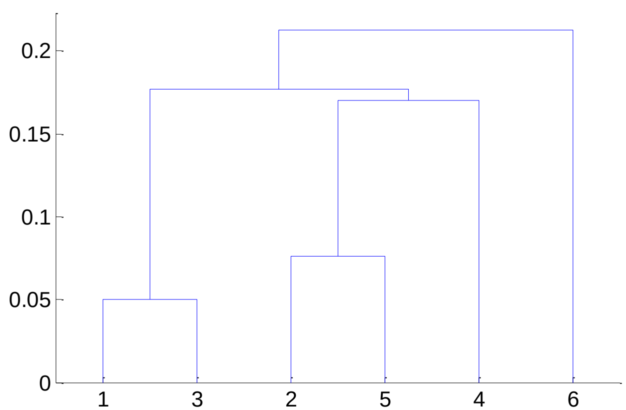
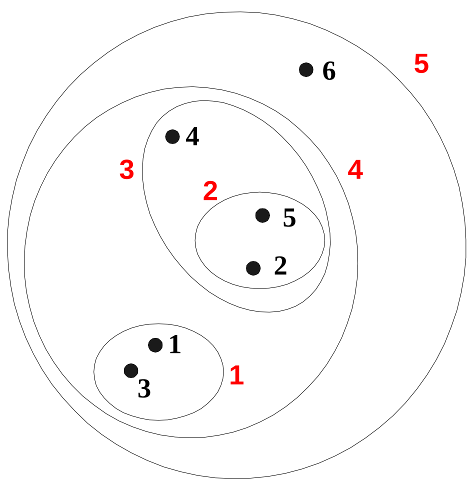
Hierarchical Clustering

Hierarchical Clustering

- Produces a set of ***nested clusters*** organized as a hierarchical tree
- Can be visualized as a **dendrogram**
  - A tree-like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- No assumptions on the number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level

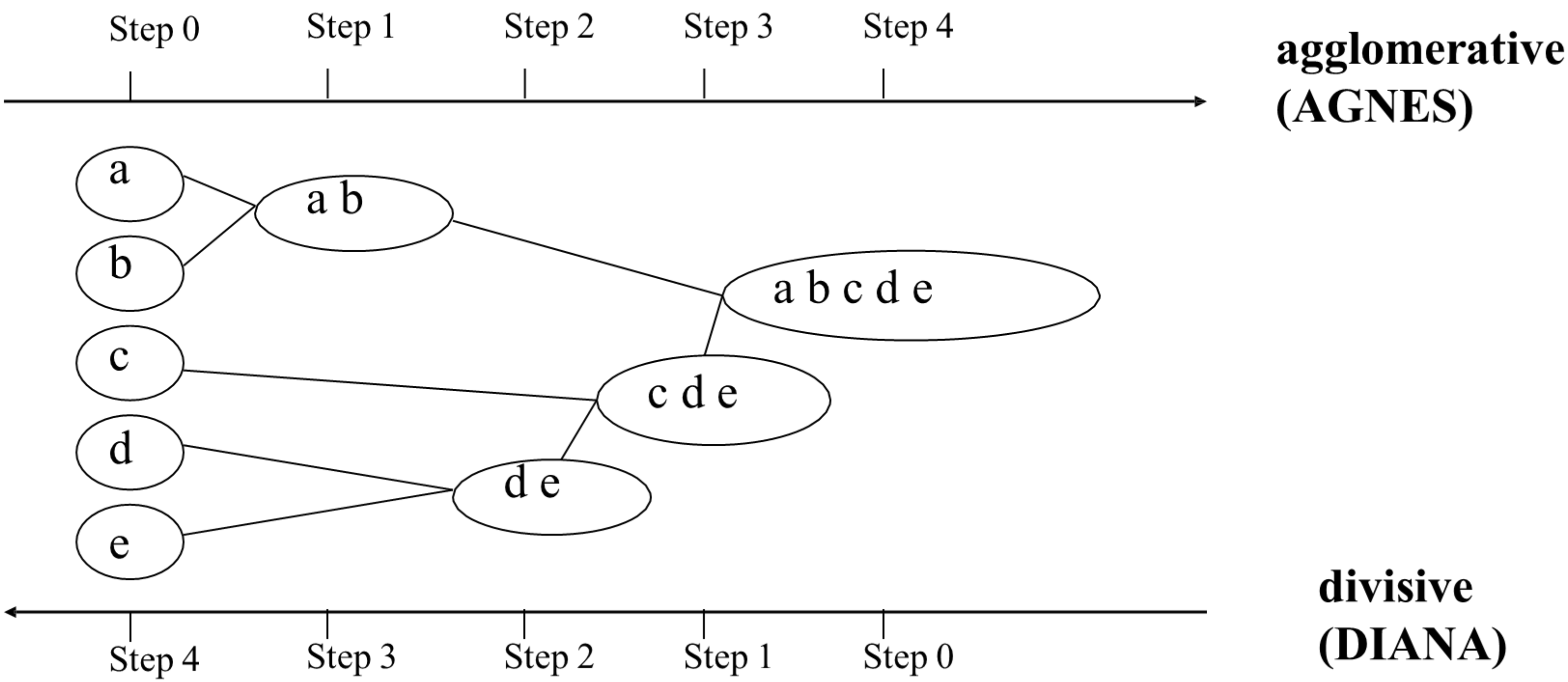
- Hierarchical clusterings may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs) etc

