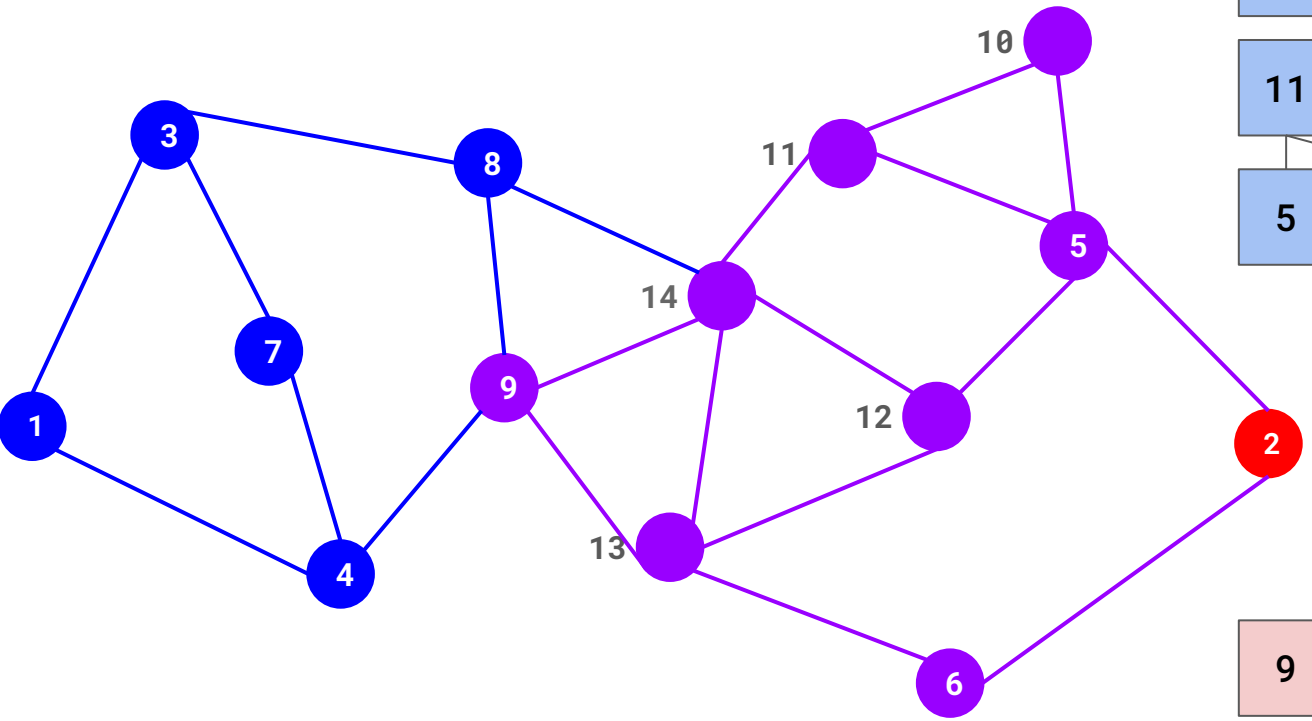
MULTI SOURCE MULTI PATH BFS

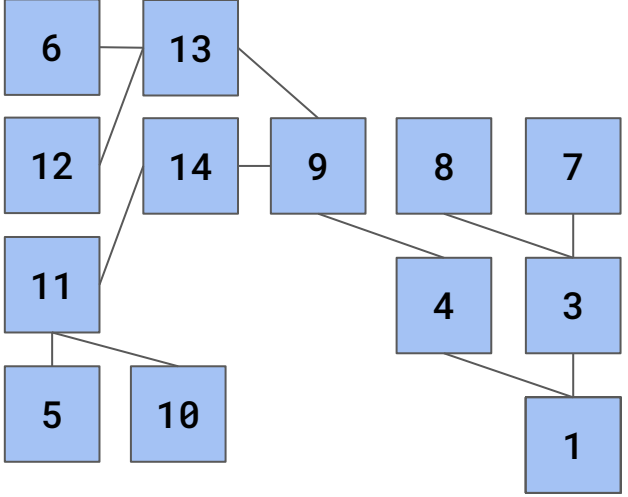


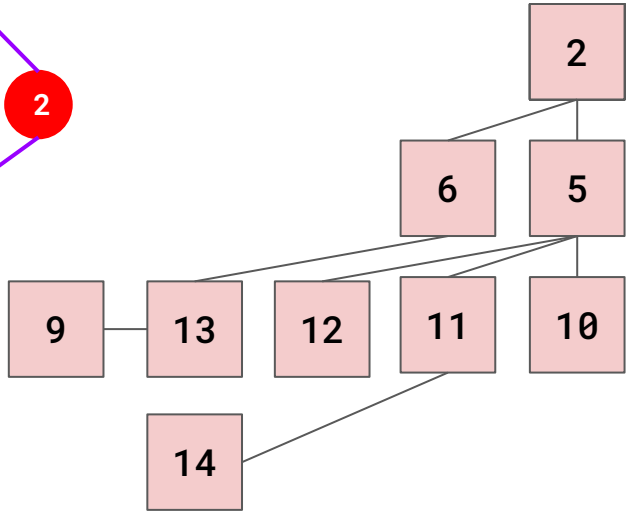




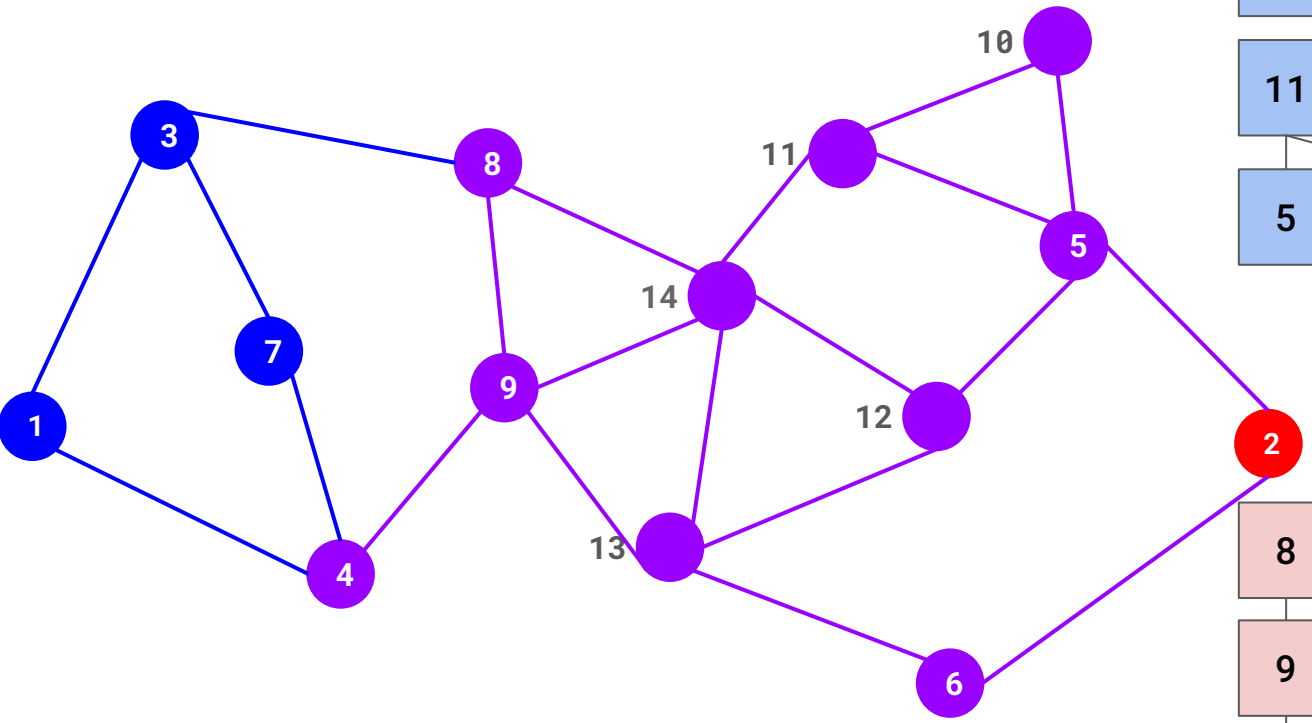
MULTI SOURCE MULTI PATH BFS

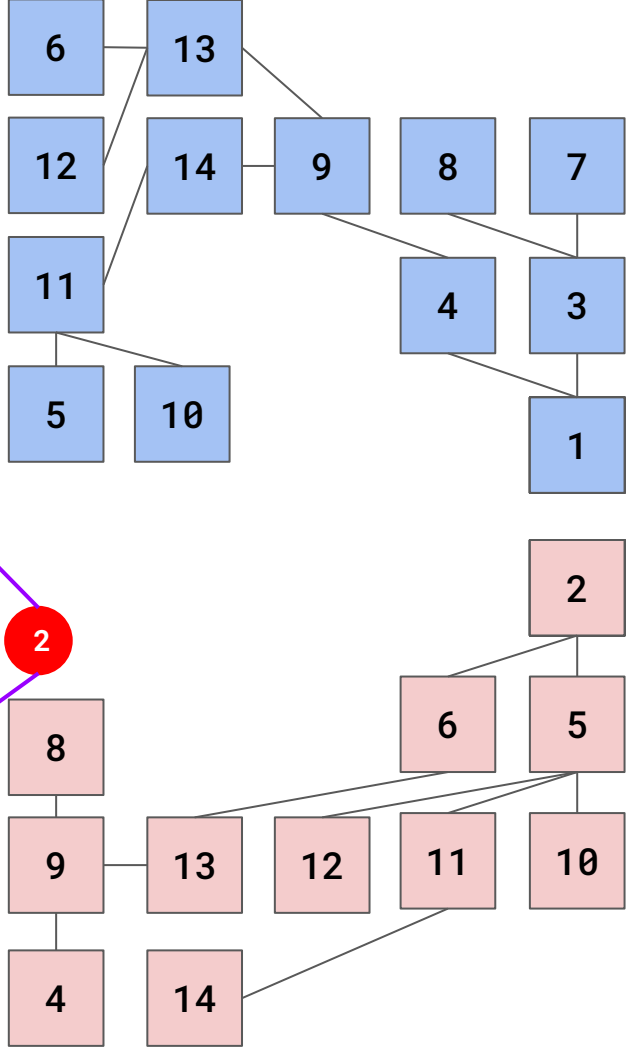




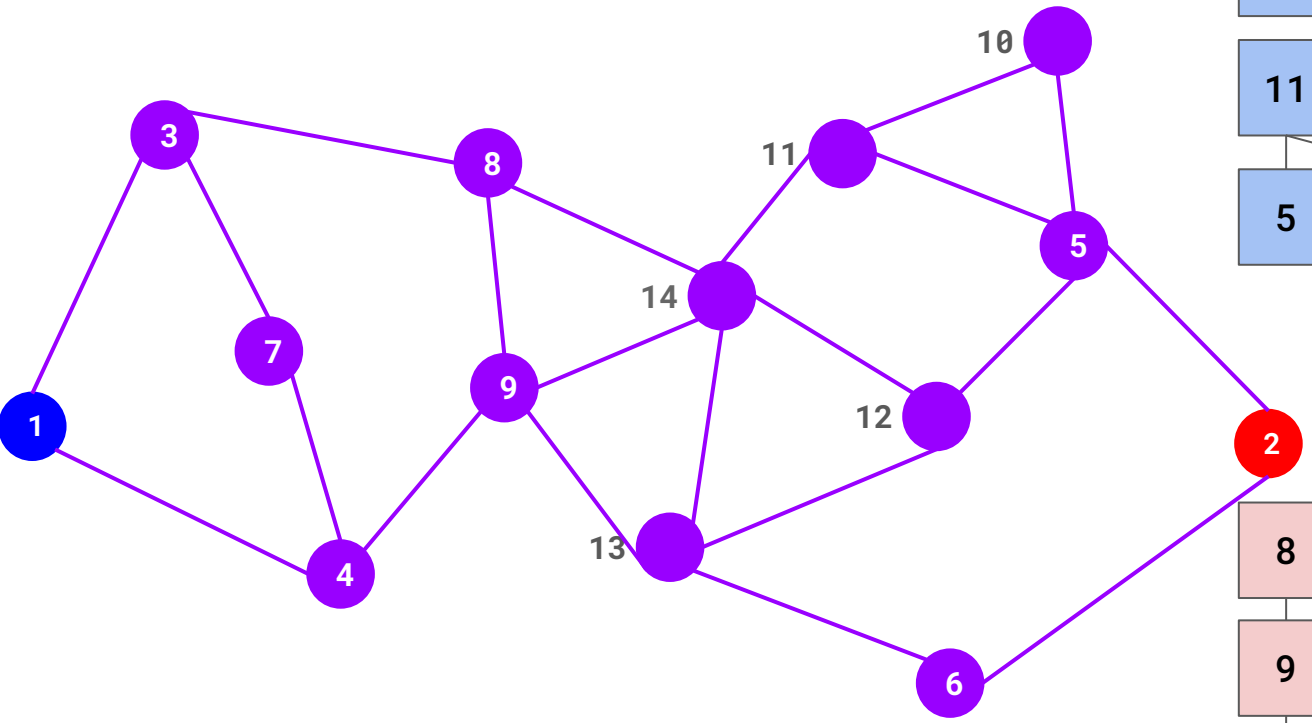


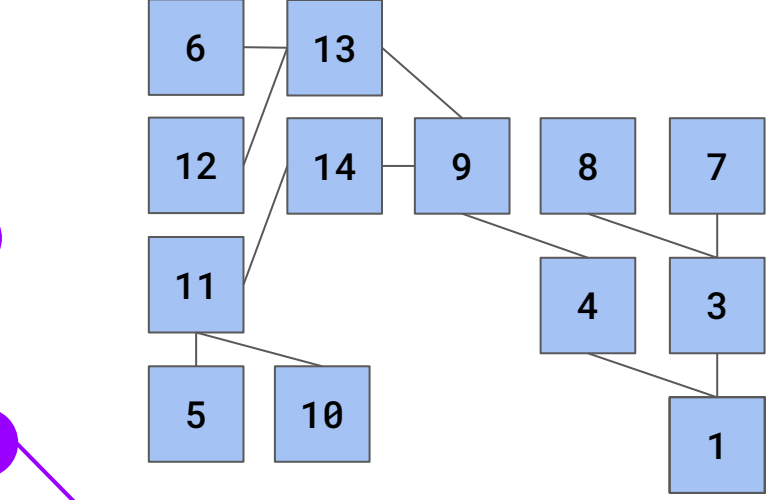
MULTI SOURCE MULTI PATH BFS

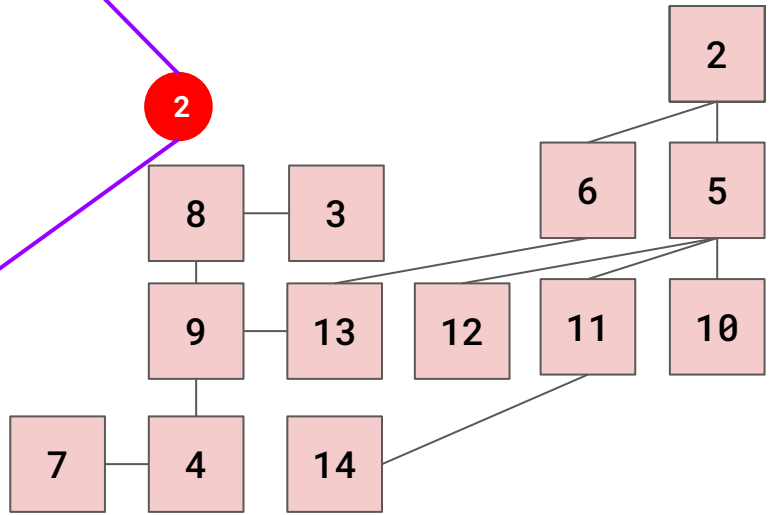


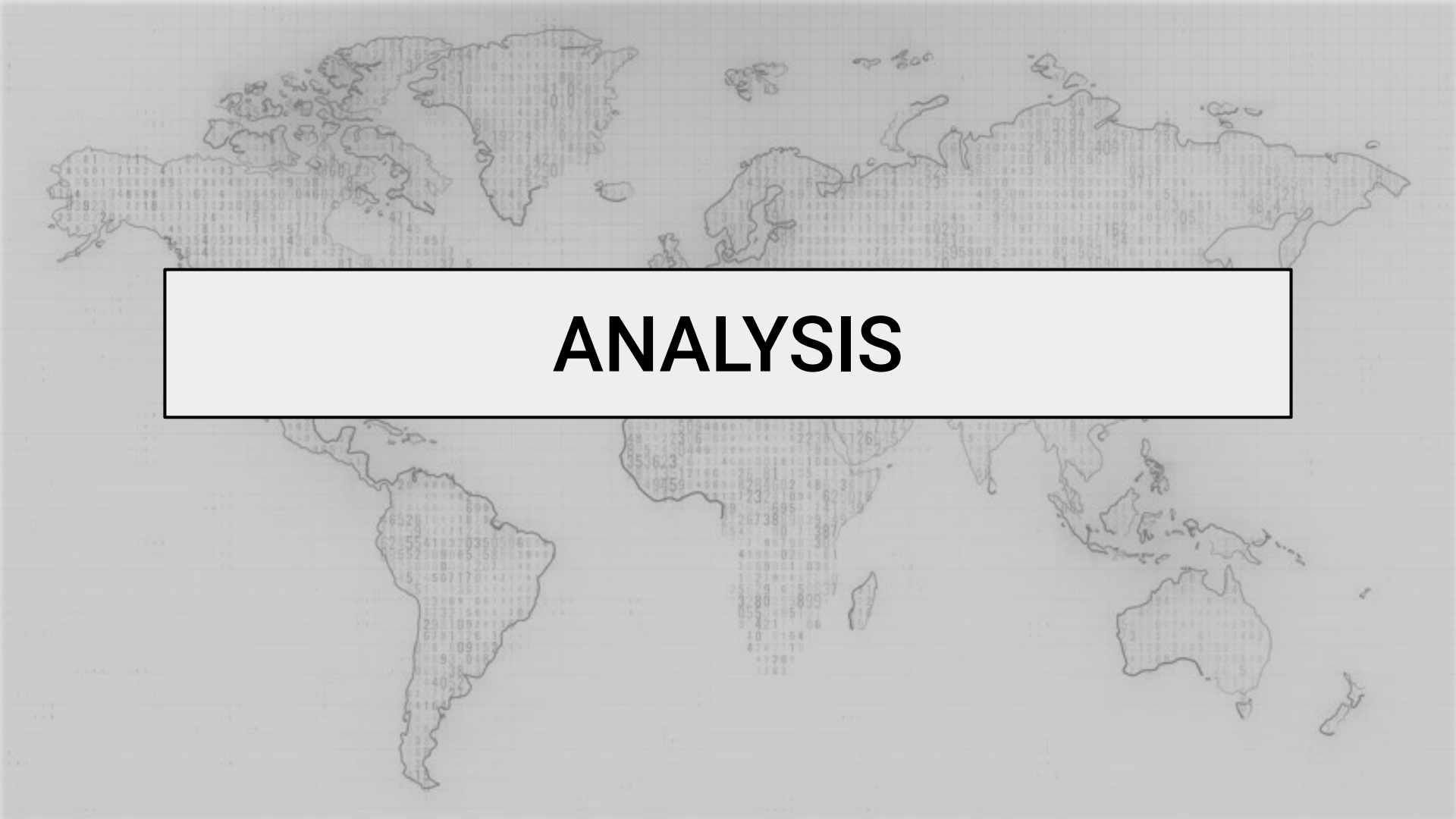


MULTI SOURCE MULTI PATH BFS








ANALYSIS

THEORETICAL ANALYSIS - TIME COMPLEXITY

Single Source
Single Path BFS

Time: O(V*E)

