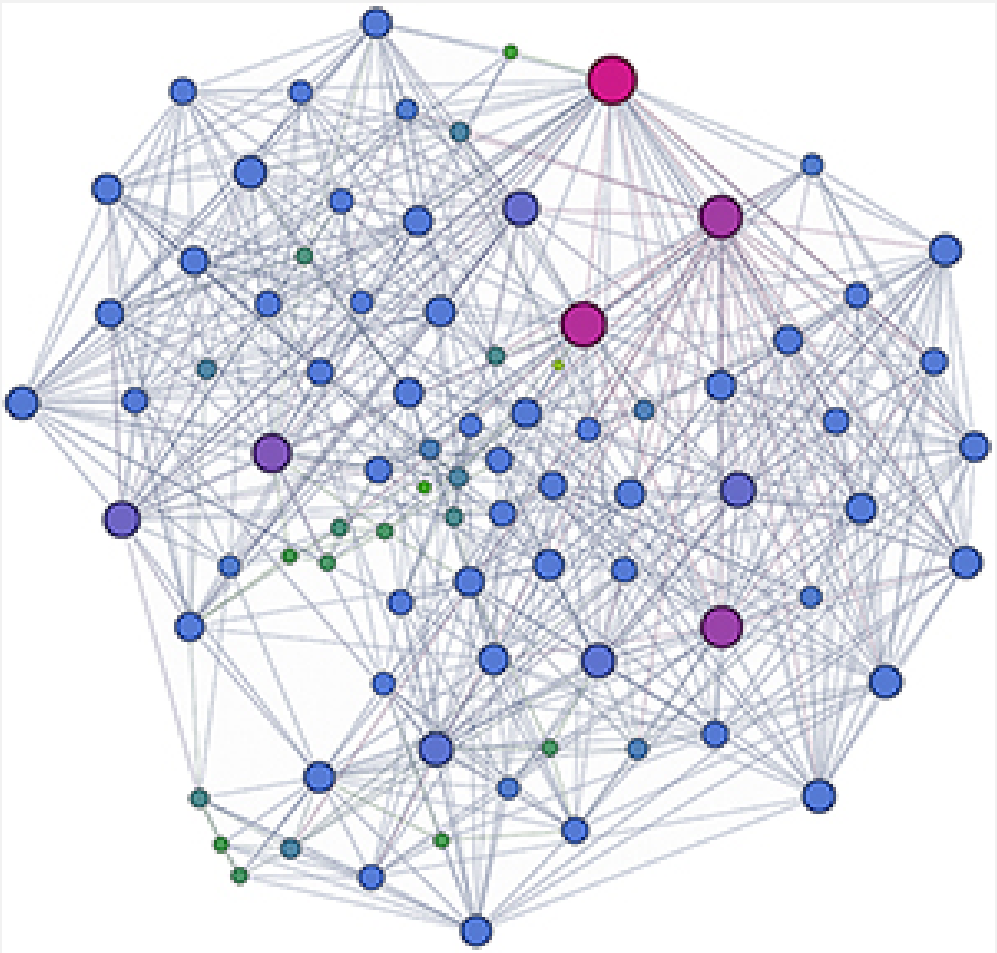
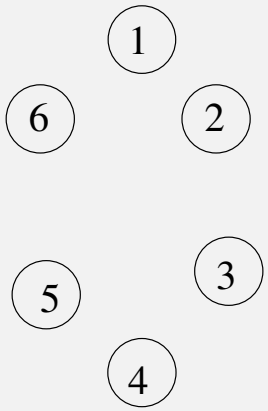
Graph Data Structure for Big Data



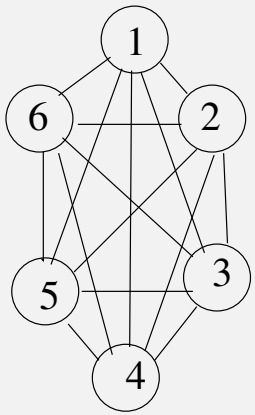
Definitions

- A graph $G = (V, E)$, is defined as a set of vertices V and a set of edges E that connect the vertices. Let the number of vertices $\mid V \mid = n$ and the number of edges $\mid E \mid = m$.
- A network is made up on nodes and links between the nodes. A graph can be seen as an abstract representation that includes networks, where vertices represent nodes and edges represent links.
- The vertices of a graph can represent servers in a distributed system, servers on the internet, cities in a road map, users in a friend network and so on. The edges represent connections between the vertices.
- Graphs can be undirected or directed, based on whether edges have direction or not.
- Graphs can have unweighted edges (we assume a weight of 1 for each edge in this case) or have weighted edges. A graph with weighted edges is known as a weighted graph.

