Project Title: Study and implement the GHS algorithm using Erlang.

Team 7 Contributions:

Rakesh Mukkara(2019101087) - Code, Report, Presentation, Project Writeup.
K.L.Bhanu(20172147) - Project Writeup.

GHS Algorithm:

GHS algorithm is used to find the distributed minimum spanning tree through asynchronous messaging.

Mainly the algorithm has 2 steps in each phase:

1. Finding the Minimum outgoing edge(MOE) of the fragment.
2. Merging the Fragments using MOE.

We will be using 8 types of messages for communication between the nodes, they are

1. wakeup:

There are three node states: the initial state **Sleeping**, the state **Find** while participating in a fragment's search for the MOE, and the state **Found** at other times.
Initially, all the nodes are in a sleep state, and messaging wakeup for a node or multiple nodes starts the algorithm. After waking up the process will set its initial parameters and send a connect message through its minimum weight to the other node.
Wakeup messages can start the nodes when they are triggered by other messages.
It does not matter how many nodes we start initially in GHS because all the nodes
will be awakened during the program.(Initial Parameters: Status(Node) = found, FindCount = 0, Level(Node) = 0, Status(MOE) = branch)

2. connect:

When a fragment finds its MOE it will send a connect message through the MOE, The recipient of this message checks the conditions required to merge or absorb the requested fragment and sends the

3. initiate:

   Initiate is like a broadcast in the fragment it makes the node find its best edge. **FindCount** is increased for node for each initiate message sent to verify the replies using the message **report**.

4. Test:

   test message receiver will check whether the edge is valid as a branch or it should be rejected depending on the fragment ID and Level of the node.

5. accept:

   accept message is received by a node when the other node checks the validity (not of the same fragment)of the test edge and sends accept to it. The accept receiver changes the best edge if its weight is lower than the bestEdge.

6. reject:

   If the receiver gets reject message it will change the edge status to reject and starts testing again.

7. report:

   After the process of the best outgoing edge in its subtree, the FindCount will be 0 so now it can send report message to its parent about the best edge. The report will send a changeRoot if the weight it received from **convergecast** is greater than the best weight it had or if the weight is infinity it will quit the processing.

8. changeRoot:

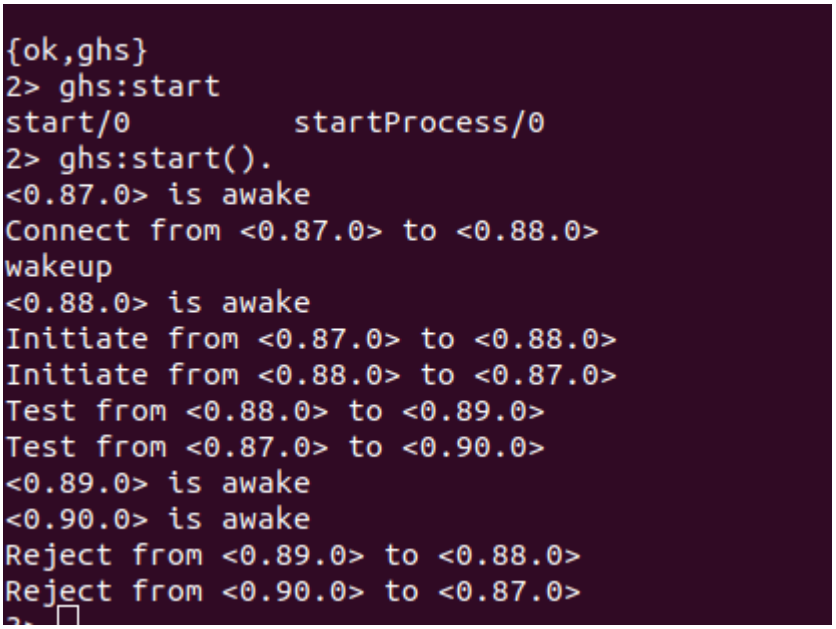
   This message passes through the best edges to MOE and then sends a connect request to the other fragment through MOE.

Test Case:

```
processData(A, [[B, 2], [D, 6]]),
processData(B, [[A, 2], [C, 3], [D, 8], [E, 5]]),
processData(C, [[B, 3], [E, 7]]),
processData(D, [[A, 6], [B, 8], [E, 9]]),
processData(E, [[B, 5], [C, 7], [D, 9]]),
```

processData is initiator for the processes which sends the Neighbours information

Output:


{ok,ghs}
2> ghs:start
start/0 startProcess/0
2> ghs:start().
<0.87.0> is awake
Connect from <0.87.0> to <0.88.0>
wakeup
<0.88.0> is awake
Initiate from <0.87.0> to <0.88.0>
Initiate from <0.88.0> to <0.87.0>
Test from <0.88.0> to <0.89.0>
Test from <0.87.0> to <0.90.0>
<0.89.0> is awake
<0.90.0> is awake
Reject from <0.89.0> to <0.88.0>
Reject from <0.90.0> to <0.87.0>
2>