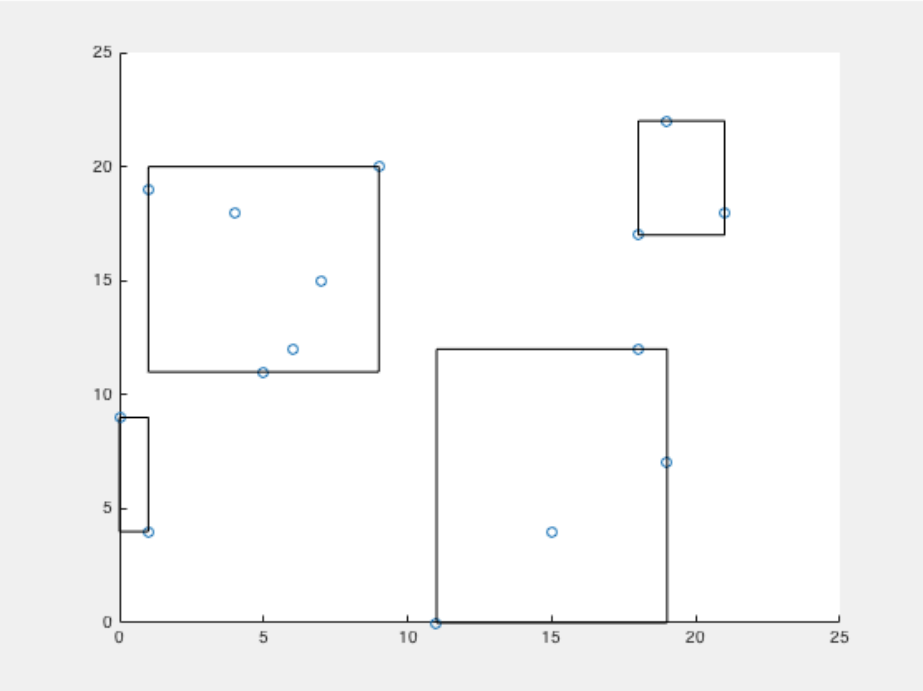
Assignment 4

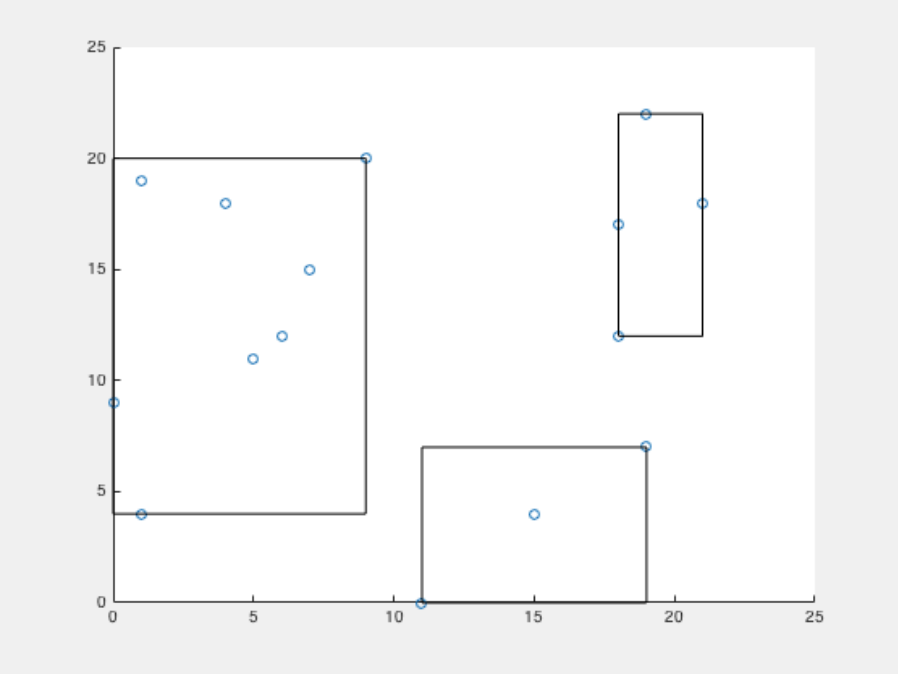
1a) Basic sequential Clustering

theta value is 12, maximum number of clusters as 4

Taking dataset in the given order. We observed 4 clusters as shown in the following diagram.



1b) using the data points in the reverse order



only 3 clusters are formed when we take data points in the reverse order.

