# Cluster Analysis

# Cluster Analysis

- Unsupervised classification

- Aims to decompose or partition a data set in to clusters such that each cluster is similar within itself but is as dissimilar as possible to other clusters.

- Inter-cluster (between-groups) distance is maximized and intra-cluster (within-group) distance is minimized.

- Different to classification as there are no predefined classes, no training data, may not even know how many clusters.

# Cluster Analysis

- Not a new field, a branch of statistics
- Many old algorithms are available
- Renewed interest due to data mining and machine learning
- New algorithms are being developed, in particular to deal with large amounts of data
- Evaluating a clustering method is difficult

# Classification Example

(From text by Roiger and Geatz)

| Sore Throat | Fever | Swollen Glands | Congestion | Headache | Diagnosis |
|---|---|---|---|---|---|
| Yes | Yes | Yes | Yes | Yes | Strep throat |
| No | No | No | Yes | Yes | Allergy |
| Yes | Yes | No | Yes | No | Cold |
| Yes | No | Yes | No | No | Strep throat |
| No | Yes | No | Yes | No | Cold |
| No | No | No | Yes | No | Allergy |
| No | No | Yes | No | No | Strep throat |
| Yes | No | No | Yes | Yes | Allergy |
| No | Yes | No | Yes | Yes | Cold |
| Yes | Yes | No | Yes | Yes | Cold |

# Clustering Example

(From text by Roiger and Geatz)

| Sore | | Swollen | | |
|---|---|---|---|---|
| Throat | Fever | Glands | Congestion | Headache |
| Yes | Yes | Yes | Yes | Yes |
| No | No | No | Yes | Yes |
| Yes | Yes | No | Yes | No |
| Yes | No | Yes | No | No |
| No | Yes | No | Yes | No |
| No | No | No | Yes | No |
| No | No | Yes | No | No |
| Yes | No | No | Yes | Yes |
| No | Yes | No | Yes | Yes |
| Yes | Yes | No | Yes | Yes |

# Cluster Analysis

- As in classification, each object has several attributes.
- Some methods require that the number of clusters is specified by the user and a seed to start each cluster be given.
- The objects are then clustered based on self-similarity.

# Questions

- How to define a cluster?
- How to find if objects are similar or not?
- How to define some concept of distance between individual objects and sets of objects?

# Types of Data

- Interval-scaled data - continuous variables on a roughly linear scale e.g. marks, age, weight. Units can affect the analysis. Distance in metres is obviously a larger number than in kilometres. May need to scale or normalise data.

- Binary data – many variables are binary e.g. gender, married or not. How to compute distance between binary variables?

# Types of Data

- Nominal data - similar to binary but can take more than two states e.g. colour, staff position. How to compute distance between two objects with nominal data?

- Ordinal data - similar to nominal but the different values are ordered in a meaningful sequence

- Ratio-scaled data - nonlinear scale data

# Distance

A simple, well-understood concept. Distance has the following properties (assume x and y to be vectors):

- distance is always positive
- distance x to x is zero
- distance x to y cannot be greater than the sum of distance x to z and z to y
- distance x to y is the same as from y to x.

Examples: absolute value of the difference ∑|x-y| (the Manhattan distance ) or ∑(x-y)**2 (the Euclidean distance).

# Distance

Three common distance measures are;

1. Manhattan Distance or the absolute value of the difference

$$D(x, y) = \sum | x-y |$$

Or

The Manhattan Distance between two points (X1, Y1) and (X2, Y2) is given by |X1 – X2| + |Y1-Y2|.

2. Euclidean distance

$$D(x, y) = ( \sum(x-y)^2 )^{½}$$

3. Maximum difference

$$D(x, y) = \max_i | x_i – y_i |$$

# Desired Features of Cluster Analysis

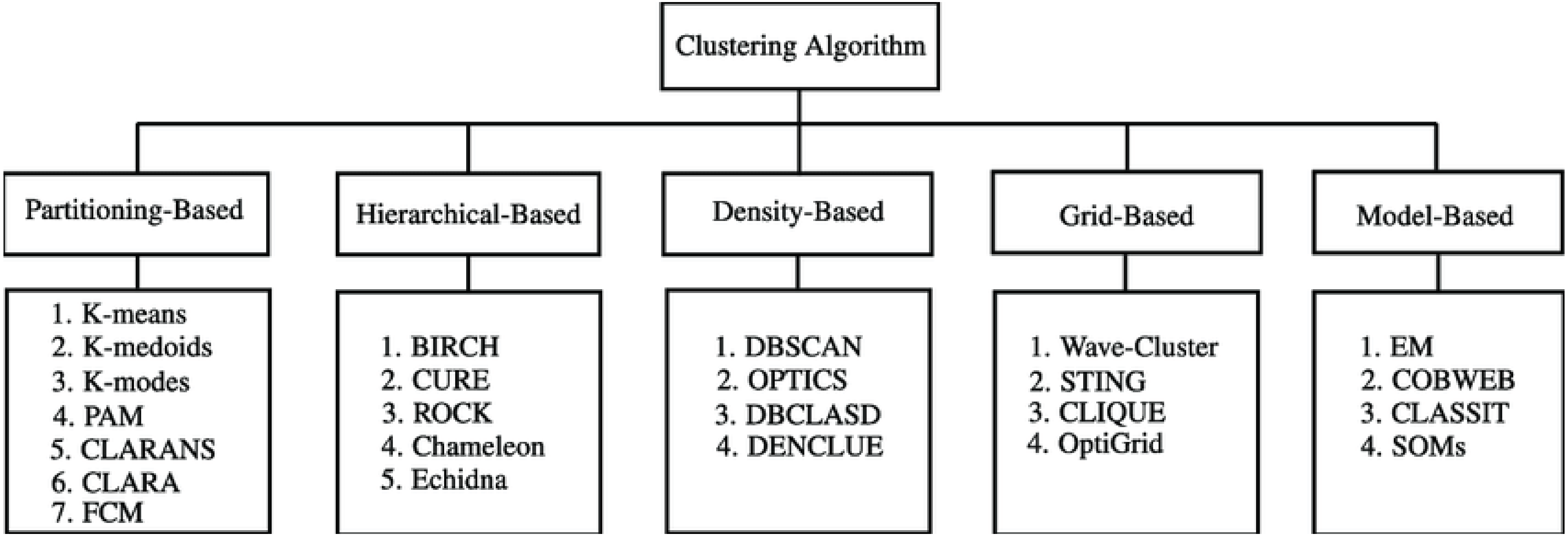
- Scalability
- Only one scan of dataset
- Ability to stop and resume
- Minimum input parameters
- Robustness
- Ability to discover different cluster shapes
- Ability to deal with Different Data Types
- Independent from the order of input

# Clustering

It is not always easy to predict how many clusters to expect from a set of data.

One may choose a number of clusters based on some knowledge of data. The result may be an acceptable set of clusters. This does not however mean that a different number of clusters will not provide an even better set of clusters. Choosing the number of clusters is a difficult problem.

# Types of Methods



Clustering Algorithm

Partitioning-Based

1. K-means
2. K-medoids
3. K-modes
4. PAM
5. CLARANS
6. CLARA
7. FCM

Hierarchical-Based

1. BIRCH
2. CURE
3. ROCK
4. Chameleon
5. Echidna

Density-Based

1. DBSCAN
2. OPTICS
3. DBCLASD
4. DENCLUE

Grid-Based

1. Wave-Cluster
2. STING
3. CLIQUE
4. OptiGrid

Model-Based

1. EM
2. COBWEB
3. CLASSIT
4. SOMs

# Types of Methods

- Partitioning methods – given n objects, make k (≤n) partitions (clusters) of data and use iterative relocation method. It is assumed that each cluster has at least one object and each object belongs to only one cluster.

