Execution:

Inside erlang shell

1) c(ghs).
2) ghs:start().

Code Explanation:

Start:

Start function will start the processes and assign them the neighbour edges, then it will wakeup a process to start the algorithm.

Wakeup:

```
       wakeup->
           [Mpid, Mwt, Mst] = findMin(Edges),
           Mpid ! {connect,self(), 0}, %pid, LN, wt
           NewEdges = [[EXA, EXB, EXC] || [EXA, EXB, EXC] <- Edges, EXA
/= Mpid] ++ [[Mpid, Mwt, branch]],
           io:fwrite("~w is awake", [self()]),
           startGhs(FN, 0, NewEdges , found, BestEdge, BestWt,
TestEdge, InBranch, 0);
```

The nodes in the sleep test will be sent a wakeup request so that they will set their initial properties LN = 0, SN = found, SE(MInimim Wt Edge) = Branch, and send connect through the Minimumweight edge.

Connect:

When node in sleep connect will wakeup the node,

```
       {connect, Upid, L} when L<LN ->
           NewEdges = [[EXA, EXB, EXC] || [EXA, EXB, EXC] <- Edges, EXA
=/= Upid] ++ [[Upid, getWt(Edges, Upid), branch]],
           Upid ! {initiate, self(), LN, FN, SN},
           if
               SN == find ->
                   startGhs(FN, LN, NewEdges , SN, BestEdge, BestWt,
TestEdge, InBranch, FindCount+1);
               true ->
                   startGhs(FN, LN, NewEdges , SN, BestEdge, BestWt,
TestEdge, InBranch, FindCount)
           end;
```

```
      {connect, Upid, L}->
          MsgEdgeStatus = getStatus(Edges, Upid),
          if
              MsgEdgeStatus == basic ->
                  self() ! {connect, Upid, L};
              true ->
                  Upid ! {initiate, self(), LN+1, getWt(Edges, Upid),
find}
          end,
          startGhs(FN, LN, Edges , SN, BestEdge, BestWt, TestEdge,
InBranch, FindCount);
```

If the level of the node is higher than the level of the received node then it will send initiate to that node with its FN, LN, SN(Fragment number, Level number, Status of node) else if the edge is basic then it will push the message to the end so that it can revisit after a level increase.
If the levels are equal and the edge is a branch then initiate will be sent with FN, LN, SN with FN = weight of the edge between them.

Initiate:

```
          BranchEdges = [[A, B, C] || [A, B, C] <- Edges, C == branch,
A /= Upid],
          NewFindCount = sendInitiates(Edges, Upid, L, F, S, 0) +
FindCount,
```

The initiate will send the initiates to its children and if it is in the state find it will increase the FindCount with number of children and then is SN = Find it should find the minimum edges with state basic and send Test on that edge, it such edge does not exist it should check its FindCount and test-edge, if its 0 then it will send report message to its parent(InBranch) and changes state to found.

Test:

If SN = sleep , it should wakeup the node when receiving test,

```
{test, Upid, L, F} when L > LN ->
          self() ! {test, Upid, L, F},
          startGhs(FN, LN, Edges , SN, BestEdge, BestWt, TestEdge,
InBranch, FindCount);
```

If LN(received) < LN(send) it should place the message at the end of the queue to revisit it after the level increase.

```
{test, Upid, L, F} when F /= FN ->
           Upid ! {accept, self()},
           startGhs(FN, LN, Edges , SN, BestEdge, BestWt, TestEdge,
InBranch, FindCount);
```

Else if they does not belong to same fragment it should send accept to the other node.
Else if status = Basic, it should change the status of the edge to rejected and then check for the other test edge.

Accept:

Accept will change its best-edge if received edge has lower weight, then it will check its FindCount and test-edge, if its 0 then it will send report message to its parent(InBranch) and changes its state to found.

Reject:

If the received edge status is basic it will change to rejected, then it will start looking for new test edge.

Report:

```
               EdgeStatus /=branch ->
                   if
                       W < BestWt ->
                           NewBestEdge = Upid,
                           NewBestWeight = W;
                       true ->
                           NewBestEdge = BestEdge,
                           NewBestWeight = BestWt
                   end,
                   if
                       FindCount == 0 andalso TestEdge == none->
                           InBranch ! {report, self(),  BestWt},
                           startGhs(FN, LN, Edges , found, NewBestEdge,
NewBestWeight, TestEdge, InBranch, FindCount);
                       true ->
                           startGhs(FN, LN, Edges , SN, NewBestEdge,
NewBestWeight, TestEdge, InBranch, FindCount)
                   end;
```

If the received message is not from a child then it will check the weights to change the best weight accordingly and then it will check its FindCount and test-edge, if its 0

then it will send report message to its parent(InBranch) and changes its state to found.
Else if SN=find it will wait until SN changes to found so it pushes the messages back received from child,
Else it will send changeRoot if received W > BestWt.
Else if W = BestWt = infinite it can exit as there are no MOE's

ChangeRoot:

```
        {changeRoot} ->
            EdgeStatus = getStatus(Edges, BestEdge),
            if
                EdgeStatus == branch ->
                    BestEdge ! {changeRoot};
                true ->
                    BestEdge ! {connect, self(), LN}
            end,
            startGhs(FN, LN, Edges , SN, BestEdge, BestWt, TestEdge,
InBranch, FindCount)
```

ChangeRoot will go through the best-edges found if status of that Edge is branch and whenever it finds a basic edge in its path it will send Connect.