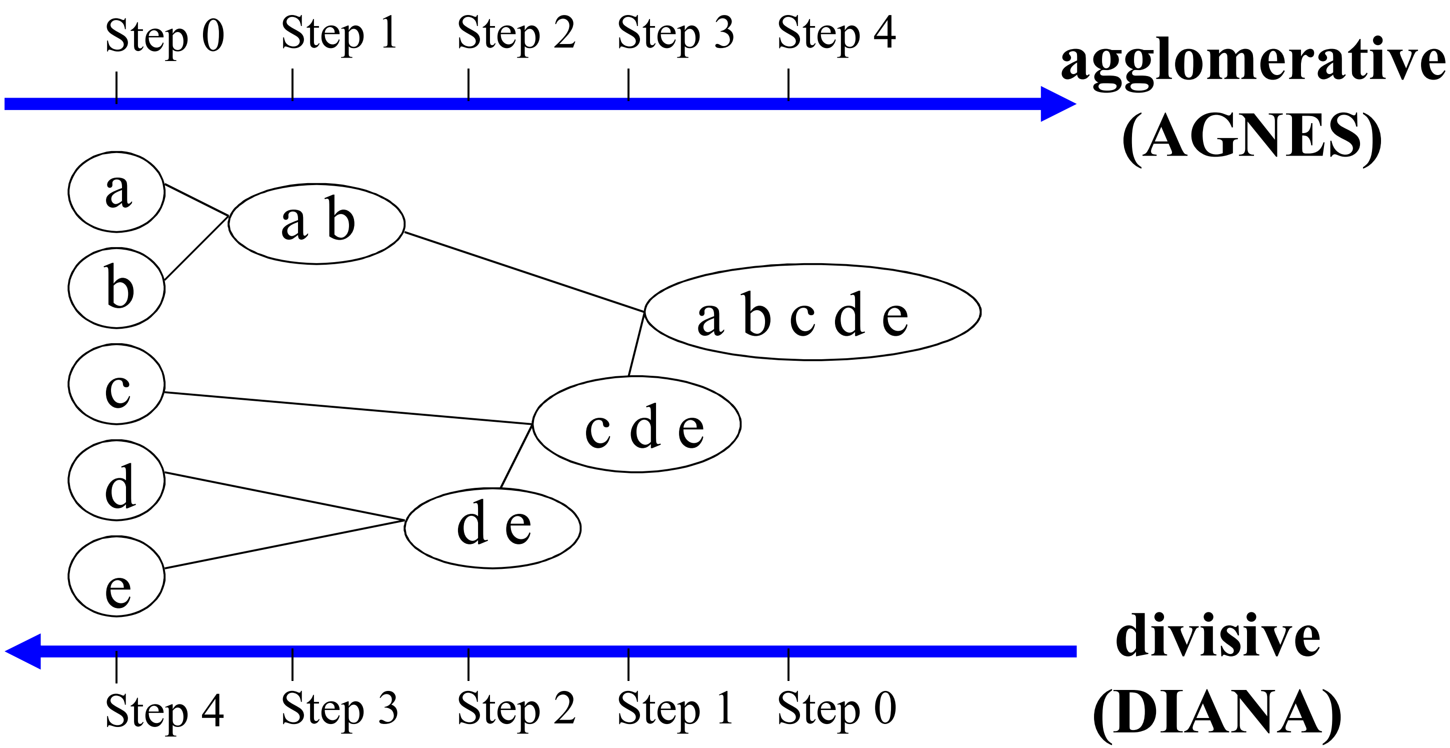
Chapter 7. Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Density-Based Methods
7. Clustering High-Dimensional Data
8. Constraint-Based Clustering
9. Outlier Analysis
10. Summary

