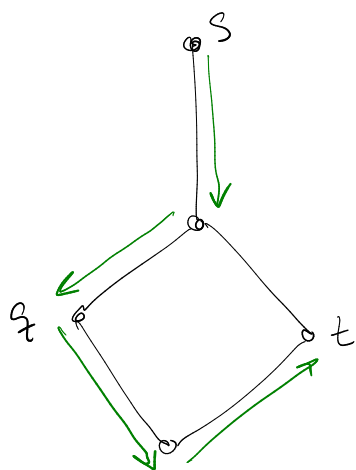
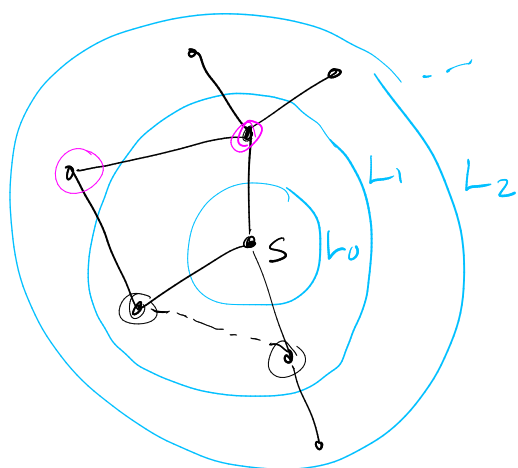


DFS cannot find shortest paths between vertices!



It will find some path,
but not necessarily the shortest.

Breadth-first search (BFS)



$$L_0 = \{s\}$$

L_k = all vertices adjacent
to vertex in L_{k-1}
that were not already
in a previous layer.

Claims:

- ① Shortest path $s \rightarrow v$ for any vertex v has length k
iff $v \in L_k$.
- ② For any edge $\{u, v\}$, layers of u, v differ by at most 1.

BFS(G, s):

Mark all vertices unvisited

$Q \leftarrow$ empty queue

Mark s visited

$Q.enqueue(s)$

invariant:
every vertex in Q
is visited.

[$parent \leftarrow$ empty dict
 $layer \leftarrow$ empty dict
 $layer[s] \leftarrow 0$.

while Q is not empty: $\leftarrow$

$u \leftarrow Q.dequeue$ $\leftarrow$

 for all v adjacent to u :

 if v is not visited:

 Mark v visited

$Q.enqueue(v)$

$parent[v] \leftarrow u$

$layer[v] \leftarrow layer[u] + 1$

By similar
argument as DFS—
takes $O(V+E)$
time.

Java: Queue / ArrayDeque

Python: from collections import deque