1c) Rand Index

Rand index to find the difference between the two clusterings:

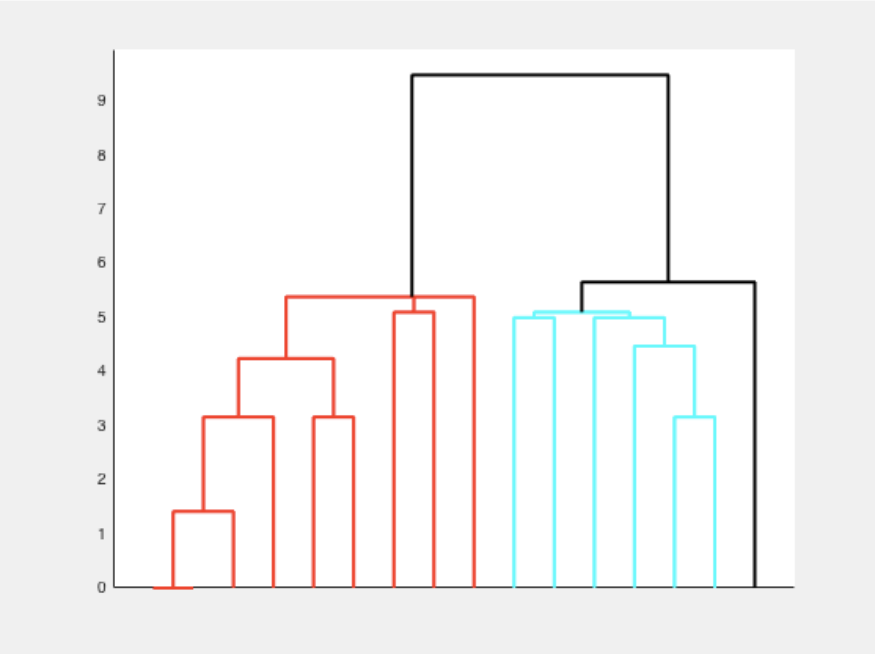
ans =

0.6800

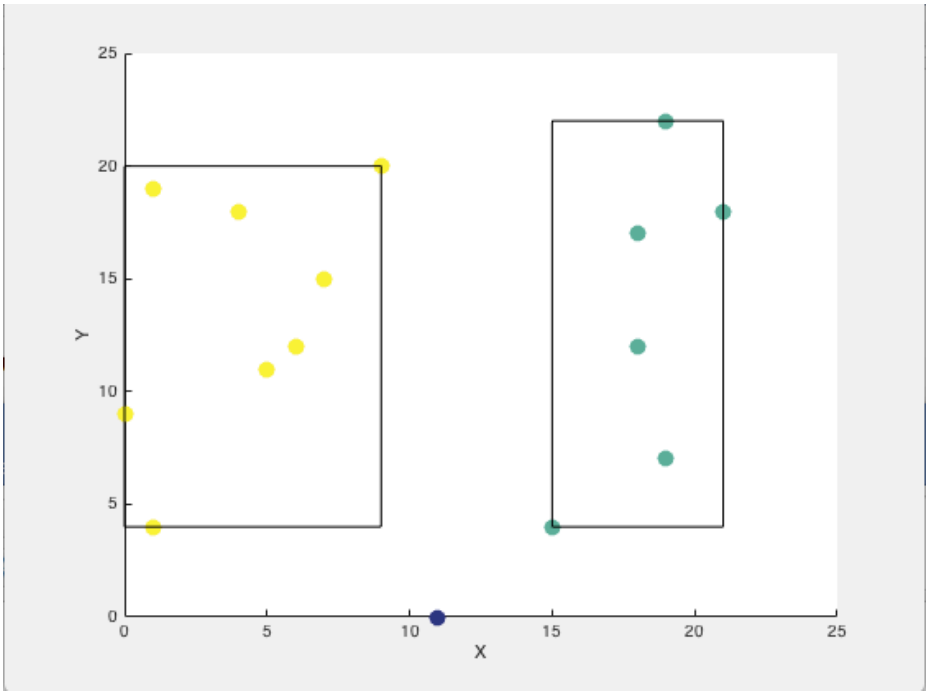
In BSAS(Basic sequential algorithm scheme), each cluster is represented by a single vector. Which are said to be global clustering criteria. Here is the mean vector represents the cluster. Calculation of mean differs when ordering is changed. The order in which the vectors are presented to the BSAS plays an important role.

