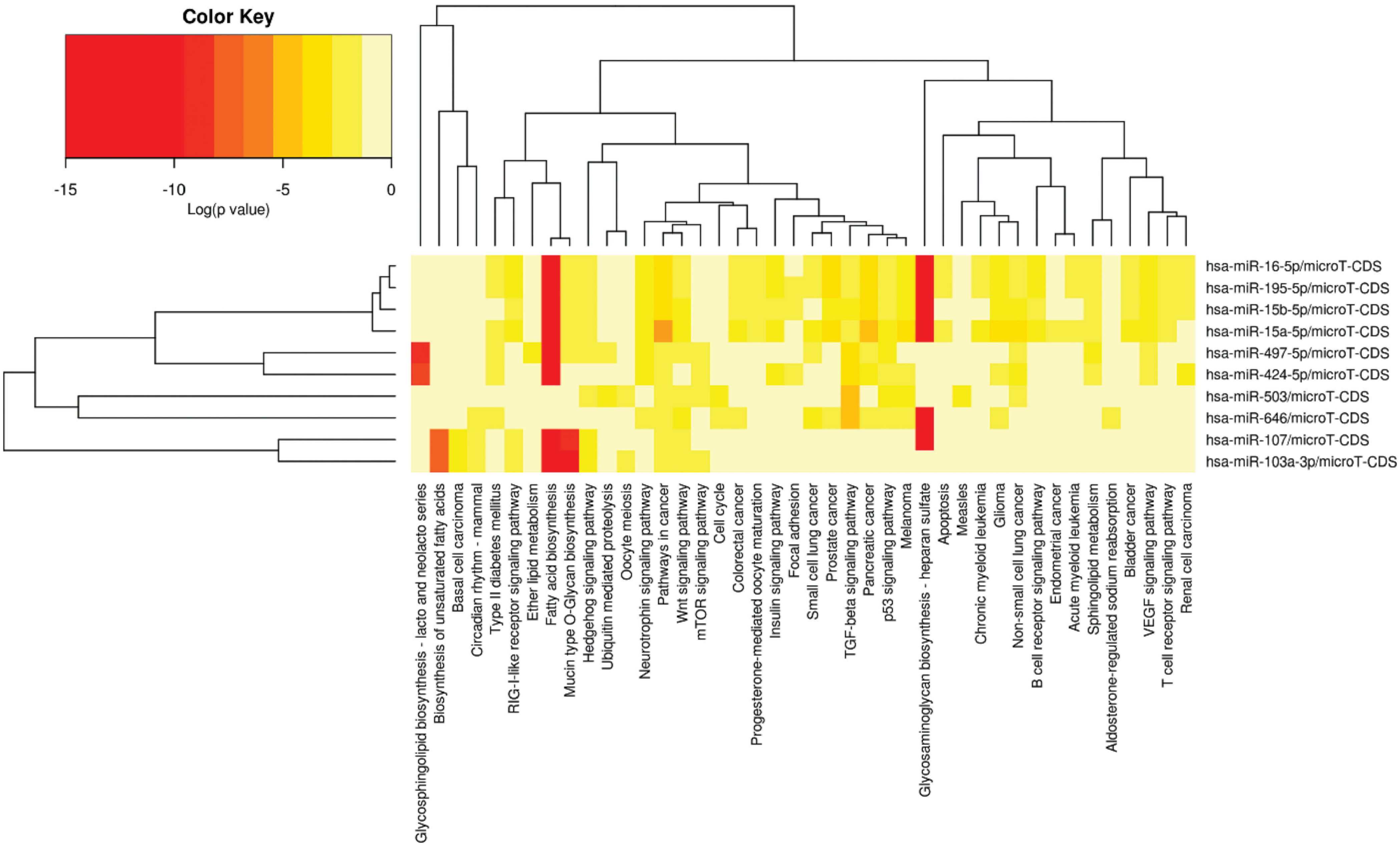
CS4801 : Hierarchical Clustering

Sahely Bhadra

26/9/2017

1. Hierarchical cluster : Agglomerative and Divisive
2. BIRCH

Hierarchical Clustering