Hierarchical Clustering

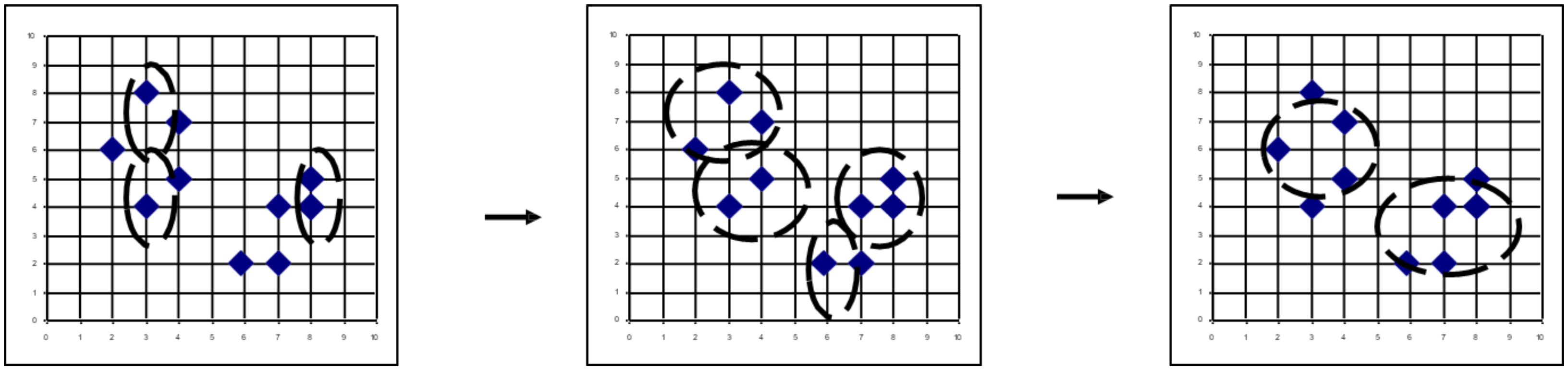
- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or **k** clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are **k** clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Hierarchical Clustering

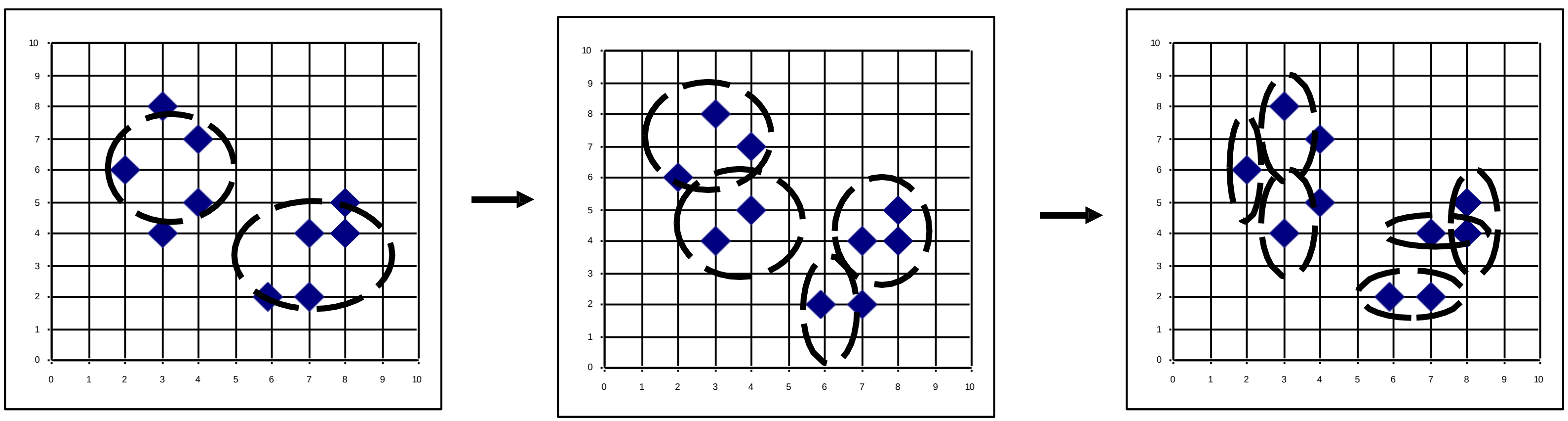
SEE
THE
ARROW



Hierarchical Clustering



AGNES (Agglomerative Nesting)



DIANA (Divisive Analysis)

Complexity of hierarchical clustering

- Distance matrix is used for deciding which clusters to merge/split

- At least quadratic in the number of data points

- Not usable for large datasets

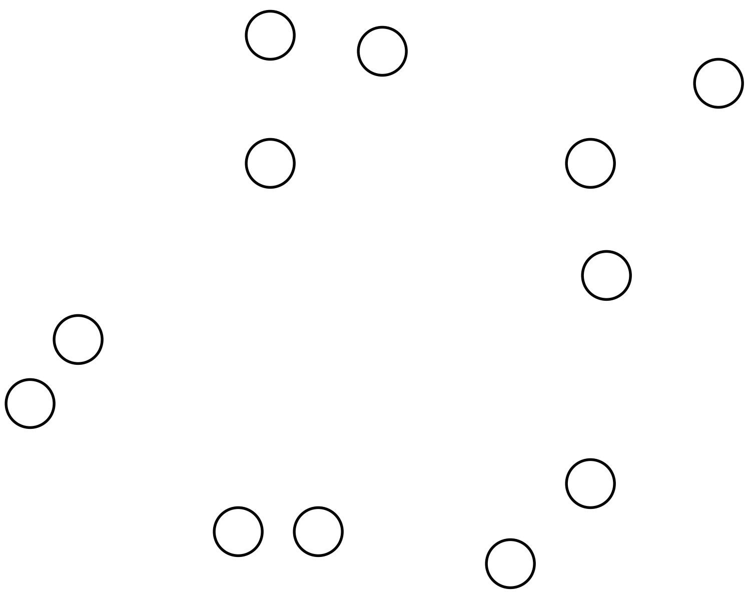
Agglomerative clustering algorithm

AGNES: Agglomerative Nesting

- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the distance between two clusters
  - Different definitions of the distance between clusters lead to different algorithms