2a). Hierarchical clustering using single-link

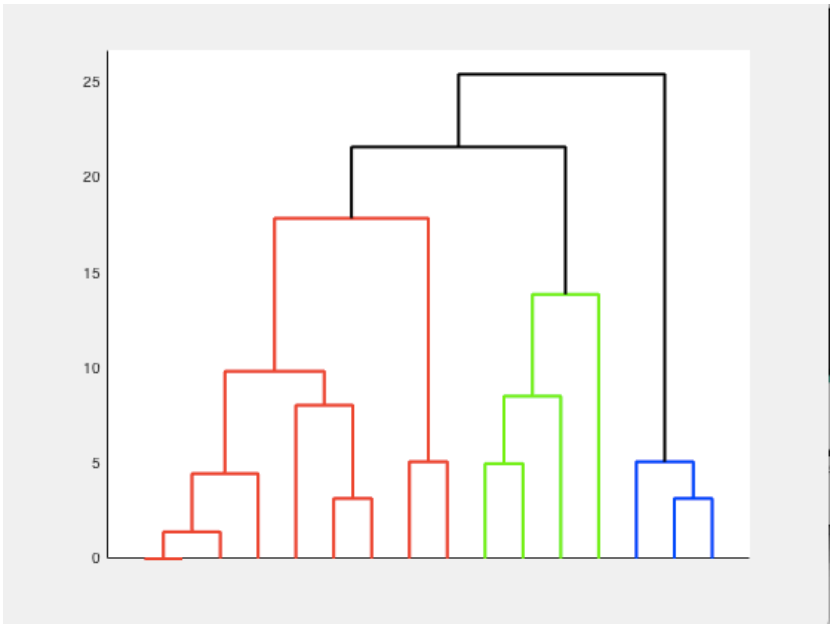
Dendrogram



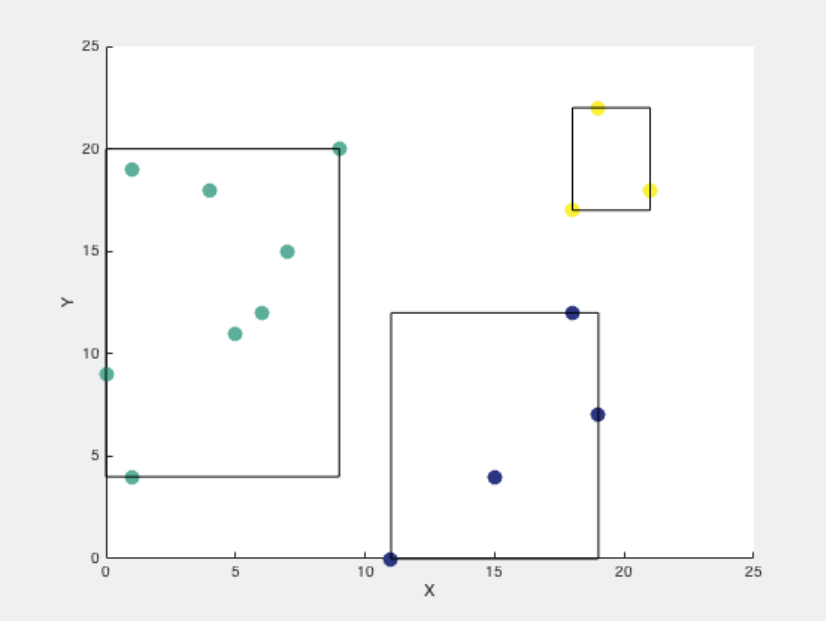
Clusters 2D plot



2b) Hierarchical clustering using complete link



Clusters 2D plot using complete link



2c) the sum of squared errors for the clusterings obtained in (a) and (b)

sum of squared errors of Hierarchical Cluster using Single Link
SSE_SingleLink = 80.9975

Cluster 3 is contributing more for the SSE

sum of squared errors of Hierarchical Cluster using Complete Link
SSE_CompleteLink = 72.4479

Cluster 2 is contributing more for the SSE

2d)