- Hierarchical methods - start with one cluster and then split it in smaller clusters or start with each object in an individual cluster and then try to merge similar clusters.

# Types of Methods

- Density-based methods - for each data point in a cluster, at least a minimum number of points must exist within a given radius

- Grid-based methods - object space is divided into a grid

- Model-based methods - a model is assumed, perhaps based on a probability distribution

# The K-Means Method

Perhaps the most commonly used method.

The method involves choosing the number of clusters and then dividing the given n objects choosing k seeds randomly as starting samples of these clusters.

Once the seeds have been specified, each member is assigned to a cluster that is closest to it based on some distance measure. Once all the members have been allocated, the mean value of each cluster is computed and these means essentially become the new seeds.

# The K-Means Method

Using the mean value of each cluster, all the members are now re-allocated to the clusters. In most situations, some members will change clusters unless the first guesses of seeds were very good.

This process continues until no changes take place to cluster memberships.

Different starting seeds obviously may lead to different clusters.

# The K-Means Method

Essentially the algorithm tries to build cluster with high level of similarity within clusters and low level of similarity between clusters.

This method requires means of variables to be computed.

The method is scalable and is efficient. The algorithm does not find a global minimum, it rather terminates at a local minimum.

# Example

| Student | Age | Marks1 | Marks2 | Marks3 |
|---|---|---|---|---|
| S1 | 18 | 73 | 75 | 57 |
| S2 | 18 | 79 | 85 | 75 |
| S3 | 23 | 70 | 70 | 52 |
| S4 | 20 | 55 | 55 | 55 |
| S5 | 22 | 85 | 86 | 87 |
| S6 | 19 | 91 | 90 | 89 |
| S7 | 20 | 70 | 65 | 65 |
| S8 | 21 | 53 | 56 | 59 |
| S9 | 19 | 82 | 82 | 60 |
| S10 | 40 | 76 | 60 | 78 |

# Example

Let the three seeds be the first three records:

| Student | Age | Mark1 | Mark2 | Mark3 |
|---|---|---|---|---|
| S1 | 18 | 73 | 75 | 57 |
| S2 | 18 | 79 | 85 | 75 |
| S3 | 23 | 70 | 70 | 52 |

Now we compute the distances between the objects based on the four attributes. K-Means requires Euclidean distances but we use the sum of absolute differences as well as to show how the distance metric can change the results. Next page table shows the distances and their allocation to the nearest neighbor.

# Distances

| Student | Age | Marks1 | Marks2 | Marks3 | Manhattan | | Euclidean | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Distance | | Distance | |
| S1 | 18 | 73 | 75 | 57 | Seed 1 | C1 | Seed 1 | C1 |
| S2 | 18 | 79 | 85 | 75 | Seed 2 | C2 | Seed 2 | C2 |
| S3 | 23 | 70 | 70 | 52 | Seed 3 | C3 | Seed 3 | C3 |
| S4 | 20 | 55 | 55 | 55 | 42/76/36 | C3 | 27/43/22 | C3 |
| S5 | 22 | 85 | 86 | 87 | 57/23/67 | C2 | 34/14/41 | C2 |
| S6 | 19 | 91 | 90 | 89 | 66/32/82 | C2 | 40/19/47 | C2 |
| S7 | 20 | 70 | 65 | 60 | 23/41/21 | C3 | 13/24/14 | C1 |
| S8 | 21 | 53 | 56 | 59 | 44/74/40 | C3 | 28/42/23 | C3 |
| S9 | 19 | 82 | 82 | 60 | 20/22/36 | C1 | 12/16/19 | C1 |
| S10 | 47 | 75 | 76 | 77 | 60/52/58 | C2 | 33/34/32 | C3 |

# Example

The means of the new clusters are also different as shown on the next slide. We now start with the new means and compute Euclidean distances. The process continues until there is no change.

# Cluster Means

| | Cluster | Age | Mark1 | Mark2 | Mark3 |
|---|---|---|---|---|---|
| | C1 | 18.5 | 77.5 | 78.5 | 58.5 |
| | C2 | 24.8 | 82.8 | 80.3 | 82 |
| | C3 | 21 | 62 | 61.5 | 57.8 |
| | Seed 1 | 18 | 73 | 75 | 57 |
| | Seed 2 | 18 | 79 | 85 | 75 |
| | Seed 3 | 23 | 70 | 70 | 52 |

# Distances

| Student | Age | Marks1 | Marks2 | Marks3 | Distance | Cluster |
|---|---|---|---|---|---|---|
| C1 | 18.5 | 77.5 | 78.5 | 58.5 | | |
| C2 | 24.8 | 82.8 | 80.3 | 82 | | |
| C3 | 21 | 62 | 61.5 | 57.8 | | |
| S1 | 18 | 73 | 75 | 57 | 8/52/36 | C1 |
| S2 | 18 | 79 | 85 | 75 | 30/18/63 | C2 |
| S3 | 23 | 70 | 70 | 52 | 22/67/28 | C1 |
| S4 | 20 | 55 | 55 | 55 | 46/91/26 | C3 |
| S5 | 22 | 85 | 86 | 87 | 51/7/78 | C2 |
| S6 | 19 | 91 | 90 | 89 | 60/15/93 | C2 |
| S7 | 20 | 70 | 65 | 60 | 19/56/22 | C1 |
| S8 | 21 | 53 | 56 | 59 | 44/89/22 | C3 |
| S9 | 19 | 82 | 82 | 60 | 16/32/48 | C1 |
| S10 | 47 | 75 | 76 | 77 | 52/63/43 | C3 |

# Comments

- The last iteration changed the cluster membership of S3 from C3 to C1. New means are now computed followed by computation of new distances. The process continues until there is no change.

- Finally
- Cluster 1: s1, s9
- Cluster 2: s2,s5,s6,s10
- Cluster 3: s3, s4, s7, s8

# Advantages of K-Means

- Relatively simple to implement
- Scale to large data set
- Guarantees Convergence
- Easily adapt to new examples
- Generalize to cluster of different shapes and size

# Disadvantages of K-Means

- Choosing K manually
- Being dependent on initial value of k
- Has trouble clustering data where clusters are of variable sizes and density
- Clustering outliers- centroid can be dragged by outliers
- Scaling with number of dimensions- need to reduce dimensions

# Hierarchical Clustering

Quite different than partitioning. Involves gradually merging different objects into clusters (called agglomerative) or dividing large clusters into smaller ones (called divisive).

We are considering one method- agglomerative.

# Agglomerative Clustering

The algorithm normally starts with each cluster consisting of a single data point. Using a measure of distance, the algorithm merges two clusters that are nearest, thus reducing the number of clusters. The process continues until all the data points are in one cluster.

The algorithm requires that we be able to compute distances between two objects and also between two clusters.

# Distance between Clusters

Single Link method – nearest neighbour - distance between two clusters is the distance between the two closest points, one from each cluster. Chains can be formed (why?) and a long string of points may be assigned to the same cluster.

Complete Linkage method – furthest neighbour – distance between two clusters is the distance between the two furthest points, one from each cluster. Does not allow chains.

# Distance between Clusters

Centroid method – distance between two clusters is the distance between the two centroids or the two centres of gravity of the two clusters.