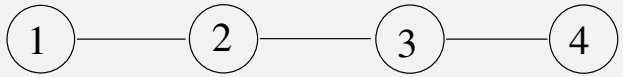
Examples of Undirected Graphs (1)



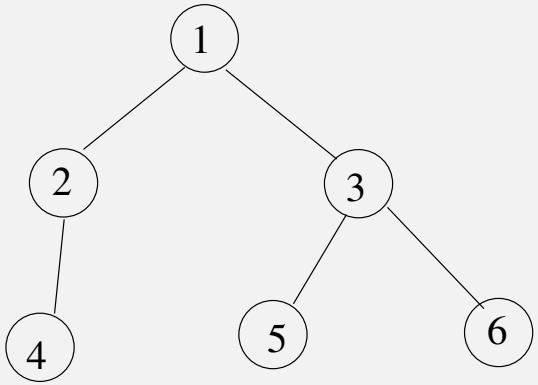
An Empty Graph



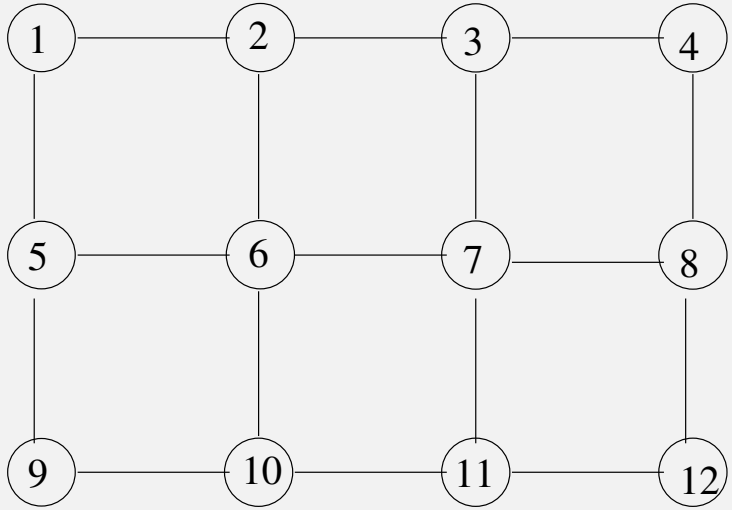
A Complete Graph



A List is a graph

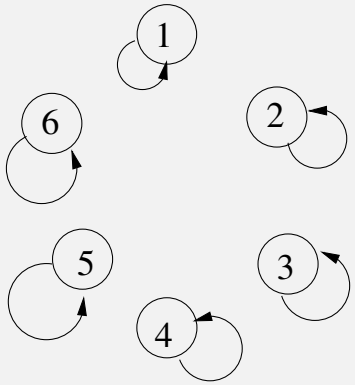


A Tree is a graph

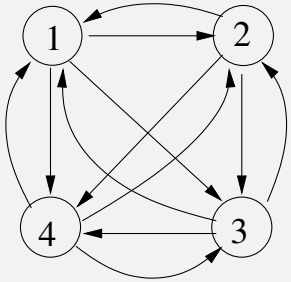


A grid or city map is a graph!

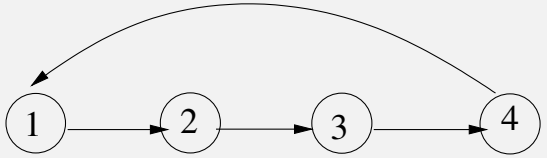
Examples of Directed Graphs (1)



