```
-- 768
--  | 384
--  |  | 192
--  |  |  | 96
--  |  |  |  | 48          {positive}
--  |  |  |  | 48          {positive}
--  |  |  | 96
--  |  |  |  | 48          {positive}
--  |  |  |  | 48          {negative}
--  |  | 192
--  |  |  | 96
--  |  |  |  | 48          {positive}
--  |  |  |  | 48          {negative}
--  |  |  | 96
--  |  |  |  | 48          {positive}
--  |  |  |  | 48          {positive}
--  | 384
--  |  | 192
--  |  |  | 96
--  |  |  |  | 48          {negative}
--  |  |  |  | 48          {negative}
--  |  |  | 96
--  |  |  |  | 48          {negative}
--  |  |  |  | 48          {negative}
--  |  | 192
--  |  |  | 96
--  |  |  |  | 48          {negative}
--  |  |  |  | 48          {negative}
--  |  |  | 96
--  |  |  |  | 48          {negative}
--  |  |  |  | 48          {negative}

local R = require
local the,egs,lib = R"the", R"egs", R"lib"
local per,cos,norm,o,fmt,rnds=lib.per,lib.cosine,lib.norm,lib.o,lib.fmt,lib.rnds
local map,any,many,sort,up1 = lib.map,lib.any, lib.many,lib.sort,lib.up1

local cluster={}
function cluster.new(top,egs1,      i,lefts,rights)
  egs1 = egs1 or top
  i   = {egs=egs1, top=top, rank=0}
  lefts, rights, i.left, i.right, i.border, i.c = cluster.half(top, egs1.rows)
  if #egs1.rows >= 2*(#top.rows)^the.leaves then
    if #lefts.rows < #egs1.rows then
      i.lefts = cluster.new(top, lefts)
      i.rights= cluster.new(top, rights) end end
  return i end
---      _  |_   _
---     _>  | |  (_)  \/\/

function cluster.show(i,   pre, front)
  pre = pre or ""
  local front = fmt("%s%s", pre, #i.egs.rows)
  if   cluster.leaf(i)
  then print(fmt("%-20s%s",front, o(rnds(egs.mid(i.egs,i.egs.cols.y)))))
  else print(front)
       if i.lefts  then cluster.show(i.lefts,  "|  "..pre)
       if i.rights then cluster.show(i.rights, "|  "..pre) end end end end

function cluster.leaf(i) return not (i.lefts or i.rights) end
---      _|  o   _  _|_
---     (_|  |  _>  |_

function cluster.dist(eg1,row1,row2)
  local function sym(c,x,y) return x==y and 0 or 1 end
  local function num(c,x,y)
    if     x=="?" then y = norm(c.lo, c.hi, y); x=y<.5 and 1 or 0
    elseif y=="?" then x = norm(c.lo, c.hi, x); y=x<.5 and 1 or 0
    else               x,y = norm(c.lo, c.hi, x), norm(c.lo, c.hi, y) end
    return math.abs(x-y) end
  local function dist(c,x,y)
    return x=="?" and y=="?" and 1 or (c.nump and num or sym)(c,x,y) end
  local d, n = 0, #eg1.cols.x
  for key,c in pairs(eg1.cols.x) do d=d+dist(c, row1[c.at], row2[c.at])^the.p en
d
  return (d/n)^(1/the.p) end

function cluster.neighbors(eg1, r1, rows)
  return sort(map(rows or eg1.rows,
              function(r2) return {cluster.dist(eg1,r1,r2),r2} end), up1) end
---      _  _  _  _  _  _  _|_  _
---     _>  (/_ |_) (/_ |  (_|  |_ (/_
---                 |

function cluster.half(eg1, rows)
  local project,far,some,left,right,c,lefts,rights,border
  rows    = rows or eg1.rows
  far     = function(r,t) return per(cluster.neighbors(eg1,r,t), the.far)[2] end
  project = function(r)
              return {cos(cluster.dist(eg1,left,r),
                          cluster.dist(eg1,right,r),
                          c),
                      r} end
  some    = many(rows,     the.some)
  left    = far(any(some), some)
  right   = far(left,      some)
  c       = cluster.dist(eg1,left,right)
  lefts,rights = egs.clone(eg1), egs.clone(eg1)
  for n, projection in pairs(sort(map(rows,project), up1)) do
    if n==#rows//2 then border = projection[1] end
    egs.add(n <= #rows//2 and lefts or rights, projection[2]) end
  return lefts, rights, left, right, border, c  end

return cluster
```