Unweighted pair-group average method –distance between two clusters is the average distance between all pairs of objects in the two clusters.  This means p*n distances need to be computed if p and n are the number of objects in each of the clusters

# Algorithm

Each object is assigned to a cluster of its own

Build a distance matrix by computing distances between every pair of objects

Find the two nearest clusters

Merge them and remove them from the list

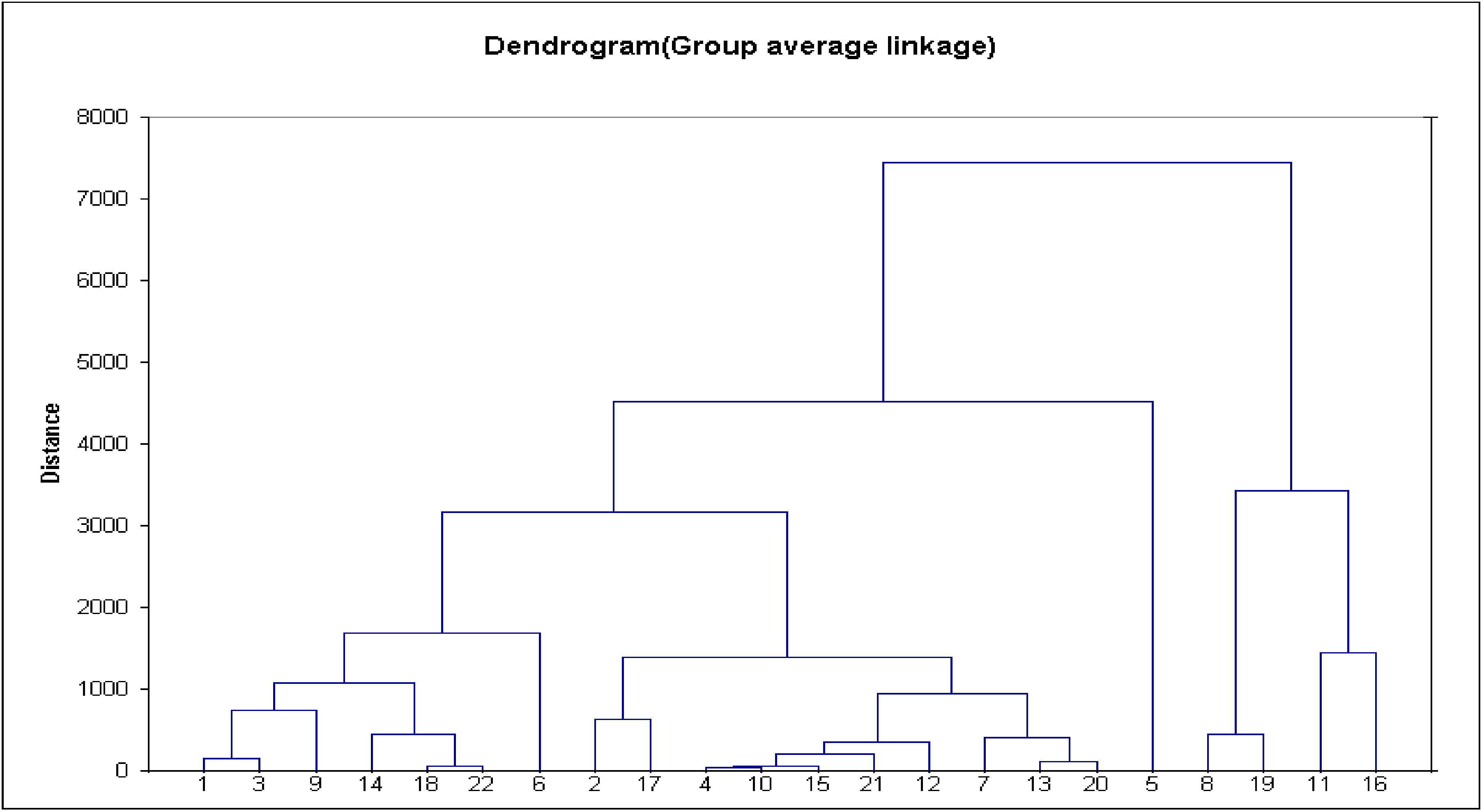
Update matrix by including distance of all objects from the new cluster

Continue until all objects belong to one cluster

The next slide shows an example of the final result of hierarchical clustering.

# Hierarchical Clustering

http://www.resample.com/xlminer/help/HClst/HClst_ex.htm



# Example

We now consider the simple example about students’ marks and use the distance between two objects as the Manhattan distance.

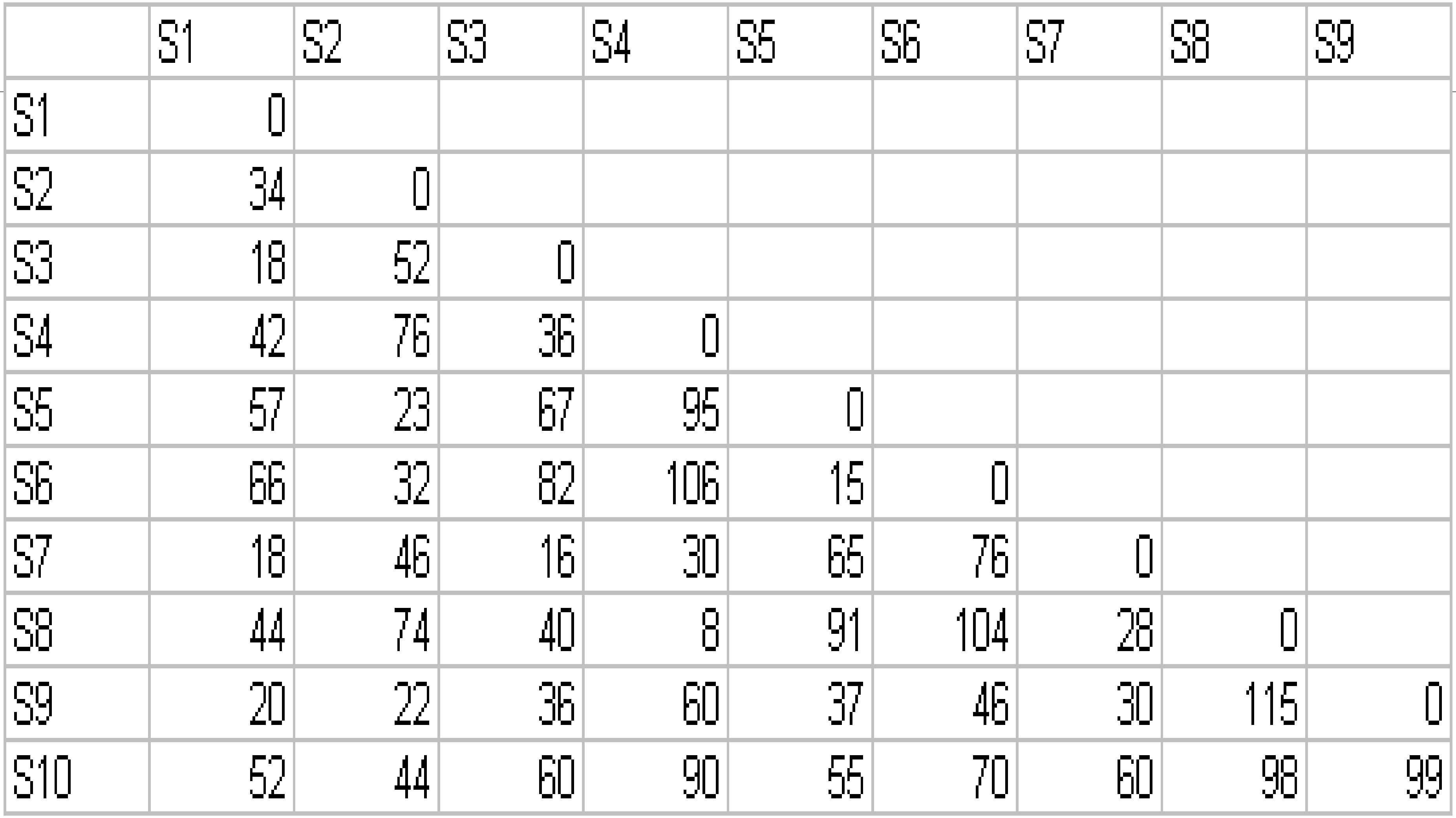
We will use the centroid method for distance between clusters.