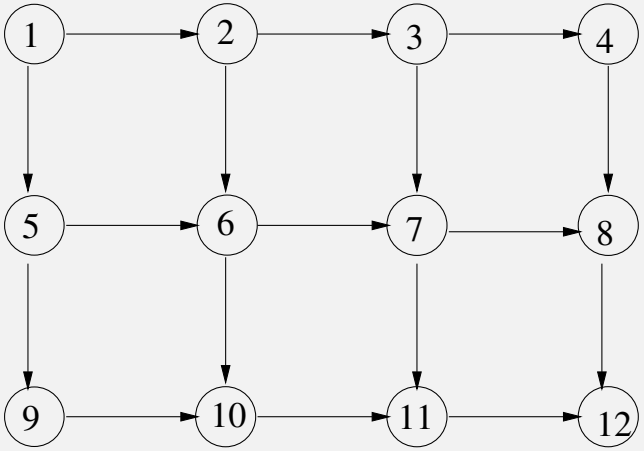
A party of narcissists



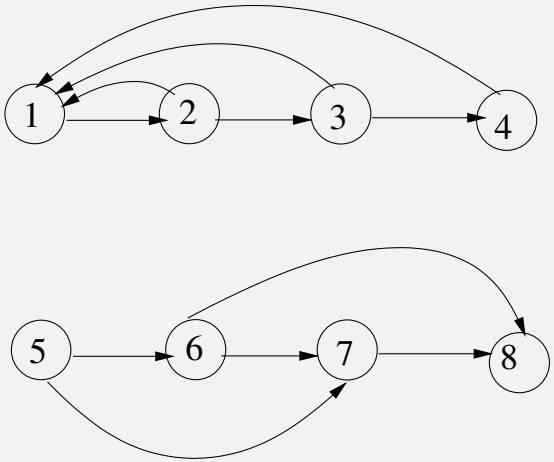
A Complete Graph



A circular list is a graph

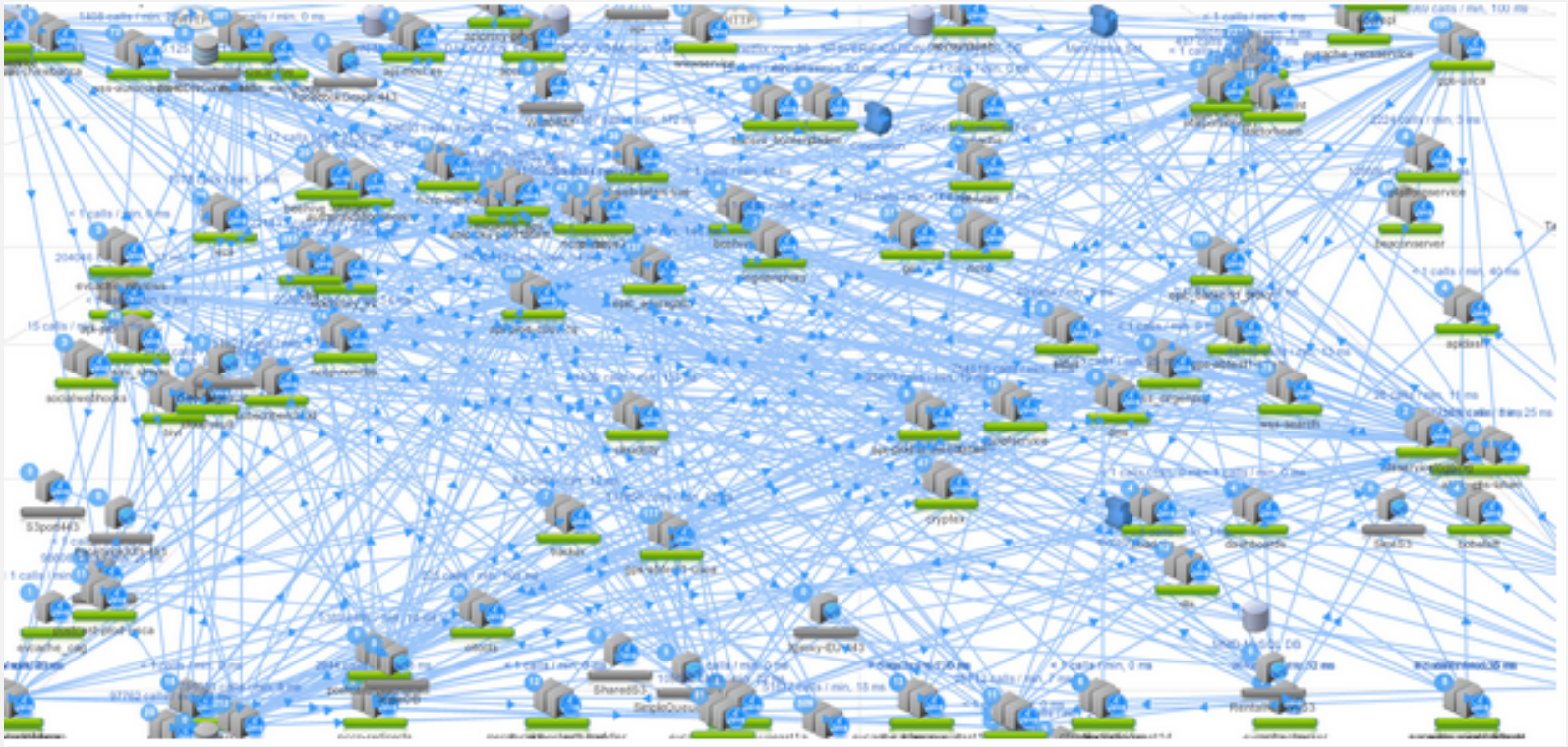


All roads lead to 12!