Correlation for single link
correlation coefficient 7.259384e-01

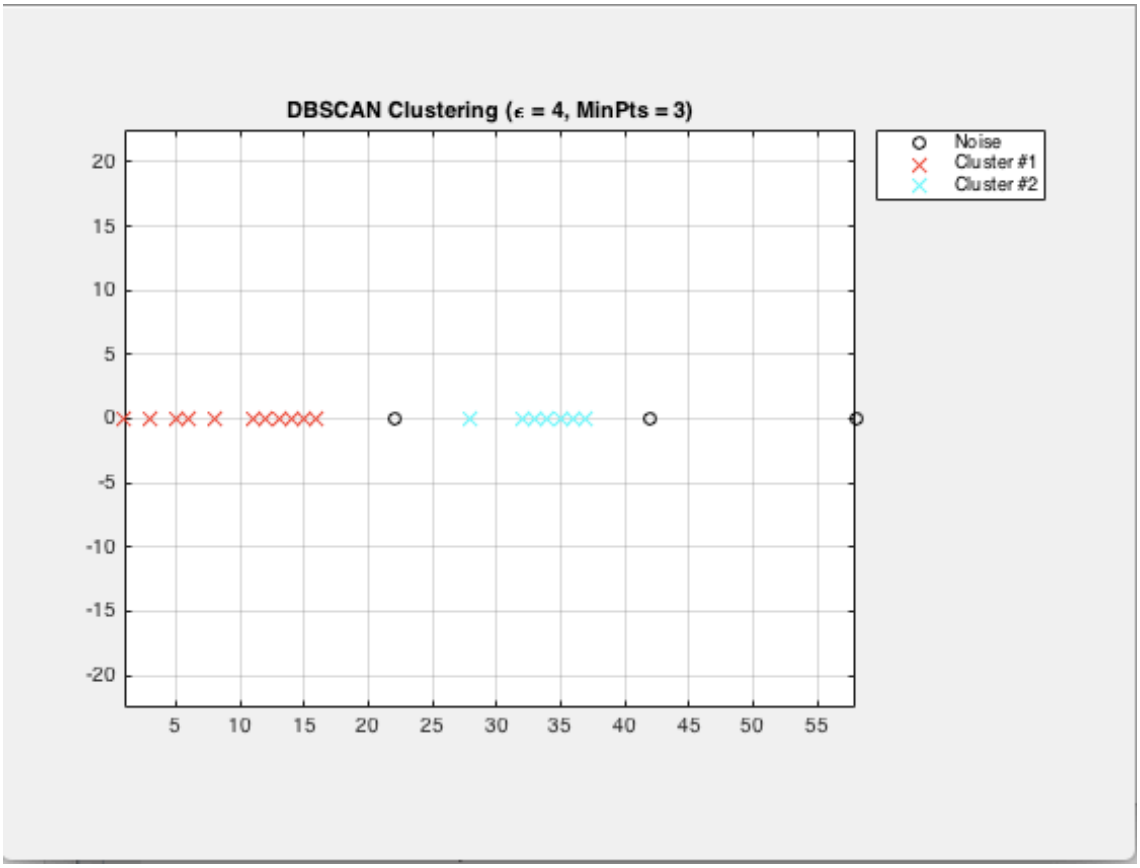
Correlation for complete link
correlation coefficient 7.600997e-01