Single Source
Multi Path BFS

Time: O(K*V*E)

Multi Source
Single Path BFS

Time: O(V+E)

Multi Source
Multi Path BFS

Time: O(K*(V+E))

THEORETICAL ANALYSIS - SPACE COMPLEXITY

Single Source
Single Path BFS

Uses 2 lists for every node

Single Source
Multi Path BFS

Uses 3 lists for every node

Multi Source
Single Path BFS

Uses 1 dictionary for 1 search done
Uses 1 tree for 1 search done

Multi Source
Multi Path BFS

Uses 1 dictionary for 1 search done
Uses h trees for 1 search done
Where h is the number of hospitals

THEORETICAL ANALYSIS - SPACE COMPLEXITY

Single Source
Single Path BFS

Space: O(2*V)

Single Source
Multi Path BFS

Space: O(3*V)

Multi Source
Single Path BFS

Space: O(V+V)

Multi Source
Multi Path BFS

Space: O(H*V+V)

THEORETICAL ANALYSIS - SPACE COMPLEXITY

Single Source Single Path BFS Space: O(V)	Single Source Multi Path BFS Space: O(V)
Multi Source Single Path BFS Space: O(V)	Multi Source Multi Path BFS Space: O(H*V)

THEORETICAL ANALYSIS

(A) DESIGN AN ALGORITHM FOR COMPUTING DISTANCE FROM EACH NODE TO ITS NEAREST HOSPITAL

MULTI SOURCE SINGLE PATH

(B) TIME COMPLEXITY SHOULD NOT DEPEND ON NUMBER OF HOSPITALS

MULTI SOURCE SINGLE PATH (FAST)
OR
SINGLE SOURCE SINGLE PATH (SLOW)

