Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)
- Transportation networks (roads, trains, flights etc)

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

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

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

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

Typical graph problems and algorithms:

- Graph search and path planning

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)
- Transportation networks (roads, trains, flights etc)
- 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

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)
- Transportation networks (roads, trains, flights etc)
- 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

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)
- Transportation networks (roads, trains, flights etc)
- 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

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)
- Transportation networks (roads, trains, flights etc)
- 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

Graph Algorithms using Map-Reduce

Graphs are ubiquitous in modern society. Some examples:

- The hyperlink structure of the web
- Social networks on social networking sites like Facebook, IMDB, email, text messages and tweet flows (like Twitter)
- Transportation networks (roads, trains, flights etc)
- 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

Big Graphs

- Big graphs are typically sparse so adjacency list representation is much more space efficient. Typical value may be $m = O(n)$, where m is the number of links and n is the number of nodes in the graph.

Big Graphs

- Big graphs are typically sparse so adjacency list representation is much more space efficient. Typical value 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 1.3 billion users but each user, on an average, may only have few hundred friends, so the graph is very sparse.

Parallel Depth-First Search

- Assume that all links have unit distance for simplification. Assume that graph is connected.

Parallel Depth-First Search

- 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.

Parallel Depth-First Search

- 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.
- Each node is represented by a node id n (an integer), its current distance (initialized to ∞) and its adjacency list data structure N.

Parallel Depth-First Search

- 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.
- Each node is represented by a node id n (an integer), its current distance (initialized to ∞) and its adjacency list data structure N.
- 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.

Parallel Depth-First Search

- 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.
- Each node is represented by a node id n (an integer), its current distance (initialized to ∞) and its adjacency list data structure N.
- 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.

Parallel Depth-First Search

- 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.
- Each node is represented by a node id n (an integer), its current distance (initialized to ∞) and its adjacency list data structure N.
- 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

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

```
class Reducer
    method Reduce(nid m, [d_1, d_2, . . .])
        d_min ← ∞
        M ← ∅
        for all d ∈ counts [d_1, d_2, . . .] do
            if IsNode(d) then
                M ← d                                   ▷ Recover graph structure
            else if d < d_min then                      ▷ Look for shorter distance
                d_min ← d
        M.Distance ← d_min                              ▷ Update shortest distance
        Emit(nid m, node M)
```

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. Or by creating an abstract class with two sub-classes.

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. Or 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).

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. Or 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.

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. Or 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.

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.
- *Hadoop: The Definitive Guide (3rd ed.)*. Tom White, 2012, O'Reilly.