```
local function optimize(egs,    cluster,leaves,row1,row2)
  cluster = CLUSTER(egs)
  local function order(a,b) return a.egs:betters(b.egs)  end
  for rank,leaf in pairs(quicksort(cluster:leaves(), order)) do
    leaf.rank = rank end
  return cluster end

function CLUSTER.project(i,row)
  return cosine(i.top:dist(row, i.left), i.top:dist(row, i.right), i.c) end

function CLUSTER.where(i,row)
  if   i:leaf() then return i end
  if   i:project(row) <= i.border
  then return i.lefts  and i.lefts:where( row) or i
  else return i.rights and i.rights:where(row) or i end end

function CLUSTER.better(i,row1,row2,    where1, where2)
  where1, where2 = i:where(row1), i:where(row2)
  if     where1.rank > where2.rank then return false
  elseif where1.rank < where2.rank then return true
  else   return where1:xbetter(row1,row2) end end

function CLUSTER.xbetter(i,row1,row2,  x1,x2)
  x1,x2 = i:project(row1), i:project(row2)
  return i.egs:better(i.left, i.right) and x1 <= x2 or x1 > x2 end

function CLUSTER.leaves(i, out)
  out = out or {}
  if i:leaf() then push(out,i) end
  if i.lefts  then i.lefts:leaves(out) end
  if i.rights then i.rights:leaves(out) end
  return out end

local function optimize(egs,    cluster,leaves,row1,row2)
 local function optimize(egs,    cluster,leaves,row1,row2)
 local function optimize(egs,    cluster,leaves,row1,row2)
  cluster = CLUSTER(egs)
  local function order(a,b) return a.egs:betters(b.egs)  end
  for rank,leaf in pairs(quicksort(cluster:leaves(), order)) do
    leaf.rank = rank end
  return cluster end

function CLUSTER.project(i,row)
  return cosine(i.top:dist(row, i.left), i.top:dist(row, i.right), i.c) end

function CLUSTER.where(i,row)
  if   i:leaf() then return i end
  if   i:project(row) <= i.border
  then return i.lefts  and i.lefts:where( row) or i
  else return i.rights and i.rights:where(row) or i end end

function CLUSTER.better(i,row1,row2,    where1, where2)
  where1, where2 = i:where(row1), i:where(row2)
  if     where1.rank > where2.rank then return false
  elseif where1.rank < where2.rank then return true
  else   return where1:xbetter(row1,row2) end end

function CLUSTER.xbetter(i,row1,row2,  x1,x2)
  x1,x2 = i:project(row1), i:project(row2)
  return i.egs:better(i.left, i.right) and x1 <= x2 or x1 > x2 end

function CLUSTER.leaves(i, out)
  out = out or {}
  if i:leaf() then push(out,i) end
  if i.lefts  then i.lefts:leaves(out) end
  if i.rights then i.rights:leaves(out) end
  return out end

 cluster = CLUSTER(egs)
  local function order(a,b) return a.egs:betters(b.egs)  end
  for rank,leaf in pairs(quicksort(cluster:leaves(), order)) do
    leaf.rank = rank end
  return cluster end

function CLUSTER.project(i,row)
  return cosine(i.top:dist(row, i.left), i.top:dist(row, i.right), i.c) end

function CLUSTER.where(i,row)
  if   i:leaf() then return i end
  if   i:project(row) <= i.border
  then return i.lefts  and i.lefts:where( row) or i
  else return i.rights and i.rights:where(row) or i end end

function CLUSTER.better(i,row1,row2,    where1, where2)
  where1, where2 = i:where(row1), i:where(row2)
  if     where1.rank > where2.rank then return false
  elseif where1.rank < where2.rank then return true
  else   return where1:xbetter(row1,row2) end end

function CLUSTER.xbetter(i,row1,row2,  x1,x2)
  x1,x2 = i:project(row1), i:project(row2)
  return i.egs:better(i.left, i.right) and x1 <= x2 or x1 > x2 end

function CLUSTER.leaves(i, out)
  out = out or {}
  if i:leaf() then push(out,i) end
  if i.lefts  then i.lefts:leaves(out) end
  if i.rights then i.rights:leaves(out) end
  return out end

clocal function optimize(egs,    cluster,leaves,row1,row2)
  cluster = CLUSTER(egs)
  local function order(a,b) return a.egs:betters(b.egs)  end
  for rank,leaf in pairs(quicksort(cluster:leaves(), order)) do
    leaf.rank = rank end
  return cluster end

function CLUSTER.project(i,row)
  return cosine(i.top:dist(row, i.left), i.top:dist(row, i.right), i.c) end

function CLUSTER.where(i,row)
  if   i:leaf() then return i end
  if   i:project(row) <= i.border
  then return i.lefts  and i.lefts:where( row) or i
  else return i.rights and i.rights:where(row) or i end end

function CLUSTER.better(i,row1,row2,    where1, where2)
  where1, where2 = i:where(row1), i:where(row2)
  if     where1.rank > where2.rank then return false
  elseif where1.rank < where2.rank then return true
  else   return where1:xbetter(row1,row2) end end

function CLUSTER.xbetter(i,row1,row2,  x1,x2)
  x1,x2 = i:project(row1), i:project(row2)
  return i.egs:better(i.left, i.right) and x1 <= x2 or x1 > x2 end

function CLUSTER.leaves(i, out)
  out = out or {}
  if i:leaf() then push(out,i) end
  if i.lefts  then i.lefts:leaves(out) end
  if i.rights then i.rights:leaves(out) end
  return out end

luster = CLUSTER(egs)
  local function order(a,b) return a.egs:betters(b.egs)  end
  for rank,leaf in pairs(quicksort(cluster:leaves(), order)) do
    leaf.rank = rank end
  return cluster end

function CLUSTER.project(i,row)
  return cosine(i.top:dist(row, i.left), i.top:dist(row, i.right), i.c) end

function CLUSTER.where(i,row)
  if   i:leaf() then return i end
  if   i:project(row) <= i.border
  then return i.lefts  and i.lefts:where( row) or i
  else return i.rights and i.rights:where(row) or i end end

function CLUSTER.better(i,row1,row2,    where1, where2)
  where1, where2 = i:where(row1), i:where(row2)
  if     where1.rank > where2.rank then return false
  elseif where1.rank < where2.rank then return true
  else   return where1:xbetter(row1,row2) end end

function CLUSTER.xbetter(i,row1,row2,  x1,x2)
  x1,x2 = i:project(row1), i:project(row2)
  return i.egs:better(i.left, i.right) and x1 <= x2 or x1 > x2 end

function CLUSTER.leaves(i, out)
  out = out or {}
  if i:leaf() then push(out,i) end
  if i.lefts  then i.lefts:leaves(out) end
  if i.rights then i.rights:leaves(out) end
  return out end


```