**GRAPHS**

**https://github.com/mission-peace/interview/tree/master/src/com/interview/graph**

1. **Depth First Search and its application in backtracking algorithms. Applications - path finding in a matrix, printing all the words that can be formed in a matrix of characters and similar problems. Note : if graph is disconnected (forest), DFS as well as BFS should be done from all the source vertices).**
2. **Breadth First Search and its application in shortest path finding (snake and ladder problem minimum no of moves, shortest path in maze), connected components in a graph, etc.**
3. **Single Source Shortest Path Algorithms in weighted graphs – Dijkstra’s algo(also called greedy BFS which can’t detect negative weight cycles and fails), Bellman Ford Algo(detects weighted cycle as well). Prior to this understand shortest path has optimal substructure and also understand generic shortest path algorithm which uses relaxation of edges to find the shortest path(MIT video on Dijkstra by Srini Ramdas). Application – find minimum weight cycle in graph.**
4. **Detecting cycles in a graph (a bit tricky as approaches differ in directed/undirected)**
5. **Note : For directed graphs, the presence of a back edge(edge from the current vertex to any of its ancestor in the DFS tree) confirms a cycle, so DFS can do inO(E+V) time. For undirected graph, since there is no concept of back edge and also a vertex can be visited more than once due to undirected nature of the edge, we need to keep track of the dfs predecessor of the vertex while traversing; during traversal if we encounter an already visited vertex which is not the parent/dfs predecessor of the current one, then this confirms the cycle in the undirected graph. For undirected, union-find algo can also be used, in O(Elogv)time by making V sets and then iterating over set of edges. Follow-up - print all the cycles present in the graph. Find minimum weight cycle in graph.**
6. **Number of connected components in a graph or forest(use of DFS/BFS, disjoint set DS)**
7. **Topological sort in a DAG (Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge E(u,v), vertex u comes before v in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG. It’s different from DFS in the sense that a vertex must be printed before its outgoing/adjacent vertices which is not a necessity in DFS. So topological sort can be implemented by modifying DFS and using an explicit stack to store ordered visiting of vertices). The trick is that one should be able to print the path in a recursion tree in visited order-> keep recursing till u reach end of the path and then push the current node on stack, when recursive calls return parent nodes in the recursion tree will get pushed on the stack, final leaving the stack with order of nodes in the proper order as in the recursion tree.**
8. **Disjoint set DS using union by rank and path compression (application in cycle detection, no of connected components, Kruskal’s algorithm for MST). Actually this algo is not limited to only graph, it’s a set algorithm and has many useful applications.One more spplicstion I in job sequencing with deadlines-** [**http://www.geeksforgeeks.org/job-sequencing-using-disjoint-set-union/**](http://www.geeksforgeeks.org/job-sequencing-using-disjoint-set-union/)**. Next - https://github.com/mission-peace/interview/blob/master/src/com/interview/graph/DisjointSet.java**
9. **Check for bipartiteness of a graph**
10. **Minimum Spanning trees algos – Prims algo(greedy and a bit complex than Kruskal). Kruskal’s algo(extremely simple and cute, uses Disjoint set concept). Steps : 1. Sort edges in non-decreasing order. 2. Make |V| disjoint sets. 3. Start processing edges, if both ends of the edge are in the same disjoint set ignore the edge else add the edge in the list of mst edges and do the union of the disjoint sets which the ends of the edge are part of.**
11. **Floyd Warshal Algorithm for all-pair shortest path( DP based algo….simple implementation)**
12. **Graph coloring problem and its application on scheduling problems……chromatic number of a graph**
13. **Articulation points, bridges in a graph and their use in detecting vulnerabilities in a network.**
14. **Euler tour and Hamiltonian tour**
15. **Travelling Salesman Problem(TSP)**
16. **Check if a graph is strongly connected – Kosaraju algo**