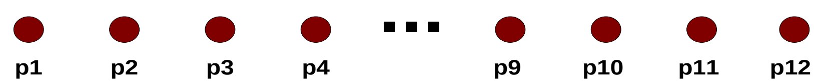
Input/ Initial setting

- Start with clusters of individual points and a distance/proximity matrix



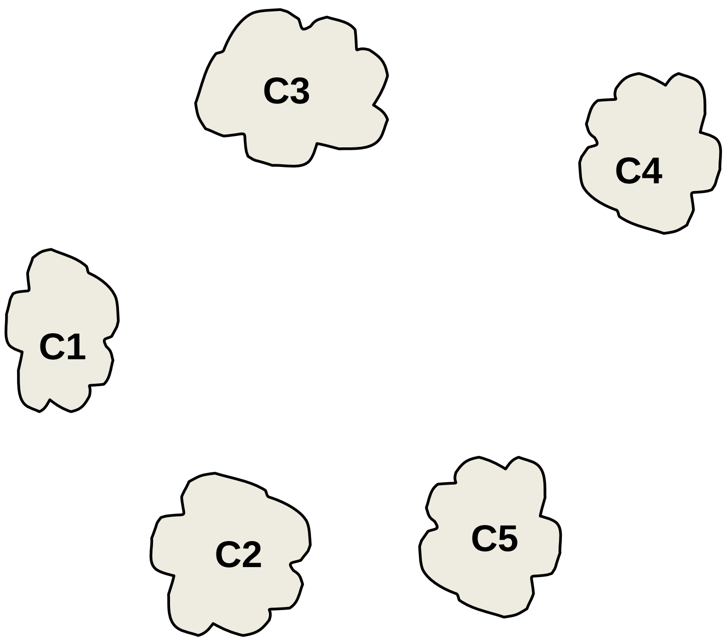
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Distance/Proximity Matrix



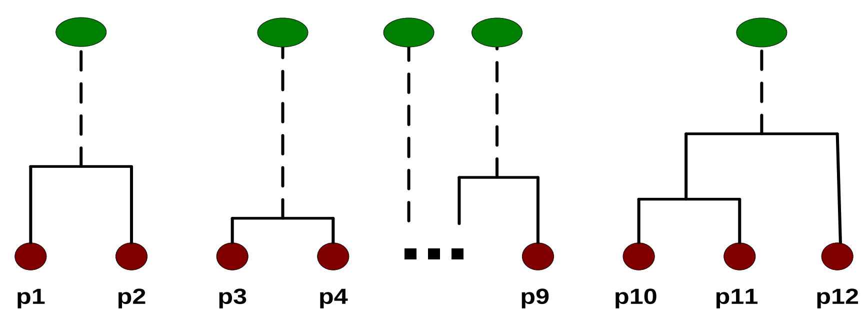
Intermediate State

- After some merging steps, we have some clusters



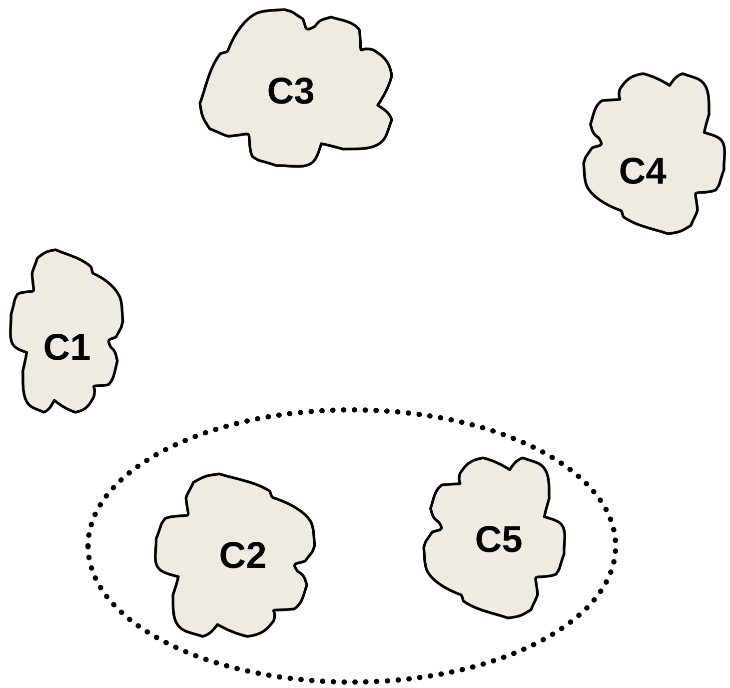
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

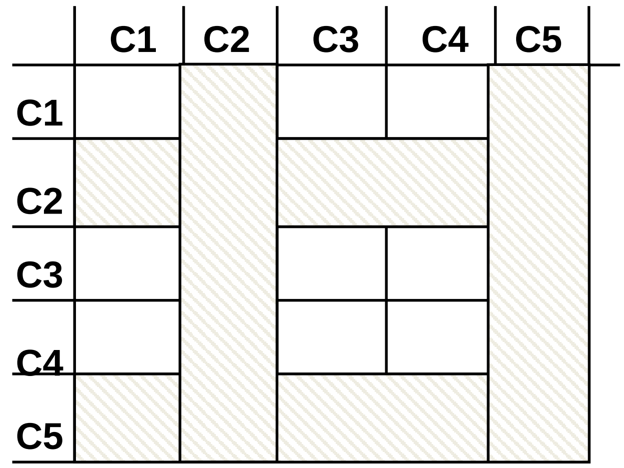
Distance/Proximity Matrix



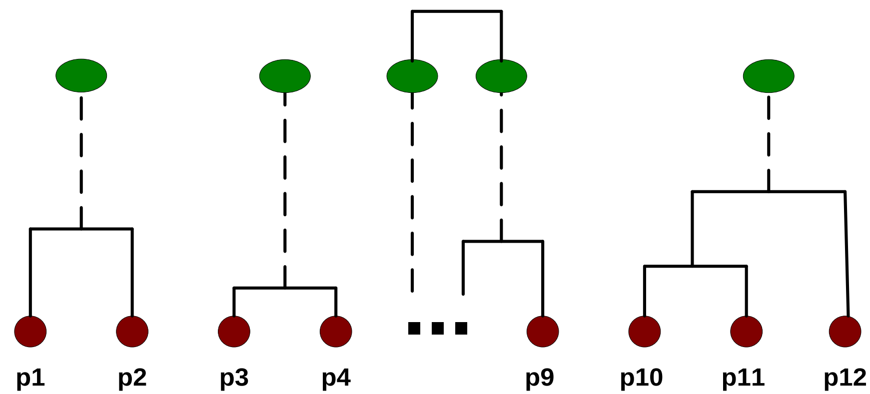
Intermediate State

- Merge the two closest clusters (C2 and C5) and update the distance matrix.



