Breadth-First Search

→ BFS traversal is more like sending many explorers out in all directions to traverse a graph in a coordinated fashion

→ BFS proceeds in rounds and subdivides the vertices into levels
- starts at vertex s at level 0
- in first round, mark as 'visited', all vertices adjacent to start vertex -- placed in level one
- in 2nd round, allow all explorers to go two steps/edges from vertex
  → placed in 2nd level and marked
- process continues until no new vertices found in a level

```
def BFS(g, s, discovered):
  level = [s]                          // first level includes only s
  while len(level) > 0:
    next_level = []
    for u in level:
      for e in incident_edges(u):      // for every outgoing edge
        v = e.opposite(u)
        if v not in discovered:        // v is unvisited
          discovered[v] = e            // e is tree edge which disc. v
          next_level.append(v)         // v further considered in next pass
    level = next_level                 // relabel 'next' level to become current
```

→ **Proposition:** Let G be undirected graph or directed graph on which a BFS traversal starting at vertex s has been performed. Then:
- traversal visits all vertices of G that are reachable from s
- for each vertex v at level i, path of BFS tree T btwn. s and v has i edges, any other path of G from s to v has at least i edges
- if (u,v) is edge not in BFS tree, the level number of v can be at most 1 greater than level number of u

→ **Proposition:** Let G be a graph w/ n vertices and m edges represented w/ adjacency list. A BFS traversal takes $O(n+m)$ time.