(C) FIND THE TOP 2 NEAREST HOSPITALS FROM EACH NODE

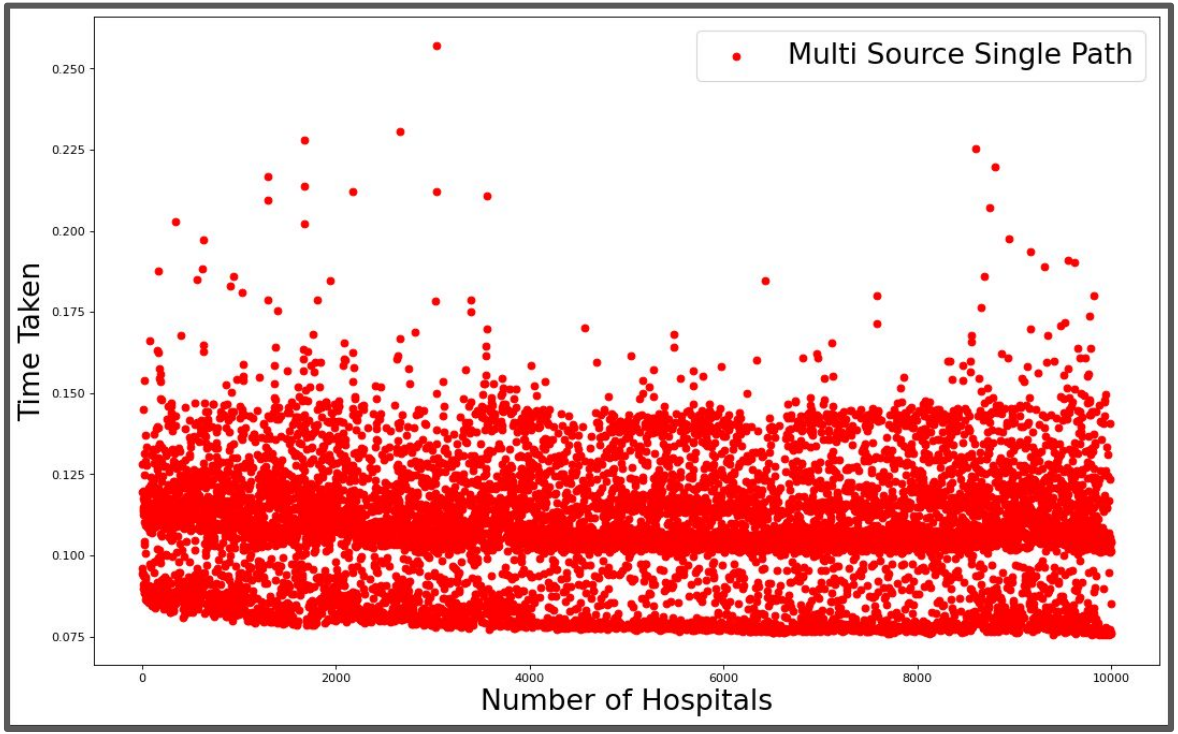
MULTI SOURCE MULTI PATH

(D) GENERAL ALGORITHM FOR COMPUTING DISTANCE FROM EACH NODE TO TOP-K NEAREST HOSPITALS FOR ANY VALUE OF K

MULTI SOURCE MULTI PATH

EMPIRICAL STUDY AND ANALYSIS

TIME COMPLEXITY SHOULD NOT DEPEND ON NUMBER OF HOSPITALS