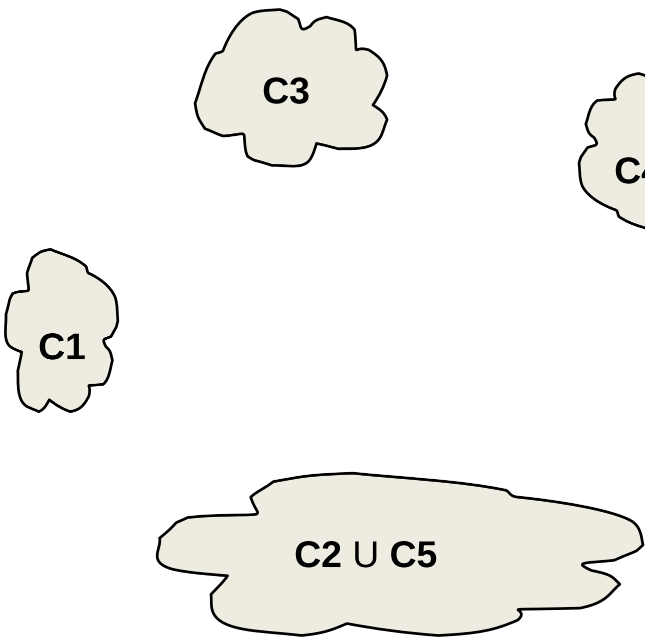


Distance/Proximity Matrix

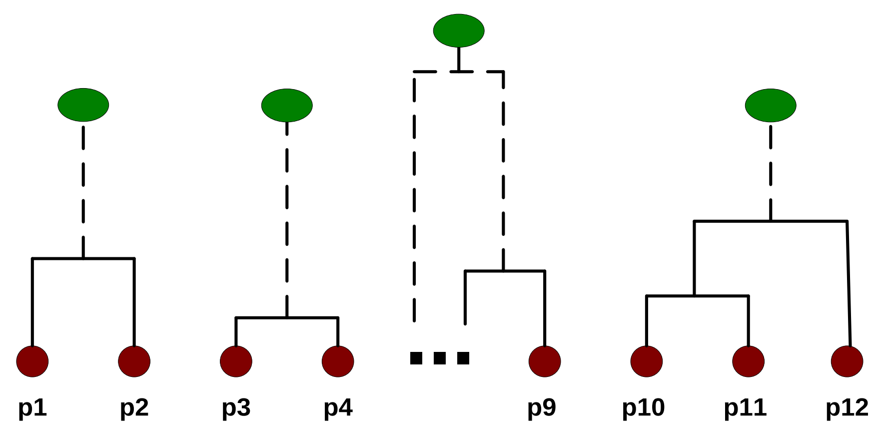


After Merging

- “How do we update the distance matrix?”



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		



Distance between two clusters

- Each cluster is a set of points
- How do we define distance between two sets of points
  - Lots of alternatives
  - Not an easy task

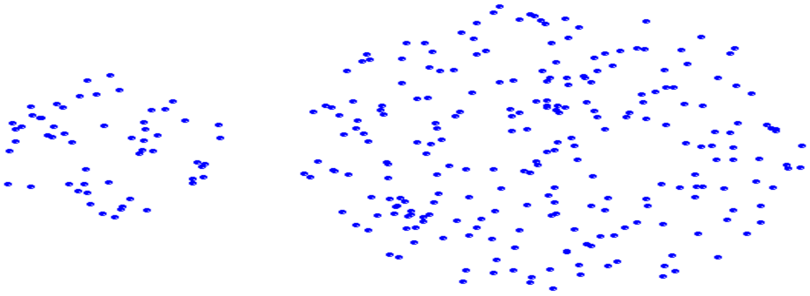
Distance between two clusters

- **Single-link distance** between clusters C_i and C_j is the ***minimum distance*** between any object in C_i and any object in C_j

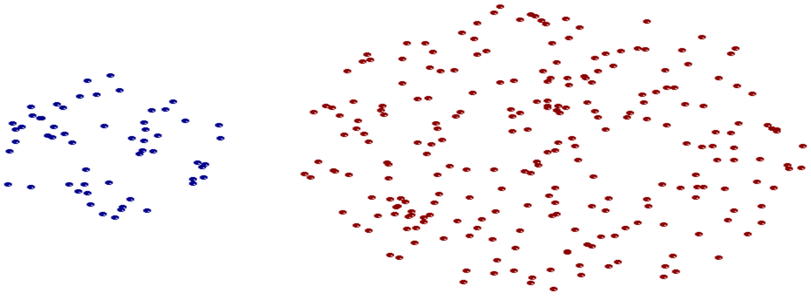
- The distance is **defined by the two most similar objects**

$$D_{sl}\left(C_i, C_j\right) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$

