Q1

Key	BFS	DFS
Definition	Stands for Breadth first Search	Stands for depth first search.
Data Struct	It uses Queue to find the Shortest path	It uses Stack to find the shortest path.
Source	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 traverse further to augment 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 edges

Q2 Stack is used to implement DFS, because in it we first traverse the whole branch of the tree & later on visit the adjacent branch, since this is similar to LIFO, 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 then 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 ~~more~~ ~~than~~ close to maximal number of edge.

if graph is dense it should be represented by adjancecy matrix

If graph is sparse it should be represented by adjaceny 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 sub sets where there is no common element b/w two sets.

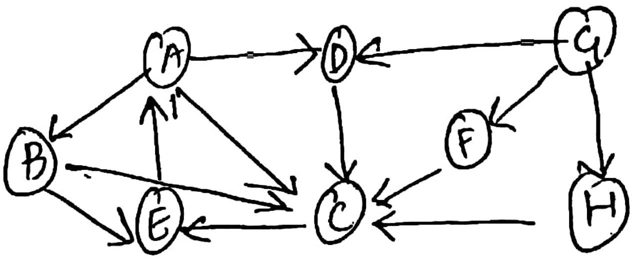
Operations are

i) Union

ii) make new set

iii Find

Q6 BFS



A → B → C → D → E,

G → ~~D →~~ H → F ~~→ G~~

DFS

A → D → C → B , G → F → H

Q7 connected components = 4

vertices = 10

Q8 Topological sort → 0-1-2-3-4-5

DFS → 5 → 2 → 3 → 1 → 0

4 can't be reached

Q 9) Yes, heap data structure can be used to create priority queue.

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

Q10) min heap → root element is the smallest

max heap → root element is the largest