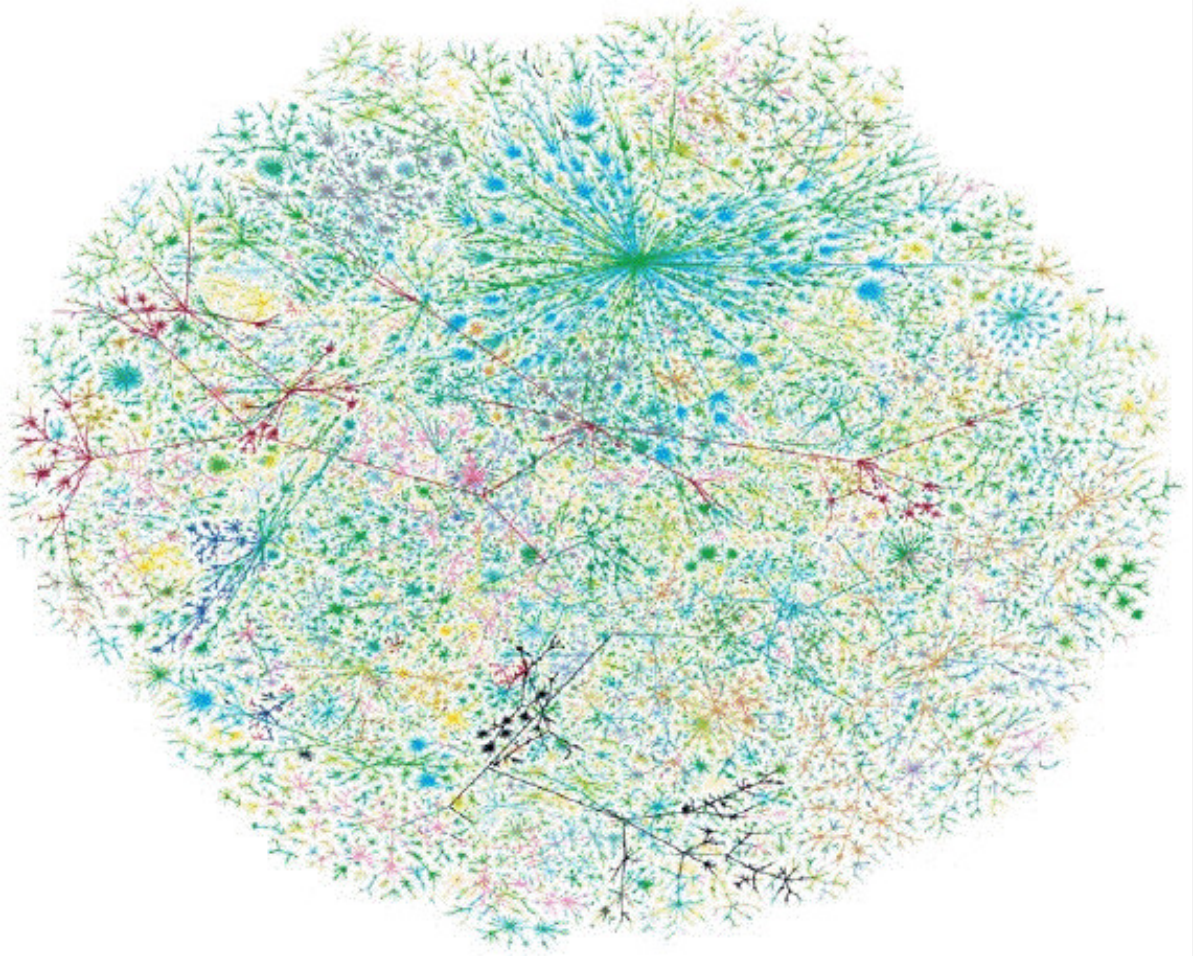


A graph with two connected components