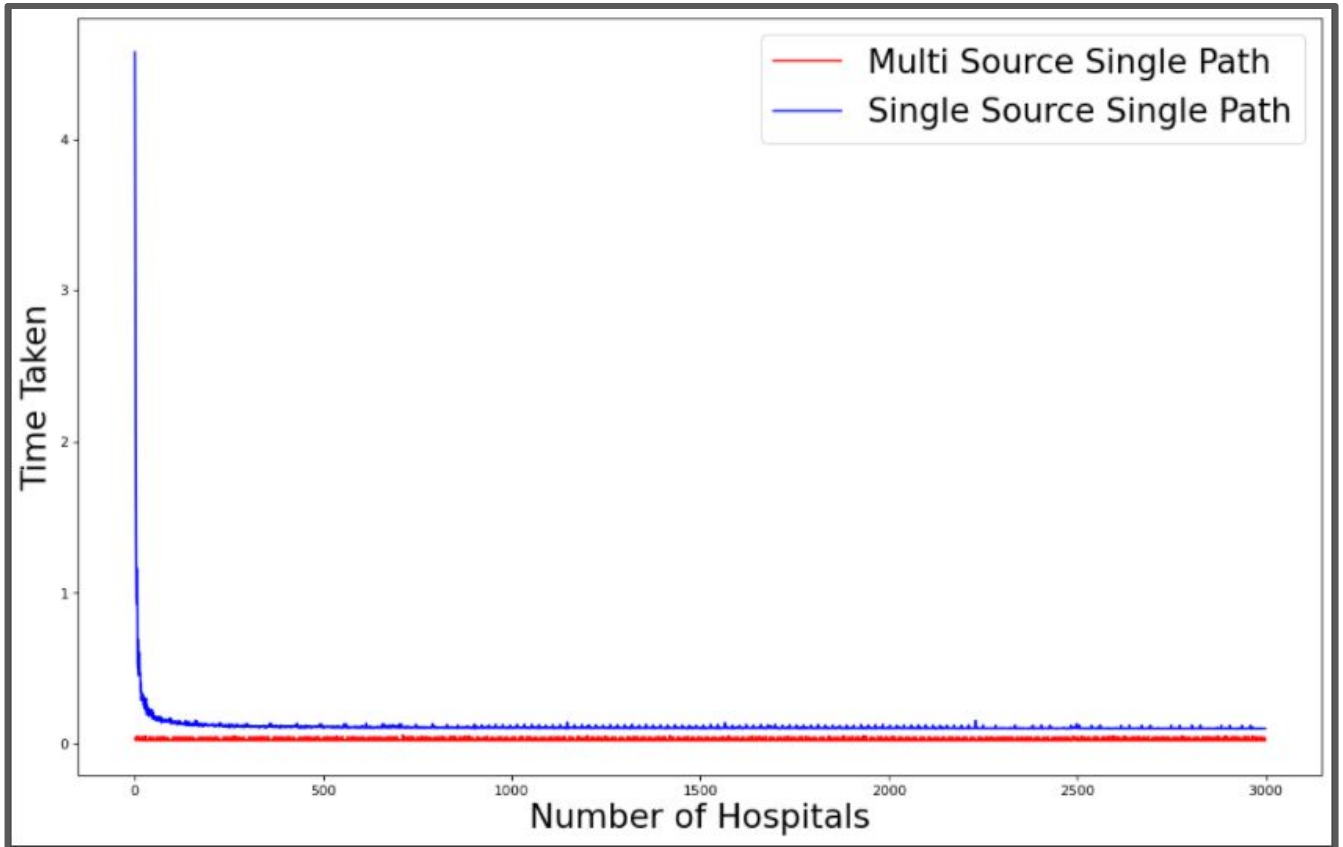


EMPIRICAL STUDY AND ANALYSIS

EFFECTS OF

NUMBER OF

HOSPITALS

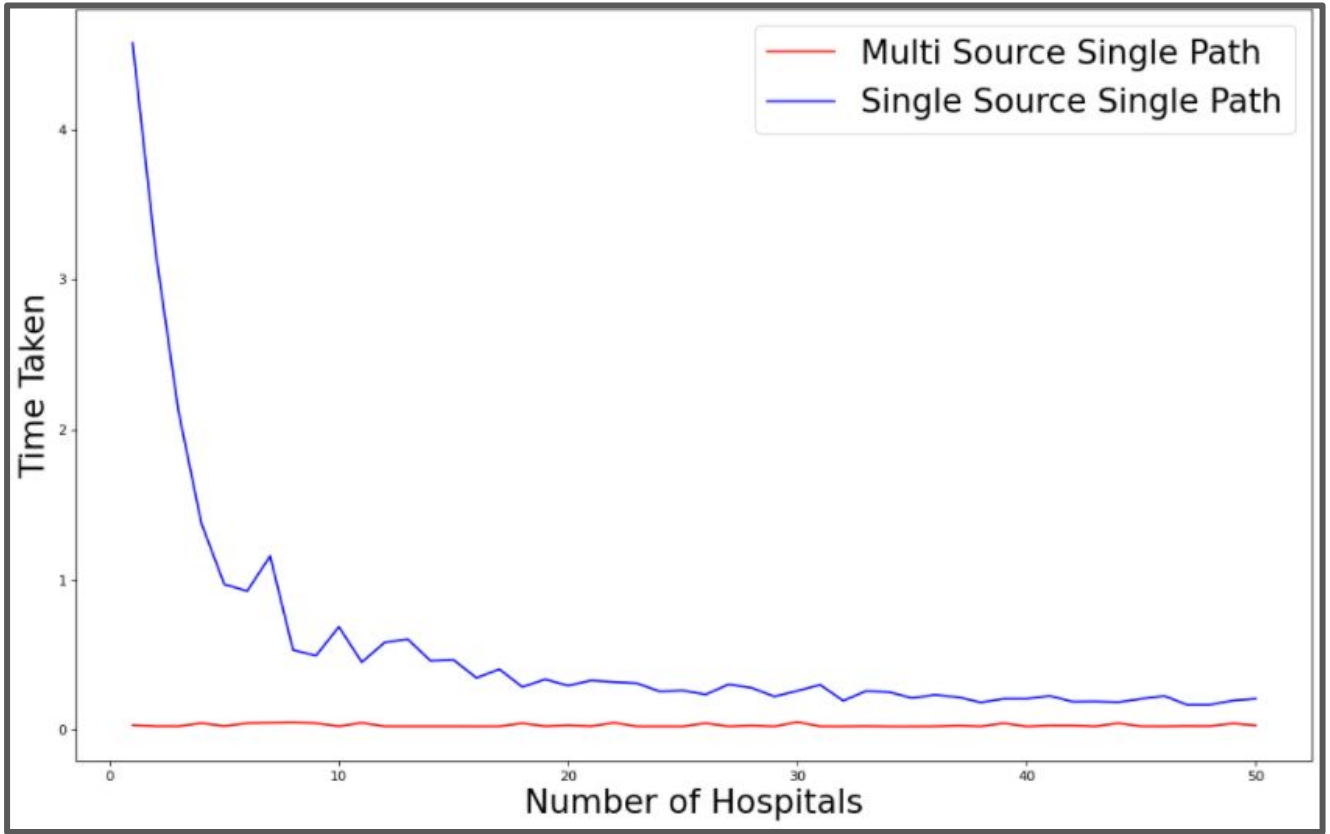


EMPIRICAL STUDY AND ANALYSIS

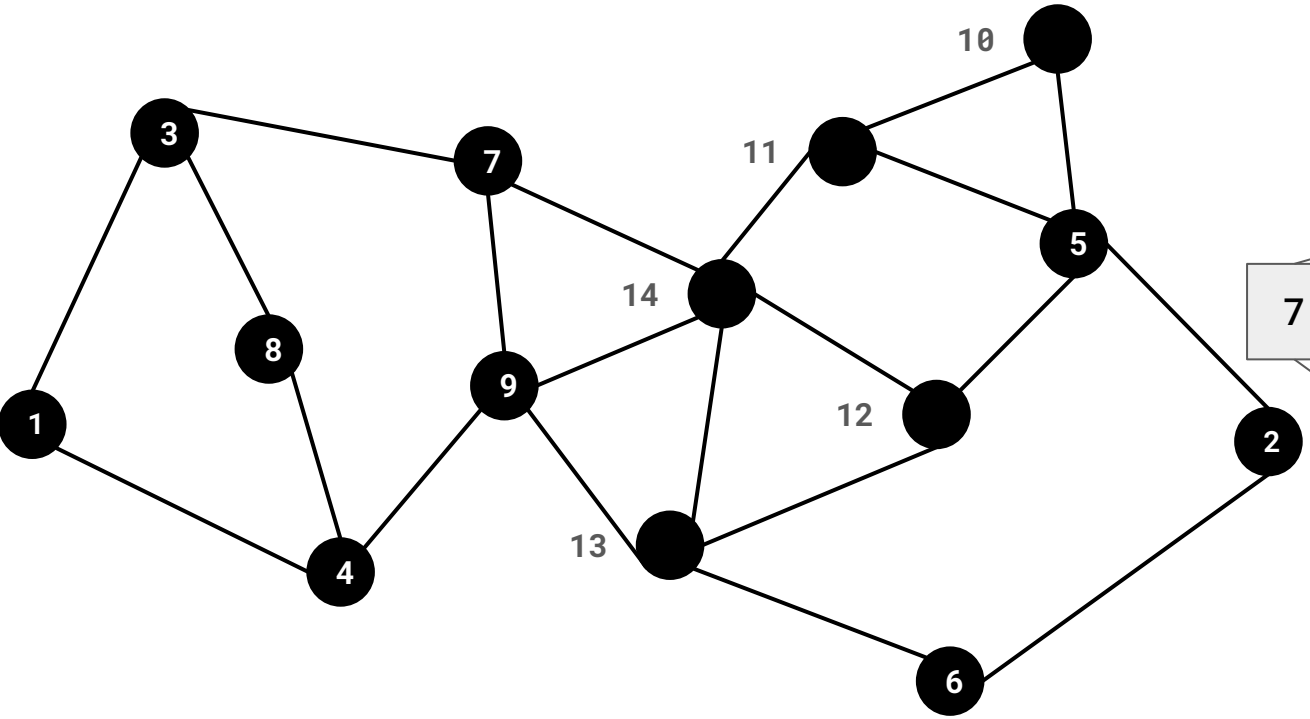
EFFECTS OF

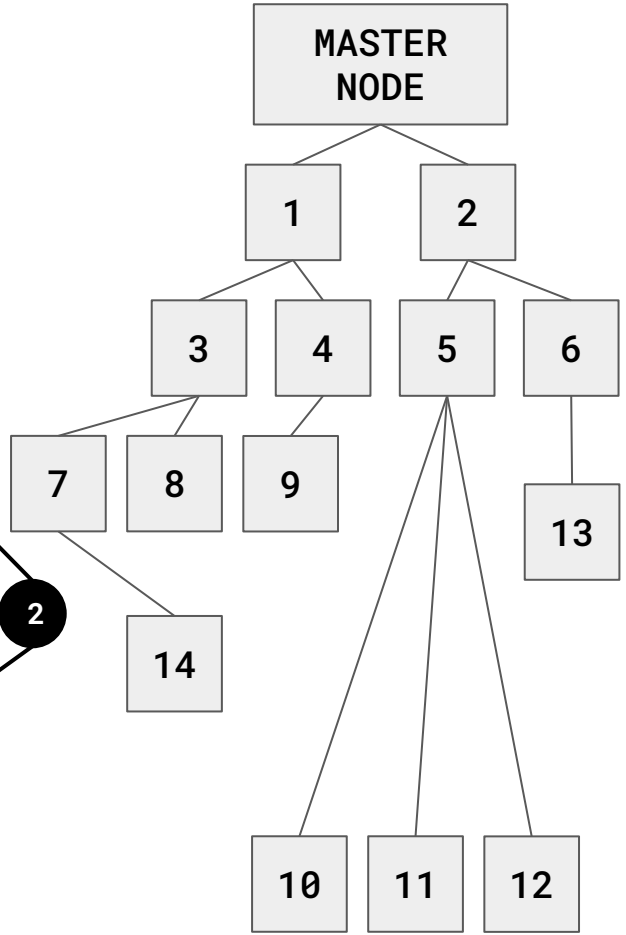
NUMBER OF

HOSPITALS

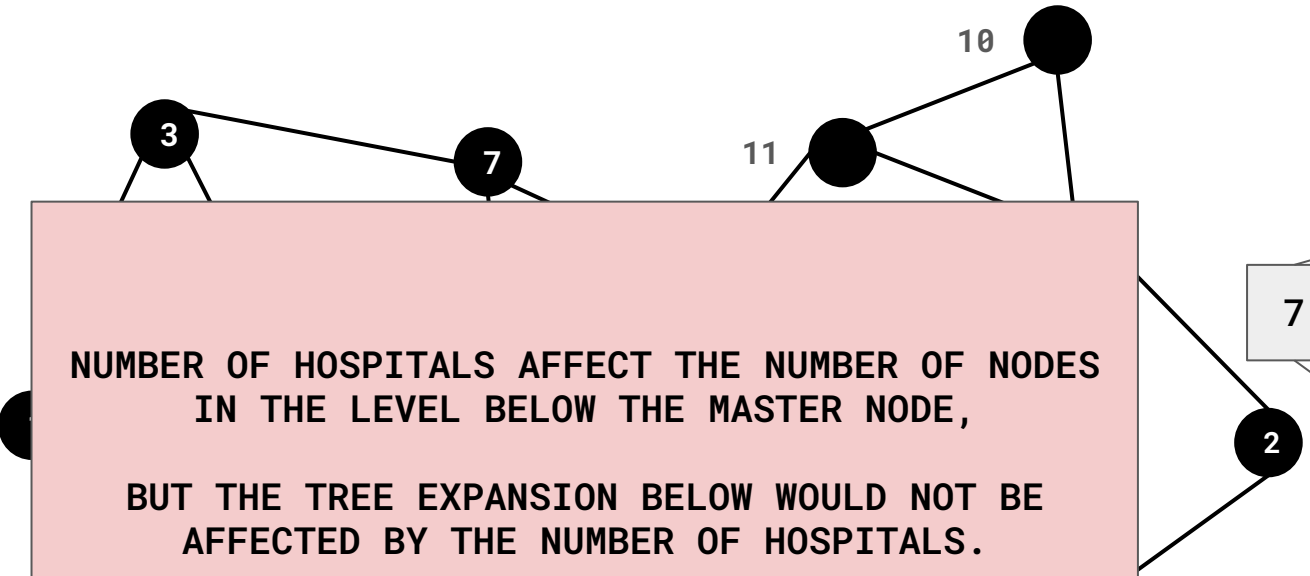