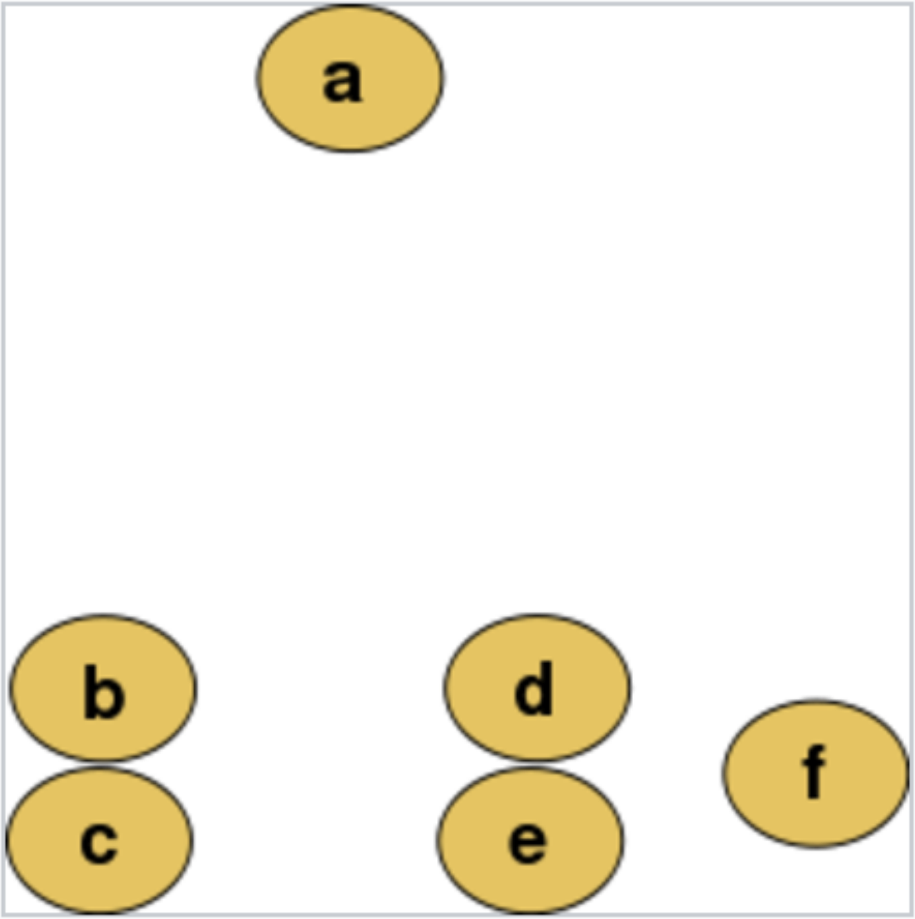
Hierarchical Clustering

- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both Agglomerative and Divisive is of no use.