Strengths of single-link clustering



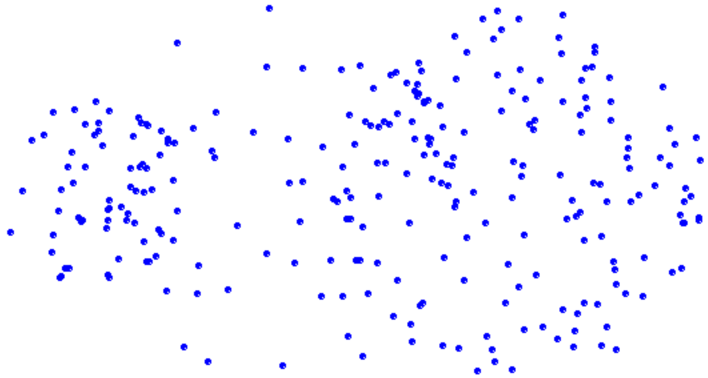
Original Points



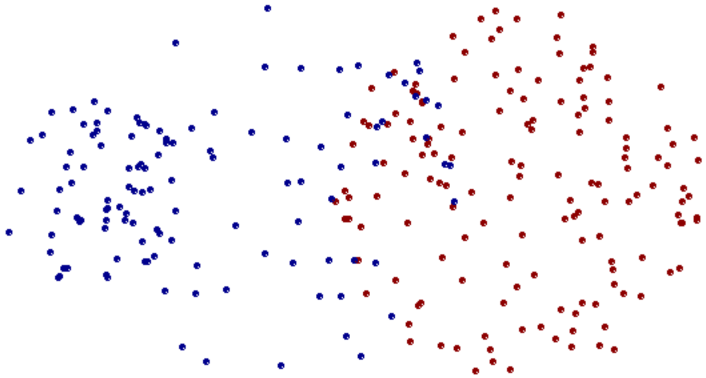
Two Clusters

- **Can handle non-elliptical shapes**

Limitations of single-link clustering



Original Points



Two Clusters

- Sensitive to noise and outliers
- It produces long, elongated clusters

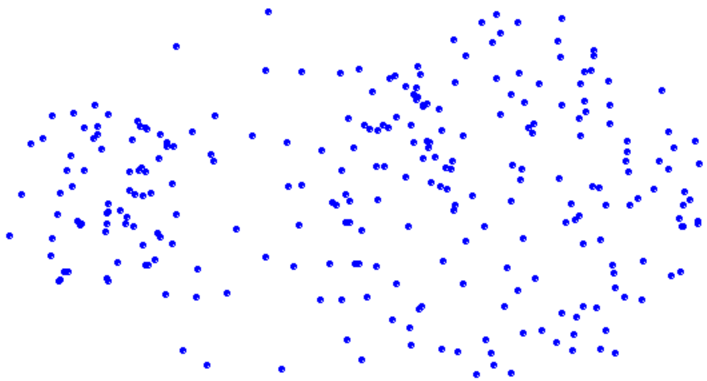
Distance between two clusters

- **Complete-link distance** between clusters C_i and C_j is the ***maximum distance*** between any object in C_i and any object in C_j

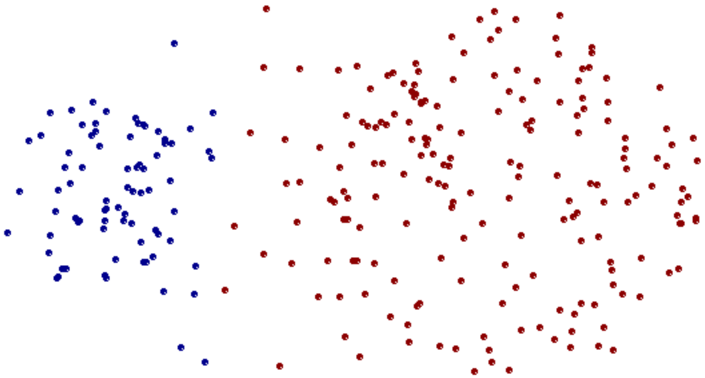
- The distance is **defined by the two most dissimilar objects**

$$D_{cl}\left(C_i, C_j\right) = \max_{x,y}\left\{d(x,y) \middle| x \in C_i, y \in C_j\right\}$$

Strengths of complete-link clustering



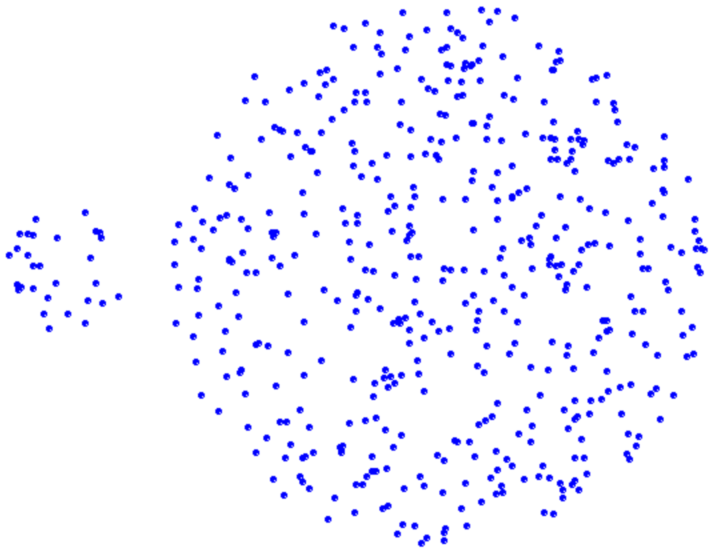
Original Points



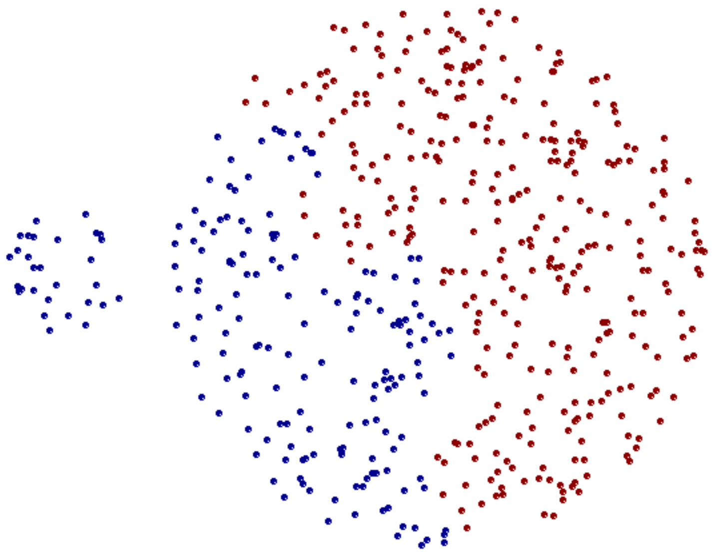
Two Clusters

- **More balanced clusters (with equal diameter)**
- **Less susceptible to noise**

Limitations of complete-link clustering



Original Points



Two Clusters

- **Tends to break large clusters**
- **All clusters tend to have the same diameter – small clusters are merged with larger ones**

Distance between two clusters

- **Group average distance** between clusters C_i and C_j is the ***average distance*** between any object in C_i and any object in C_j

$$D_{avg}\left(C_i, C_j\right) = \frac{1}{\left|C_i\right| \times \left|C_j\right|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

Average-link clustering: discussion

- Compromise between Single and Complete Link

- Strengths
  - Less susceptible to noise and outliers

- Limitations
  - Biased towards globular clusters

Distance between two clusters

- **Centroid distance** between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{centroids}(C_i, C_j) = d(r_i, r_j)$$

Distance between two clusters

- **Ward's distance** between clusters C_i and C_j is the ***difference*** between the ***total within cluster sum of squares for the two clusters separately***, and the ***within cluster sum of squares resulting from merging the two clusters*** in cluster C_{ij}