We first calculate a matrix of distances.

# Example

| Student | Age | Marks1 | Marks2 | Marks3 |
|---|---|---|---|---|
| 971234 | 18 | 73 | 75 | 57 |
| 994321 | 18 | 79 | 85 | 75 |
| 965438 | 23 | 70 | 70 | 52 |
| 987654 | 20 | 55 | 55 | 55 |
| 968765 | 22 | 85 | 86 | 87 |
| 978765 | 19 | 91 | 90 | 89 |
| 984567 | 20 | 70 | 65 | 60 |
| 985555 | 21 | 53 | 56 | 59 |
| 998888 | 19 | 82 | 82 | 60 |
| 995544 | 47 | 75 | 76 | 77 |

# Distance Matrix



| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 |
|---|---|---|---|---|---|---|---|---|---|
| S1 | 0 | | | | | | | | |
| S2 | 34 | 0 | | | | | | | |
| S3 | 18 | 52 | 0 | | | | | | |
| S4 | 42 | 76 | 36 | 0 | | | | | |
| S5 | 57 | 23 | 67 | 95 | 0 | | | | |
| S6 | 66 | 32 | 82 | 106 | 15 | 0 | | | |
| S7 | 18 | 46 | 16 | 30 | 65 | 76 | 0 | | |
| S8 | 44 | 74 | 40 | 8 | 91 | 104 | 28 | 0 | |
| S9 | 20 | 22 | 36 | 60 | 37 | 46 | 30 | 115 | 0 |
| S10 | 52 | 44 | 60 | 90 | 55 | 70 | 60 | 98 | 99 |

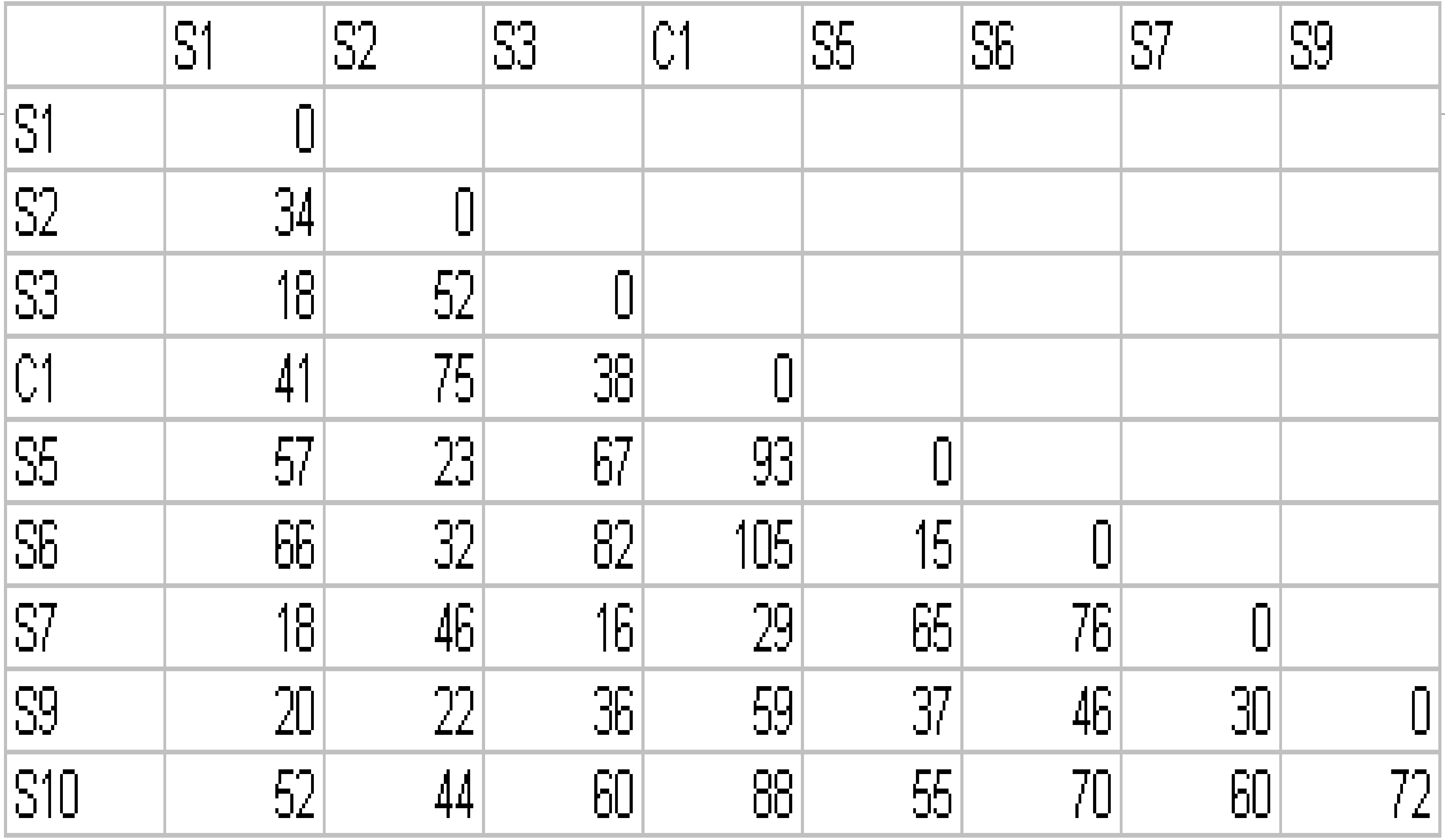
# Example

The matrix gives distance of each object with every other object.

The smallest distance of 8 is between object 4 and object 8. They are combined and put where object 4 was.

Compute distances from this cluster and update the distance matrix.

