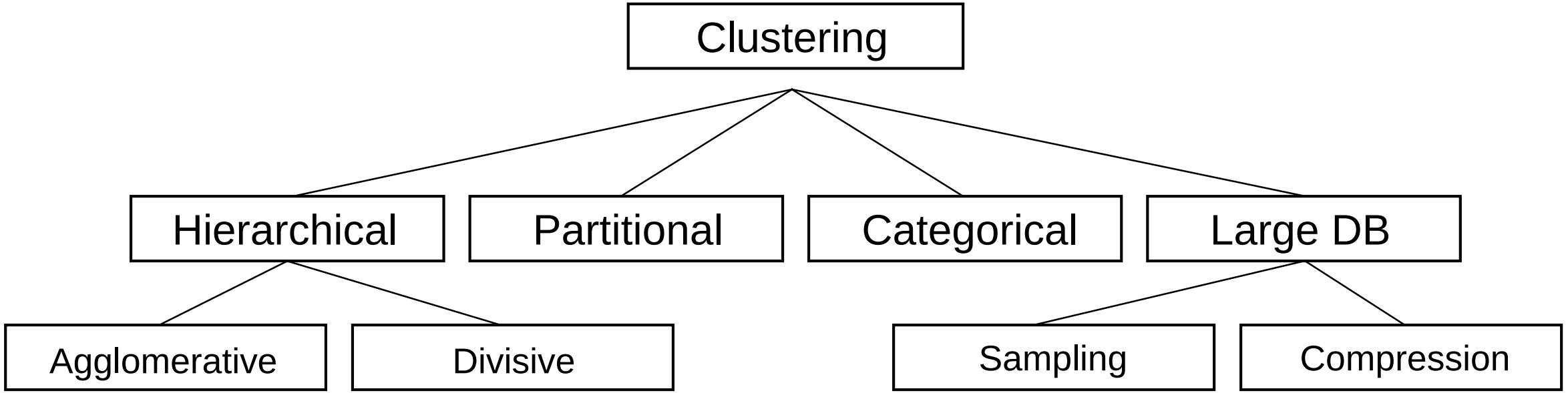
CLUSTERING

Clustering Problem – Formal Definition

- Assume the result of clustering is the generation of $K = \{K_1, K_2, \ldots, K_k\}$
- Given a data set $D=\{t_1,t_2,\ldots,t_n\}$ of tuples and an integer value k, the ***Clustering Problem*** is to define a mapping $f:D \rightarrow \{1,..,k\}$ where each t_i is assigned to one cluster K_j, $1<=j<=k$.
- A ***Cluster***, K_j, contains precisely those tuples mapped to it.
- K forms a partition of D.

Clustering Approaches



Types of Clustering

- ***Hierarchical*** – A set of nested clusters organized as a hierarchical tree
- ***Partitional*** – A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
  - One set of clusters created
- ***Overlapping/Non-overlapping***

Cluster Parameters

Given a cluster $K_m = \{ t_{m1}, t_{m2}, \ldots, t_{mn} \}$ of n points.

$$centroid = C_m = \frac{\sum_{i=1}^{N} (t_{mi})}{N}$$

$$radius = R_m = \sqrt{\frac{\sum_{i=1}^{N} (t_{mi} - C_m)^2}{N}}$$

$$diameter = D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (t_{mi} - t_{mj})^2}{(N)(N-1)}}$$

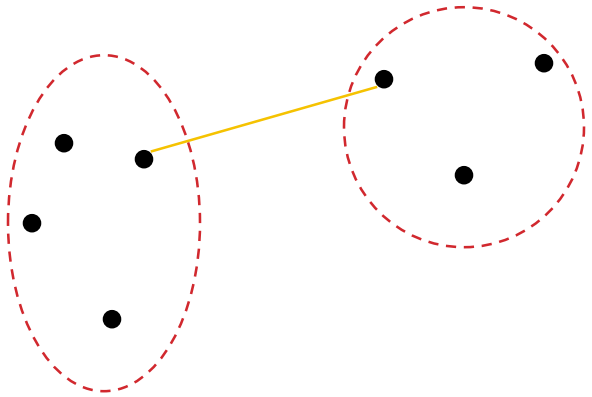
Medoid = M_m = an actual point in the cluster closest to the centroid

Calculating the Distance between Clusters K_i and K_j

- ***Single link***: smallest distance between an element in one cluster and an element in the other,

  i.e., $dis(K_i, K_j) = min(t_{ip}, t_{jq})$

$$dist(K_i, K_j) = \min\{ dist(t_{il}, t_{jm})\} \forall\ t_{il} \in K_i \notin K_j\ and \forall\ t_{jm} \in K_j \notin K_i$$

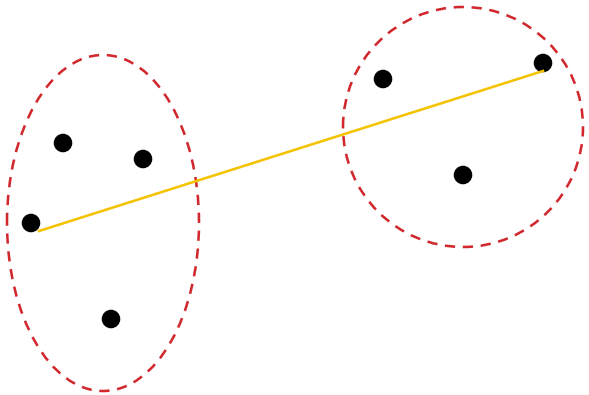


Calculating the Distance between Clusters K_i and K_j

- ***Complete link***: largest distance between an element in one cluster and an element in the other,

  i.e., $dis(K_i, K_j) = max(t_{ip}, t_{jq})$

$$dist(K_i, Kj) = \max\{ dist(t_{il}, tjm)\} \forall t_{il} \in Ki \notin Kj \; and \; \forall \; tjm \in Kj \notin Ki$$

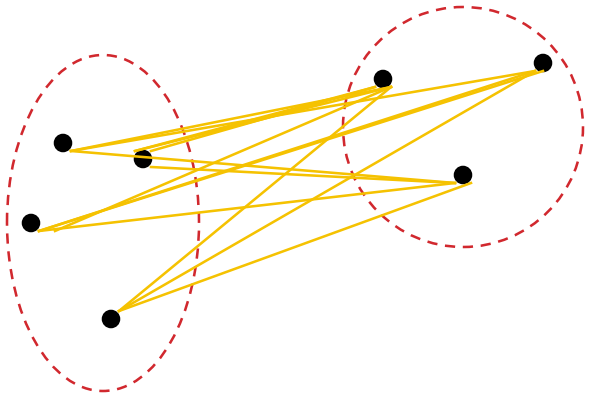


Calculating the Distance between Clusters K_i and K_j

- ***Average link***: avg distance between an element in one cluster and an element in the other,

  i.e., $dis(K_i, K_j) = avg(t_{ip}, t_{jq})$

$$\text{dist}(K_i, Kj) = \text{average}\{\text{dist}(t_{il}, tjm)\}\ \forall t_{il} \in Ki \notin Kj\ and\ \forall\ tjm \in Kj \notin Ki$$