Correlation for complete link is higher than the single link

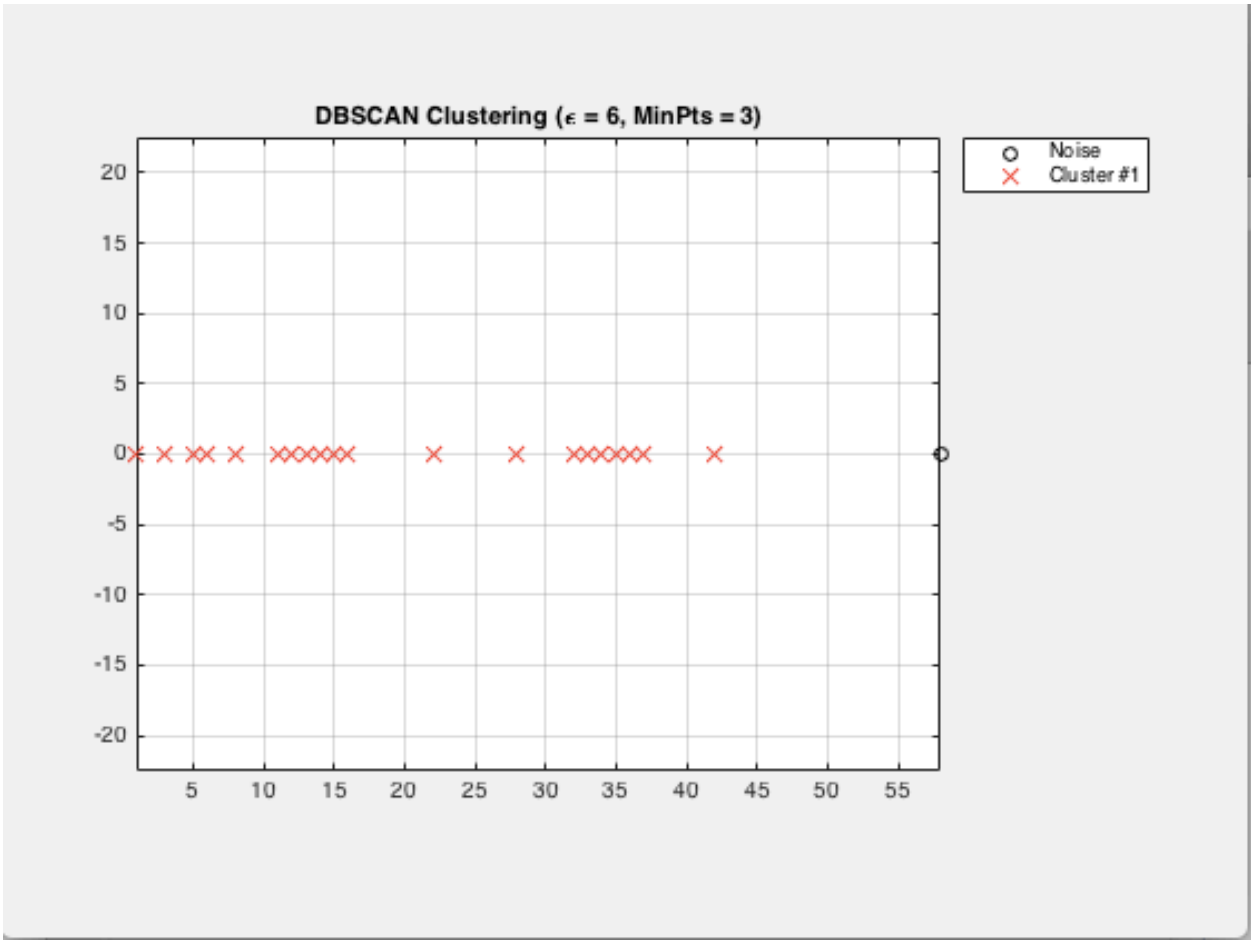
correlation between the proximity martix and binary martix for single link 7.269757e-01
correlation between the proximity martix and binary martix complete link 6.901631e-01

single link uses minimal distance between cluster points and other point to decide whether to merge in the cluster. Shortest Euclidian distance measure gives the high correlation among the cluster points.

3a) Use DBSCAN algorithm epsilon value is 4 and MinPoints value of 3



3b) Do the same as in (a) above but use Epsilon value of 6.



3c) Compare the two clustering's using Rand Index and show all your work.

RandIndex =

0.4524

Density-based clustering uses local density of points to determine the clusters rather than using only the distance between points.

ε-*neighborhood* of **x**, too large, denser clusters may be merged together. ε-*neighborhood* of **x**, too small, sparse clusters will be categorized as noise.

Source Code:

Main.m

```
% [6, 12], [19, 7], [15, 4], [11, 0], [18, 12], [9, 20], [19, 22],
% [18, 17], [5, 11], [4, 18], [7, 15], [21, 18], [1, 19], [1, 4], [0, 9], [5,
11].

dsCell={[6, 12], [19, 7], [15, 4], [11, 0], [18, 12], [9, 20], [19, 22],[18,
17],...
    [5, 11], [4, 18], [7, 15], [21, 18], [1, 19], [1, 4], [0, 9], [5, 11]};

%1a Basic Sequence Algorithm Scheme
m=1;
theta=12;
q=4;
obj=BasicSequentialAlgo(theta,q);
[Ck,Cm]=PartitionDatasetIntoClusters(obj,dsCell);
[bsSet]=plotClusters(Ck);

%1b Basic Sequence Algorithm Scheme the data points in the reverse order
[HCk,HCm]=PartitionDatasetIntoClusters(obj,fliplr(dsCell));
[hbsSet]=plotClusters(HCk);

%1c.Rand Index Calculation
N=nchoosek(length(dsCell),2);
a=0;
b=0;
c=0;
d=0;
for i=1:(length(dsCell)-1)
    x1=dsCell{i};
    for j=i+1:length(dsCell)
        x2=dsCell{j};
        samebsCluster=0;
        samehbsCluster=0;

        for rdinx=1:length(bsSet)
            if((nnz(intersect(x1,bsSet{rdinx},'rows'))==2) &&
(nnz(intersect(x2,bsSet{rdinx},'rows'))==2))
                samebsCluster=1;
                break;
            end
        end
        for dinx=1:length(hbsSet)
            if((nnz(intersect(x1,hbsSet{dinx},'rows'))==2) &&
(nnz(intersect(x2,hbsSet{dinx},'rows'))==2))
                samehbsCluster=1;
                break;
            end
        end
```

```
        if(samebsCluster==1 && samehbsCluster==1)
            a=a+1;
        elseif(samebsCluster==0 && samehbsCluster==0)
            b=b+1;
        elseif(samebsCluster==1 && samehbsCluster==0)
            c=c+1;
        elseif(samebsCluster==0 && samehbsCluster==1)
            d=d+1;
        else
        end


    end
end

fprintf('Rand index to find the difference between the two clusterings:\n')
(a+b)/N




X=[6 12; 19 7; 15 4; 11 0; 18 12; 9 20; 19 22;18 17;
    5 11; 4 18; 7 15; 21 18; 1 19; 1 4; 0 9; 5 11];

NUM=3;%No. of Clusters
PairDistance = pdist(X);

% 2a. Hierarichal clustering using single link
singleLink = linkage(PairDistance,'single');
singleLinkClusterData = clusterdata(X,'linkage','single','maxclust',NUM);

figure();
singleLinkDendogram = dendrogram(singleLink, 0,
'colorthreshold',mean(singleLink(end-NUM+1:end-NUM+2,3)));
set(singleLinkDendogram,'LineWidth',2)
set(gca, 'XTickLabel',[], 'TickLength',[0 0])

singleLinkClustPtsCell={};

% dividing points into clusters
for idxClust=1:NUM
singleLinkClustPtsCell(idxClust)={X(find(singleLinkClusterData==idxClust),:)}
;
end



figure();
scatter(X(:,1),X(:,2),100, singleLinkClusterData, 'filled')
xlabel X, ylabel Y
hold on
for rectId=1:length(singleLinkClustPtsCell)
rectangle('Position', [min(singleLinkClustPtsCell{rectId}(:,1))
min(singleLinkClustPtsCell{rectId}(:,2))...
    max(singleLinkClustPtsCell{rectId}(:,1))-
```

```
min(singleLinkClustPtsCell{rectId}(:,1))...
    max(singleLinkClustPtsCell{rectId}(:,2))-
min(singleLinkClustPtsCell{rectId}(:,2))]);

end



% 2b. Hierarichal clustering using complete link
completeLink = linkage(PairDistance,'complete');
completeLinkClusterData = clusterdata(X,'linkage','complete','maxclust',NUM);

figure();
completeLinkDendogram = dendrogram(completeLink, 0,
'colorthreshold',mean(completeLink(end-NUM+1:end-NUM+2,3)));
set(completeLinkDendogram,'LineWidth',2)
set(gca, 'XTickLabel',[], 'TickLength',[0 0])

completeLinkClustPtsCell={};

% dividing points into clusters
for idxClust=1:NUM
completeLinkClustPtsCell(idxClust)={X(find(completeLinkClusterData==idxClust)
,:)};
end



figure();
scatter(X(:,1),X(:,2),100, completeLinkClusterData, 'filled')
xlabel X, ylabel Y
hold on
for rectId=1:length(completeLinkClustPtsCell)
rectangle('Position', [min(completeLinkClustPtsCell{rectId}(:,1))
min(completeLinkClustPtsCell{rectId}(:,2))...
    max(completeLinkClustPtsCell{rectId}(:,1))-
min(completeLinkClustPtsCell{rectId}(:,1))...
    max(completeLinkClustPtsCell{rectId}(:,2))-
min(completeLinkClustPtsCell{rectId}(:,2))]);

end



%2c Calculating Sum of squared errors
fprintf('sum of squared errors of Hierarchical Cluster using Single Link');
SSE_SingleLink=sumofSquaredErrors(singleLinkClustPtsCell)

fprintf('sum of squared errors of Hierarchical Cluster using Complete Link')
SSE_CompleteLink=sumofSquaredErrors(completeLinkClustPtsCell)



%2d.Use the correlation analysis to determine which of these two
% clusterings (obtained in (a) and (b)) has higher correlation
```

```
% Compute Spearman's rank correlation between the
% dissimilarities and the cophenetic distances

fprintf('\nCorrelation for single link \n');
[Cophenetic_Cor_SingleLink,D] = cophenet(singleLink,PairDistance);
r1 = corr(PairDistance',D','type','spearman');
fprintf('Cophenetic correlation coefficient %d\n',Cophenetic_Cor_SingleLink);

fprintf('\nCorrelation for complete link \n');
[Cophenetic_Cor_CompleteLink,D] = cophenet(completeLink,PairDistance);
r2 = corr(PairDistance',D','type','spearman');
fprintf('Cophenetic correlation coefficient
%d\n',Cophenetic_Cor_CompleteLink);


% correlation between the proximity martix and binary martix
fprintf('correlation between the proximity martix and binary martix for
single link %d\n',r1);
fprintf('correlation between the proximity martix and binary martix complete
link %d\n',r2);



%3a). DBSCAN algorithm
Xi=[1, 3, 5, 6, 8, 11, 12, 13, 14, 15, 16, 22, 28, 32, 33, 34, 35, 36, 37,
42, 58];
Xi=Xi';
Yi=repmat(0,1,21);
Yi=Yi';
Res=[Xi Yi];
epsilon=4;
MinPts=3;
IDX1=dbscan(Res,epsilon,MinPts);
figure();
PlotClusterinResult(Res, IDX1);
title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ', MinPts = '
num2str(MinPts) ')']);


%3b.Do the same as in (a) above but use Epsilon value of 6.

epsilon=6;
MinPts=3;
IDX=dbscan(Res,epsilon,MinPts);
figure();
PlotClusterinResult(Res, IDX);
title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ', MinPts = '
num2str(MinPts) ')']);


%3c)Compare the two clusterings using Rand Index and show all your work.
a=0;
b=0;
for indx=1:length(IDX1)-1
    x1=IDX1(indx);
    y1=IDX(indx);
```

```
    for jndx=indx+1:length(IDX)
        x2=IDX1(jndx);
        y2=IDX(jndx);
        if((x1==x2) && (y1==y2))
            a=a+1;
        elseif((x1~=x2) && (y1~=y2))
            b=b+1;
        end
    end

end

N1=nchoosek(length(IDX1),2);
RandIndex=(a+b)/N1
```

