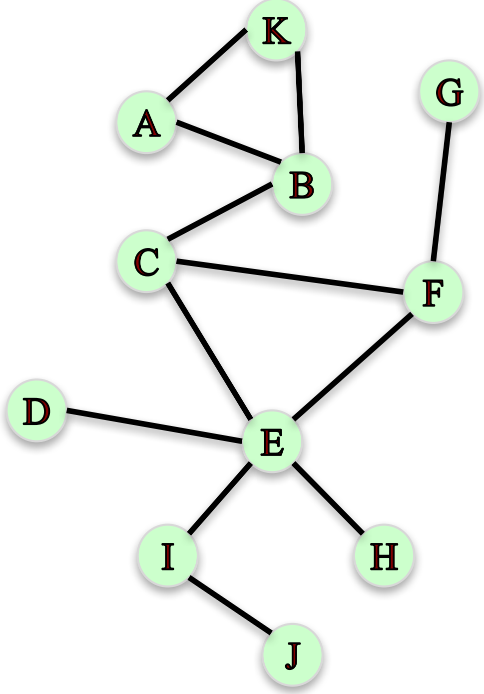
Distance

How "far" is node A from node H?

Are nodes far away or close to each other in this network?

Which nodes are "closest" and "farthest" to other nodes?

We need a sense of distance between nodes to answer these questions



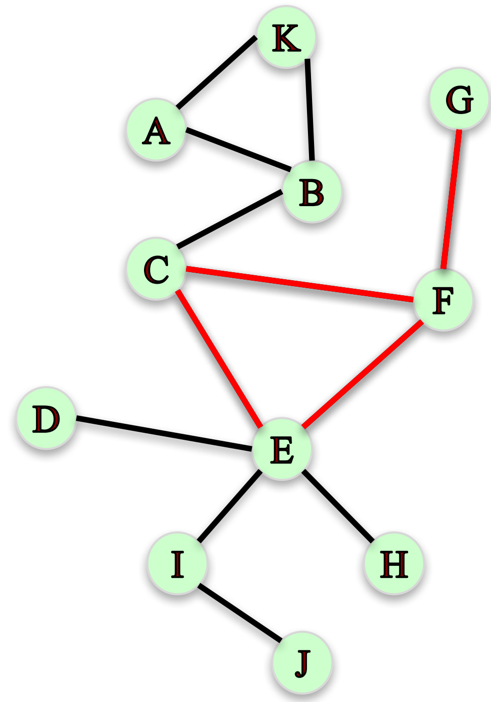
Paths

Path: A sequence of nodes connected by an edge.

Find two paths from node G to node C:

G – F – C

G – F – E – C



Distance

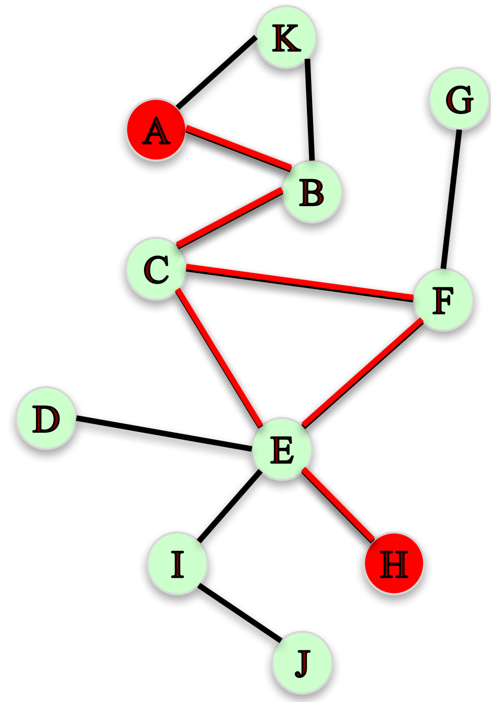
How far is node A from node H?

Path 1: A – B – C – E – H (4 "hops")

Path 2: A – B – C – F – E – H (5 "hops")

Path length: Number of steps it contains from beginning to end.