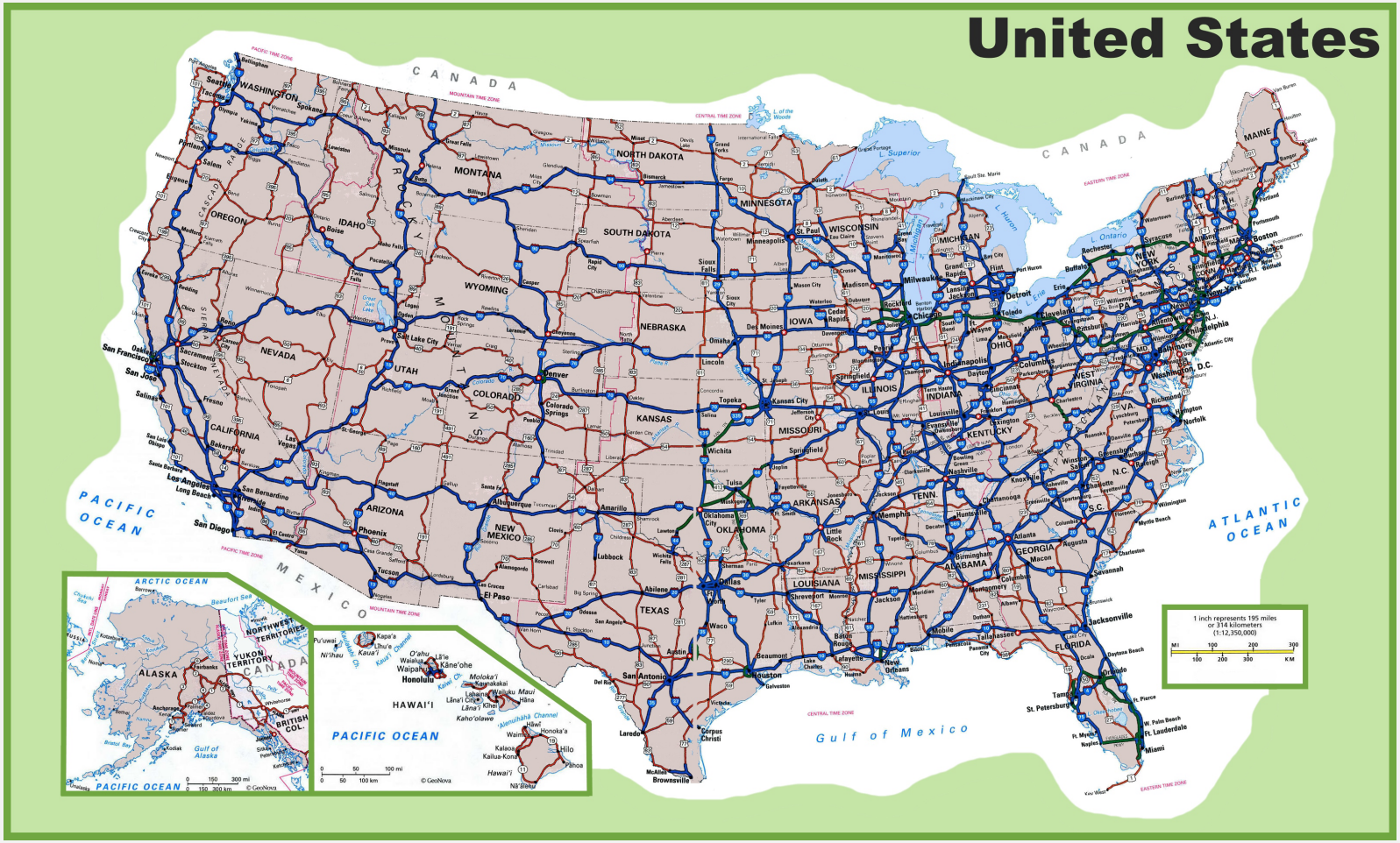
A graph of Netflix servers!



