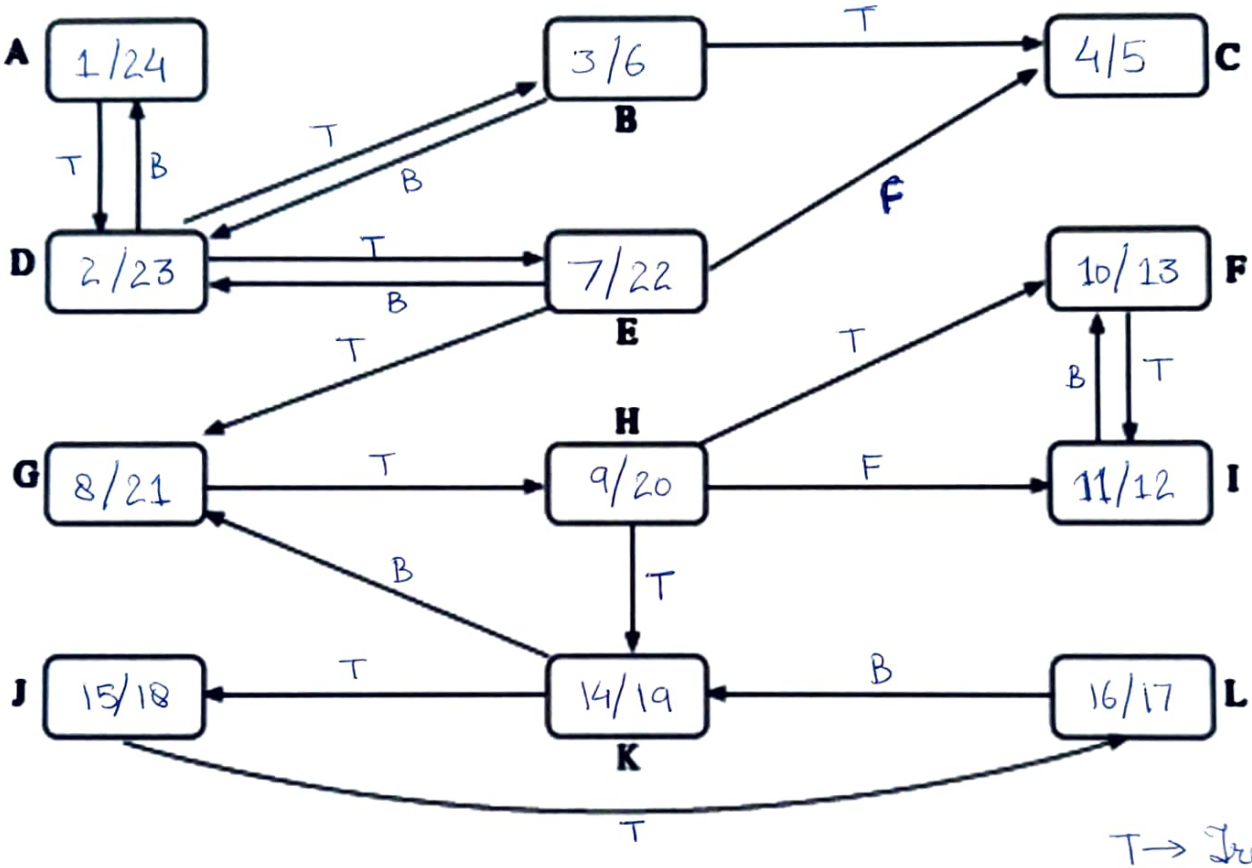
