Q1

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
Source	It is better when target is close to source	It is better when target is far from source
Suitable for decision tree	It considers all neighbour so it is not suitable for decision tree used in puzzle games.	It is more suitable as with one decision we need to traverse further to augment the decision
Speed	It is slower than DFS	It is faster than BFS
Time complexity	$O(V+E)$ where V is vertices & E is edges	$O(V+E)$ where V is vertex and E is edges

Q2

Stack is used to implement DFS, because in it we first traverse the whole branch of the tree & later on visit the adjoint Branch, since this is similar to FIFO, therefore stack is used.

Queue is used to implement BFS, it is because queue is used as a FIFO instead because BFS is to test the immediate children first & after all immediate children ~~are~~ tested, to them return to those children & check their children & so forth.

Q3

Sparse graph -> graph when 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 adjacency matrix

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

Q4 BFS

In undirected graph, do a BFS traversal on given graph, for each visited vertex V, if there is an adjacent 'u' such that 'v' 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 visited & that neighbour is not the parent of that current node then there exist a cycle in the graph.

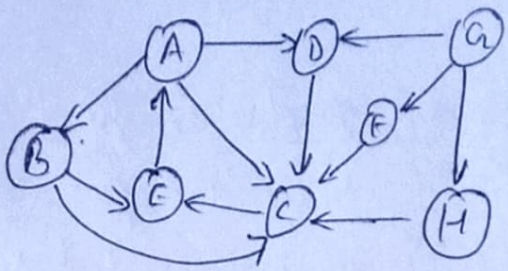
Q5 Disjoint Set data Structure

The disjoint set can be defined as the subsets where there is no common element b/w two sets

Operation are

i) union
ii) make new set
iii) find.

6 BFS



A → B → C → D → E

G → H → F

DFS

A → D → C → B, G → F → H.

Q7 connected component = 1

vertices = 10

Q8 Topological sort → 0 - 1 - 2 - 3 - 4 - 5

DFS → 5 → 2 → 3 → 1 → 0

4 can't be reached

Q9 yes, heap data structures can be used to create priority queue

- Dijkstra to find shortest path
- Prim's algo
- Huffman algo

Q10

min heap → root element is the smallest

max heap → root element is the largest