# Updated Matrix



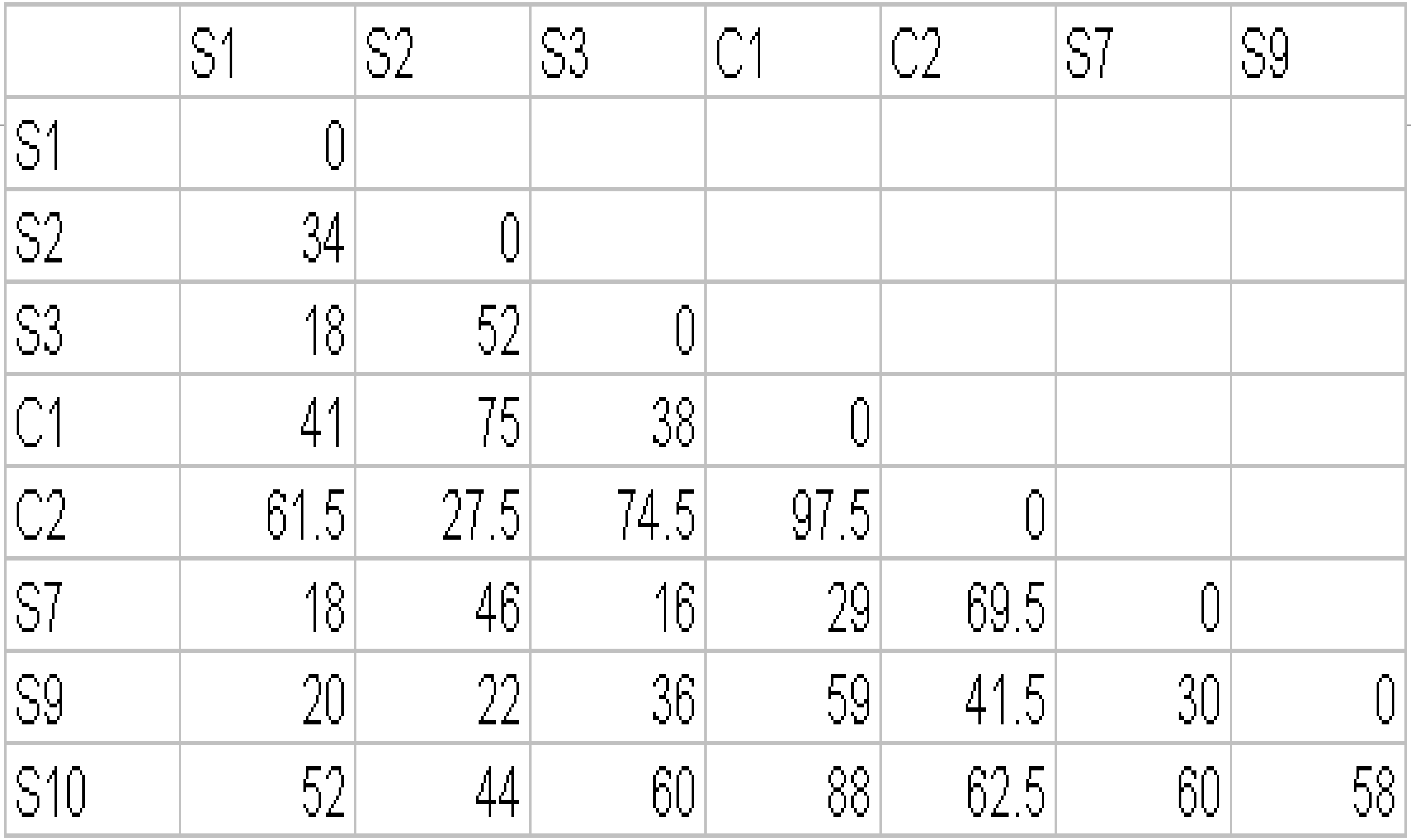
| | S1 | S2 | S3 | C1 | S5 | S6 | S7 | S9 |
|---|---|---|---|---|---|---|---|---|
| S1 | 0 | | | | | | | |
| S2 | 34 | 0 | | | | | | |
| S3 | 18 | 52 | 0 | | | | | |
| C1 | 41 | 75 | 38 | 0 | | | | |
| S5 | 57 | 23 | 67 | 93 | 0 | | | |
| S6 | 66 | 32 | 82 | 105 | 15 | 0 | | |
| S7 | 18 | 46 | 16 | 29 | 65 | 76 | 0 | |
| S9 | 20 | 22 | 36 | 59 | 37 | 46 | 30 | 0 |
| S10 | 52 | 44 | 60 | 88 | 55 | 70 | 60 | 72 |

# Note

The smallest distance now is 15 between the objects 5 and 6. They are combined in a cluster and 5 and 6 are removed.

Compute distances from this cluster and update the distance matrix.

# Updated Matrix



| | S1 | S2 | S3 | C1 | C2 | S7 | S9 |
|---|---|---|---|---|---|---|---|
| S1 | 0 | | | | | | |
| S2 | 34 | 0 | | | | | |
| S3 | 18 | 52 | 0 | | | | |
| C1 | 41 | 75 | 38 | 0 | | | |
| C2 | 61.5 | 27.5 | 74.5 | 97.5 | 0 | | |
| S7 | 18 | 46 | 16 | 29 | 69.5 | 0 | |
| S9 | 20 | 22 | 36 | 59 | 41.5 | 30 | 0 |
| S10 | 52 | 44 | 60 | 88 | 62.5 | 60 | 58 |

# Next

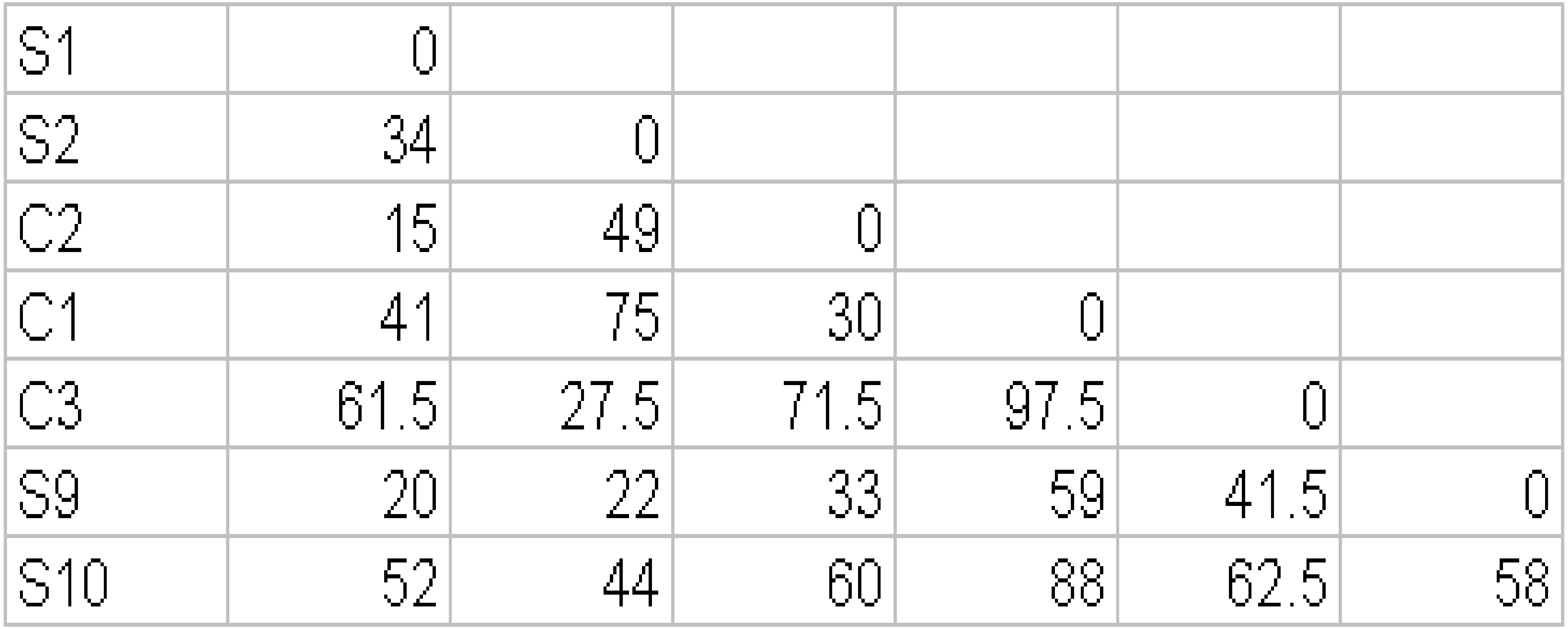
Look at shortest distance again.

S3 and S7 are at a distance 16 apart. We merge them and put C3 where S3 was.

The updated distance matrix is given on the next slide. It shows that C2 and S1 have the smallest distance and are then merged in the next step.