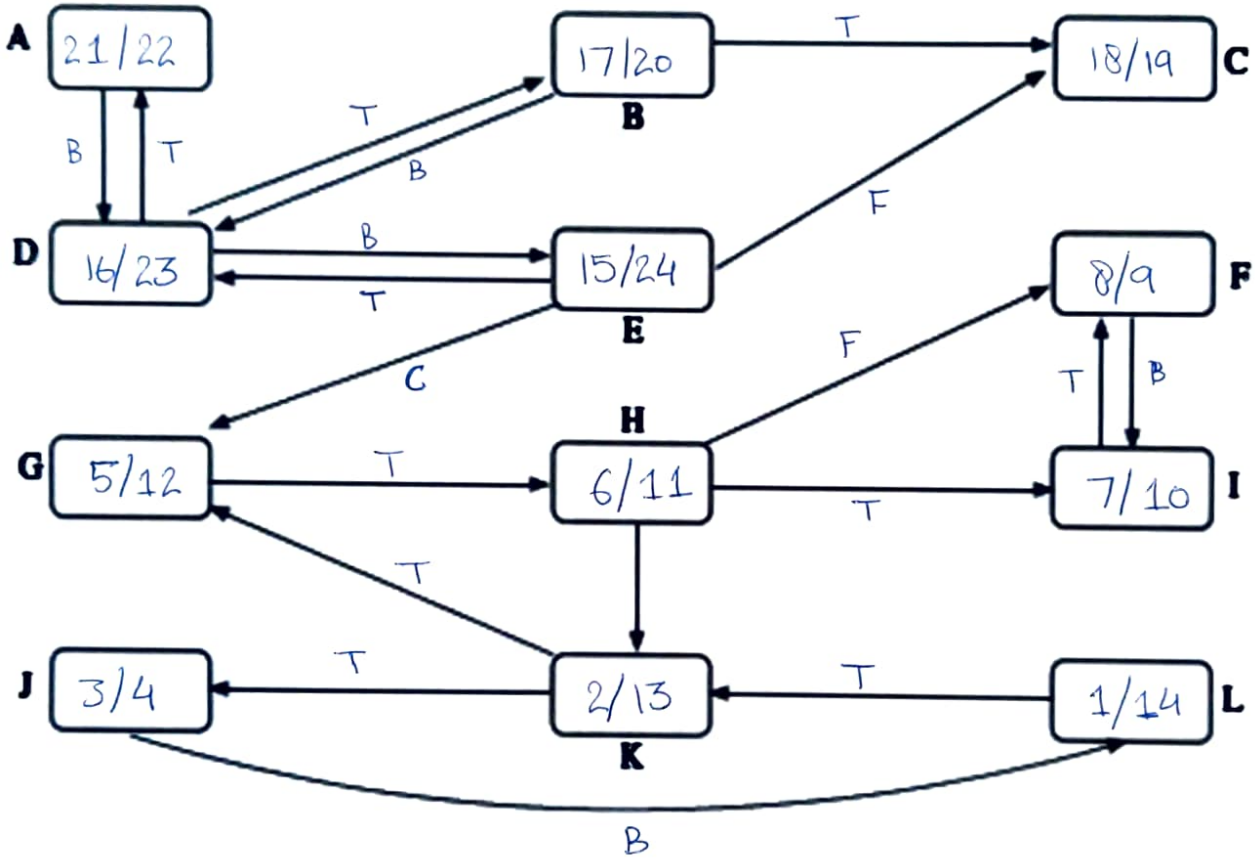
a)