Path 1 has length 4, Path 2 has length 5



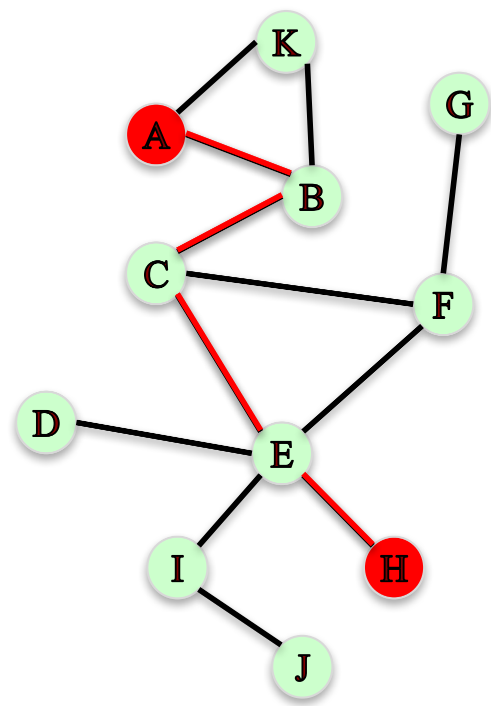
Distance

Distance between two nodes: the length of the shortest path between them.

The distance between node A and H is 4

In: nx.shortest_path(G,'A', 'H')
Out: ['A', 'B', 'C', 'E', 'H']

