**WHITE PAPER CONSTRUCTION – V1.0**

**GROUP 8**

Graph representation:

(A) d/u [ 1|=2, 2|=3]

d: directed

u: undirected

= 2 way edge, directed

- 1 way edge

-- undirected

(B) fully connected graph with 3 nodes: 1, 2, 3,4

[1, [2, [1,3,[1,2,4,[1,2,3]],4],3,4]

Graph interface

1. d/u (directed/undirected) (mutable?)
2. number of nodes
3. number of edges
4. Adjacency list (out-degree related)
5. Anti adjacency list (in-degree related)
6. Edge exists
7. Node exists
8. Distance in hops (path exists)
9. Dfs tree
10. Bfs tree
11. Topological sort/level order traversal
12. cluster based on user-input function

EX.: cluster(pathExists(node1,node2))

1. merge (merge 2 nodes/merge 2 edges)

merge(lambda1(nodes), lambda2(edges))

1. block [edge between 2 nodes => the two nodes cannot connect]]
   1. Ex: block [a-b] => a has blocked b
2. Groups (list of nodes; user-defined)
3. Type of node (if there are multiple types of nodes in a single graph)

Properties of nodes

1. associative array (user-specified)
   1. Ex. Property[‘name’] = val
   2. Restrictions:
      1. values always strings(?).
      2. Do not allow a property to be a node itself (?)
2. Node id
3. List of graph ids (graphs to which it belongs to it)

Properties of edges

1. Id (GraphId + NodePair)
2. graph id
3. associative array (user-specified)
   1. Ex. Property[‘type’], Property[‘weight’]

Operators

‘-‘ single direction

‘=’ bothways directed

Add node to graph - graphid = graphid+node

Add edge to graph - graphid = graphid +edge[node1[-/=]node2]

Delete node/edge ‘!-’ graph = graph !- node/edge