MULTI SOURCE SINGLE PATH BFS





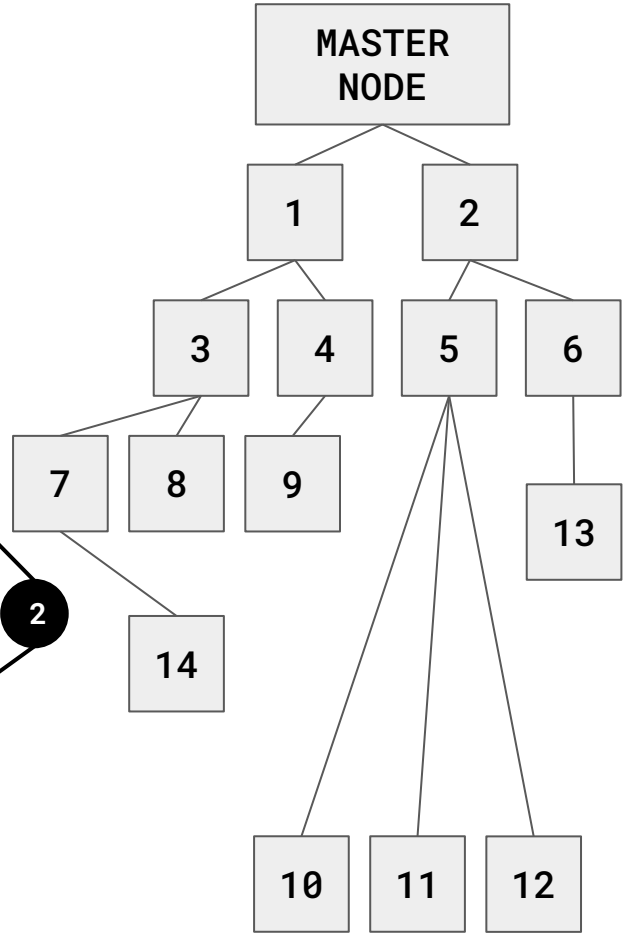
MULTI SOURCE SINGLE PATH BFS



NUMBER OF HOSPITALS AFFECT THE NUMBER OF NODES IN THE LEVEL BELOW THE MASTER NODE,

BUT THE TREE EXPANSION BELOW WOULD NOT BE AFFECTED BY THE NUMBER OF HOSPITALS.

AS IT WOULD STOP EXPANDING WHEN ALL THE REACHABLE NODES ARE REACHED.



