algoritmo grahamsScan()

min <- 0

para i <- 1 mientras i < this.xPoints.length hacer i <-i+1

si yPoints[i] == yPoints[min] entonces

si xPoints[i] < xPoints[min] entonces

min = i;

fin si

sino si yPoints[i] < yPoints[min] entonces

min = i;

fin si

fin para

al <- Lista

ang<-0

dist<-0

para i <- 0 mientras i < tamaño(xPoints) hacer i<-i+1

si i != min entonces

ang <- atan((yPoints[i]-yPoints[min])/(xPoints[i]-xPoints[min]))//arco tangenete de un angulo

si ang < 0 entonces

ang <- ang + valorPI

fin si

dist <-raizCudadrada((xPoints[i] - xPoints[min])\*(xPoints[i] - xPoints[min]) + (yPoints[i] - yPoints[min])\*(yPoints[i] - yPoints[min]))

agregar[al]<- (pData(i, ang, dist))

fin si

fin para

Ordena\_fusion(al)

stack <- arreglo[tamaño(xPoints)+1]

j <- 2

para i <-0 mientras i<tamaño[al] hacer i<-i+1

data <- get[al]<-j-2

stack[j] <- index[data]

j<-j+1

fin para

stack[0] <- stack[tamaño[xPoints]]

stack[1] <- min

M <- 2

////aqui se calcula el producto de los vectores (x2-x1)\*(y3-y1)-(y2-y1)\*(x3-x1)

para i <- 3 mientras i <= tamaño(xPoints) hacer i++

mientras (xPoints[stack[M]] - xPoints[stack[M-1]])\*(yPoints[stack[i]] - yPoints[stack[M-1]]) - (yPoints[stack[M]] - yPoints[stack[M-1]])\*(xPoints[stack[i]] - xPoints[stack[M-1]])<=0 hacer M<-M-1

fin mientras

M<-=M+1

tmp1<-stack[i]

stack[i]<-stack[M]

stack[M]<-tmp1

fin para

num <- M

xPoints2 <- arreglo[num]

yPoints2 <- arreglo[num]

para i <- 0 mientras i < num hacer i++

xPoints2[i] <- xPoints[stack[i+1]]

yPoints2[i] <- yPoints[stack[i+1]]

fin para

Fin grahamsScan