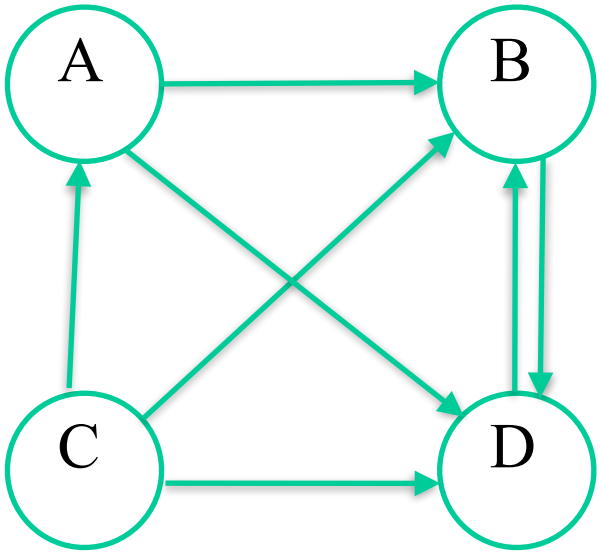
# Graphs

# Graphs

- Consist of nodes and edges
  - edges may be
    - weighted or unweighted
    - Directed or undirected
- No distinguished starting location
- Loops allowed




A graph with 4 nodes and unweighted, directed edges

# Adjacency Lists

- Each node holds list of edges leaving the node
  - Add an ArrayList of edges to the node definition
- Edge need only store destination
- How do you store bi-directional links?

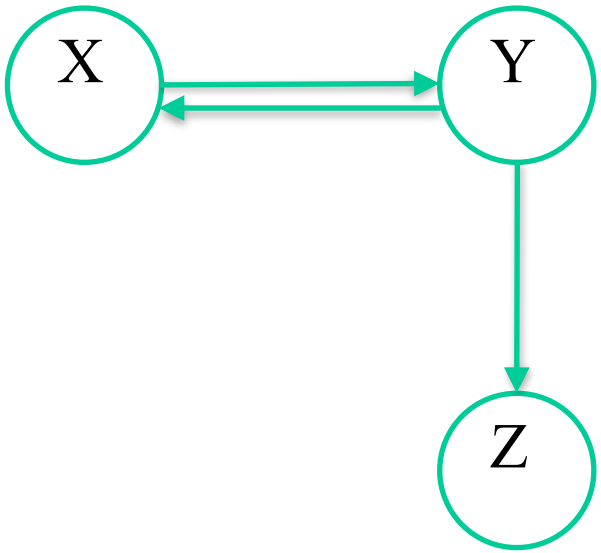
```
private class Node<H> {
    // Node content
    public H payload;
    // hold the list
    public ArrayList<Edge<G>> edges;

    public Node(H payl) {
        this.payload = payl;
        this.edges = new ArrayList<Edge
    }

    public void addEdge(Node<G> n, doub
        edges.add(new Edge<G>(n, w));
    }
```

# Graph Navigation

- Can I get from Node X to Node Z?
  - Adj List representation?



# Path Exists

```
boolean pathExists(Starting, ending)
  Stack s <- new Stack
  add staring point to stack
  while stack not empty
     n <- pop stack
     if n is destination
        return true
     With each edge from n
        add end to stack
   return false
```

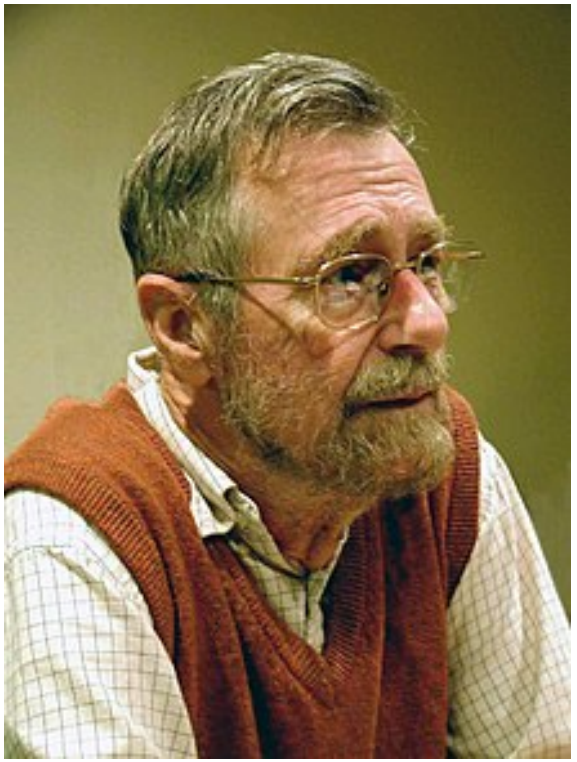
- Problem: loops
  - How to handle?
- A “depth first” traversal

# Shortest unweighted path

- Change path exists to from stack to Queue
  - need to store paths

# Shortest Weighted Path

- Edsger W. Dijkstra
- "Dijsktra's shortest path algorithm" (1956)
- non-negative weights
- A "greedy" algorithm
  - Do the best thing you can based on local info and hope you get a global best.



1930-2002
PhD in CS (1959)
Pioneered structured programing
Seminal work in distributed computing
Curmudgeon

# Finding Groups

- Suppose undirected links
- Question: Identify groups
  - A group is all the nodes in a graph that can be reached each other

- How does this problem change when you have directed links?