In: nx.shortest_path_length(G,'A', 'H')
Out: 4

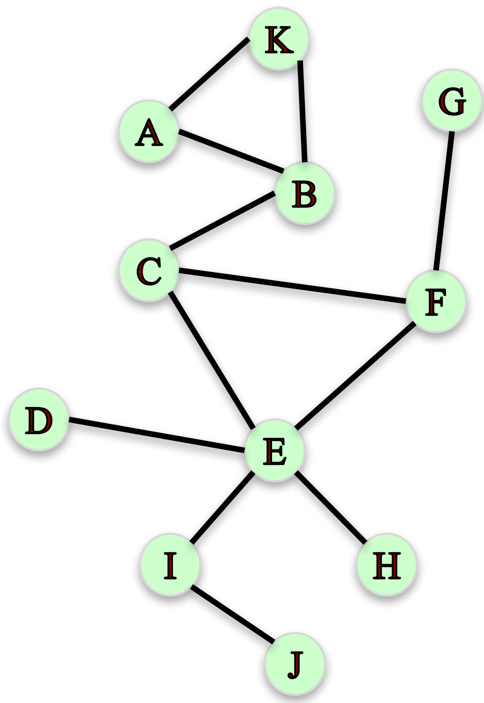


Distance

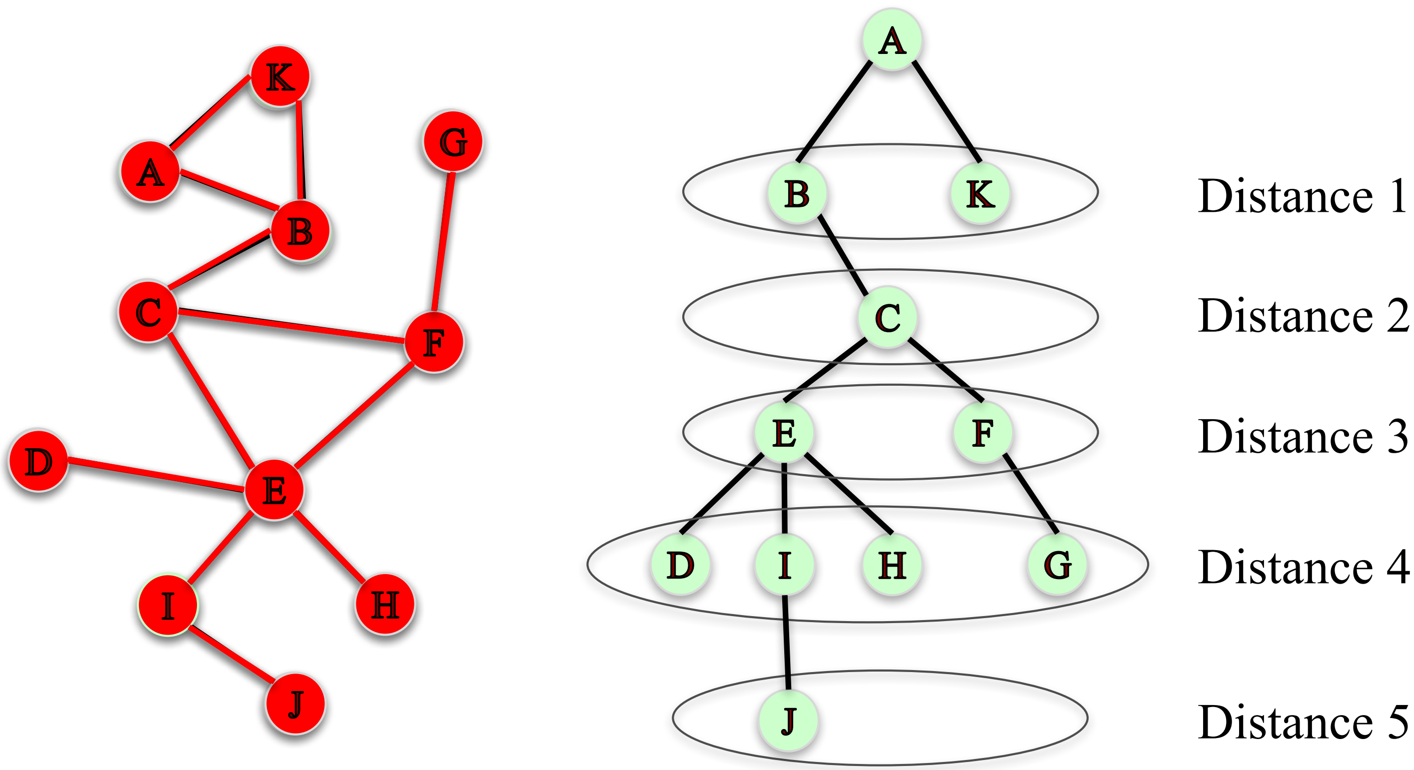
Finding the distance from node A to every other node.

Easy to do manually in small networks but tedious in large (real) networks.

Breadth-first search: a systematic and efficient procedure for computing distances from a node to all other nodes in a large network by "discovering" nodes in layers.



Breadth-First Search



Breadth-First Search

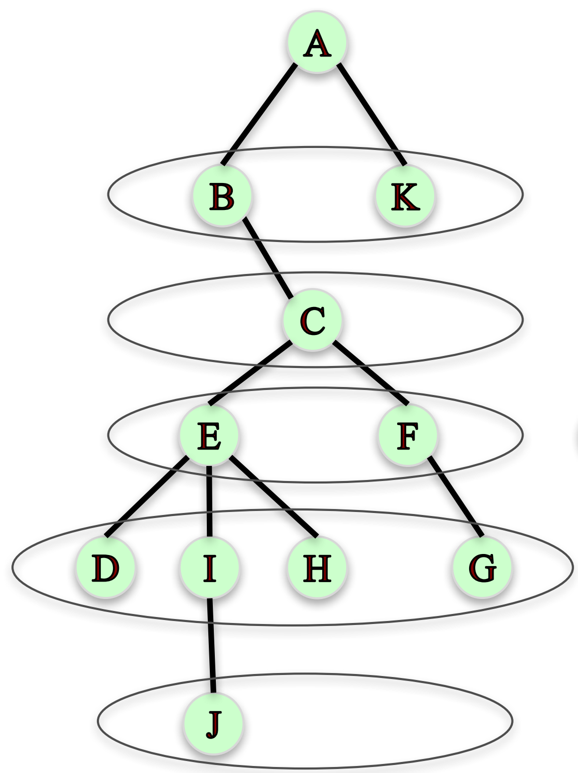
In: T = nx.bfs_tree(G, 'A')