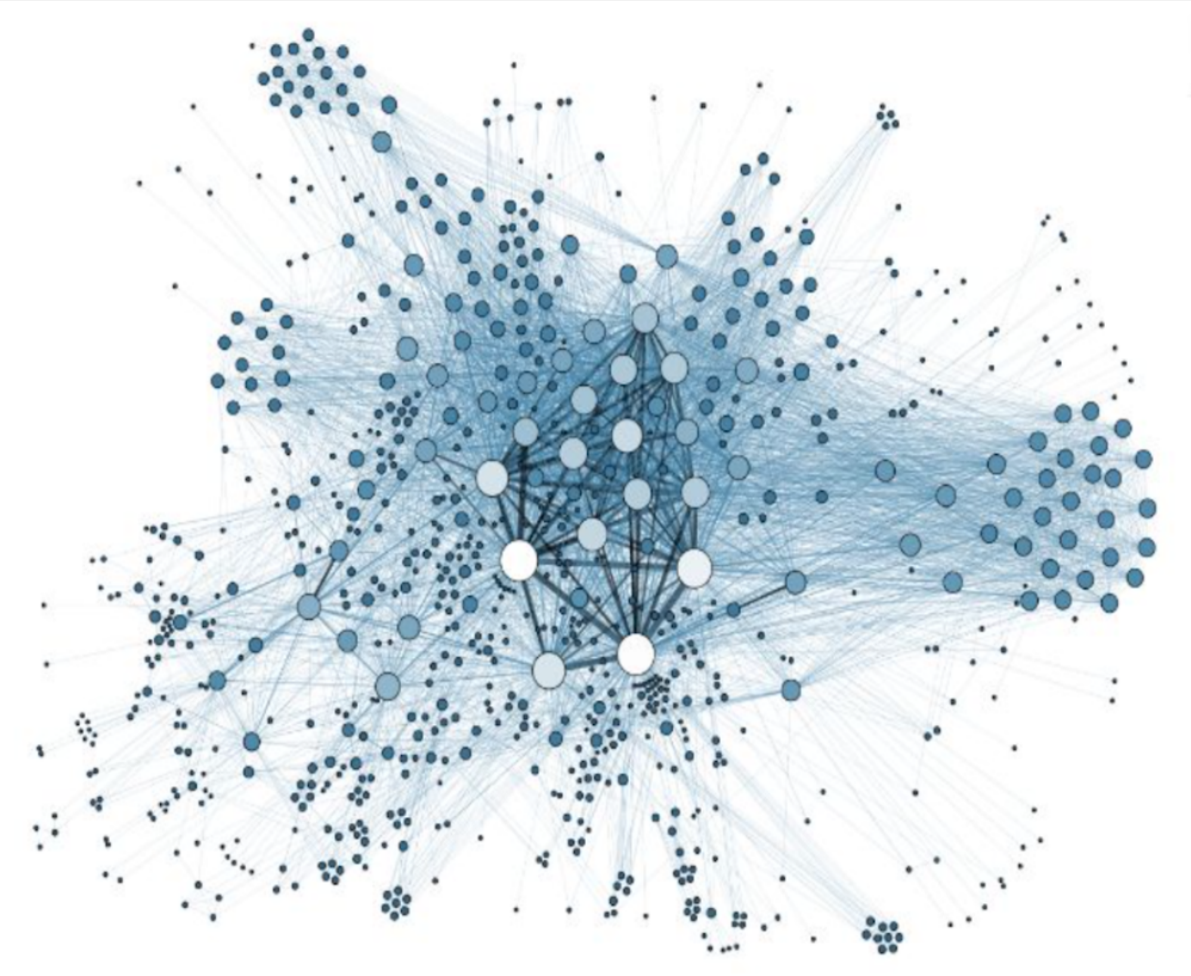
A graph representing the Internet



A graph representing the US road map



Facebook Friend Graph



Graph Algorithms using Map-Reduce

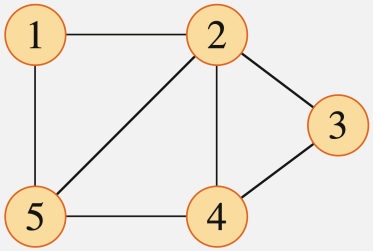
Graphs are ubiquitous in modern society. Some examples:

