Quesl sol :-

key	BFS	DFS
Definition	Stands for Breadth first Search	Stands for depth first Search.
Data Structure	It uses Queue to find the Shortest path.	It uses Stack to find the Shortest path.
Sources	It is better when target is closer to source	It is better when target is far from source.
Suitable for decision tree	It considers all neighbours so it is not suitable for decision tree used in Puzzle games.	It is more suitable as with one decision we need to traversal further to the decision.
Speed	It is slower the DFS	It is faster than BFS.
Time complexity	O(V+E) where V is Vertices & E is edges	O(V+E) where V is Vertices & E is edge.

Q2 sol!— Stack is used to Implement DFS, because in it we first traversal the whole branch of the tree & later on visit the adjacent branch, since this is similar to LIFO, therefore stacks used.

Queue is used to Implement BFS, It is because queue is use as a FIFO instead because BFS is to test the immediat children first & after all immediate children are tested, to these return to those children & check their children

Ques3 sol!— Sparse Graph - Graph where no of edges is much less than the possible number of edges.

Dense Graph — where number of edges is much close to maximal number of edges.

if graph is dense it should be represented by adjancey matrix

if graph is sparse it should be represented by adjacency list.

Qu 4 BFS - In undirected graph, do a BFS traversal on given graph, for each visited vertex v, if there is an adjacent 'u' such that 'u' is already visited & 'u' is not parent of 'v' then there is cycle in a graph.

DFS - Run DFS from a node and mark this node as visited now for any other Vertice if its neighbour is already visite & that neighbour is not the parent of that current node then there exist a cycle in the graph.

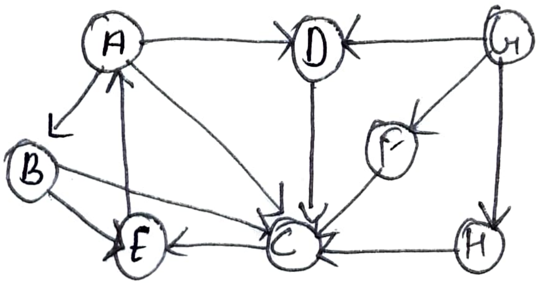
Ques5 Disjoint Set Data Structure -
soln! - The disjoint set can be defined as the sub sets where there is no common element b/w two sets.

Operations are -

- union
- Make new set
- find.

Ques6 soln! - BFS

$A \to B \to C \to D \to E$
$G \to H \to F$



DFS

$A \to D \to C \to B$, $G \to F \to H$

Ques7 soln! - Connected Components = 4
Vertices = 10

Q8 soln! - Topological Sort $\to$ 0-1-2-3-4-5
DFS $\to 5 \to 2 \to 3 \to 1 \to 0$
4 can't be reached

Q9 soln! - Yes, heap data structure can be used to create priority queue.

- Dijkstra to find shortest path.
- Prim's Algo

Q10 soln! - Min heap - root element is the smallest
Max heap - root element is the largest.