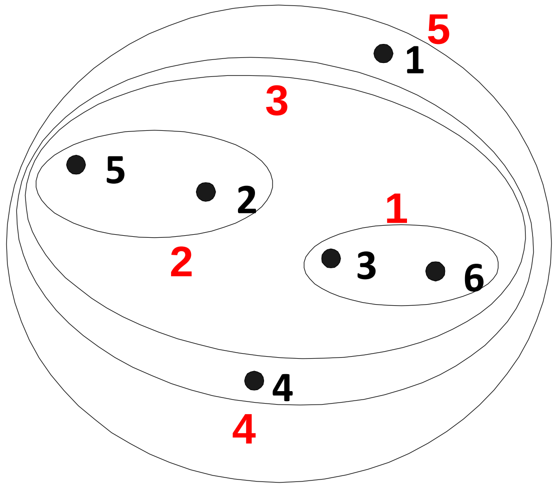
$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

- r_i: centroid of C_i
- r_j: centroid of C_j
- r_{ij}: centroid of C_{ij}

Ward's distance for clusters

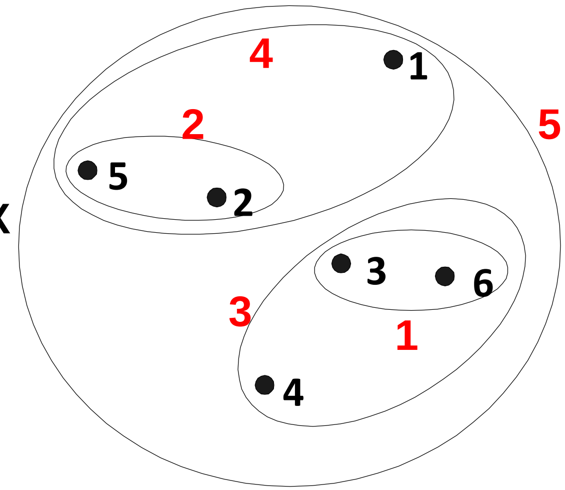
- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
  - Can be used to initialize k-means

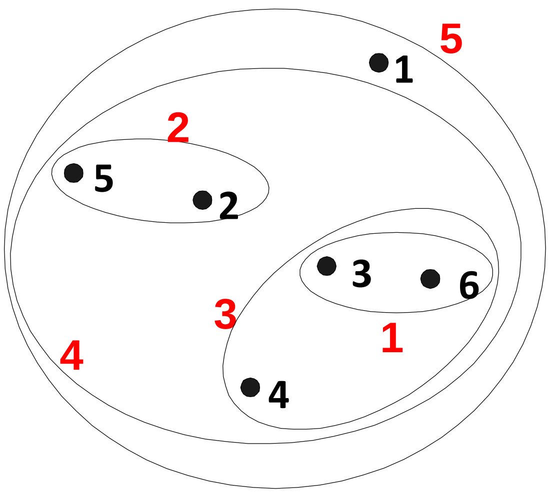
Hierarchical Clustering: Comparison



MIN

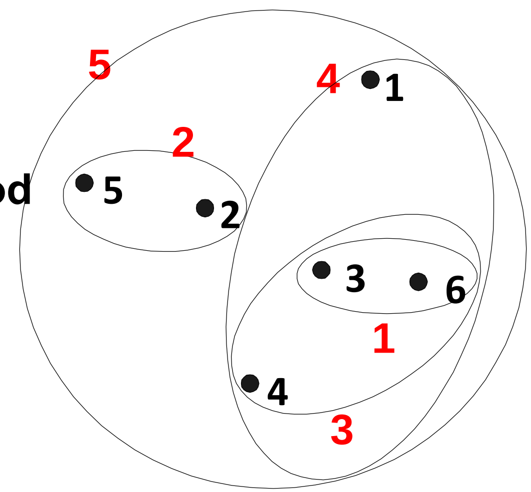
MAX





Group Average

Ward's Method



Example of converting data points into distance matrix

- Clustering analysis with agglomerative algorithm

	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

distance matrix

$$d_{AB} = \left((1-1.5)^2 + (1-1.5)^2\right)^{\frac{1}{2}} = \sqrt{\tfrac{1}{2}} = 0.7071$$

$$d_{DF} = \left((3-3)^2 + (4-3.5)^2\right)^{\frac{1}{2}} = 0.5$$

Euclidean distance

Agglomerative Hierarchical Clustering - Numerical

Consider the following set of 6 one dimensional data points:
18, 22, 25, 42, 27, 43

- Apply the agglomerative hierarchical clustering algorithm to build the hierarchical clustering dendogram.
- Merge the clusters using Min distance and update the proximity matrix accordingly.
- show the proximity matrix corresponding to each iteration of the algorithm.

DATA POINTS

	18	22	25	27	42	43
18	0	4	7	9	24	25
22	4	0	3	5	20	21
25	7	3	0	2	17	18
27	9	5	2	0	15	16
42	24	20	17	15	0	1
43	25	21	18	16	1	0

MERGING 42,43

	18	22	25	27	42	43
18	0	4	7	9	24	25
22	4	0	3	5	20	21
25	7	3	0	2	17	18
27	9	5	2	0	15	16
42	24	20	17	15	0	1
43	25	21	18	16	1	0

(42, 43)

MERGED 42,43

	18	22	25	27	42, 43
18	0	4	7	9	24
22	4	0	3	5	20
25	7	3	0	2	17
27	9	5	2	0	15
42, 43	24	20	17	15	0

MERGING 25,27

	18	22	25	27	42, 43
18	0	4	7	9	24
22	4	0	3	5	20
25	7	3	0	2	17
27	9	5	2	0	15
42, 43	24	20	17	15	0

(42, 43), (25, 27)