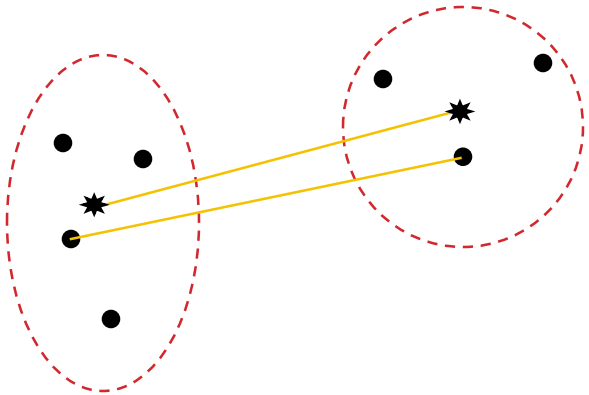
Calculating the Distance between Clusters K_i and K_j

- ***Centroid distance***: distance between the centroids of the two clusters

$$\text{dist}(K_i, Kj) = \text{dist}(C_i, C_j)\}$$

- ***Medoid distance***: distance between the medoids of the two clusters

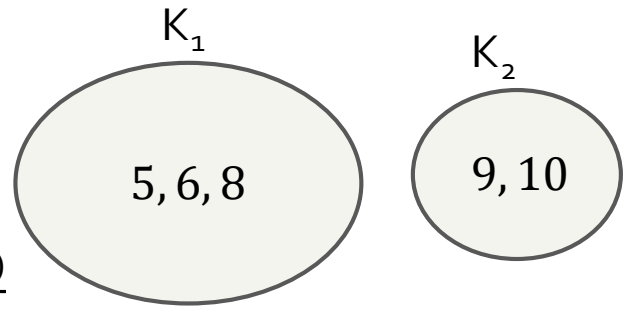
$$\text{dist}(K_i, Kj) = \text{dist}(M_i, M_j)\}$$



Example

- $K_1 = \{5, 6, 8\}$, $K_2 = \{9, 10\}$
- Single link distance = $\text{dist}(8, 9) = 9 - 8 = 1$
- Complete link distance = $\text{dist}(5, 10) = 10 - 5 = 5$
- Average link distance $= \frac{(9-5)+(9-6)+(9-8)+(10-5)+(10-6)+(10-8)}{6}$

$$= \frac{4+3+1+5+4+2}{6} = \frac{19}{6} = 3.167$$



- Centroid distance $= \text{dist}(C_1, C_2)$

$C_1 = \frac{5+6+8}{3} = \frac{19}{3} = 6.34$

$C_2 = \frac{9+10}{2} = \frac{19}{2} = 9.5$

$\text{dist}(C_1, C_2) = dist(9.5, 6.34) = 3.16$

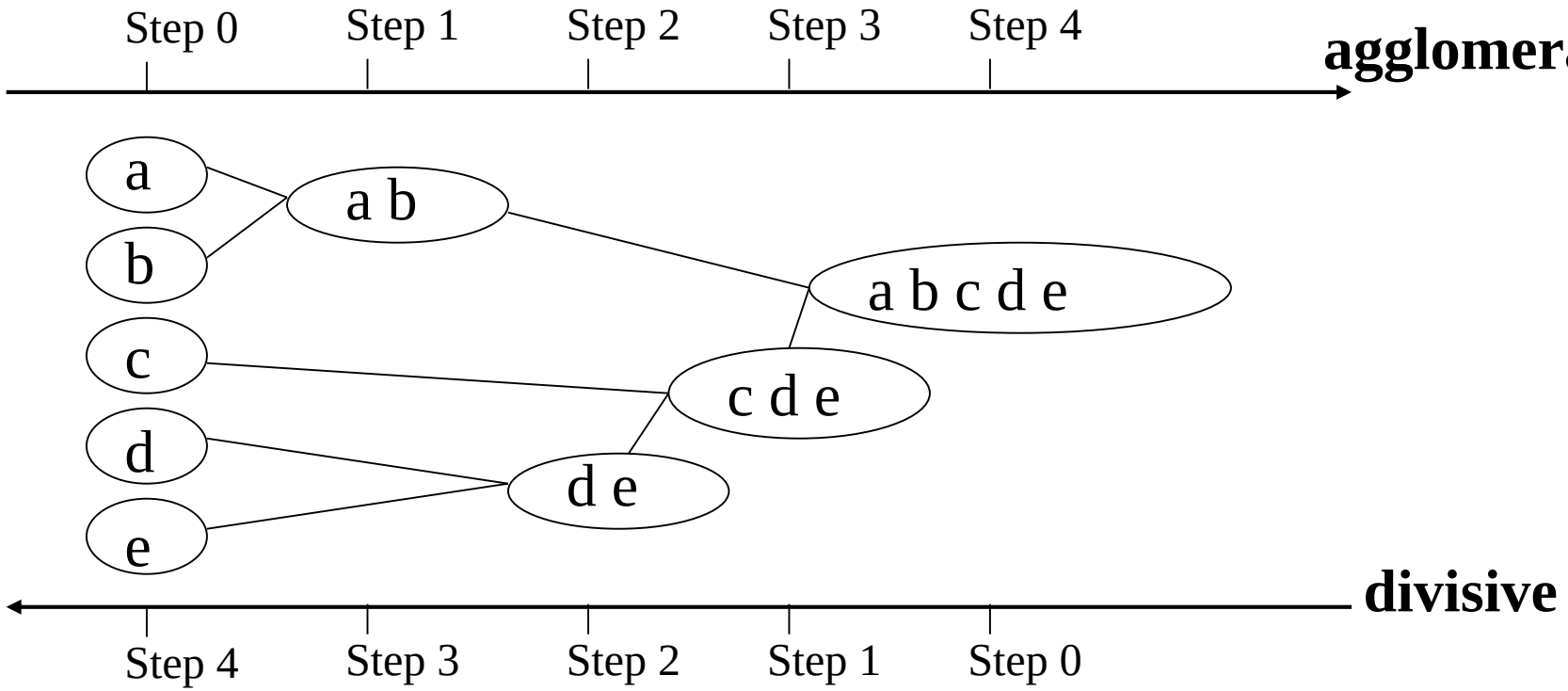
- Medoid distance $= \text{dist}(M_1, M_2) = dist(6, 9) = 3$

Hierarchical Clustering

- Nested set of clusters is created
- User decides which set of clusters to use
- ***Agglomerative***
  - Bottom-up approach
  - Initially each item in its own cluster
  - Iteratively clusters are merged together
- ***Divisive***
  - Top-down approach
  - Initially all items in one cluster
  - Clusters are successively divided

Hierarchical Clustering

- Use distance matrix
- This method does not require the number of clusters ***k*** as an input, but may need a termination condition