- Transportation networks (roads, trains, flights etc)
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)
- The hyperlink structure of the web
- Human body can be seen as a graph of genes, proteins, cells etc

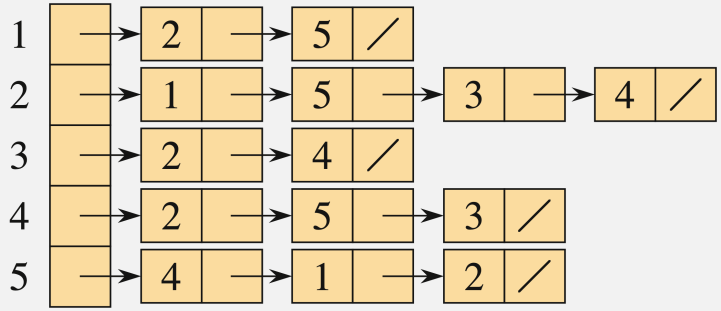
Typical graph problems and algorithms:

- Graph search and path planning
- Graph clustering
- Minimum spanning trees
- Bipartite graph matching
- Maximum flow
- Finding "special" nodes

Representing Undirected Graphs: Adjacency Lists and Adjacency Matrix



(a)

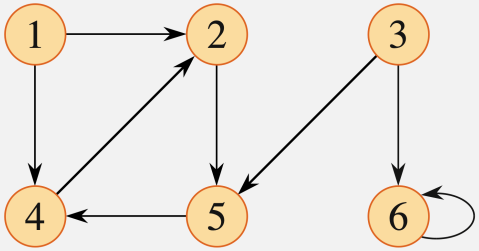


(b)

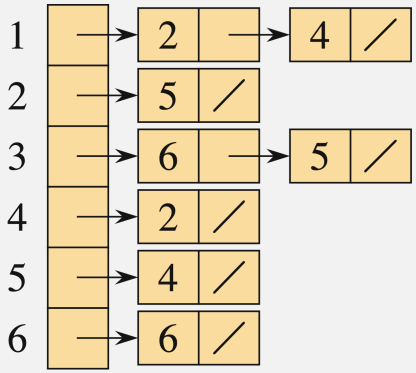
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

Representing Directed Graphs: Adjacency Lists and Adjacency Matrix



(a)



(b)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

Big Graphs

- Big graphs are typically sparse so adjacency list representation is much more space efficient. Typical space requirement may be $m = O(n)$, where m is the number of links and n is the number of nodes in the graph.
- Example: Facebook has around 2.7 billion users but each user, on an average, may only have few hundred friends, so the graph is very sparse.

Parallel Breadth-First Search (1)

- Assume that all links have unit distance for simplification. Assume that graph is connected.
- We are interested in finding the shortest distance from a source vertex to all other vertices. Since the distances are all one, this is the same as a breadth-first search.
- The largest shortest distance (starting from any node) is known as the diameter of the graph.
- Each node is represented by a node id n (an integer), its current distance (initialized to ∞) and its adjacency list data structure N. The source vertex has its current distance initialized to be 0.

Parallel Breadth-First Search (2)

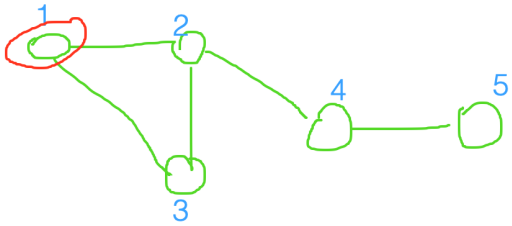
- Each mapper emits a key-value pair for each neighbor on the node's adjacency list. The key contains the node id of the neighbor, and the value is the current distance to the node plus one.
- After shuffle and sort, reducers will receive keys corresponding to the destination node ids and distances corresponding to all paths leading to that node.The reducer will select the shortest of these distances and then update the distance in the node data structure.
- Each iteration of the map-reduce algorithm expands the "search frontier" by one hop, and, eventually, all nodes will be discovered with their shortest distances (assuming a fully-connected graph).