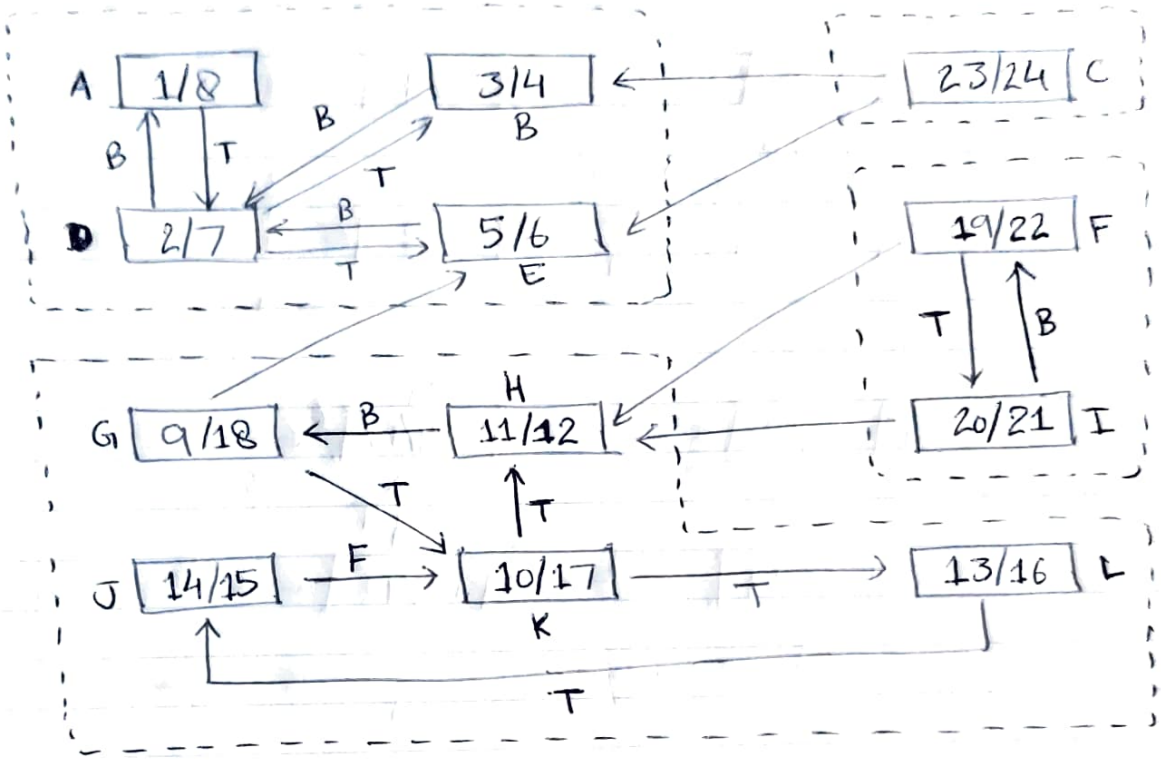


T → Tree edge
B → Back edge
F → Forward edge
C → Cross edge

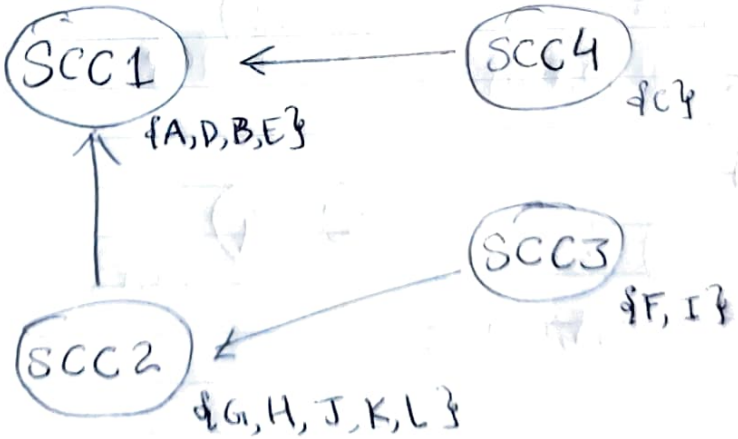
b)