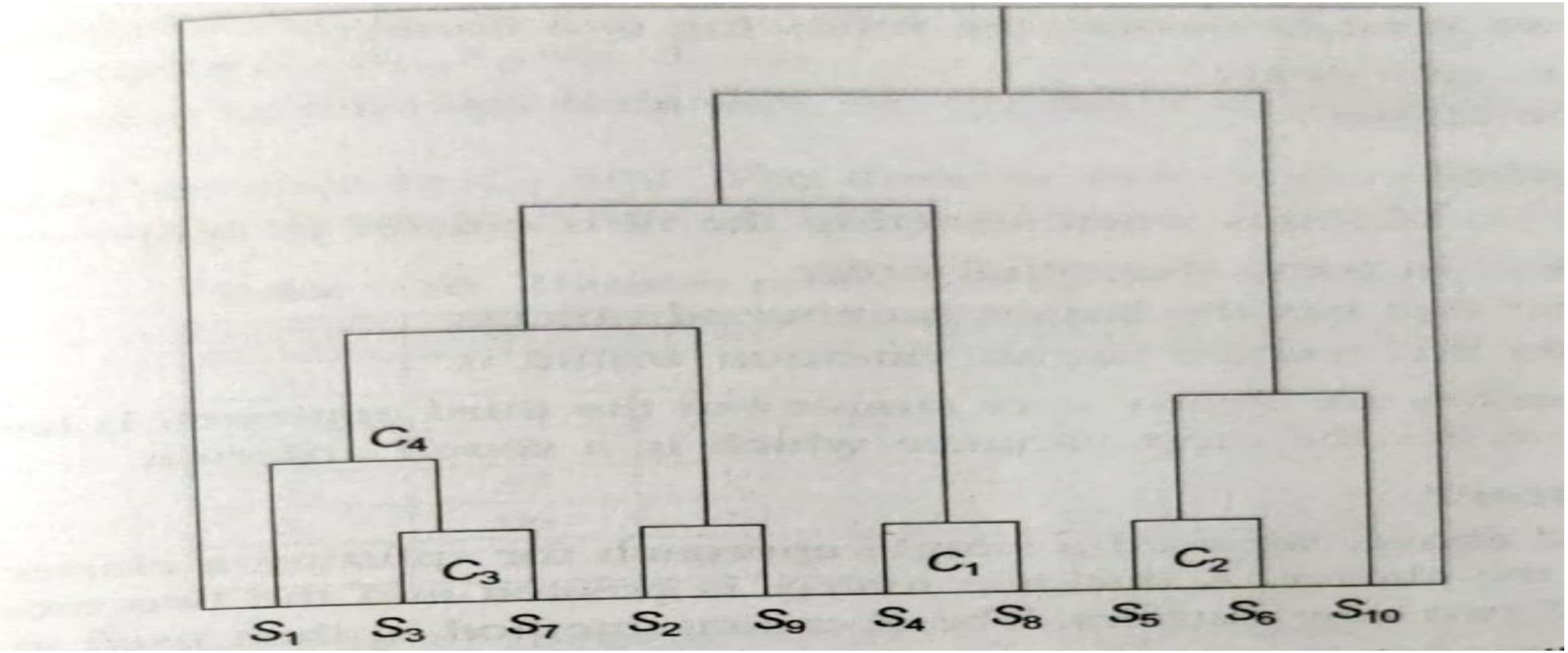
We stop short of finishing the example.

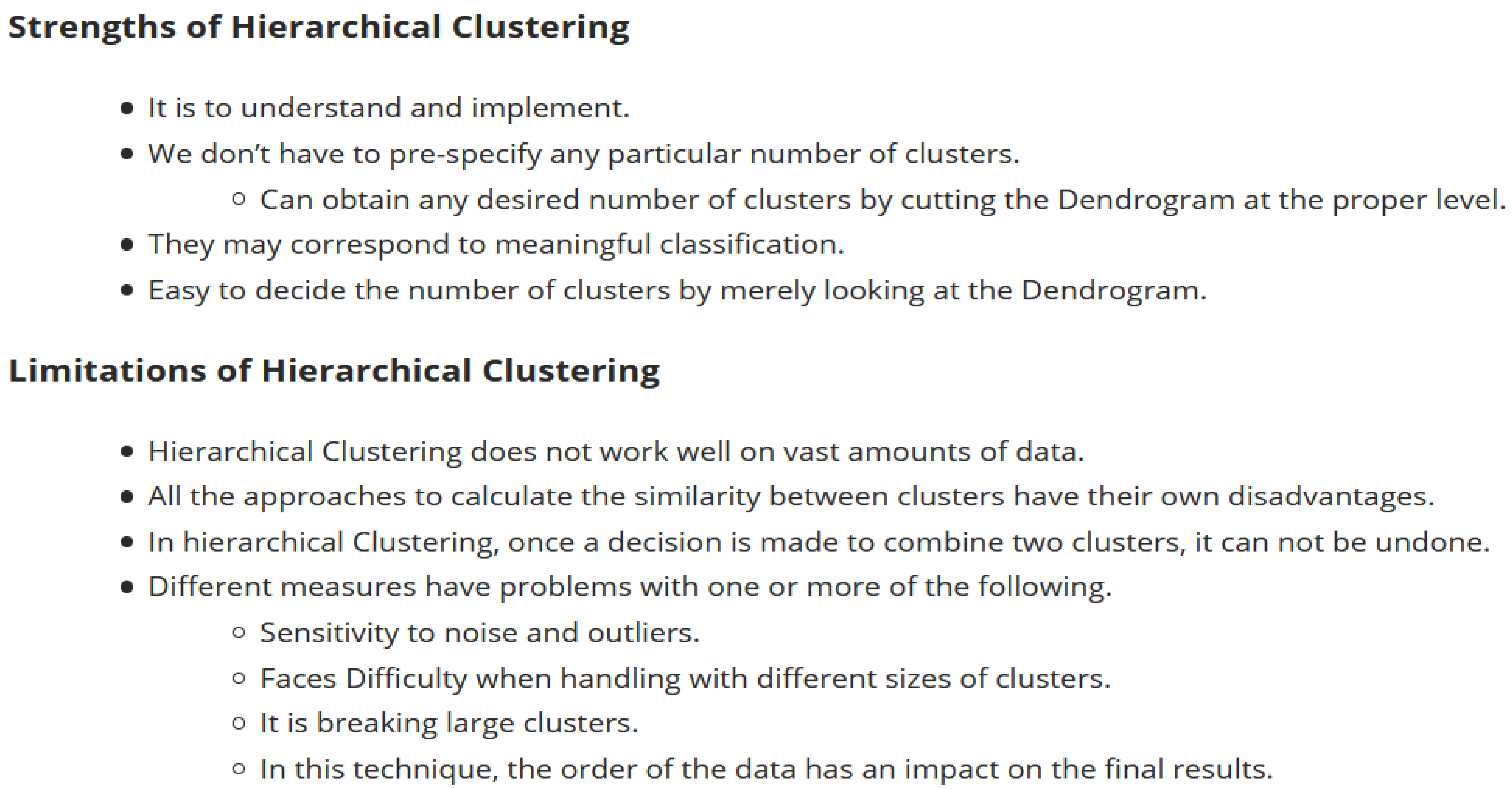
# Updated matrix



| | | | | | | |
|---|---|---|---|---|---|---|
| S1 | 0 | | | | | |
| S2 | 34 | 0 | | | | |
| C2 | 15 | 49 | 0 | | | |
| C1 | 41 | 75 | 30 | 0 | | |
| C3 | 61.5 | 27.5 | 71.5 | 97.5 | 0 | |
| S9 | 20 | 22 | 33 | 59 | 41.5 | 0 |
| S10 | 52 | 44 | 60 | 88 | 62.5 | 58 |

# Final Result





## Strengths of Hierarchical Clustering

- It is to understand and implement.
- We don't have to pre-specify any particular number of clusters.
  - Can obtain any desired number of clusters by cutting the Dendrogram at the proper level.
- They may correspond to meaningful classification.
- Easy to decide the number of clusters by merely looking at the Dendrogram.

## Limitations of Hierarchical Clustering

- Hierarchical Clustering does not work well on vast amounts of data.
- All the approaches to calculate the similarity between clusters have their own disadvantages.
- In hierarchical Clustering, once a decision is made to combine two clusters, it can not be undone.
- Different measures have problems with one or more of the following.
  - Sensitivity to noise and outliers.
  - Faces Difficulty when handling with different sizes of clusters.
  - It is breaking large clusters.
  - In this technique, the order of the data has an impact on the final results.

