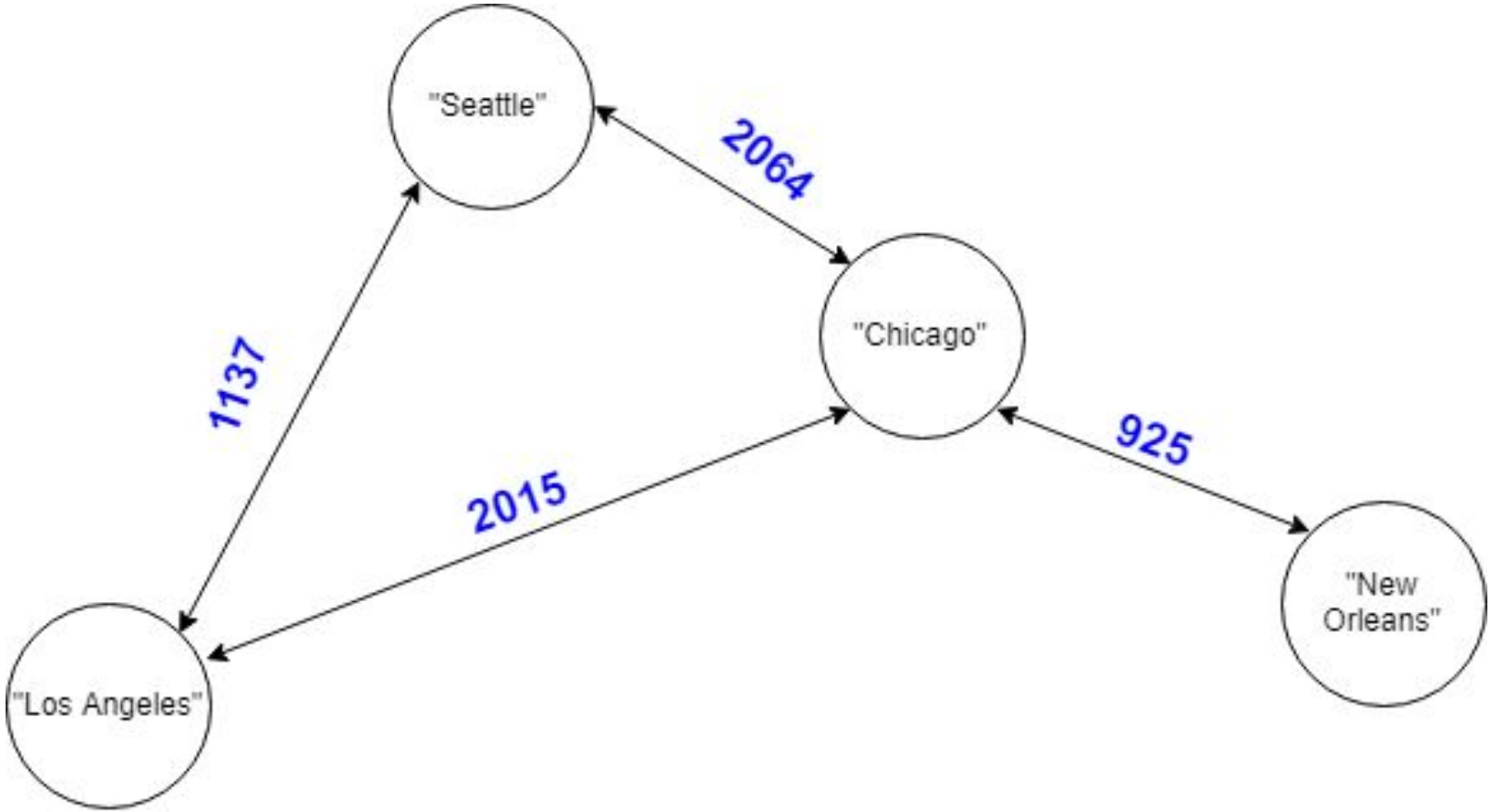
9.2 Social Network

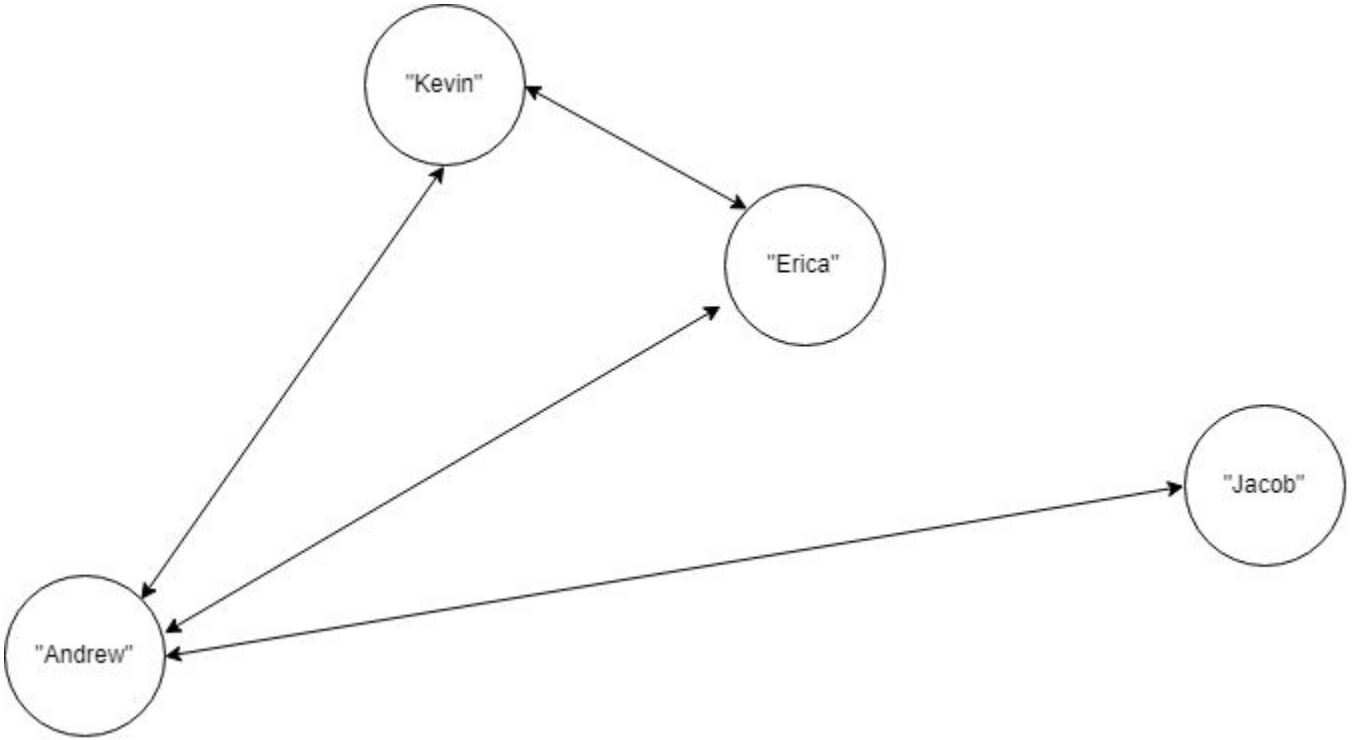
How would you design the data structures for a social network like Facebook? Describe how you would design an algorithm to show the shortest path between two people (e.g., Me-> Bob-> Susan-> Jason-> You).

Reference:
https://tianrunhe.wordpress.com/2012/04/08/design-the-data-structure-for-large-social-network/