Parallel Breadth-First Search Pseudo-code

```
1: class Mapper
2:     method Map(nid n, node N)
3:         d ← N.Distance
4:         Emit(nid n, N)                          ▷ Pass along graph structure
5:         for all nodeid m ∈ N.AdjacencyList do
6:             Emit(nid m, d + 1)                  ▷ Emit distances to reachable nodes

1: class Reducer
2:     method Reduce(nid m, [d1, d2, ...])
3:         d_min ← ∞
4:         M ← ∅
5:         for all d ∈ counts [d1, d2, ...] do
6:             if IsNode(d) then
7:                 M ← d                           ▷ Recover graph structure
8:             else if d < d_min then              ▷ Look for shorter distance
9:                 d_min ← d
10:        M.Distance ← d_min                      ▷ Update shortest distance
11:        Emit(nid m, node M)
```

Parallel Breadth-First Search Example



1 —> 2,3. (1, (0, (2,3)))
2 —> 1,3,4. (2, (x, (1,3,4)))
3 —> 1,2. (3, (x, (1,2)))
4 —> 2,5. (4, (x, (2,5)))
5 —> 4. (5, (x, (4)))

Map round 1

(1, (0, (2,3))) (2,1) (3,1). (2, (x, (1,3,4))) (1,x) (3, x) (4,x) (3, (x, (1,2))) (1,x) (2,x). (4, (x, (2,5))) (2,x) (5,x) (5, (x,(4))) (4,x)

Reduce round 1

((1,x)(1,x),(1,(0, (2,3))). (2,1)(2, (x, (1,3,4))) (2,x). (3, (x, (1,2)))(3,1)(3,x). (4, (x, (2,5)))(4,x)(4,x). (5, (x,(4)))(5,x)

(1, (0, (2,3))) (2, (1,(1,3,4))) (3, (1, (1,2))) (4, (x, (2,5))) (5, (x,(4)))

Map round 2

(1, (0, (2,3))) (2,1)(3,1). (2, (1,(1,3,4)))(1,2)(3,2)(4,2) (3, (1, (1,2)))(1,2)(2,2). (4, (x, (2,5)))(2,x)(5,x). (5, (x,(4)))(4,x)

Reduce round 2

(1, (0, (2,3)))(1,2)(1,2). (2, (1,(1,3,4)))(2,1)(2,2) (3, (1, (1,2)))(3,1)(3,2). (4, (x, (2,5)))(4,2)4,x). (5, (x,(4)))(5,x)

(1, (0, (2,3))) (2, (1,(1,3,4))) (3, (1, (1,2))) (4, (2, (2,5))) (5, (x,(4)))

Implementation Tips

- Note that in this algorithm we are overloading the value type, which can either be a distance (integer) or a complex data structure representing a node. This can be done in Hadoop by creating a wrapper object with an indicator variable specifying the type of the content.
- What is a more object-oriented way of doing the above? By creating an abstract class with two sub-classes.
- Since the graph is connected, all nodes are reachable, and since all edge distances are one, all discovered nodes are guaranteed to have the shortest distances (i.e., there is not a shorter path that goes through a node that hasn't been discovered).
- Global Hadoop counters can be defined to count the number of nodes that have distances of ∞. At the end of the job, the driver program can access the final counter value and check to see if another iteration is necessary.
- Most real-life graphs have a small diameter. Search for "six degrees of freedom" or Facebook's experiment that showed 4.74 degrees of freedom over a large set of users.

Shortest Paths in Weighted Graphs

- Instead of $d + 1$, the mapper now emits $d + w$, where w is the weight of the edge.
- Termination will be different: The algorithm can terminate when shortest distances at every node no longer change. The worst-case is that the number of iterations equals the number of nodes. But most real-life graph will terminate for iterations somewhere close to the diameter of the graph.

References

- Chapter 5 in *Data-Intensive Text Processing with MapReduce* by Jimmy Lin and Chris Dyer.