1 Graph-Traversal

1.1 Breadth-First Search

Given a graph $G = (V, E)$, we want to discover all nodes a distance $dis(s, v)$. For BFS, we maintain a set S of all the nodes that we have travelled to. We push all of the vertices in Adj[i]. Also, we maintain a the parent of every node we find the shortest path of. Since we only place each node once in the set S, in total our algorithm will take $O(n + m)$.

1.2 Max trees

If we run BFS algo twice from an arbitrary node, the greatest dis[d] will be one of the points on the pair (o_1, o_2). Why? If there exists a longer path than (o_1, o_{i2}), then there has to be some contradiction or cycle, which is not allowed by the definition of a tree.

1.3 DFS-Visit

In DFS, we move through the most recent currently unexplored node. As soon as we find a node that is not yet explored, we mark it as "discovered", and immediately move to explore its edges. When we finish exploring every node, we mark it as "finished". However, DFS-visit only acts on the nodes reachable from s.

1.4 DFS

If we want to act on all the nodes of the graph, we need to consider the nodes not reachable from s.

For two nodes (u, v), if there is overlap between $[u.d, v.d]$and$[u.f, v.f]$, then one is the ancestor of the other. Why is DFS said to create a *forest*? Since not all the elements might be reached with just the initial call of DFS, we might need to call it on other nodes originally unreachable from our starting node.

1.5 Edges

Four types w.r.t. DFS-forest F

- **tree edges**: edges $(u, v) \in$ F.
- **back edges**: edges (u, v) connecting u to an ancestor.
- **forward edges**: edges (u, v) connecting node to descendant.
- **cross edges**: all other edges(b/w disjoint trees....)