```
function build_inconsistency_list(edge_list,G,property):
//insertions
for each e in edge_list
  VertexProperty[e.src].tri_count = 0
  VertexProperty[e.dst].tri_count = 0
  inconsistency_list =
  inconsistency_list U{e.src,e.dst,DEFAULT_PRIORITY}
 G'= G' U (e.src, e.dst)
  for each neighbor x of e.src
     inconsistency_list =
     inconsistency_list U{x,DEFAULT_PRIORITY}
    G' = G' U (e.src, x)
  for each neighbor y of e.src
     inconsistency_list =
     inconsistency_list U {y,DEFAULT_PRIORITY}
    G' = G' U (e.dst,y)
return (inconsistency_list,G')

function property_guard(degree: DE,threshold_fraction f):
if fraction of inconsistent vertices with
(degree > DE > n∗f) > f
   run static re–computation
 else run incremental algorithm

function frontier_activate(G',inconsistency_list):
// Using extract operation on inconsistency_list
for every edge e in G'
  activate(e.src)
  activate(e.dst)

function update_inconsistency_list(G',inconsistency_list,
new_inconsistency=NULL)
if(G.activity.empty())
   inconsistency_list.clear()

//I–GAS computation loop
function I–GAS(inconsistency_list,G'):
While(!inconsistency_list.empty())
  if(itr = 1)
     frontier_activate(G',inconsistency_list)
  Modified–GAS(G')
  update_inconsistency_list(G',inconsistency_list)

function merge_state(inconsistency_list) :
for each v in inconsistency_list
  VertexProperty[v].tri_count += tri_count_old
```