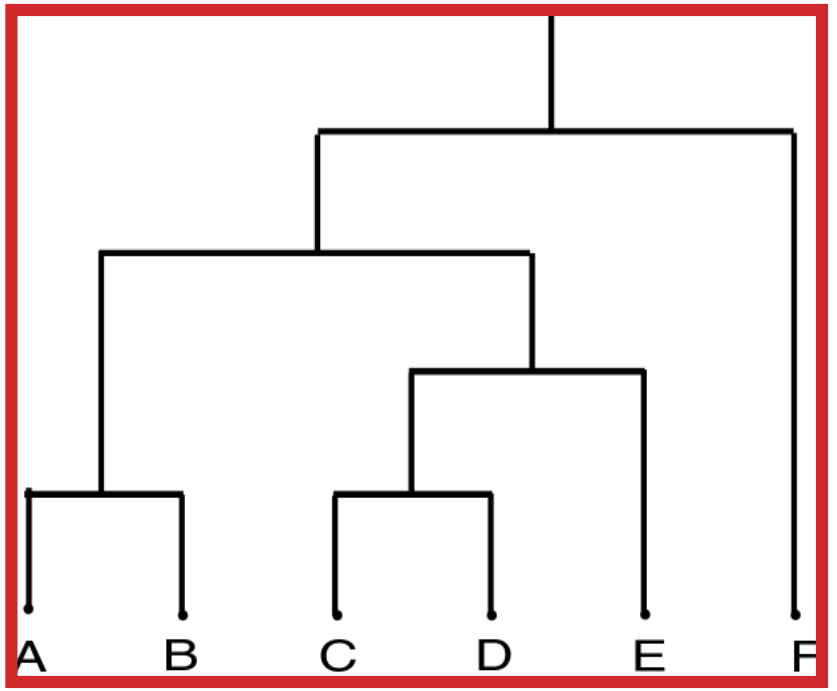


Hierarchical Algorithms

- Single Link
- MST Single Link
- Complete Link
- Average Link

Dendrogram

- ***Dendrogram:*** a tree-like data structure that illustrates hierarchical clustering techniques
- It shows the different set of clusters obtained at each level of the algorithm
- Each level shows clusters for that level
  - Leaves – individual clusters
  - Root – one cluster
- A cluster at level i is the union of its children clusters at level i+1
- Each level of the tree corresponds to a different set of clusters that may be used

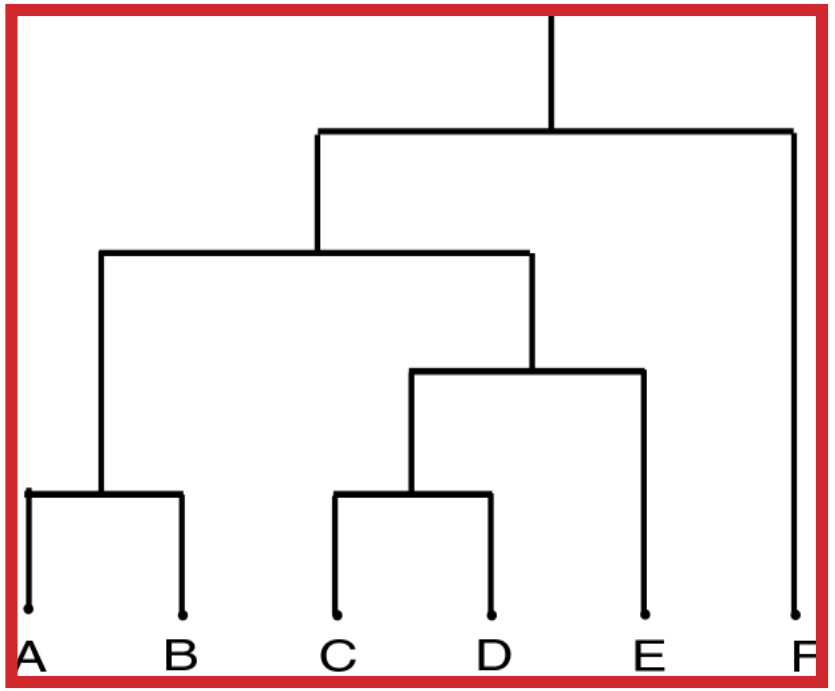


Dendrogram – Formal Definition

- A ***dendrogram*** is a set of ordered triples $< d, k, K >$, where
  - d is the threshold distance
  - k is number of clusters
  - K is et of clusters

Dendrogram

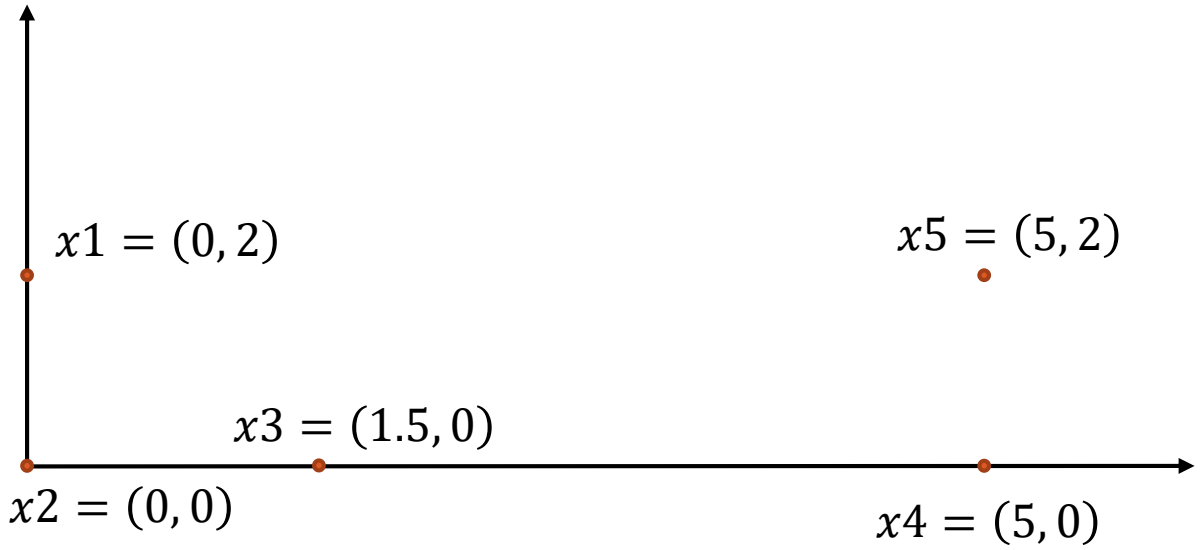
- Each level of the dendrogram corresponds to a different set of clusters
- A clustering is obtained by cutting the dendrogram at a certain distance
- Each connected components forms a cluster



Example (1 – 6)

- Consider the points $x1 = (0, 2), x2 = (0, 0), x3 = (1.5, 0), x4 = (5, 0)$, and $x5 = (5, 2)$ in the $2D$ space. Show the dendrogram that results from clustering these points using the single-link agglomerative hierarchical clustering method. Use Euclidean distance to measure distances between points.