THEORETICAL ANALYSIS

(A) DESIGN AN ALGORITHM FOR COMPUTING DISTANCE FROM EACH NODE TO ITS NEAREST HOSPITAL

MULTI SOURCE SINGLE PATH

(B) TIME COMPLEXITY SHOULD NOT DEPEND ON NUMBER OF HOSPITALS

MULTI SOURCE SINGLE PATH

(C) FIND THE TOP 2 NEAREST HOSPITALS FROM EACH NODE

MULTI SOURCE MULTI PATH

(D) GENERAL ALGORITHM FOR COMPUTING DISTANCE FROM EACH NODE TO TOP-K NEAREST HOSPITALS FOR ANY VALUE OF K

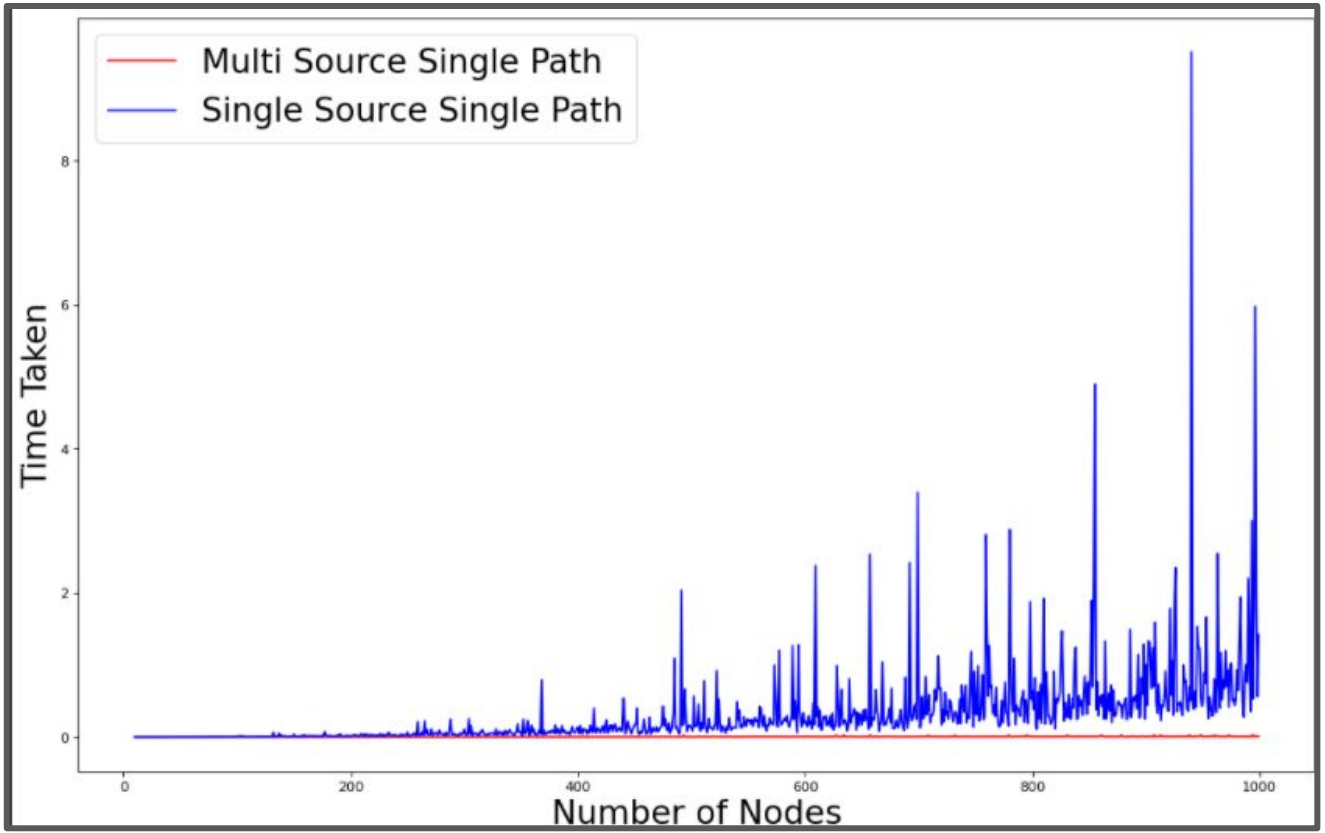
MULTI SOURCE MULTI PATH

EMPIRICAL STUDY AND ANALYSIS

EFFECTS OF

NUMBER OF NODES IN THE GRAPH

Single Source Single Path BFS rate of Increase is much faster than **Multi Source Single Path BFS**