BasicSequentialAlgo.m

```
classdef BasicSequentialAlgo
    properties
        theta=[];       % threshold value
        q=[];           % maximum number of clusters constriant
        dataset={};     % dataset cell
        Cm={};          % dataset contains clusters means
        Ck={};          % dataset contains clusters points
        m=0;

    end

    methods
        %   Input:  threshold value and maximum no. of clusters
        %   Output: returns the BSAS obj

        function self=BasicSequentialAlgo(theta,q)
            self.m=1;
            self.theta=theta;
            self.q=q;
            self.Cm={};
            self.Ck={};
        end



        %   Input:  threshold value and maximum no. of clusters
        %   Output: returns the BSAS obj
        function [Ck,Cm] = PartitionDatasetIntoClusters(self,ds)
            self.dataset=ds;
            self.Cm(self.m)=self.dataset(1);
             self.Ck{self.m}=self.dataset(1);

            for i=2:length(self.dataset)
```

```
[distance,index]=findMinDistancetoMean(self,self.dataset(i),self.Cm);

%                 if(size(self.Ck,2)>=index)
%                     self.Ck{self.m}=[self.Ck{index},self.Cm(index)];
%
%                 else
%                     self.Ck(index)= {self.Cm(index)};
%                 end
%


                if((distance>self.theta) && (self.m<self.q))
                    self.m=self.m+1;
                    self.Cm(self.m)=self.dataset(i);
                     self.Ck{self.m}=self.dataset(i);
                else

                    if(size(self.Ck,2)>=self.m)
                        self.Ck{index}=[self.Ck{index},self.dataset(i)];
                    else
                        self.Ck(index)={self.dataset{i}};
                    end
                    %Find mean values of Clusters and change the mean of
                    %Clusters
                    for indx=1:length(self.Ck)
                        meanSet=self.Ck{indx};
                        x=0,y=0;
                        for subidx=1:length(meanSet)
                            x=x+meanSet{subidx}(1);
                            y=y+meanSet{subidx}(2);
                        end
                        newMean=[x/length(meanSet) y/length(meanSet)];
                        self.Cm(indx)={newMean};
                    end

                end
            end
            Ck=self.Ck;
            Cm=self.Cm;
        end

%   Input: instance vector and means set of clusters
        %   Output: returns minimum distance and its index value

        function [distance,index]=findMinDistancetoMean(self,Xi,Cm)

            for j=1:self.m
                curMean=Cm(j);
                distances(1,j)=sqrt((curMean{1}(1,1)-Xi{1}(1,1))^2+
(curMean{1}(1,2)-Xi{1}(1,2))^2);
            end

            [distance,index]=min(distances);
        end

    end
end
```

