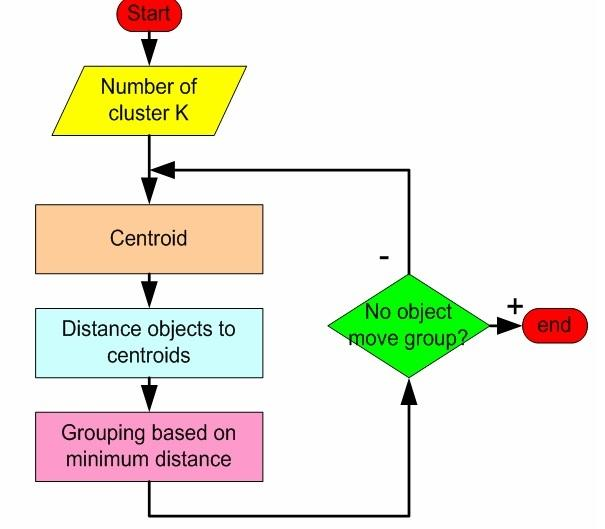
# **K MEANS**

**FLOW CHART**



**PSEUDO CODE**

Begin

Do

1. Assign query to random clusters
2. Calculate the mean of the clusters using Euclidean distance
3. Calculate centroids and form new clusters
4. Calculate distance of the query from the new centroids
5. Assign query to cluster with least distance
6. If(query still in initial cluster)

Stop

Else

Repeat from step 2

End

**ALGORITHM AND IMPLEMENTATION**

**Centroid coordinates**

def calculateCentroid(self):

reduce\_coord = lambda i:reduce(lambda x,p : x + p.coords[i],self.points,0.0)

centroid\_coords = [reduce\_coord(i)/len(self.points) for i in range(self.n)]

return Point(centroid\_coords)

**Determine distance**

def getDistance(a, b):

if a.n != b.n: raise Exception("ILLEGAL: non comparable points")

ret = reduce(lambda x,y: x + pow((a.coords[y]-b.coords[y]), 2),range(a.n),0.0)

return math.sqrt(ret)

**Cluster creation**

for i in range(len(clusters)):

shift = clusters[i].update(lists[i])

biggest\_shift = max(biggest\_shift, shift)

if biggest\_shift < cutoff:

break

return clusters

**Screenshot**