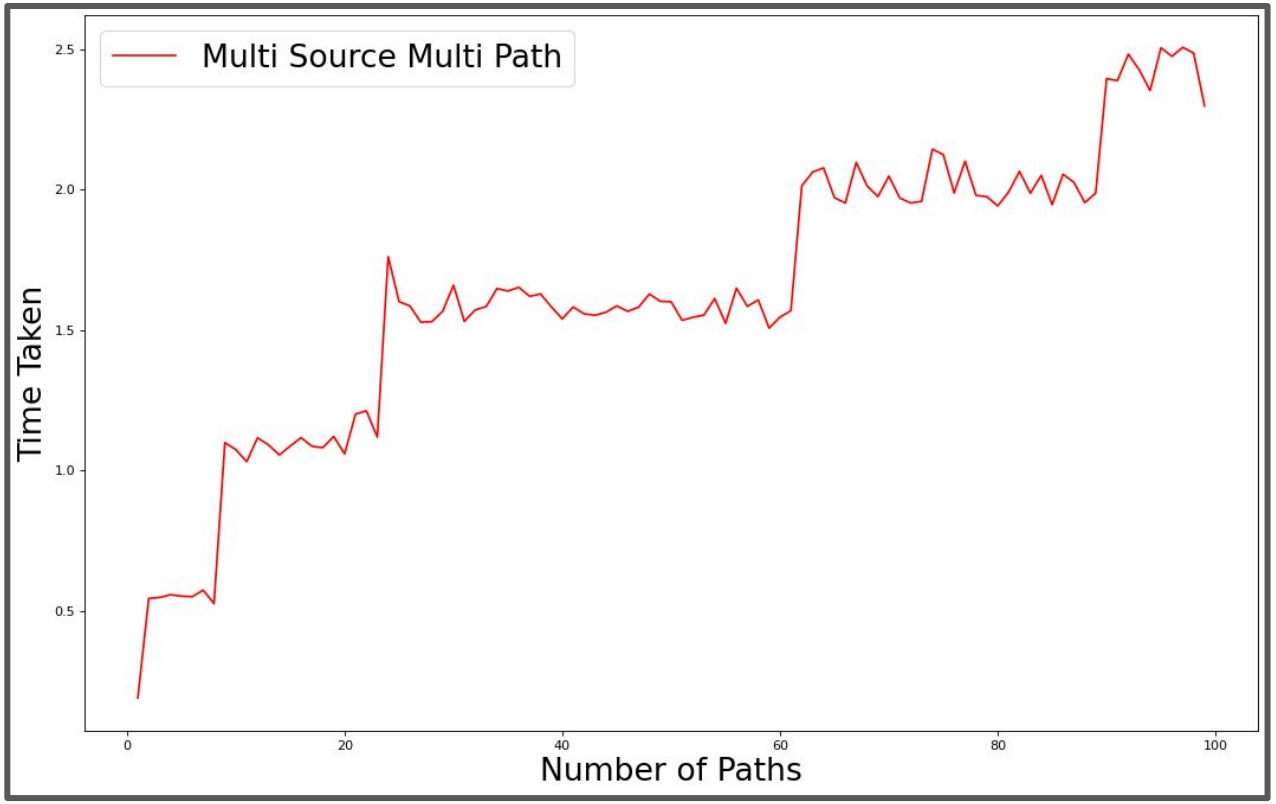
EMPIRICAL STUDY AND ANALYSIS

EFFECTS OF K

K → 1 to H

TIME TAKEN HAS STAIRCASE LIKE INCREASE IN TIME TAKEN DUE TO OUR ALGORITHM

FULLY EXPLORES, THEN CHECKS

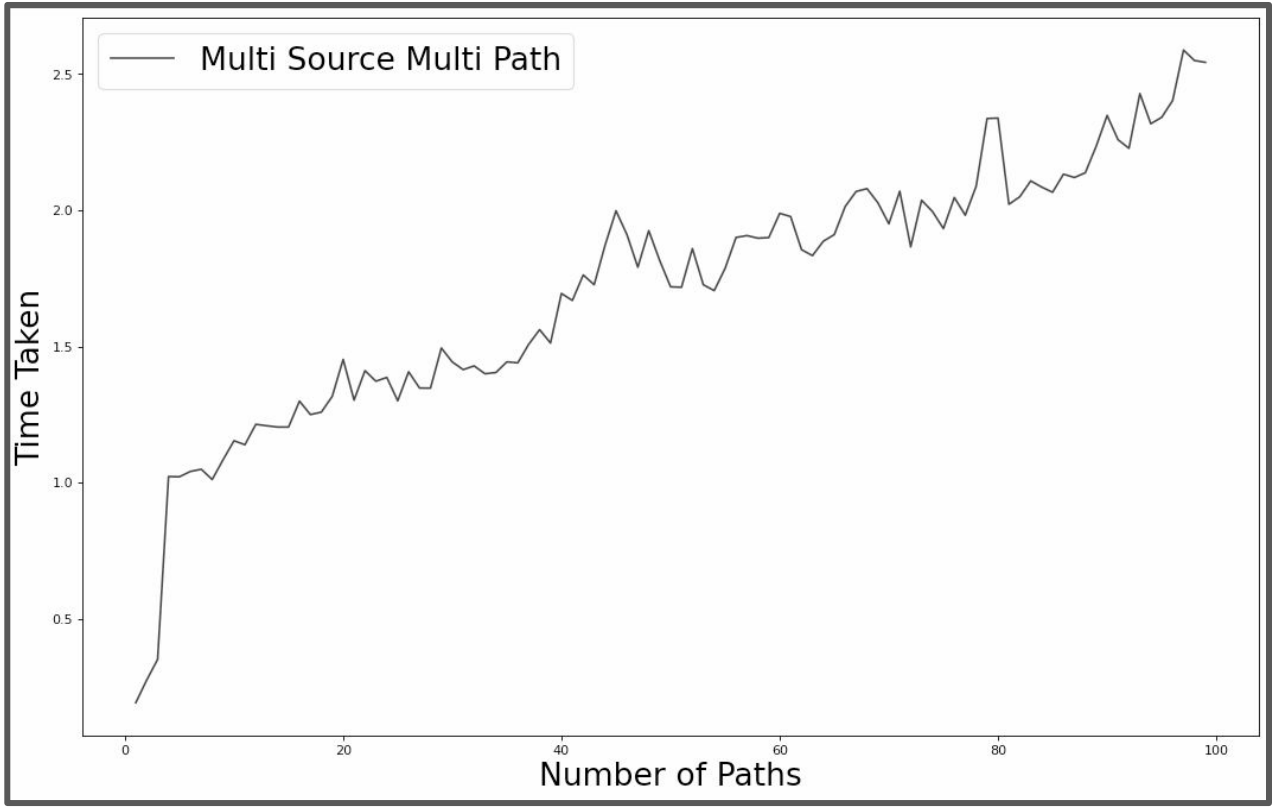


EMPIRICAL STUDY AND ANALYSIS

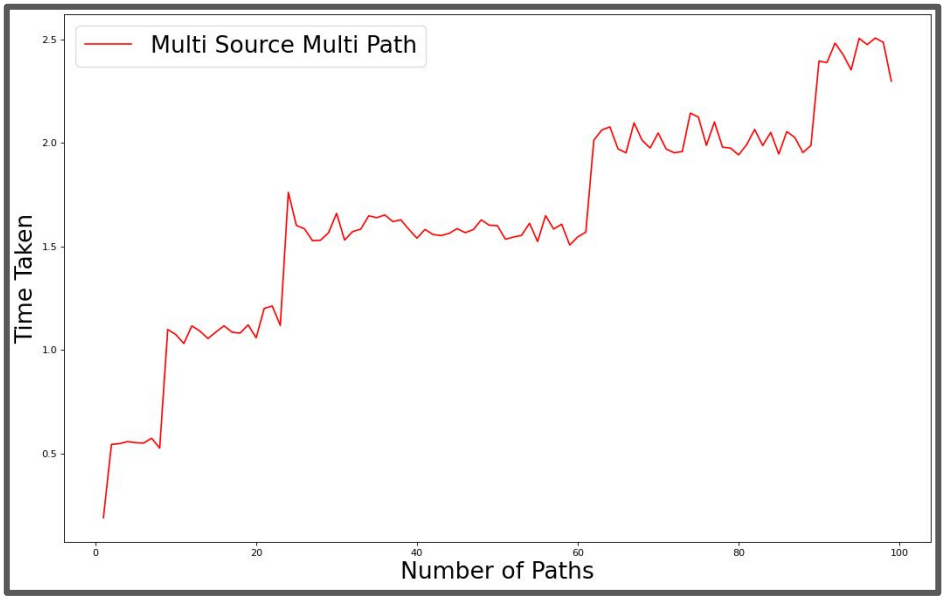
EFFECTS OF K

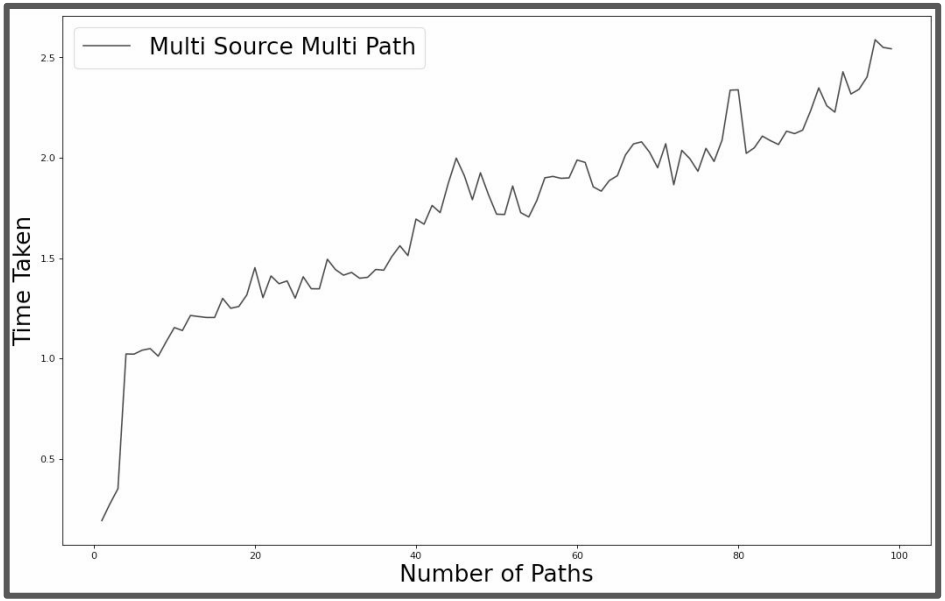
K → 1 to H

CAN BE A MORE CURVED SHAPE IF MORE CHECKS ARE CONDUCTED BETWEEN EXPLORATION



EMPIRICAL STUDY AND ANALYSIS





ONE ROUND OF CHECK AFTER A FULL SET OF EXPANSION.

ADDITIONAL CHECKS POTENTIALLY CAUSING SOME OVERHEADS.

ONE ROUND OF CHECK AFTER AN EXPANSION FOR EVERY HOSPITAL.

FINAL CONCLUSION

(A) DESIGN AN ALGORITHM FOR COMPUTING DISTANCE FROM EACH NODE TO ITS NEAREST HOSPITAL

MULTI SOURCE SINGLE PATH

(B) TIME COMPLEXITY SHOULD NOT DEPEND ON NUMBER OF HOSPITALS

MULTI SOURCE SINGLE PATH

(C) FIND THE TOP 2 NEAREST HOSPITALS FROM EACH NODE

MULTI SOURCE MULTI PATH

(D) GENERAL ALGORITHM FOR COMPUTING DISTANCE FROM EACH NODE TO TOP-K NEAREST HOSPITALS FOR ANY VALUE OF K

MULTI SOURCE MULTI PATH

WHAT ELSE COULD WE HAVE DONE

For road networks we could have built a graphical heuristics so as to give direction to searches done:

- Saving time ➜ lesser expansions needed
- In cases with many hospitals ➜ saving space