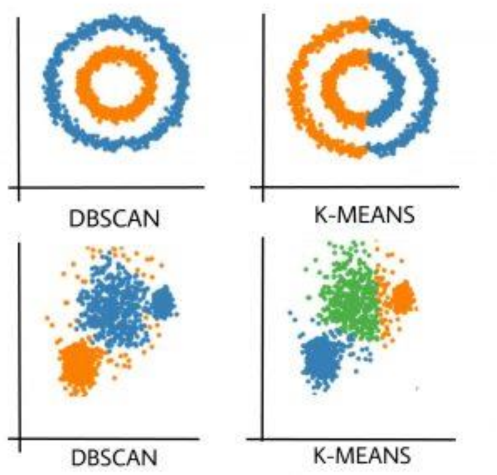
# DBSCAN (Density-based spatial clustering of applications with noise)

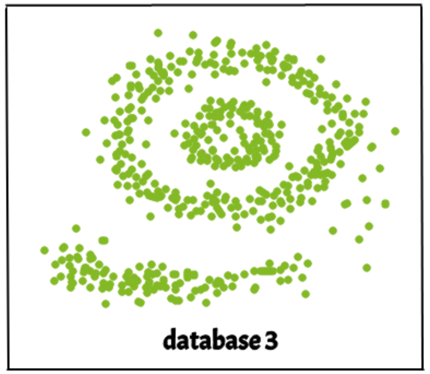
Clusters are dense regions in the data space, separated by regions of the lower density of points.

The **DBSCAN algorithm** is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

# Why DBSCAN?

- Partitioning methods (K-means) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters.
- In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.
- Real life data may contain irregularities, like: Clusters can be of arbitrary shape such as those shown in the figure below, Data may contain noise, Data set may contain nonconvex clusters
- Difficult to handle with K-means
- K-Means forms spherical clusters only. This algorithm fails when data is not spherical ( i.e. same variance in all directions).





# DBSCAN algorithm requires two parameters:

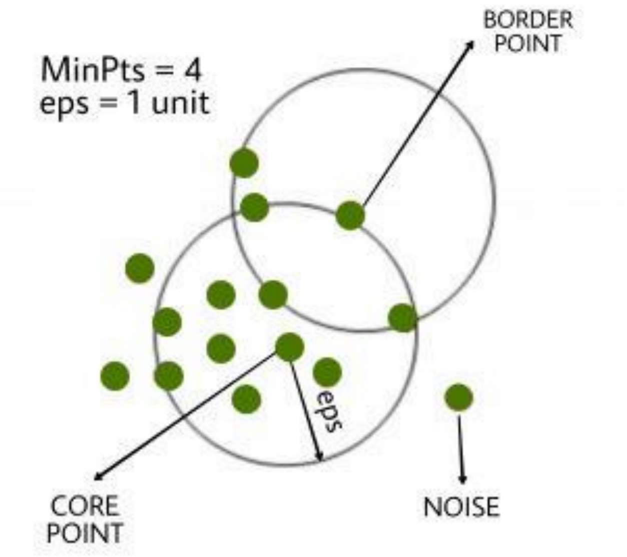
- **eps :** It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to ‘eps’ then they are considered as neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and majority of the data points will be in the same clusters. One way to find the eps value is based on the k-distance graph.
- **MinPts:** Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

# *In this algorithm, we have 3 types of data points.*

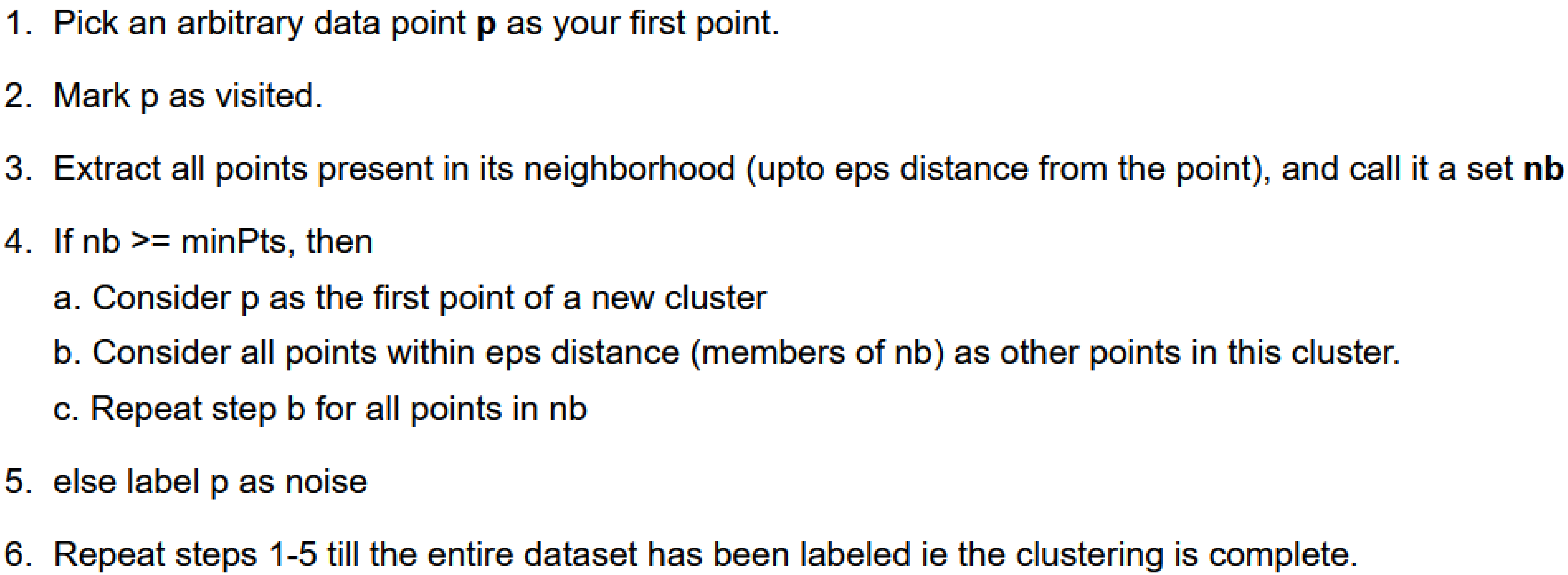
**Core Point:** A point is a core point if it has more than MinPts points within eps.