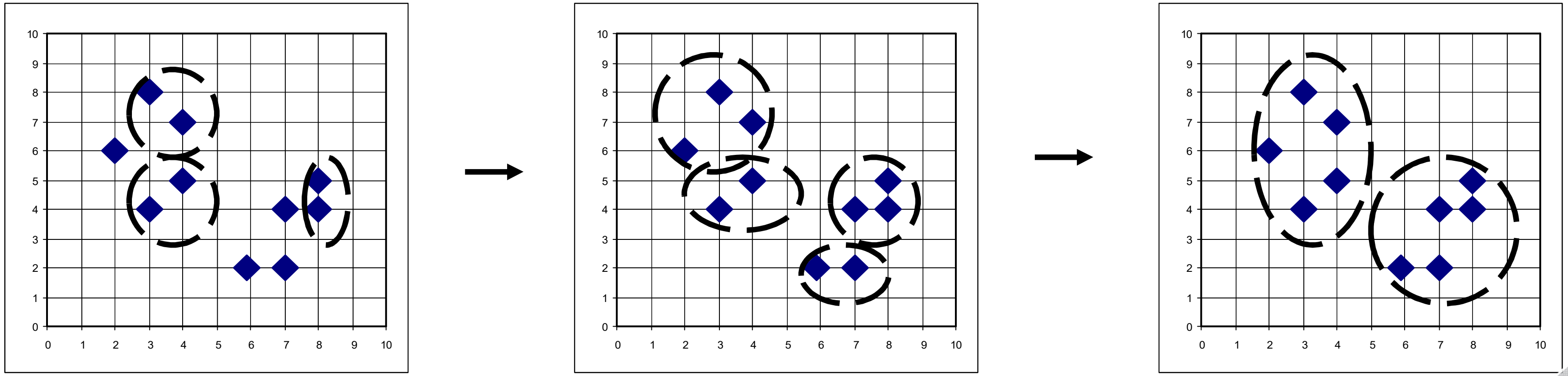
Hierarchical Clustering

- Use a distance matrix as clustering criteria
- Does not require the number of clusters ***k*** as an input, but needs a termination condition