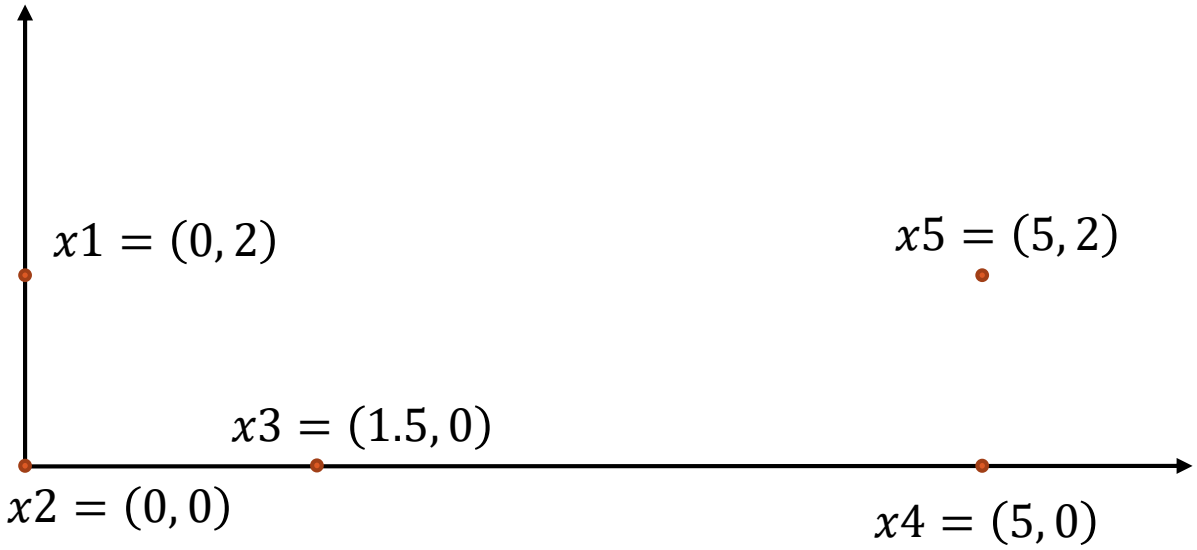
	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**



Example (cont. 2 – 6)

- Using agglomerative single-link
- <d, k, K> = $< 0,\ 5,\ \{\{x1\},\ \{x2\},\ \{x3\},\{x4\},\{x5\}\} >$
- $x2$ and $x3$ get merged at distance 1.5: <d, k, K> = $< 1.5,\ 4,\ \{\{x1\},\ \{x2, x3\ \},\{x4\},\{x5\}\} >$

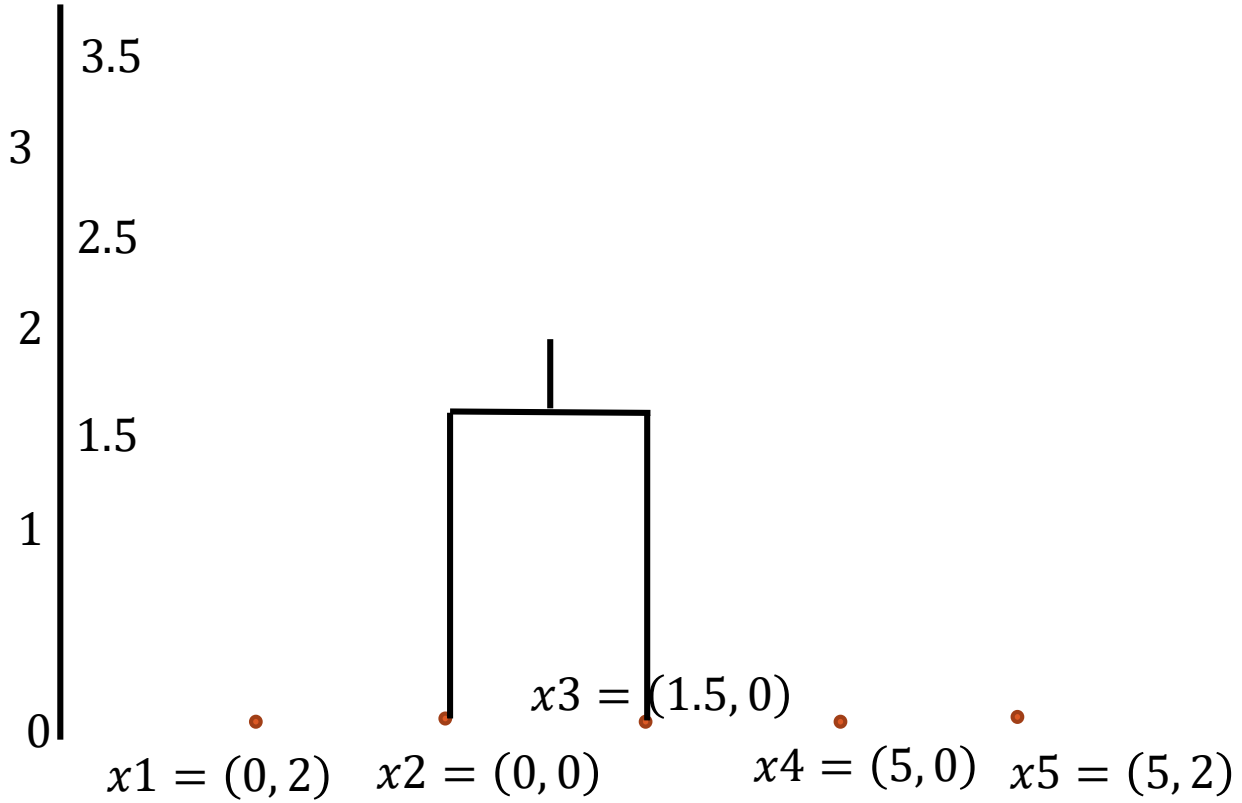
	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**



Example (cont. 3 – 6)

- Using agglomerative single-link
- <d, k, K> = $< 0,\ 5,\ \{\{x1\},\ \{x2\},\ \{x3\},\{4\},\{x5\}\} >$
- $x2$ and $x3$ get merged at distance 1.5: <d, k, K> = $< 1.5,\ 4,\ \{\{x1\},\ \{x2, x3\ \},\{x4\},\{x5\}\} >$

