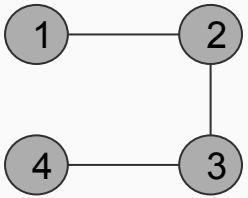
Adjacency Lists

Adjacency Lists

- Working with adjacency lists is actually exactly like matrices, and in some ways they're even easier.
- We do not need to worry about going out of bounds anymore.
- We still need to keep track of visited nodes, especially if we are working with an undirected graph.
- Demo
    - Has Path
      Shortest Path

What if I need to construct the adjacency list myself?

- Sometimes we aren't given an adjacency list, and instead we are given a list of edges.
  - Example:
    - edges = [[1,2], [2,3], [3,4]]
    - adjList ={1: [2],
      2: [1,3]
      3: [2,4]
      4: [3]}




```
const constructGraph = (edges) => {
  const graph = {};
  for (let edge of edges) {
    const node1 = edge[0];
    const node2 = edge[1];
    if (!graph[node1]) graph[node1] = new Set();
    if (!graph[node2]) graph[node2] = new Set();
    graph[node1].add(node2);
    graph[node2].add(node1);
  }
  return graph;
}
```

Adjacency List DFS Example

```
const hasPath = (graph, src, dst) => {
  if (src === dst) return true;
  for (let neighbor of graph[src]) {
    if (hasPath(graph, neighbor, dst) === true) return true;
  }
  return false;
}
```

Test Case:

```
const graph = {
  f: ['g', 'i'],
  g: ['h'],
  h: [],
  i: ['g', 'k'],
  j: ['i'],
  k: []
};

hasPath(graph, 'f', 'j'); // false
```

Adjacency List BFS Example

```
const shortestPath = (edges, nodeA, nodeB) => {
  const graph = constructGraph(edges);
  const visited = new Set();
  const queue = [{val: nodeA, level: 0}];

  while (queue.length > 0) {
    const current = queue.shift();
    visited.add(current.val)
    if (current.val === nodeB) return current.level;
    for (let neighbor of graph[current.val]) {
        if (!visited.has(neighbor)) {
              queue.push({val: neighbor, level: current.level+1});
        }
    }
  }
  return -1;
};
```

```
const constructGraph = (edges) => {
  const graph = {};

  for (let edge of edges) {
    const node1 = edge[0];
    const node2 = edge[1];
    if (!graph[node1]) graph[node1] = new Set();
    if (!graph[node2]) graph[node2] = new Set();
    graph[node1].add(node2);
    graph[node2].add(node1);
  }
  return graph;
}
```

Questions?

Let’s practice!

- Review
  - Course Schedule
  - Is Graph Bipartite?
- Bonus
  - Find Eventual Safe States
  - Course Schedule II