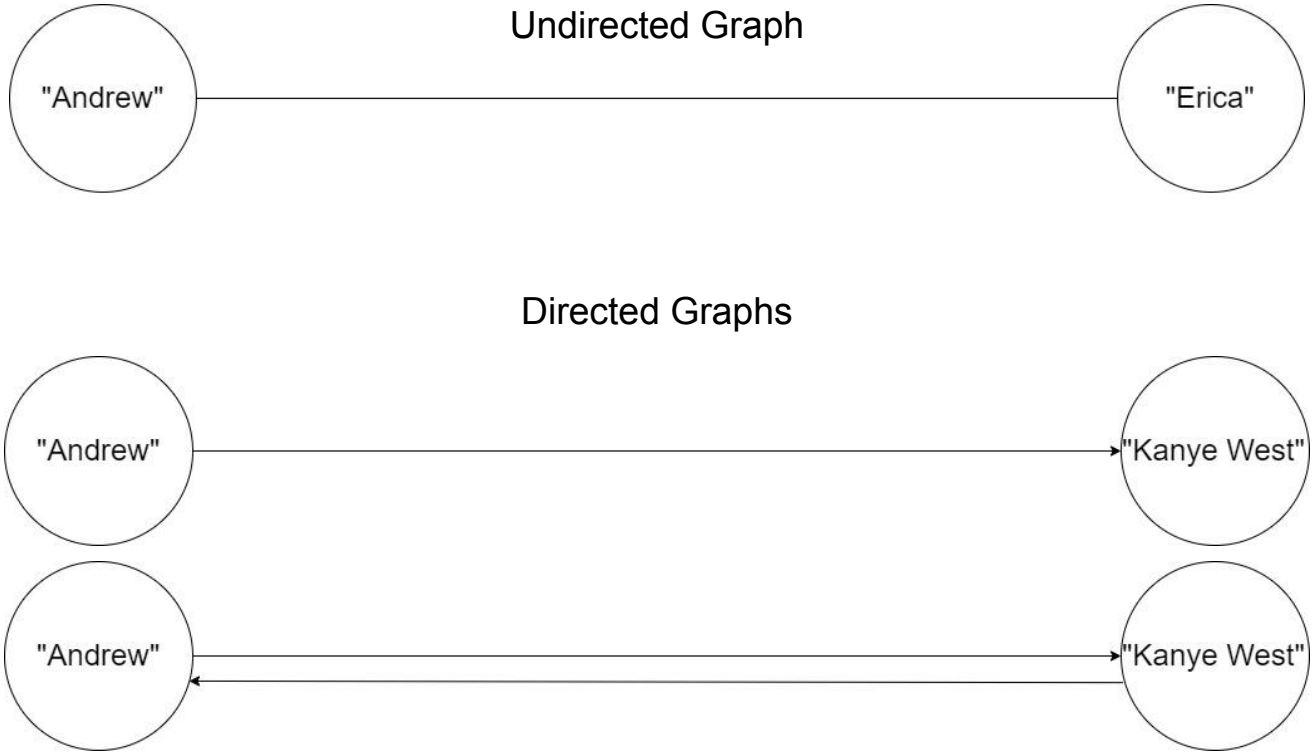
Using a Graph to represent a Map



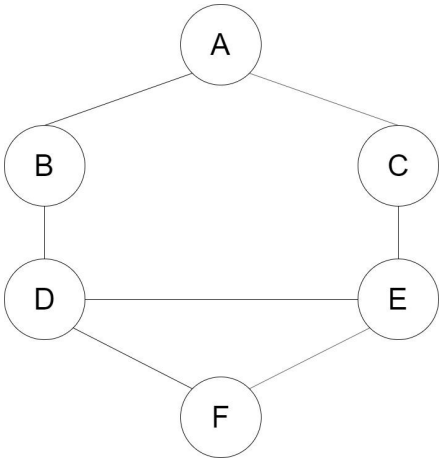
Using a Graph to represent a Social Network



Undirected versus Directed Graph Recap



DFS

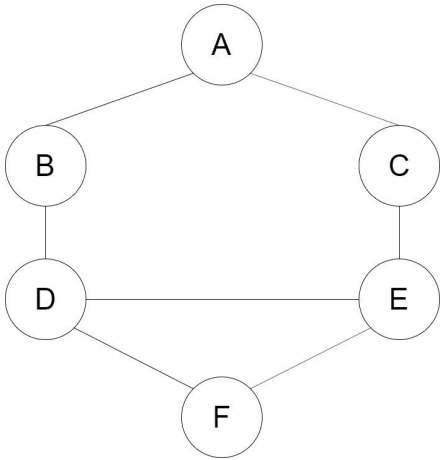


Possible DFS Results:

[A, C, E, F, D, B]

[A, B, D, E, C, F]

BFS



BFS Result:

[A, B, C, D, E, F]

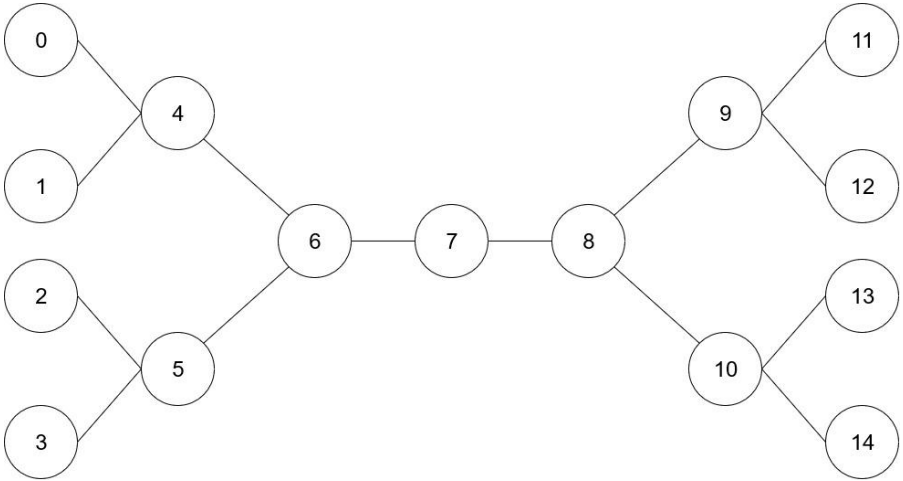
9.2 Social Network

We will use a modified BFS instead of a DFS to show the shortest path between two people.

Why not a DFS? DFS would just find a path, and not necessarily the shortest path. Two users might be one degree of separation apart, but it could search millions of nodes in their subtrees before finding this immediate connection.

We will do a bidirectional BFS. This means doing two simultaneous BFS', one from the source and one from the destination. When the searches collide, we have found a path.

Bidirectional BFS



Breadth First Search

Single search from 0 to 14 that collides after six levels.

Bidirectional Search

Two searches (one from 0 and one from 14) that collide after six levels total (three levels each).

Bidirectional BFS

Suppose every person has k friends, and our desired path of length q.

Time Complexity
BFS: $O(k^q)$
Bidirectional BFS: $0(k^{q/2} + k^{q/2}) \rightarrow 0(k^{q/2})$

The difference between $O(k^q)$ and $0(k^{q/2})$ is HUGE.
The difference between 10^10 and 10^5 is HUGE.

However, bidirectional BFS requires having access to both the source and destination node, which is not always possible.