	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**

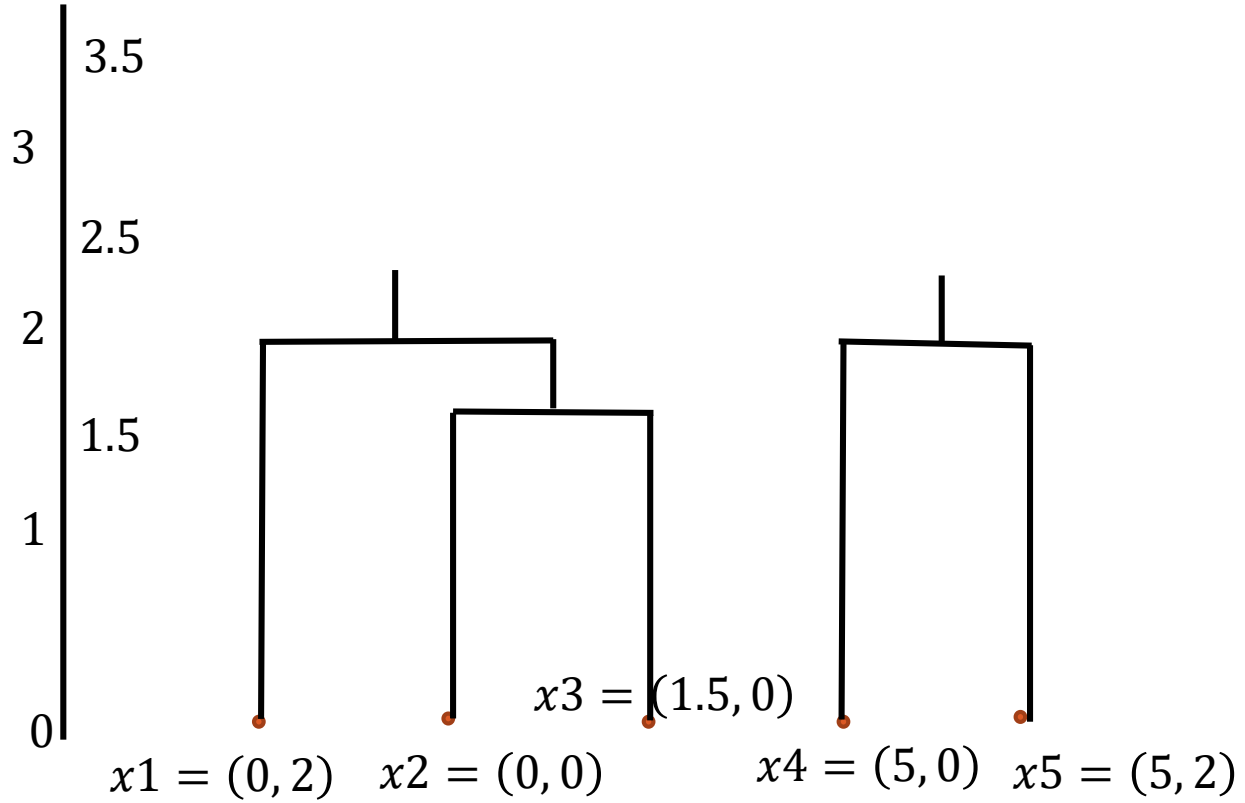


Example (cont. 4 – 6)

- Next, $x4$ and $x5$ are merged at distance 2 into the cluster $\{x4, x5\}$
- At the same level, $x1$ is merged with $\{x2, x3\}$

<d, k, K> = $< 2,\ 2,\ \{\{x1, x2, x3\}, \{x4, x5\}\} >$

	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**

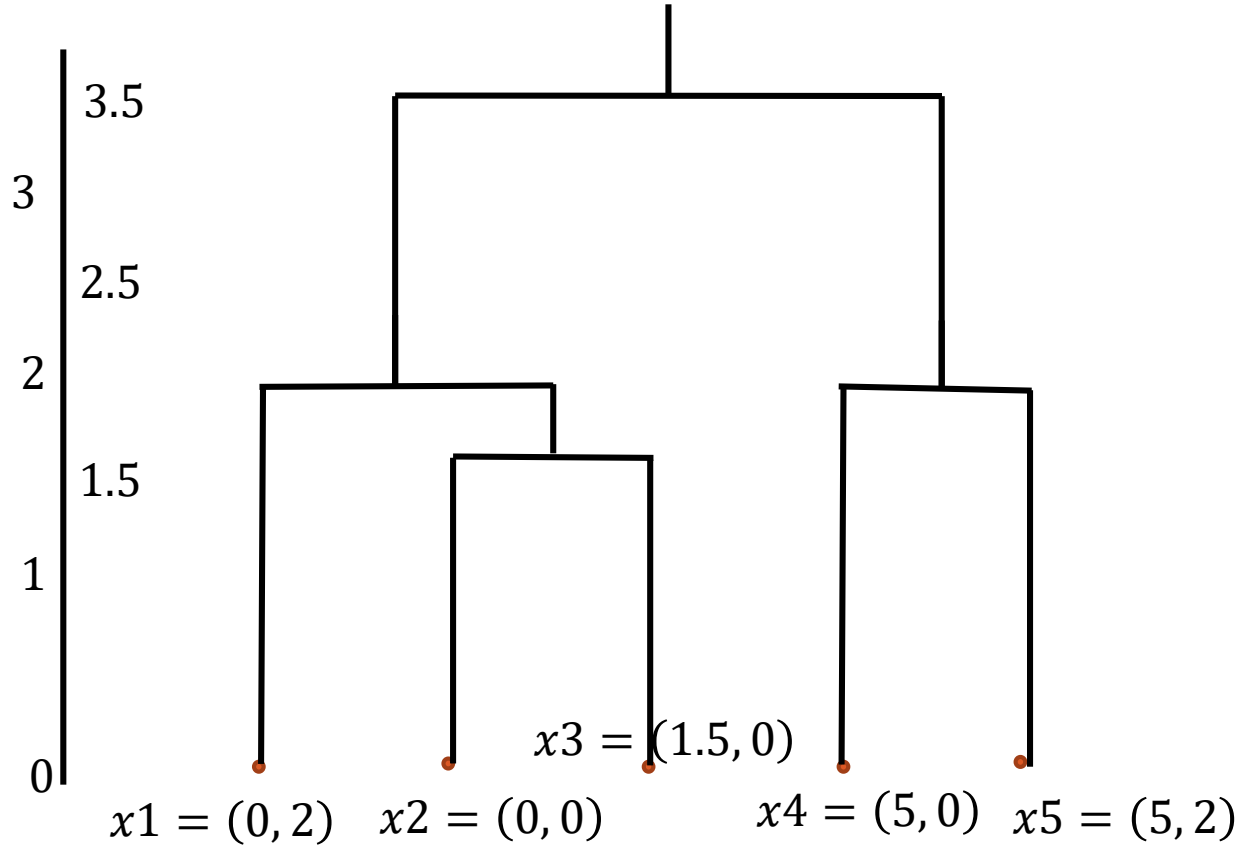


Example (cont. 5 – 6)

- Then, the clusters $\{x4, x5\}$ and $\{x1, x2, x3\}$ get merged at distance 3.5

