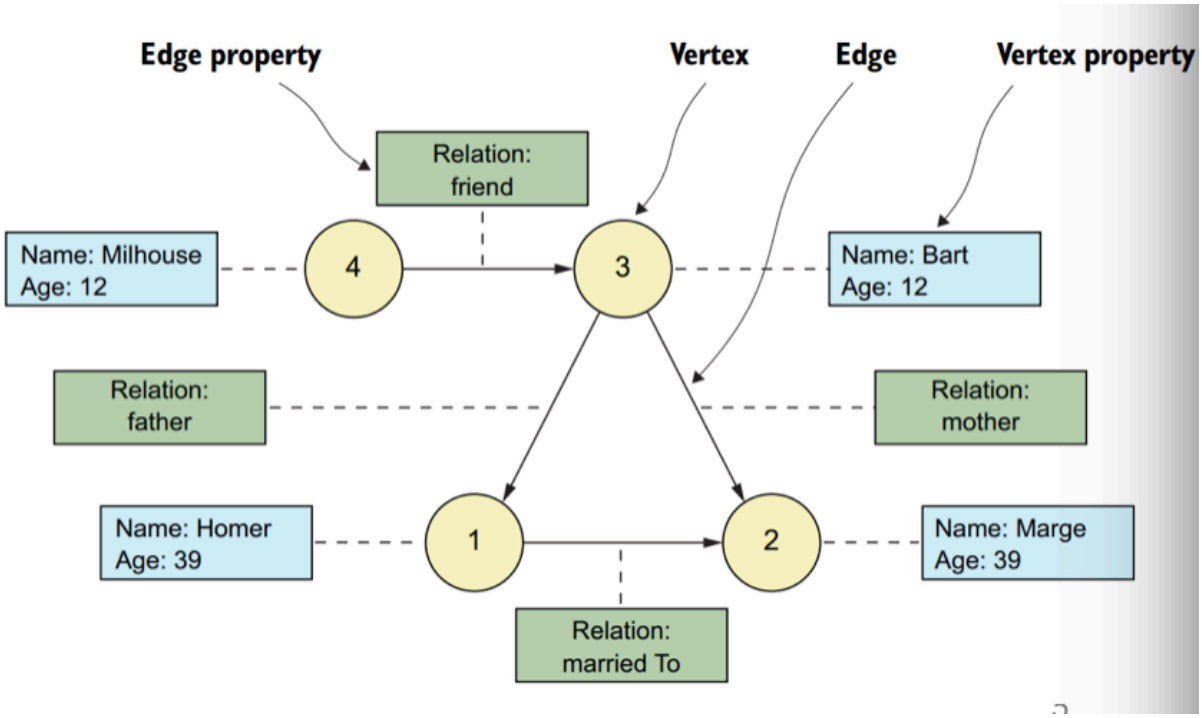
- Graph
  - a collection of vertices and connected edges.
  - storage: adjacency list or adjacency matrix
  - directed and undirected graph
    - directed graph: the order of the two vertices in an edge matters
    - undirected graph: the order doesn't matter

Property Graph

- it is a directed multigraph with user defined objects (or properties) attached to each vertex and edge.
  - note: it's possible to have multiple edges between the same two vertices, because two vertices may have different relationships, friends and coworkers, or multiple flights between two vertices



Graph Computation Model: BSP

BSP: Bulk Synchronous Processing
- a programming model and computation framework for parallel computing
- multiple computing processors, servers of cores
- computation is divided into sequences of supersteps
- each superstep, a set of processors, running the same code, executes concurrently and creates messages that are sent to other processes.
- superstep ends when all the computation in the superstep is complete and all messages have been sent

- a barrier synchronization at the end of the superstep
- the next superstep begins
- until the program terminates(reach mad num of iterations or converges)

- Many graph algorithms are performed in different iterations (pagerank or shortest paths)
- in a superstep (iteration):
  - every node will perform a compute() function based on the neighbor info received (update some info for pagerank or shortest path)
  - the node will sent the new message to its neighbors
- after every node finishes the compute(), the next superstep starts

Computing the max:

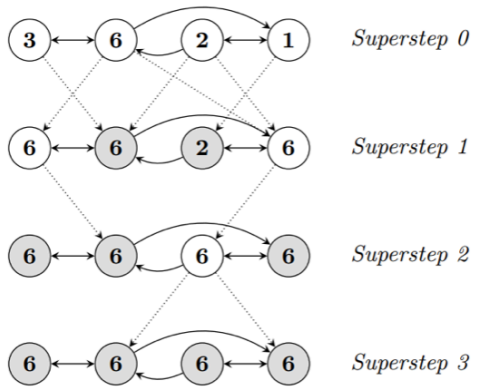
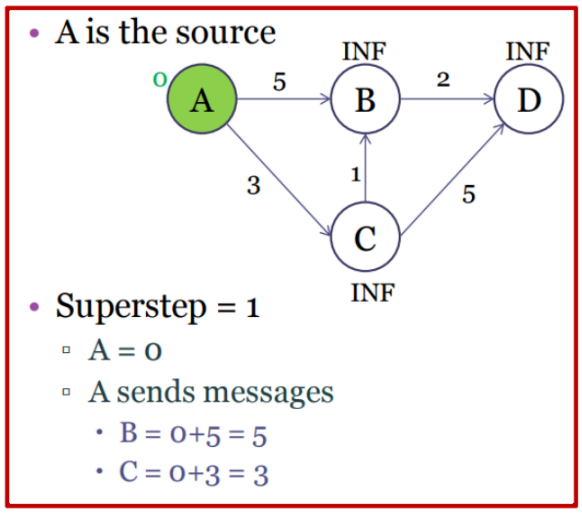