A
21/22
B
T
T
B
17/20
B
T
18/19
C
D
16/23
B
T
15/24
E
F
C
8/9
F
T
B
H
G
5/12
T
6/11
T
7/10
I
T
J
3/4
T
2/13
K
T
1/14
L
B

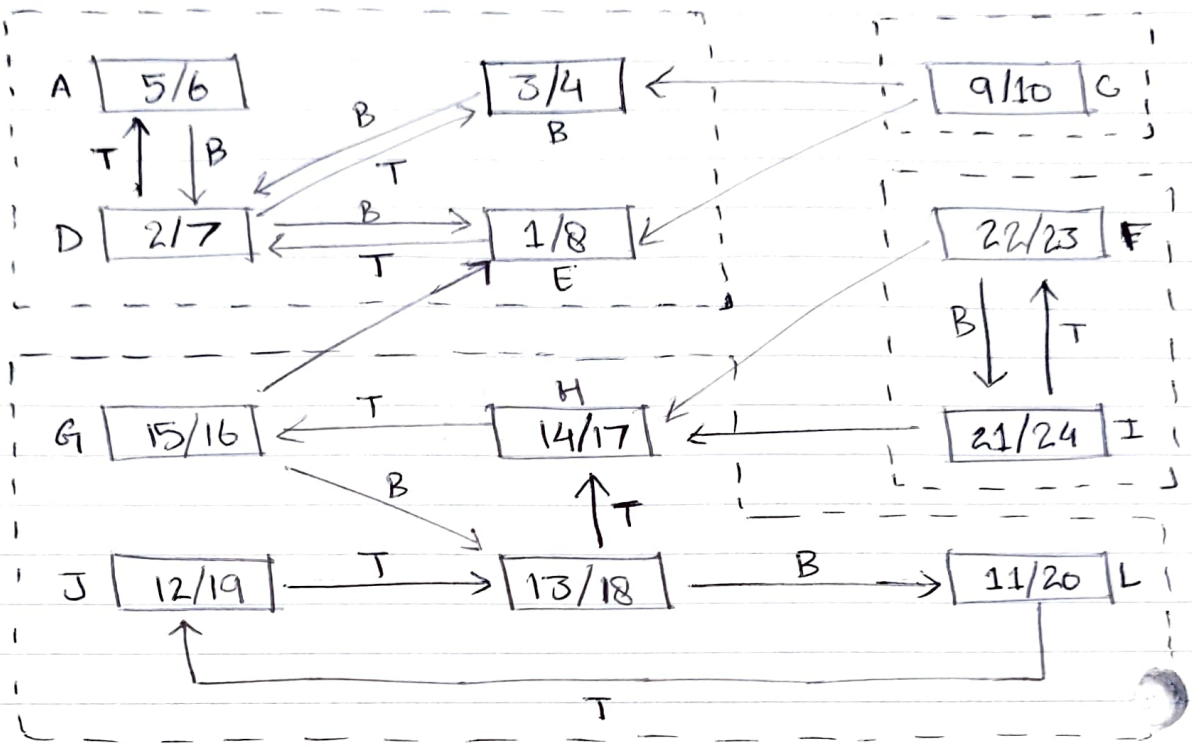
c)



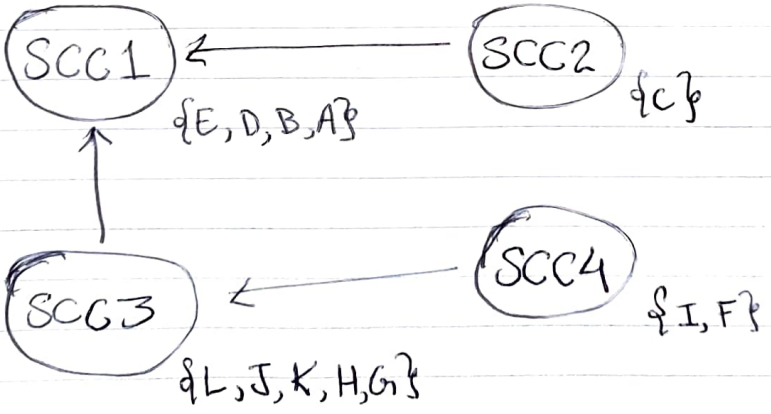
T- Tree edge
B - Back edge
F - Forward edge



d)



T → Tree edge
B → Back edge
F → Forward edge



Homework Assignment 5

Abstract

Searching a graph means systematically following the edges of the graph to visit the vertices of the graph. A graph-searching algorithm can discover much about the structure of a graph. Breadth-first search (BFS) and Depth First Search (DFS) are the simplest two graph search algorithms. The main goal of this report to study the traversal techniques of the BFS and DFS algorithms and analyze the results of BFS and DFS traversal on a graph.