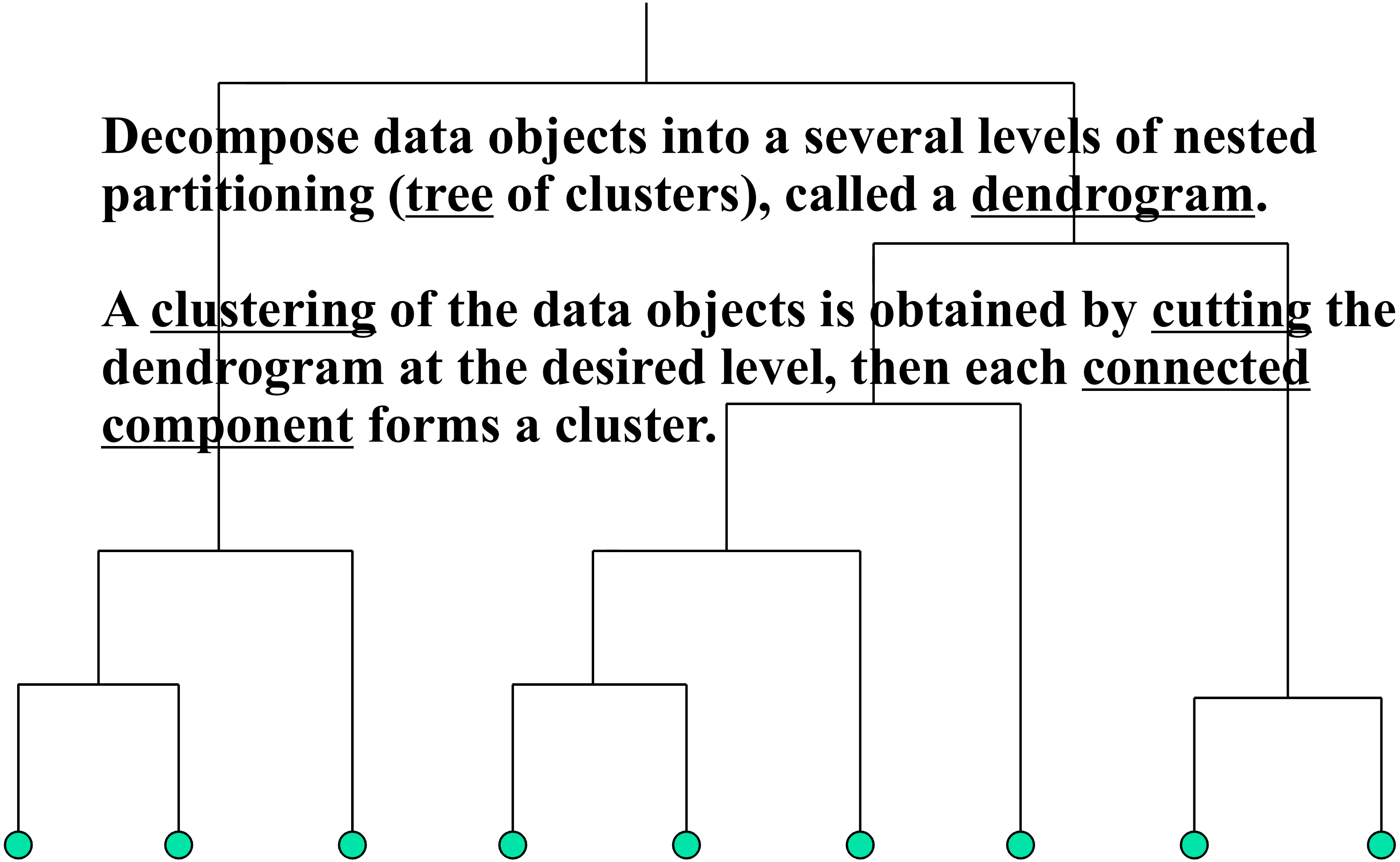


AGNES (Agglomerative Nesting)

- Introduced in Kaufmann and Rousseeuw (1990)
  - Implemented in statistical analysis packages, Splus
- Use the single-link method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Dendrogram: How the Clusters are Merged

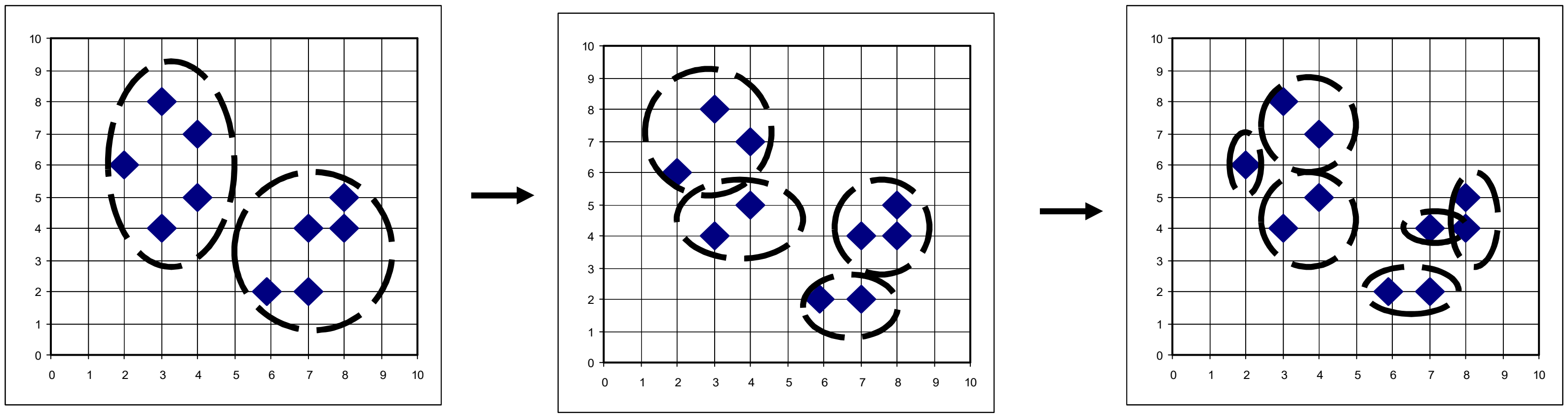


Decompose data objects into a several levels of nested partitioning (<u>tree</u> of clusters), called a <u>dendrogram</u>.

A <u>clustering</u> of the data objects is obtained by <u>cutting</u> the dendrogram at the desired level, then each <u>connected component</u> forms a cluster.

DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, Splus
- Inverse order of AGNES
- Eventually each node forms a cluster on its own

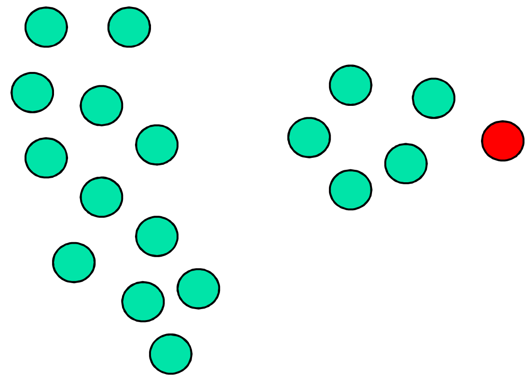


DIANA (Divisive Analysis)

- Outline
  - Initially, there is one large cluster consisting of all *n* objects
  - At each subsequent step, the largest available cluster is split into two clusters
    - Until finally all clusters comprise of a single object.
    - Thus, the hierarchy is built in *n*-1 steps.
- Complexity in the first step
  - Agglomerative method: $\frac{n(n-1)}{2}$ possible combinations
  - Divisive method: $2^{n-1}-1$ possible combinations
    - Considerably larger than an agglomerative method

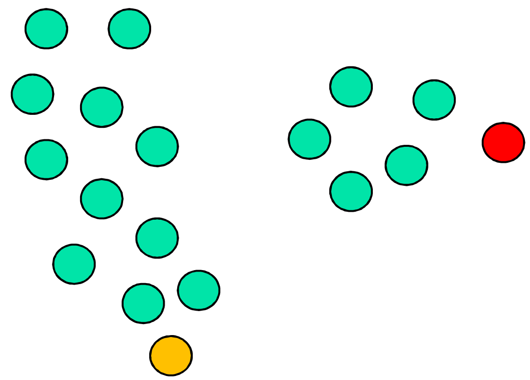
DIANA (Divisive Analysis)

- To avoid considering all possibilities, the algorithm proceeds as follows.
  1. Find the object, which has the highest average dissimilarity to all other objects. This object initiates a new cluster– a sort of a *splinter group*.
  2. For each object *i* outside the *splinter group,* compute
     $D_i = [average\ d(i,j)\ j \notin R_{splinter\ group}] - [average\ d(i,j)\ j \in R_{splinter\ group}]$
  3. Find an object *h* for which the difference D_h is the largest. If D_h is positive, then *h* is, on the average close to the splinter group. Put h into the splinter group.



DIANA (Divisive Analysis)

- To avoid considering all possibilities, the algorithm proceeds as follows.
  1. Repeat *Steps* 2 and 3 until all differences D_h are negative. The data set is then split into two clusters.
  2. Select the cluster with the largest diameter. The diameter of a cluster is the largest dissimilarity between any two of its objects. Then divide this cluster, following steps 1-4.
  3. Repeat *Step* 5 until all clusters contain only a single object.



Advacned Hierarchical Clustering Methods

- Major weakness of agglomerative clustering methods
  - do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
- Integration of hierarchical with distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - ROCK (1999): clustering categorical data by neighbor and link analysis
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

BIRCH (1996)

- Birch: Balanced Iterative Reducing and Clustering using Hierarchies (Zhang, Ramakrishnan & Livny, SIGMOD'96)
- Incrementally construct a CF (Clustering Feature) tree (cf. B-tree), a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- *Scales linearly*: finds a good clustering with a single scan and improves the quality with a few additional scans
- *Weakness:* handles only numeric data, and sensitive to the order of the data records

Clustering Feature Vector in BIRCH

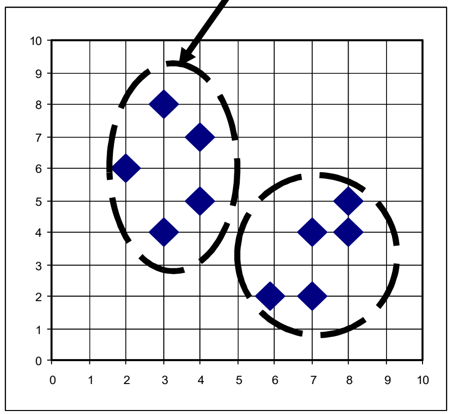
Clustering Feature: $CF = (N, \overrightarrow{LS}, SS)$