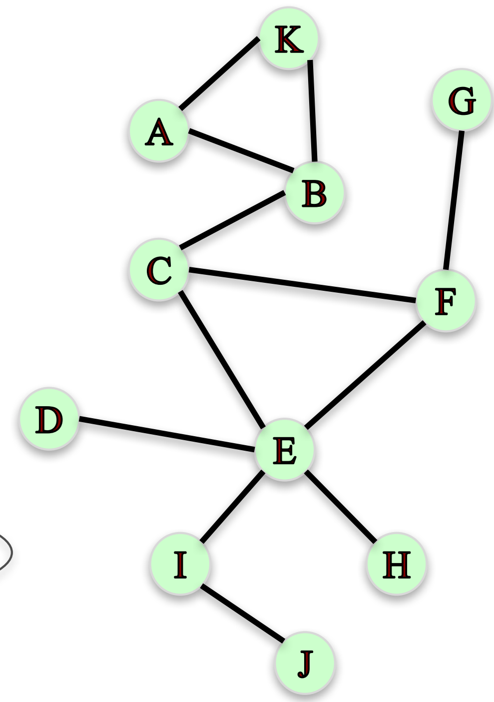
In: T.edges()

Out: [('A', 'K'), ('A', 'B'), ('B', 'C'), ('C', 'E'), ('C', 'F'), ('E', 'I'), ('E', 'H'), ('E', 'D'), ('F', 'G'), ('I', 'J')]

In: nx.shortest_path_length(G,'A')

Out: {'A': 0, 'B': 1, 'C': 2, 'D': 4, 'E': 3, 'F': 3, 'G': 4, 'H': 4, 'I': 4, 'J': 5, 'K': 1}





Distance Measures

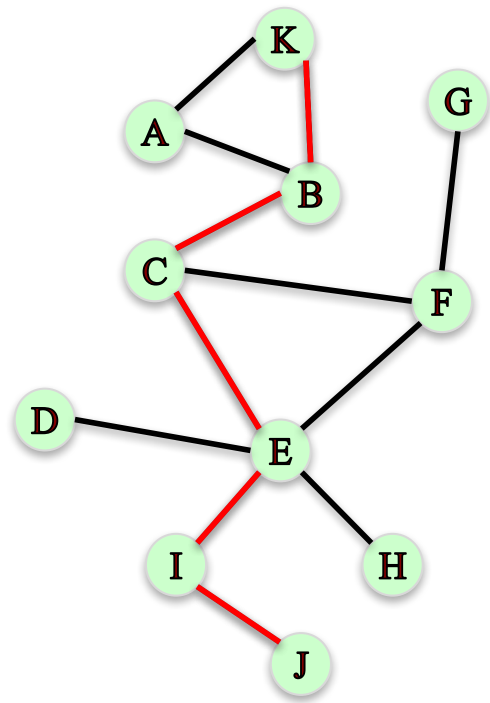
How to characterize the distance between all pairs of nodes in a graph?

Average distance between every pair of nodes.

In: nx.average_shortest_path_length(G)
Out: 2.52727272727

Diameter: maximum distance between any pair of nodes.

In: nx.diameter(G)
Out: 5



Distance Measures

How to summarize the distances between all pairs of nodes in a graph?

The **Eccentricity** of a node *n* is the largest distance between *n* and all other nodes.

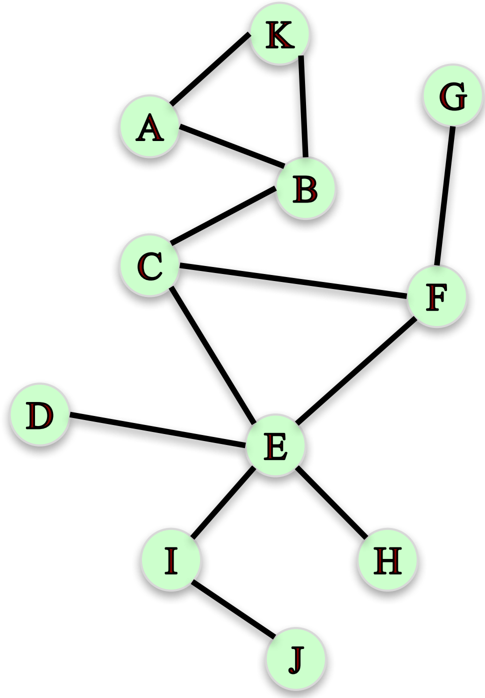
In: nx.eccentricity(G)

Out: {'A': 5, 'B': 4, 'C': 3, 'D': 4, 'E': 3, 'F': 3, 'G': 4, 'H': 4, 'I': 4, 'J': 5, 'K': 5}

The **radius** of a graph is the minimum eccentricity.