MERGING (25,27) , 22

	18	22, 25, 27	42, 43
18	0	4	24
22, 25, 27	4	0	15
42, 43	24	15	0

	18, 22, 25, 27	42, 43
18, 22, 25, 27	0	15
42, 43	15	0

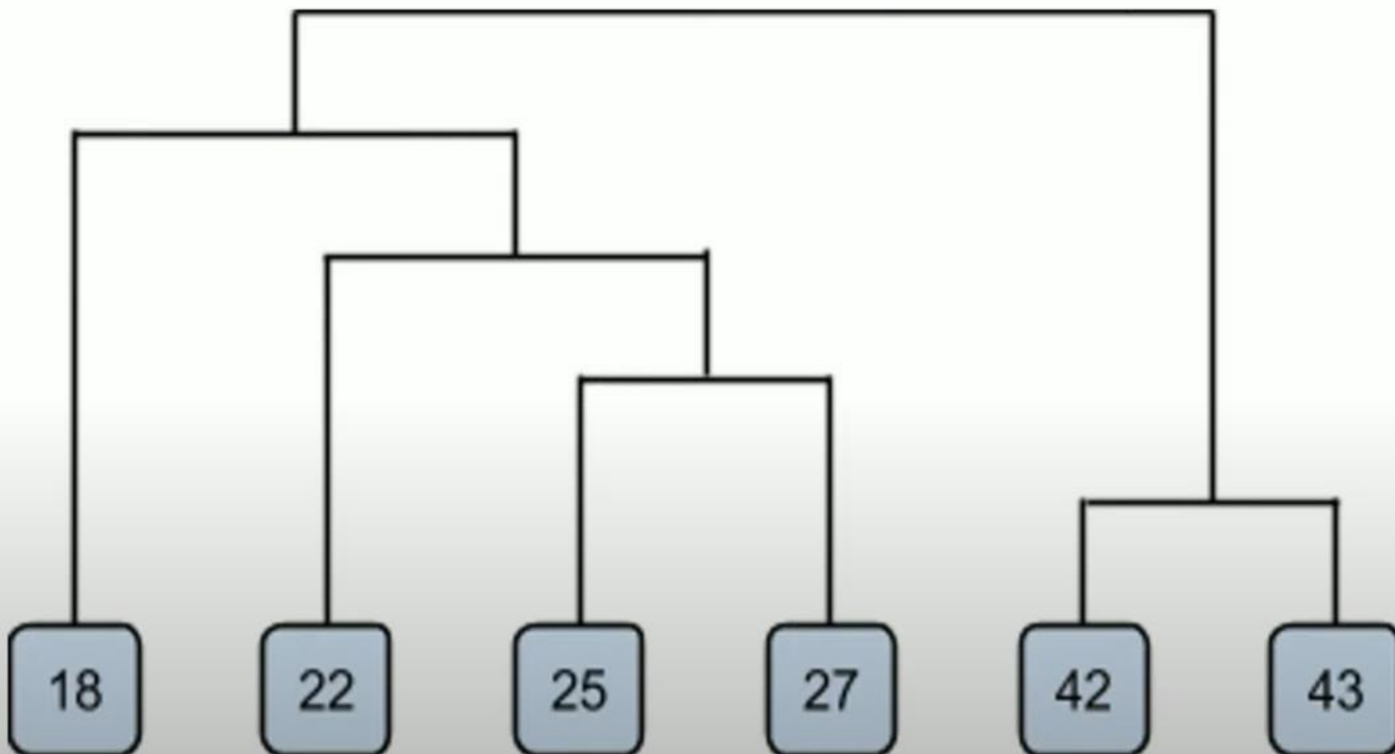
MERGING ((25,27) , 22),18 & MERGING 42,43

MERGING ALL

	18, 22, 25, 27, 42, 43
18, 22, 25, 27, 42, 43	0

Dendrogram

((42, 43), (((25, 27), 22), 18))



Hierarchical Clustering: Time and Space requirements

- For a dataset **X** consisting of **n** points
- **$O(n^2)$ space**; it requires storing the distance matrix
- **$O(n^3)$ time** in most of the cases
  - There are **n** steps and at each step the size **n^2** distance matrix must be updated and searched
  - Complexity can be reduced to **$O(n^2 \log(n))$** time for some approaches by using appropriate data structures

Divisive hierarchical clustering

DIANA: Divisive Analysis

- Start with a single cluster composed of all data points
- Split this into components
- Continue recursively
- Any inter cluster distance measure can be used
- Computationally intensive, less widely used than agglomerative methods

Divisive hierarchical clustering

DIANA: Divisive Analysis

Consider the following set of 6 one dimensional data points:
18, 22, 25, 42, 27, 43

	A	**B**	**C**	**D**	**E**	**F**
A	0	4	7	9	24	25
B	4	0	3	5	20	21
C	7	3	0	2	17	18
D	9	5	2	0	15	16
E	24	20	17	15	0	1
F	25	21	18	16	1	0

Divisive hierarchical clustering

DIANA: Divisive Analysis

Consider the following set of 6 one dimensional data points:
18, 22, 25, 42, 27, 43

Step 1: Initialize C_L={a, b, c, d, e, f}
Step 2: Initialize C_I = C_L and C_J = {}
Step 3: Initial Iteration

- Calculate the average dissimilarities of objects in C_I with other objects in C_I

- **Average Dissimilarity of a**
- a=1/5*(d(a, b) + d(a, c) + d(a, d) + d(a, e) + d(a, f))
- a=1/5(4+7+9+24+25)
- a=69/5
- =13.8

- b=10.6, c=9.4, d=9.4, e=15.4, f=16.2

	A	B	C	D	E	F
A	0	4	7	9	24	25
B	4	0	3	5	20	21
C	7	3	0	2	17	18
D	9	5	2	0	15	16
E	24	20	17	15	0	1
F	25	21	18	16	1	0

Divisive hierarchical clustering

DIANA: Divisive Analysis

- The positive highest dissimilarity is 16.2 (if tie occurs choose arbitrary/random)
- Move f from C_I to C_J
- Now we have , C_I={a, b, c, d, e} and C_J={f}

Step 3: Remaining Iterations

- Calculate the average dissimilarities of objects in C_I with other objects in C_I

- **Average Dissimilarity of a**
- a=1/4*(d(a, b) + d(a, c) + d(a, d) + d(a, e)) – 1/1(d(a, f))
- a=1/4(4+7+9+24)-25
- a=11-25
- a=-14

- b=-13, c=-10.75, d=-8.25, e=18

	A	B	C	D	E	F
A	0	4	7	9	24	25
B	4	0	3	5	20	21
C	7	3	0	2	17	18
D	9	5	2	0	15	16
E	24	20	17	15	0	1
F	25	21	18	16	1	0

- The +ve highest dissimilarity is 18 (if tie occurs choose arbitrary/random)
- Move e from C_I to C_J
- Now we have , C_I={a, b, c, d} and C_J={f, e}