N: **Number of data points**

LS: $\sum_{i=1}^{N} = \overrightarrow{X_i}$

SS: $\sum_{i=1}^{N} = \overrightarrow{X_i}^2$

CF = (5, (16,30),(54,190))

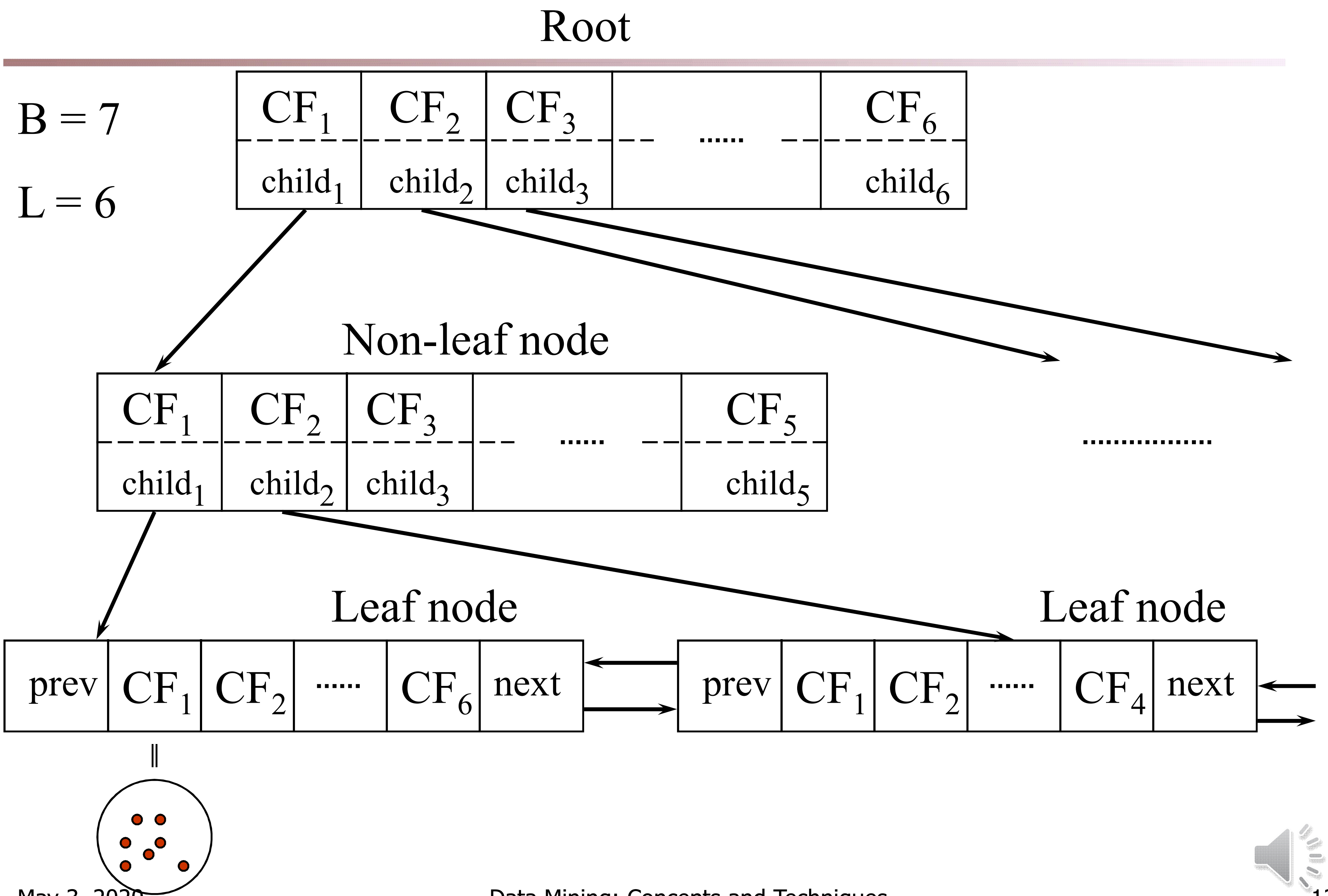


(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

CF-Tree in BIRCH

- Clustering feature:
  - Summary of the statistics for a given cluster: the 0-th, 1st and 2nd moments of the cluster from the statistical point of view
  - Registers crucial measurements for computing cluster and utilizes storage efficiently
- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
  - A non-leaf node in a tree has descendants or "children"
  - A non-leaf node stores the sum of the CFs of their children
- A CF tree has two parameters
  - Branching factor: specify the maximum number of children
  - threshold: max diameter of a cluster stored at the leaf node

The CF Tree Structure



Clustering Categorical Data: The ROCK Algorithm

- ROCK: RObust Clustering using linKs, ICDE'99
- Major ideas
  - Use the notion of *links* to measure similarity/proximity
    - Not distance-based

Similarity Measure in ROCK

- Traditional measures for categorical data may not work well, e.g., Jaccard coefficient
- *Jaccard coefficient*-based similarity function:

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

  - Ex. Let T_1 = {a, b, c}, T_2 = {c, d, e}

$$Sim(T_1, T_2) = \frac{|\{c\}|}{|\{a, b, c, d, e\}|} = \frac{1}{5} = 0.2$$

Similarity Measure in ROCK

- Example: Two groups (clusters) of transactions
  - C_1. <a, b, c, d, e>: {a, b, c}, {a, b, d}, {a, b, e}, {a, c, d}, {a, c, e}, {a, d, e}, {b, c, d}, {b, c, e}, {b, d, e}, {c, d, e}
  - C_2. <a, b, f, g>: {a, b, f}, {a, b, g}, {a, f, g}, {b, f, g}