Introduction

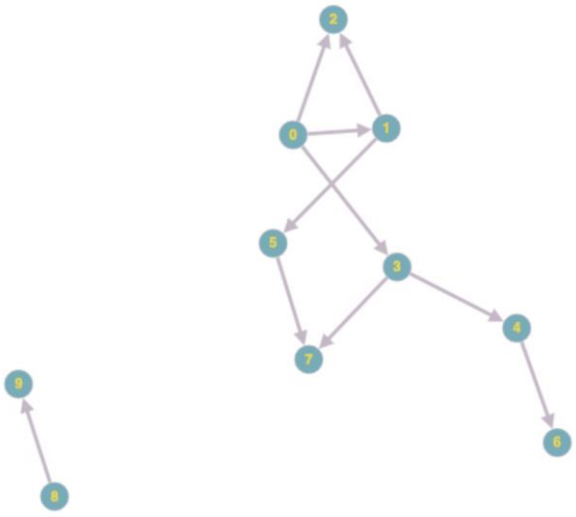
Breadth-first search is one of the simplest algorithms for searching a graph. Given a graph G= (V, E) and a distinguished source vertex s, breadth-first search systematically explores the edges of G to discover every vertex that is reachable from s. For any vertex v reachable from s, the simple path in the breadth-first tree from s to v corresponds to a shortest path from s to v in G.

Depth-first search explores edges out of the most recently discovered vertex v that still has unexplored edges leaving it. Once all of v's edges have been explored, the search backtracks to explore edges leaving the vertex from which v was discovered. This process continues until we have discovered all the vertices that are reachable from the original source vertex. If any undiscovered vertices remain, then depth-first search selects one of them as a new source, and it repeats the search from that source. The algorithm repeats this entire process until it has discovered every vertex.

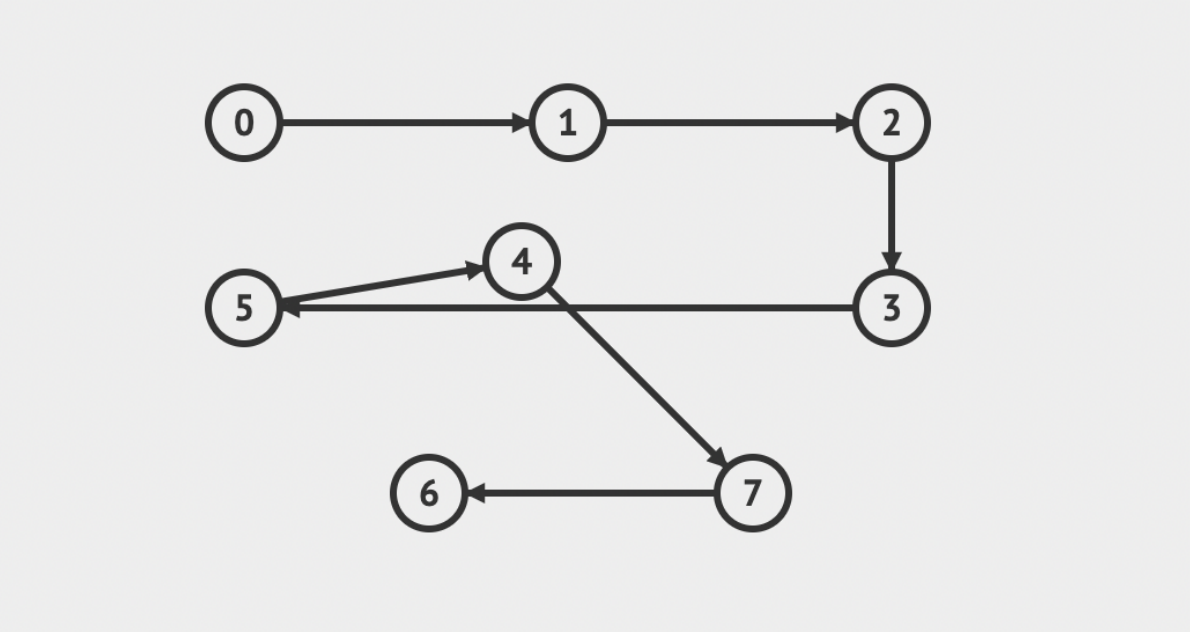
We'll perform the BFS and DFS traversal using the adjacency matrix representation of a graph and present out results in the next section.