Divisive hierarchical clustering

DIANA: Divisive Analysis

Step 3: Remaining Iterations

- Calculate the average dissimilarities of objects in C_I with other objects in C_I

- **Average Dissimilarity of a**
- a=1/3*(d(a, b) + d(a, c) + d(a, d))– 1/2(d(a, f) + d(a, e))
- a=1/3(4+7+9)-1/2(24+25)
- a=-17.83

- b=-16.5, c=-13.5, d=-10.16

	A	B	C	D	E	F
A	0	4	7	9	24	25
B	4	0	3	5	20	21
C	7	3	0	2	17	18
D	9	5	2	0	15	16
E	24	20	17	15	0	1
F	25	21	18	16	1	0

- The +ve highest dissimilarity is not available
- Stop and construct clusters C_I and C_J

C_I={a, b, c, d} and C_J={f, e}

Calculate diameter of C_I and C_J

Diameter of C_I = max(d(a, b), d(a, c), d(a, d), d(b, c), d(b, d), d(c, d)) = 9

Diameter of C_J = max(d(f, e)) = 1

Divisive hierarchical clustering

DIANA: Divisive Analysis

Choose cluster with the highest Diameter (i.e. C_I) and start repeating from step 2

Step 2: Initialize C_I = C_I={a, b, c, d} and C_J = {}

Step 3: Remaining Iterations

- Calculate the average dissimilarities of objects in C_L with other objects in C_I

- **Average Dissimilarity of a**
- a=1/3*(d(a, b) + d(a, c) + d(a, d))
- a=1/3(4+7+9)
- a=6.67

- b=4, c=4, d=5.33

	A	**B**	**C**	**D**	**E**	**F**
A	0	4	7	9	24	25
B	4	0	3	5	20	21
C	7	3	0	2	17	18
D	9	5	2	0	15	16
E	24	20	17	15	0	1
F	25	21	18	16	1	0

- The +ve highest dissimilarity is a
- Move a from C_I to C_J
- Now we have , C_I={b, c, d} and C_J={a}

Divisive hierarchical clustering

DIANA: Divisive Analysis

Step 3: Remaining Iterations

- Calculate the average dissimilarities of objects in C_I with other objects in C_I

- **Average Dissimilarity of b**
- b=1/2*(d(b, c) + d(b, d)) – 1/1(d(b, a))
- b=1/2(3+5)-4
- b=4-4
- b=0

- c=-4.5, d=-5.5

	A	B	C	D	E	F
A	0	4	7	9	24	25
B	4	0	3	5	20	21
C	7	3	0	2	17	18
D	9	5	2	0	15	16
E	24	20	17	15	0	1
F	25	21	18	16	1	0

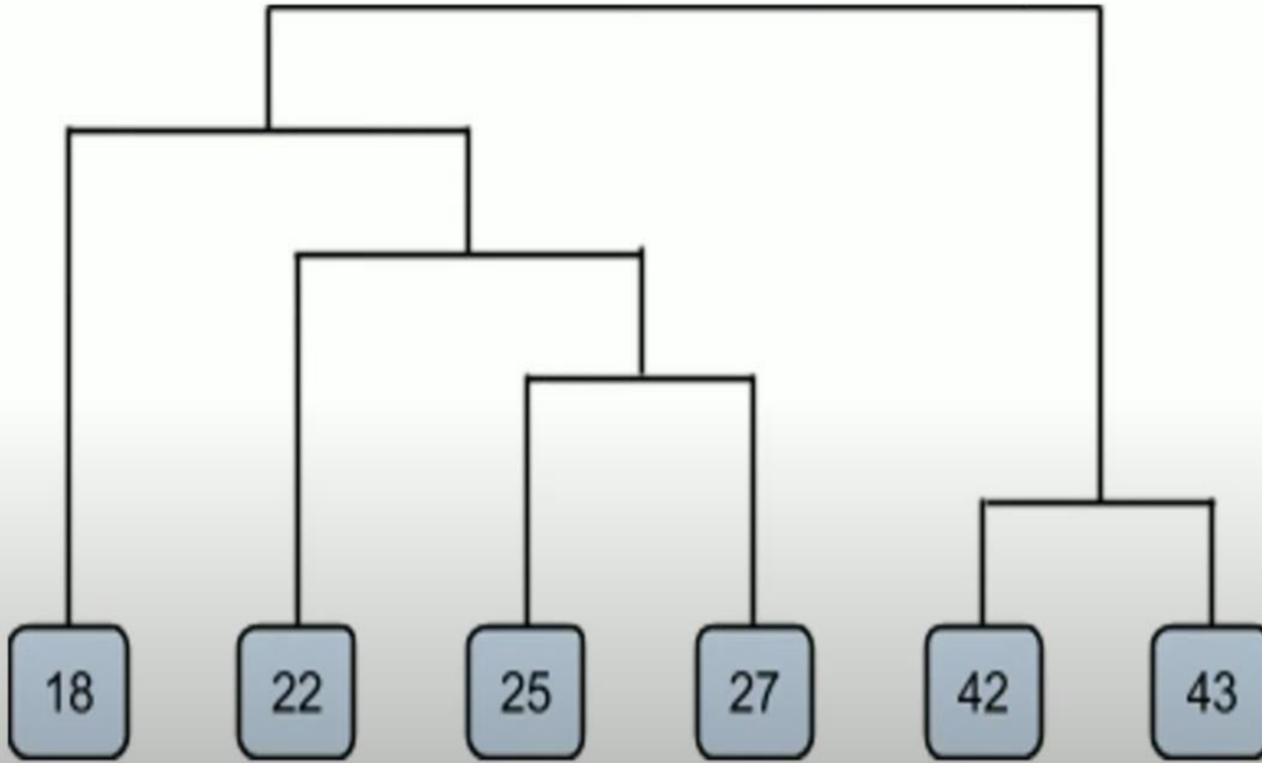
- The +ve highest dissimilarity is not available
- Stop and construct clusters C_I and C_J

C_I={b, c, d} and C_J={a}

Calculate diameter of C_I and C_J

Dendrogram

((42, 43), (((25, 27), 22), 18))



Practice Problem

	a	b	c	d	e
a	0	9	3	6	11
b	9	0	7	5	10
c	3	7	0	9	2
d	6	5	9	0	8
e	11	10	2	8	0