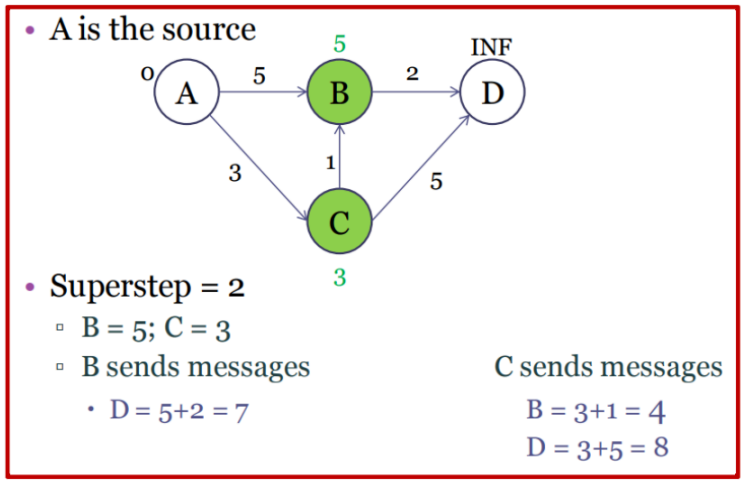
Figure 2: Maximum Value Example. Dotted lines are messages. Shaded vertices have voted to halt.

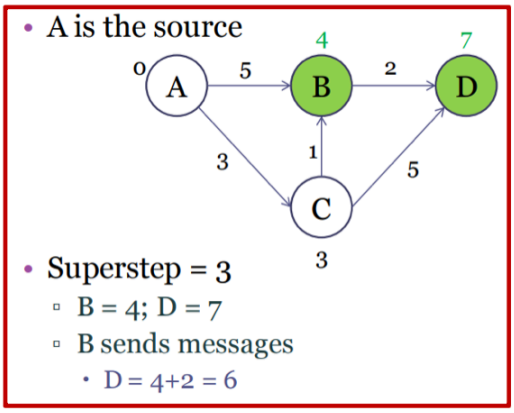
- Superstep 0: initialization
  - Every node has its initial value
  - It'll send its current value to all the neighbors
- Superstep 1
  - Every node will compare its current value with the values received from its neighbors
  - Node A receives 6 from its neighbor B and compares 6 with its old value 3, and then changes to 6
  - If a node doesn't change the value (or status), it'll become inactive and will not send new messages to neighbors
- Superstep 2
  - It a node doesn't receive new messages, it'll become inactive
- The process continues until every node is inactive (or reaches the max iterations)

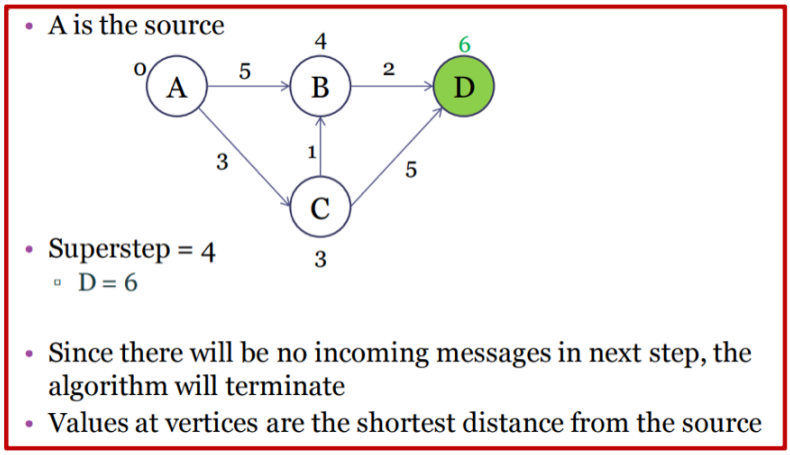
Single Source Shortest Paths:

- Finding shortest path between a single source vertex and every other vertex in the graph.
- Each vertex stores a value denoting the distance from source vertex to this vertex
- Value at each vertex is initialized to INF
- In each superstep
  - receives messages from its neighbors with updated potential minimum distances from source vertex
  - if minimum of these updated values is less than the current minimum distance of the vertex, value is updated and potential updates are sent to the neighbors (current value + outgoing edge weight)
- In first superstep, only source vertex will update its value to zero and send update messages to its neighbors
- algorithm terminates when no more updates









Spark GraphX

- GraphX is a graph processing system built inside Spark
- It relies on RDDs as the building blocks and implements many graphs algorithms for large scale data based on the BSP model

provides APIs

- graph construction
- graph transformation
- graph algorithms

Graph Construction:

- Based on vertexRDD and EdgeRDDs

VertexRDDs: contain tuples, which consist of two elements; a vertex ID of type Long and a property object of an arbitrary type

EdgeRDDs: contain edge objects, which consist of source and destination vertex ID (sourceID and destinationId) and a property object of an arbitrary type(attr, field)

- You can create a Spark graph using VertexRDD and Edge RDD (there are other construction methods)

Shortest Path Algorithm

- Spark implements the shortest-path algorithm with the ShortestPaths object.
- It has only one method, called run, which takes a graph and a sequence of landmark vertex IDs
- The returned graph's vertices contain a map with the shortest path to each of the landmarks, where the landmark vertex ID is the key and the shortest-path length is the value.