<d, k, K> = $< 3.5,\ 1,\ \{\{x1, x2, x3, x4, x5\}\} >$

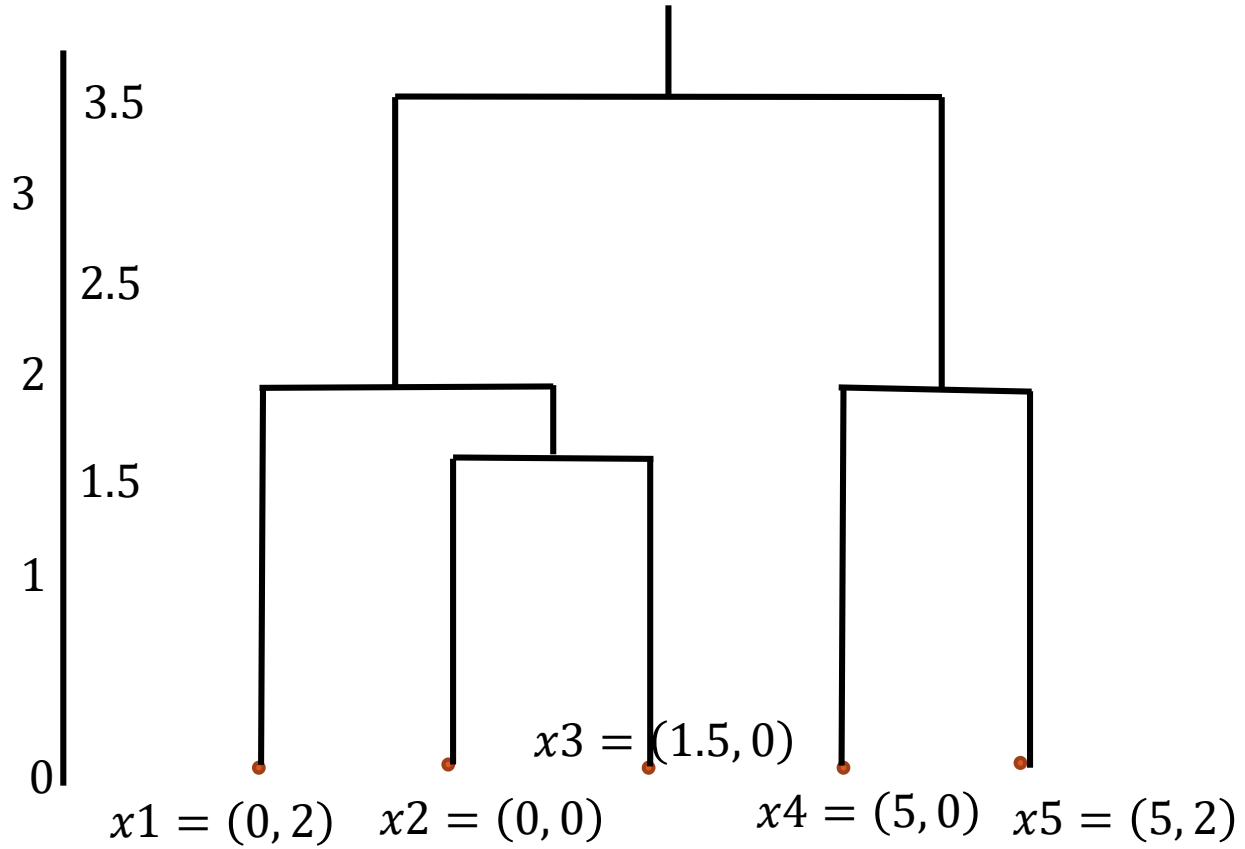
	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**



Example (cont. 6 – 6)

- The dendrogram <d, k, K>:
  $\{< 0,\ 5,\ \{\{x1\},\ \{x2\},\ \{x3\},\{4\},\{x5\}\} >,\ < 1.5,\ 4,\ \{\{x1\},\ \{x2, x3\},\{x4\},\{x5\}\} >,$
  $< 2,\ 2,\ \{\{x1, x2, x3\},\{x4, x5\}\},\ < 3.5,\ 1,\ \{\{x1, x2, x3, x4, x5\}\} >\}$

	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**



Agglomerative Algorithms

- Bottom-up approach
- Initially each item in its own cluster
- Iteratively clusters are merged together
- Basic idea is the same. Difference is the dissimilarity measure used
- Algorithm next slide gives the basic idea

Agglomerative Algorithm

Input:
$D = \{t_1, t_2, ..., t_n\}$ // **Set of elements**
A // **Adjacency matrix showing distance between elements.**
Output:
DE // **Dendrogram represented as a set of ordered triples.**
Agglomerative Algorithm:
$d = 0$;
$k = n$;
$K = \{\{t_1\}, ..., \{t_n\}\}$;
$DE = \{< d, k, K >\}$; // **Initially dendrogram contains each element in its own cluster.**
repeat
$oldk = k$;
$d = d + 1$;
$A_d =$ **Vertex adjacency matrix for graph with threshold distance of** d;
$< k, K >= NewClusters(A_d, D)$;
if $oldk \neq k$ **then**
$DE = DE \cup < d, k, K >$; // **New set of clusters added to dendrogram.**
until $k = 1$

Remarks

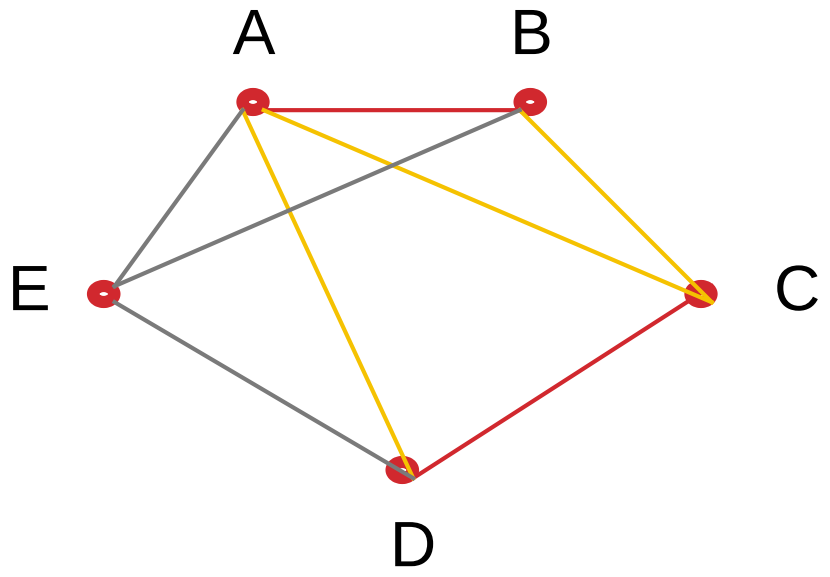
- Algorithm outputs the dendrogram, DE
- Algorithm is not incremental
- Algorithm uses NewCusters(A_d, D) to determine how clusters are merged
- Implementation of NewCusters(A_d, D) is based on the dissimilarity measure used by the algorithm (single-link, complete-link, … etc.)

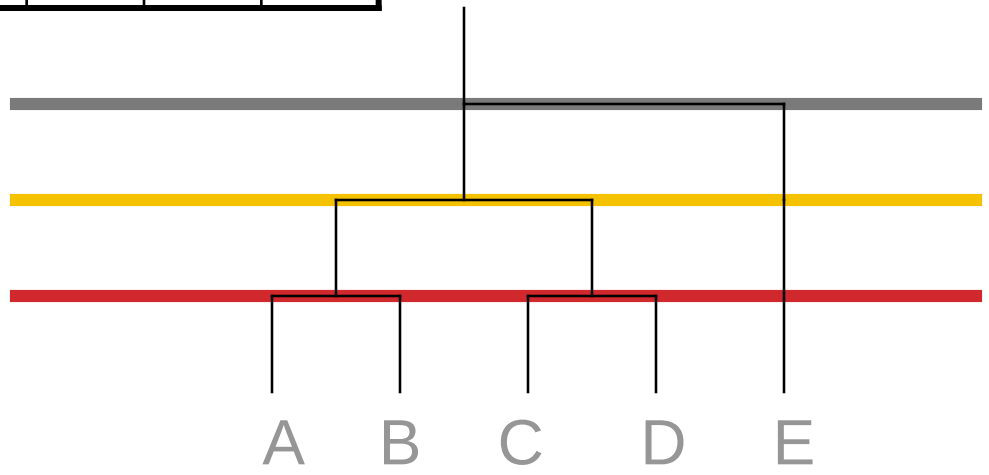
Using the Single Link Approach

- Two clusters K_i, K_j are merged if the distance between them is $\leq$ threshold value, d.
  $dis(K_i, K_j) = min(t_{ip}, t_{jq})$
- How can we find such clusters K_i and K_j ?
  - Input is given in the form of an adjacency matrix
  - As if we have a graph with objects in D as nodes
  - A weighted edge corresponds to distance/dissimilarity between the two objects
- Finds maximal connected components in this graph
  - weight on any edge is $\leq$ d
- Two clusters are merged if there is an edge with weight $\leq$ d that connects them
- Uses threshold distances at each level

