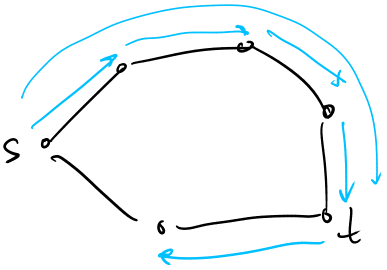
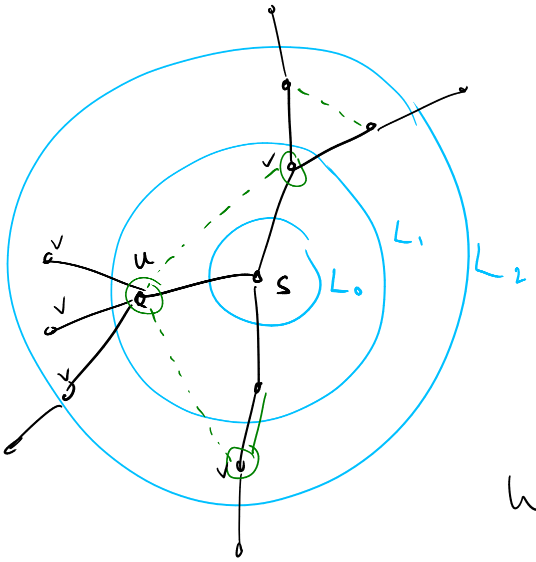
Q: What is the shortest path between vertices s, t?

DFS does not answer this Q:



Instead, we can use breadth-first search (BFS).

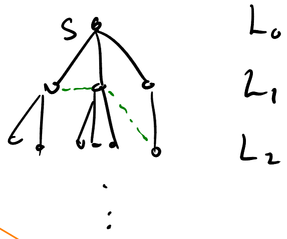


Properties:

① The shortest path from s to some vertex v has length k iff $v \in L_k$.

② For each edge (u, v), the layers of u and v differ by at most 1.

We can think of this as making a rooted tree:



```
BFS(G, s):
    Q ← empty queue
    [ parent ← empty dict ]
    [ layer ← empty dict  ]
    All vertices start UNVISITED
    Mark s VISITED
    Add s to Q
    layer[s] ← 0
    While Q is not empty:
        remove u from front of Q
        for each neighbor v of u:
            if v is UNVISITED:
                Mark v VISITED
                Add v to back of Q.
                parent[v] ← u.
                layer[v] ← layer[u] + 1.
    return parent, layer
```

invariant: anything in Q is VISITED, and has its layer filled in.

O(V + E), same as DFS.

Python: use deque (from collections import deque)

Java: ArrayDeque