Results and Evaluation:

The graph to be traversed represented using the adjacency matrix is shown below.

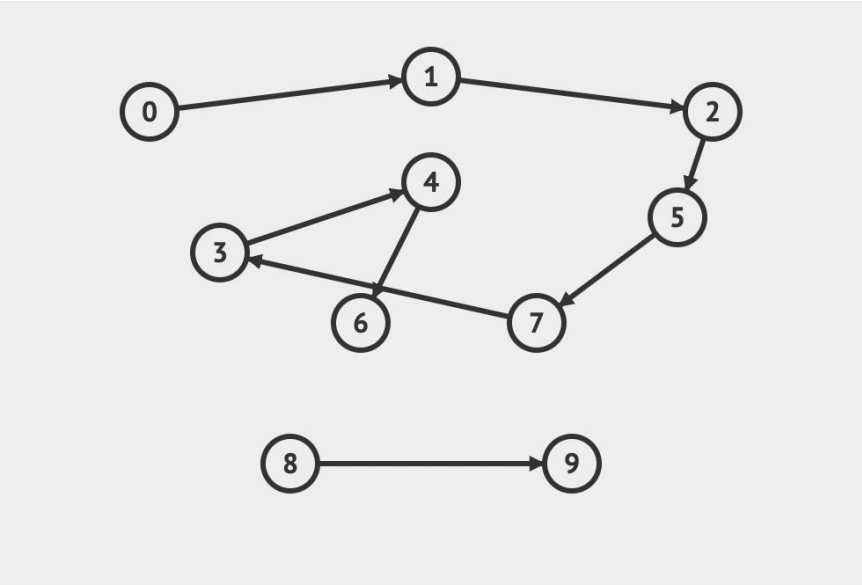


After performing the BFS traversal of the above graph, we get the breadth-first tree shown below.



From the above diagram, we can see that the nodes 8 and 9 are unreachable using the BFS traversal and hence are not part of the breath-first tree. The time complexity of BFS is O(V + E) when Adjacency List is used and $O(V^2)$ when Adjacency Matrix is used, where V stands for vertices and E stands for edges

After performing the DFS traversal of the above graph, we get the below depth-first tree.



From the above diagram, we can see that the nodes 8 and 9 are reachable using the DFS traversal even though they are not connected to the start node 0. The subgraph of a depth-first search forms a depth-first forest comprising several depth-first trees**.** The time complexity of DFS is also O(V + E) when Adjacency List is used and $O(V^2)$ when Adjacency Matrix is used, where V stands for vertices and E stands for edges.

Conclusion:

We're successfully able to implement and study the traversal behavior of BFS and DFS graph searching algorithms. From the above results, we can see the breadth-first search systematically explores the edges of G to discover every vertex that is reachable from s and hence siblings are visited before the children. DFS on the other hand, searches deeper in the graph whenever possible and hence children are visited before the siblings.