Example

	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0





Threshold of

1 2 3

Single-Link Approach & Complete-Link Approach

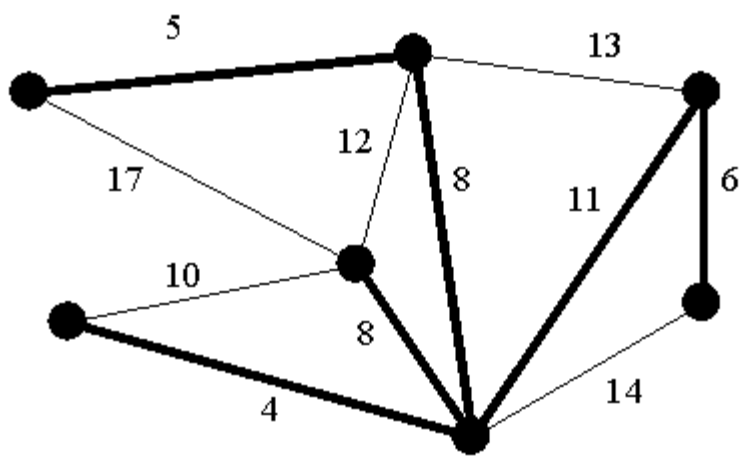
- Single link algorithm, replace NewCusters(A_d, D) by a procedure to find maximal connected components
- Problems with single link:

1. Time to find maximal connected components is $O(n^2)$
2. Can generate elongated clusters - long chains of points
   - Solution: merge the two clusters if the distance between any point in the first cluster and any point in the second cluster is $\leq$ threshold, d
   - This gives the ***complete-link alternative*** of the algorithm

- For complete-link, replace NewCusters(A_d, D) by a procedure to find cliques in the graph where distance between any two nodes is $\leq$ threshold, d
- Still $O(n^2)$ algorithm but solves the long chains problem

MST Based Single-Link Algorithm

- A variation of single-link algorithm is based on using a Minimum Spanning Tree (MST)
- A MST subgraph that is a tree that spans all nodes and has minimal total weight
- Two clusters are merged if the distance between them is smallest as described in a MST of the graph



MST Single Link Algorithm

Input:

$D = \{t_1, t_2, ..., t_n\}$ // **Set of elements**

A // **Adjacency matrix showing distance between elements.**

Output:

DE // **Dendrogram represented as a set of ordered triples.**

MST Single Link Algorithm:

$d = 0$;

$k = n$;

$K = \{\{t_1\}, ..., \{t_n\}\}$;

$DE =< d, k, K >$; // **Initially dendrogram contains each element in its own cluster.**

$M = MST(A)$;

repeat

$oldk = k$;

$K_i, K_j =$ **two clusters closest together in MST;**

$K = K - \{K_i\} - \{K_j\} \cup \{K_i \cup K_j\}$;

$k = oldk - 1$;

$d = dis(K_i, K_j)$;

$DE = DE \cup < d, k, K >$; // **New set of clusters added to dendrogram.**

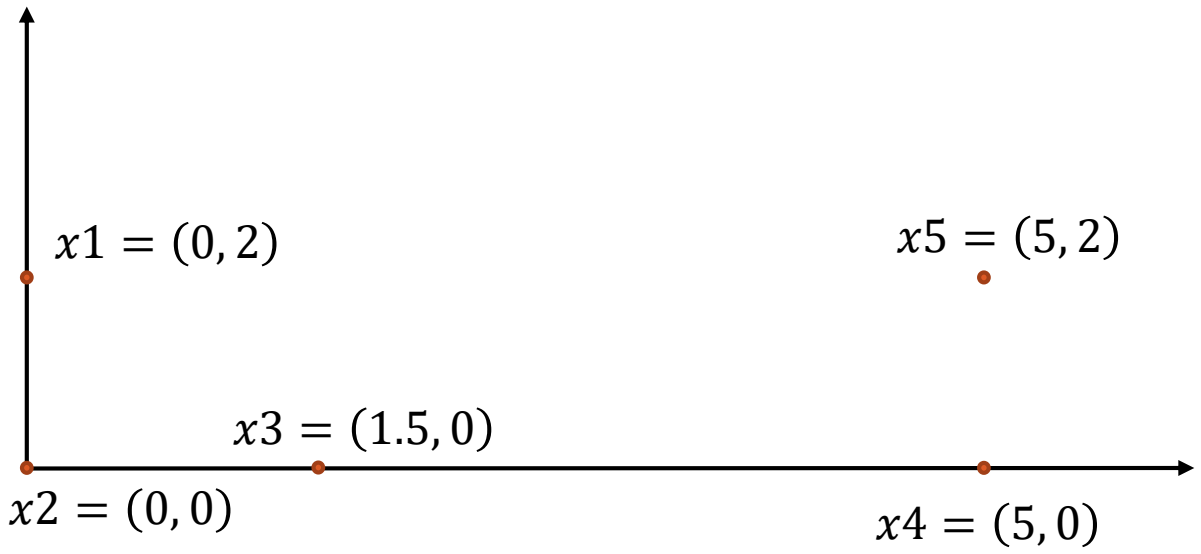
$dis(K_i, K_j) = \infty$;

until $k = 1$

Example (1 – 3)

- Apply the MST single-link approach to the dataset of 5 points in the 2D space: $x1 = (0, 2), x2 = (0, 0), x3 = (1.5, 0), x4 = (5, 0)$, and $x5 = (5, 2)$.