In: nx.radius(G)

Out: 3



Distance Measures

How to summarize the distances between all pairs of nodes in a graph?

The **Periphery** of a graph is the set of nodes that have eccentricity equal to the diameter.

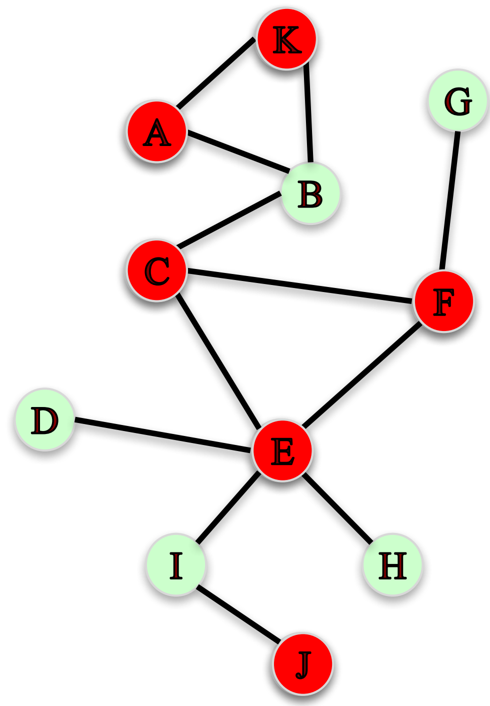
In: nx.periphery(G)

Out: ['A', 'K', 'J']

The **center** of a graph is the set of nodes that have eccentricity equal to the radius.

In: nx.center(G)

Out: ['C', 'E', 'F']



Karate Club Network

```
G = nx.karate_club_graph()
G = nx.convert_node_labels_to_integers(G,first_label=1)
```

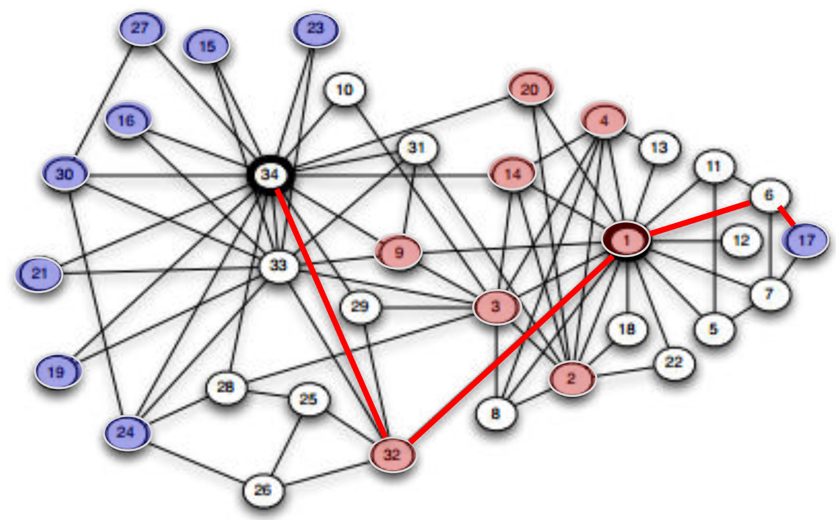
Average shortest path = 2.41

Radius = 3

Diameter = 5

Center = [1, 2, 3, 4, 9, 14, 20, 32]

Periphery: [15, 16, 17, 19, 21, 23, 24, 27, 30]



Friendship network in a 34-person karate club

Node 34 looks pretty "central". However, it has distance 4 to node 17

Summary

Distance between two nodes: length of the shortest path between them.

Eccentricity of a node *n* is the largest distance between *n* and all other nodes.

Characterizing distances in a network:

Average distance between every pair of nodes.

Diameter: maximum distance between any pair of nodes.

Radius: the minimum eccentricity in the graph.

Identifying central and peripheral nodes:

The **Periphery** is the set of nodes with eccentricity = diameter.

The **center** is the set of nodes with eccentricity = radius.