**Border Point:** A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.

**Noise or outlier:** A point which is not a core point or border point



# DBSCAN algorithm can be abstracted in the following steps :



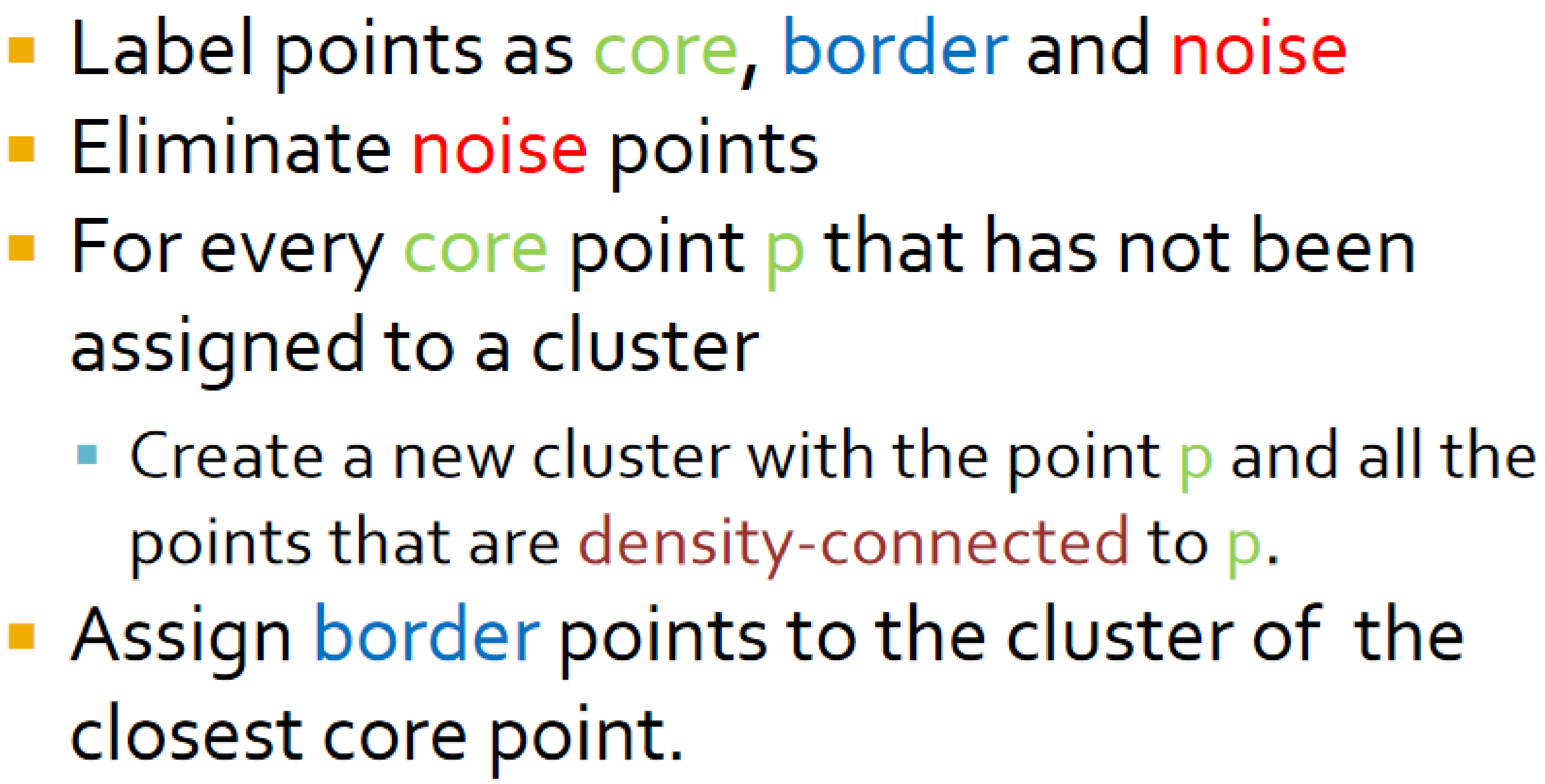
1. Pick an arbitrary data point **p** as your first point.
2. Mark p as visited.
3. Extract all points present in its neighborhood (upto eps distance from the point), and call it a set **nb**
4. If nb >= minPts, then

   a. Consider p as the first point of a new cluster

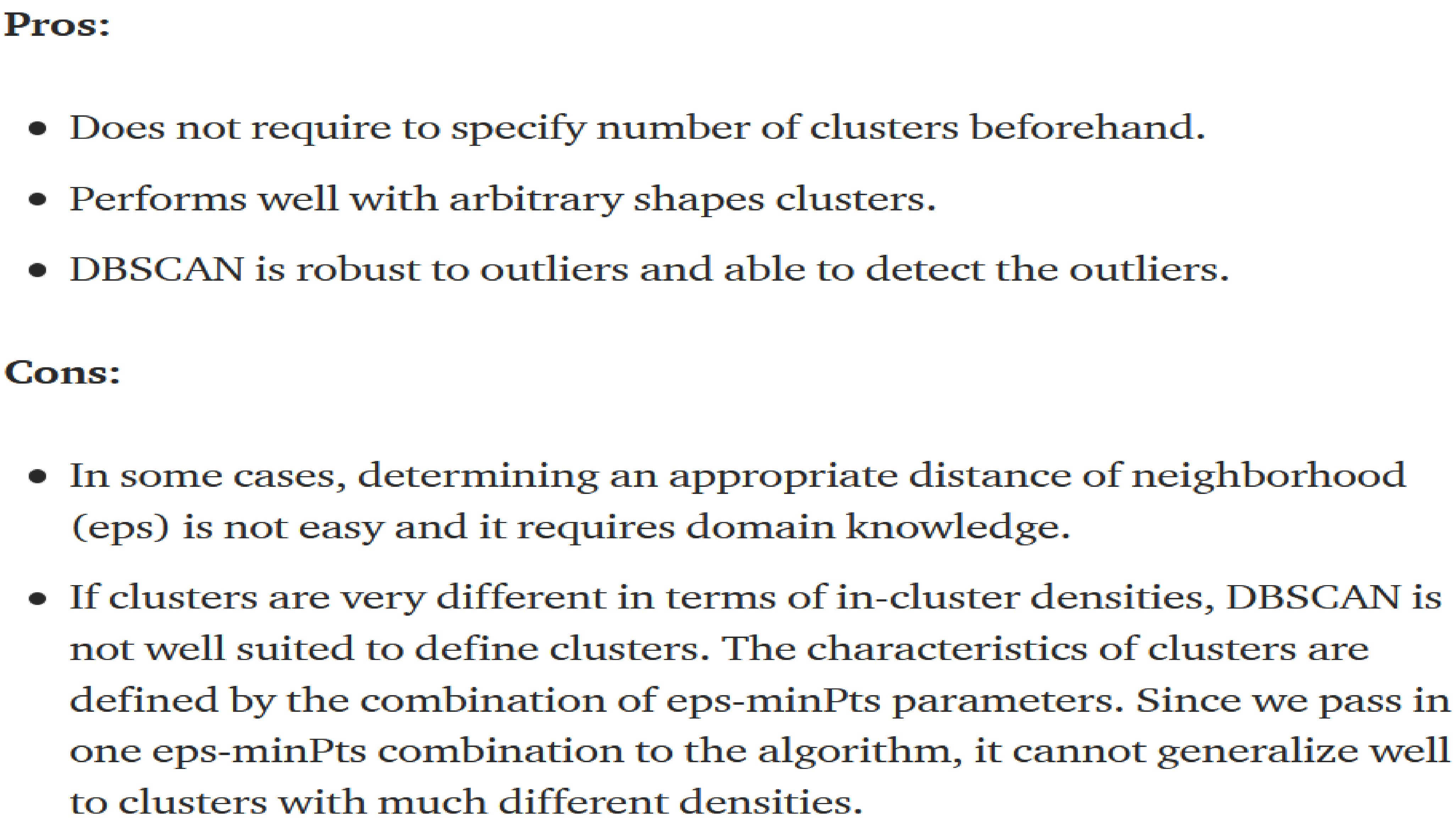
   b. Consider all points within eps distance (members of nb) as other points in this cluster.

   c. Repeat step b for all points in nb
5. else label p as noise
6. Repeat steps 1-5 till the entire dataset has been labeled ie the clustering is complete.

# Or In simple step



- Label points as core, border and noise
- Eliminate noise points
- For every core point p that has not been assigned to a cluster
  - Create a new cluster with the point p and all the points that are density-connected to p.
- Assign border points to the cluster of  the closest core point.



**Pros:**

- Does not require to specify number of clusters beforehand.
- Performs well with arbitrary shapes clusters.
- DBSCAN is robust to outliers and able to detect the outliers.

**Cons:**

- In some cases, determining an appropriate distance of neighborhood (eps) is not easy and it requires domain knowledge.
- If clusters are very different in terms of in-cluster densities, DBSCAN is not well suited to define clusters. The characteristics of clusters are defined by the combination of eps-minPts parameters. Since we pass in one eps-minPts combination to the algorithm, it cannot generalize well to clusters with much different densities.