- Jaccard coefficient may lead to a wrong clustering result
  - C_1: 0.2 ({a, **b**, c}, {**b**, d, e}} to 0.5 ({**a, b**, c}, {**a, b**, d})
  - C_1 & C_2: could be as high as 0.5 ({**a, b**, c}, {**a, b**, f})

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

Link Measure in ROCK

- Links: # of common *neighbors* (threshold = 0.5 in jC)
  - C_1 <a, b, c, d, e>: **{a, b, c}**, {a, b, d}, {a, b, e}, {a, c, d}, {a, c, e}, {a, d, e}, {b, c, d}, {b, c, e}, {b, d, e}, **{c, d, e}**
  - C_2 <a, b, f, g>: **{a, b, f}**, {a, b, g}, {a, f, g}, {b, f, g}
- Let T_1 = {**a**, **b**, **c**}, T_2 = {**c**, d, e}, T_3 = {**a**, **b**, f}
  - link(T_1, T_2) = 4, *since they have 4 common neighbors*
    - {a, c, d}, {a, c, e}, {b, c, d}, {b, c, e}
  - link(T_1, T_3) = 3, *since they have 3 common neighbors*
    - {a, b, d}, {a, b, e}, {a, b, g}
- Thus, link is a better measure than Jaccard coefficient

CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- CHAMELEON: by G. Karypis, E.H. Han, and V. Kumar'99
- Measures the similarity based on a dynamic model
  - Two clusters are merged only if the *interconnectivity* and *closeness (proximity)* between two clusters are high
    - ***Relative*** *to* the internal interconnectivity of the clusters and internal closeness of items within the clusters

CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- Draw a k-nearest neighbor graph first
  - Node: object, edge: k-nearest neighbor's link, weight: similarity
- A two-phase algorithm
  - Use a graph partitioning algorithm:
    - Cluster objects into a large number of relatively small sub-clusters
  - Use an agglomerative hierarchical clustering algorithm:
    - Find the genuine clusters by repeatedly combining these sub-clusters

CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

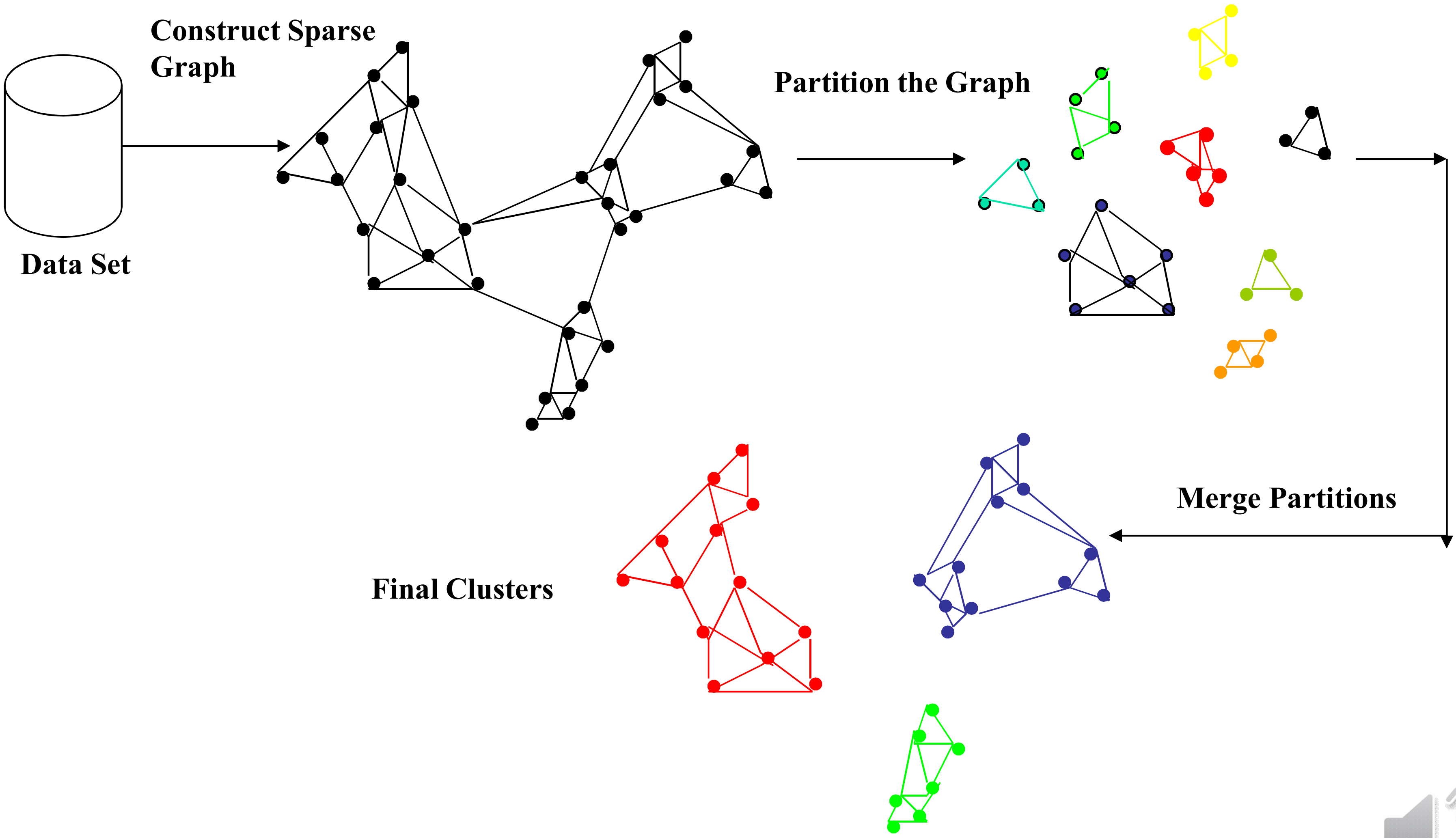
- Partitioning
  - To minimize the edge cut (**METIS**)
    - Tries to split a graph into two subgraphs of nearly equal sizes
- Relative interconnectivity

$$RI(C_i, C_j) = \frac{|EC_{\{C_i, C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)},$$

- Relative closeness

$$RC(C_i, C_j) = \frac{\bar{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC_{C_j}}},$$

Overall Framework of CHAMELEON



CHAMELEON (Clustering Complex Objects)