plotClusters.m

```
function[newSets]=plotClusters(Ck)
newSets={};
X=[],Y=[];
for idx=1:length(Ck)
    subSet=Ck{idx};
    x=[];
    y=[];
    for sidx=1:length(subSet)
        x(length(x)+1,1)=subSet{sidx}(1);
        y(length(y)+1,1)=subSet{sidx}(2);
    end
    newSets{idx}=[x y];
    X=vertcat(X,x);
    Y=vertcat(Y,y);
end
figure();
scatter(X,Y);
hold on
for rectId=1:length(newSets)
rectangle('Position', [min(newSets{rectId}(:,1)) min(newSets{rectId}(:,2))...
    max(newSets{rectId}(:,1))-min(newSets{rectId}(:,1))
max(newSets{rectId}(:,2))-min(newSets{rectId}(:,2))]);

end

end
```

PlotClusterinResult.m

```
function PlotClusterinResult(X, IDX)


    k=max(IDX);

    Colors=hsv(k);

    Legends = {};
    for i=0:k
        Xi=X(IDX==i,:);
        if i~=0
            Style = 'x';
            MarkerSize = 8;
            Color = Colors(i,:);
            Legends{end+1} = ['Cluster #' num2str(i)];
        else
            Style = 'o';
            MarkerSize = 6;
            Color = [0 0 0];
            if ~isempty(Xi)
                Legends{end+1} = 'Noise';
```

```
            end
        end
        if ~isempty(Xi)

plot(Xi(:,1),Xi(:,2),Style,'MarkerSize',MarkerSize,'Color',Color);
        end
        hold on;
    end
    hold off;
    axis equal;
    grid on;
    legend(Legends);
    legend('Location', 'NorthEastOutside');

end
```

sumofSquaredErrors.m

```
function [SSE]=sumofSquaredErrors(singleLinkClustPtsCell)
arrSSE=[];
for indx=1:length(singleLinkClustPtsCell)
    meanSet=singleLinkClustPtsCell{indx};

    newMean=[sum(meanSet(:,1))/size(meanSet,1)
sum(meanSet(:,2))/size(meanSet,1)];

    meanDist=0;
    for subidx=1:size(meanSet,1)

        distn=sqrt((meanSet(subidx,1)-newMean(1,1))^2+ (meanSet(subidx,2)-
newMean(1,2))^2);

        meanDist=meanDist+distn;
    end
    arrSSE(1,indx)=meanDist;
end
 SSE=sum(arrSSE);
end
```

Dbscan.m

```
function [IDX, isnoise]=dbscan(X,epsilon,MinPts)
    C=0;
    n=size(X,1);
    IDX=zeros(n,1);

    D=pdist2(X,X);

    visited=false(n,1);
    isnoise=false(n,1);

    for i=1:n
        if ~visited(i)
            visited(i)=true;

            Neighbors=RegionQuery(i);
            if numel(Neighbors)<MinPts
                % X(i,:) is NOISE
                isnoise(i)=true;
            else
                C=C+1;
                ExpandCluster(i,Neighbors,C);
            end

        end

    end
    function ExpandCluster(i,Neighbors,C)
        IDX(i)=C;
        k = 1;
        while true
            j = Neighbors(k);

            if ~visited(j)
                visited(j)=true;
                Neighbors2=RegionQuery(j);
                if numel(Neighbors2)>=MinPts
                    Neighbors=[Neighbors Neighbors2];   %#ok
                end
            end
            if IDX(j)==0
                IDX(j)=C;
            end

            k = k + 1;
            if k > numel(Neighbors)
                break;
            end
        end
    end

    function Neighbors=RegionQuery(i)
        Neighbors=find(D(i,:)<=epsilon);
    end
end
```