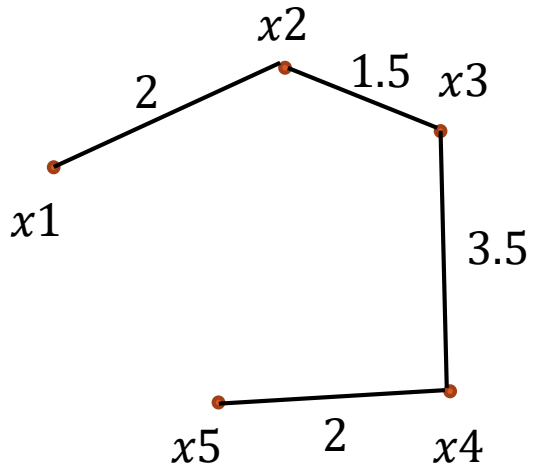
	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**



Example (2 − 3)

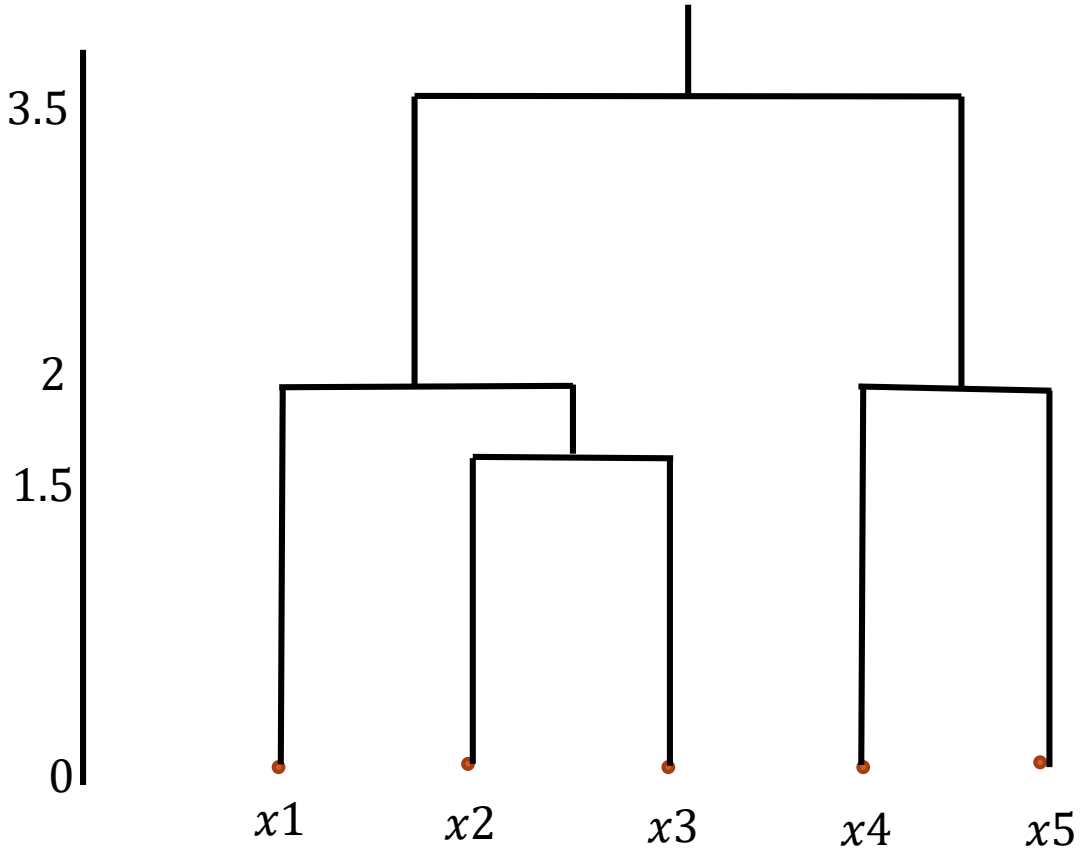
- Apply the MST single-link approach to the dataset of 5 points in the 2D space: $x1 = (0, 2), x2 = (0, 0), x3 = (1.5, 0), x4 = (5, 0)$, and $x5 = (5, 2)$.

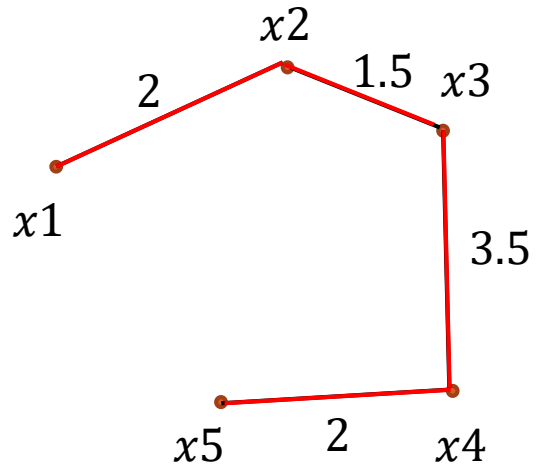
	$x1$	$x2$	$x3$	$x4$	$x5$
$x1$	**0**	**2**	**2.5**	**5.39**	**5**
$x2$		**0**	**1.5**	**5**	**5.39**
$x3$			**0**	**3.5**	**4.03**
$x4$				**0**	**2**
$x5$					**0**



Example (3 − 3)

- The dendrogram <d, k, K>:
  $\{< 0,\ 5,\ \{\{x1\},\ \{x2\},\ \{x3\},\{4\},\{x5\}\} >,\ < 1.5,\ 4,\ \{\{x1\},\ \{x2, x3\ \},\{x4\},\{x5\}\} >,$
  $< 2,\ 3,\ \{\{x1\},\ \{x2, x3\ \},\{x4, x5\}\},< 2,\ 2,\ \{\{x1, x2, x3\ \},\{x4, x5\}\},< 3.5,\ 1,$
  $\{\{x1, x2, x3\ , x4, x5\}\} >\}$





Complete-Link Algorithm

- This was described before
- Two clusters are merged if the distance between them is the smallest when considering the complete-link distances among clusters
- Complete-link finds compact clusters
- A variant of complete-link is called the furthest neighbor algorithm while single-link is sometimes called the nearest neighbor algorithm

The Average-Link Algorithm

- Two clusters are merged if the average distance between them $\leq$ threshold value, d

  (i.e., distance between any two points $x_i \in K_i$ and $x_j \in K_j$) $\leq$ threshold value, d

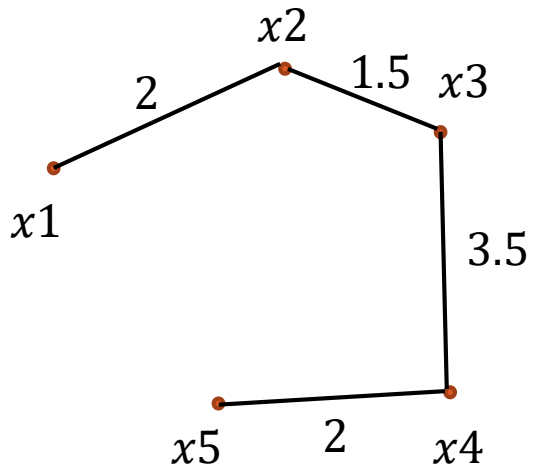
Divisive Clustering Algorithms

- Top-down approach
- Initially all items in one cluster
- Clusters are successively divided until either each object is in its own cluster or a termination condition is reached
- Reverse of the agglomerative algorithms
- One approach uses the MST version of the single link algorithm
- Clusters are split by removing edges from the MST starting with the edge with the largest weight

Example

- Apply the MST single-link divisive approach to the dataset of 5 points:

$x1 = (0, 2), x2 = (0, 0), x3 = (1.5, 0), x4 = (5, 0)$, and $x5 = (5, 2)$.



Major Weaknesses of Hierarchical Clustering Methods

- Do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
- Not incremental
- Can not undo what was done previously