Hierarchical Agglomerative Clustering (HAC) Algorithm

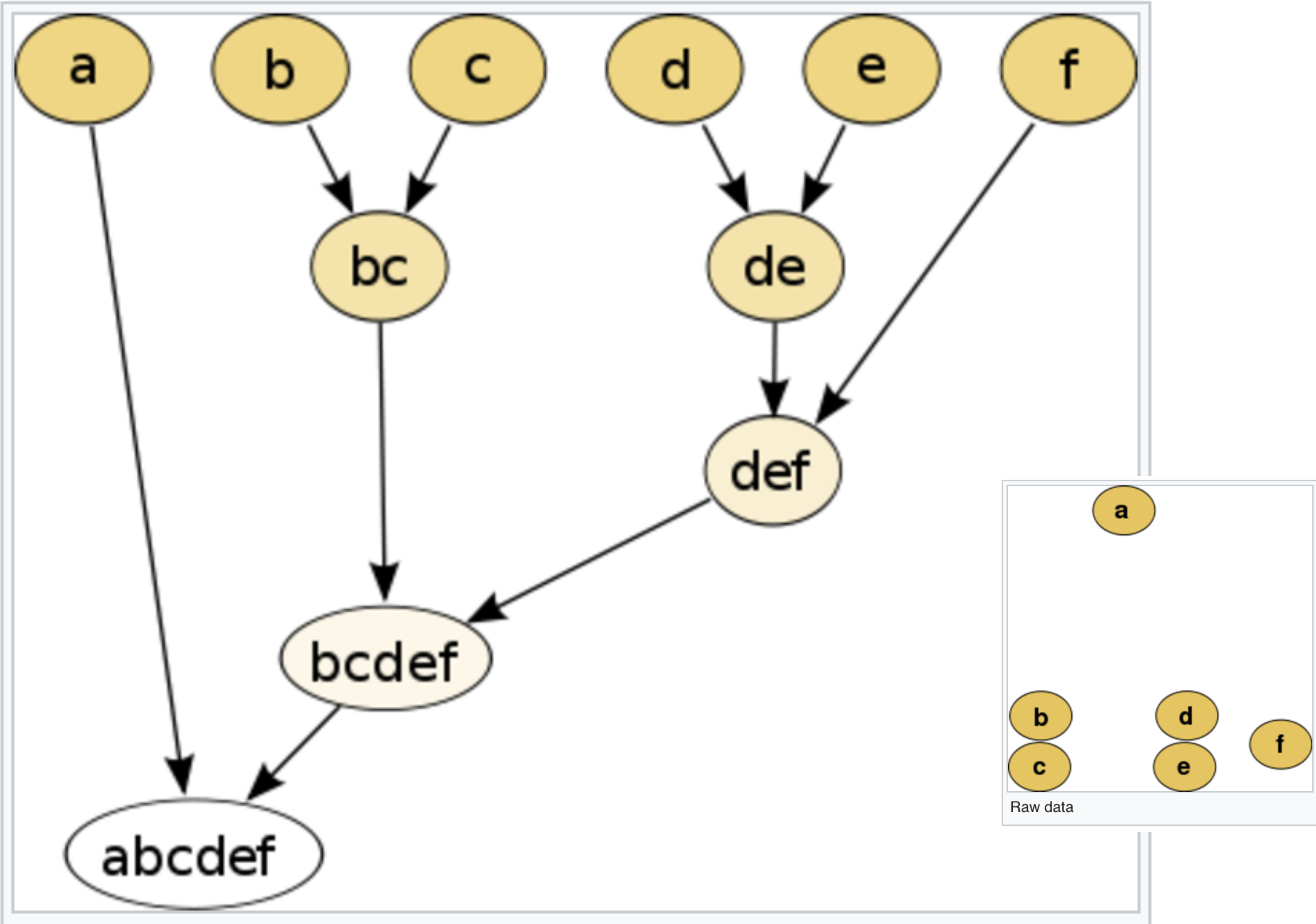
Start with all instances in their own cluster.
Until there is only one cluster:
  Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.
  Replace c_i and c_j with a single cluster $c_i \cup c_j$

Data points



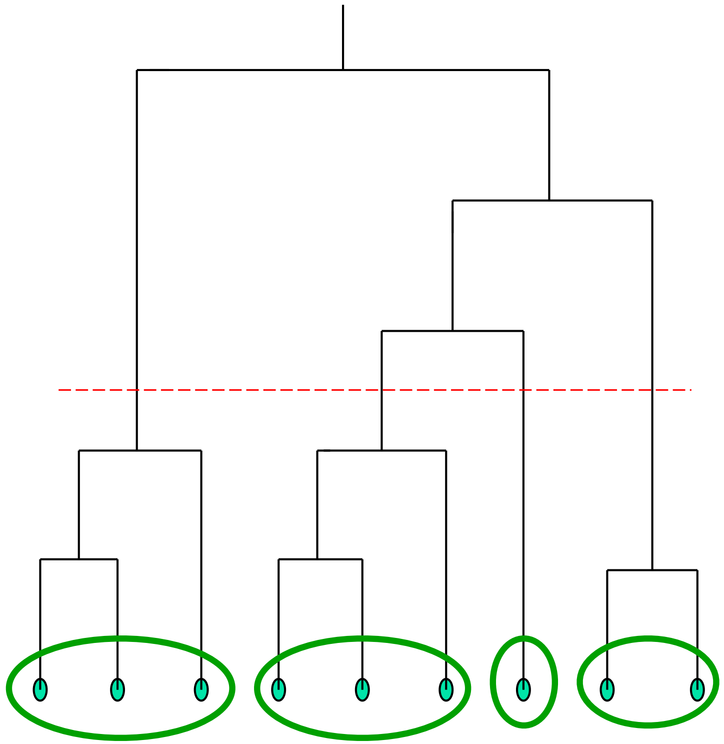
Raw data

Agglomerative Dendrogram

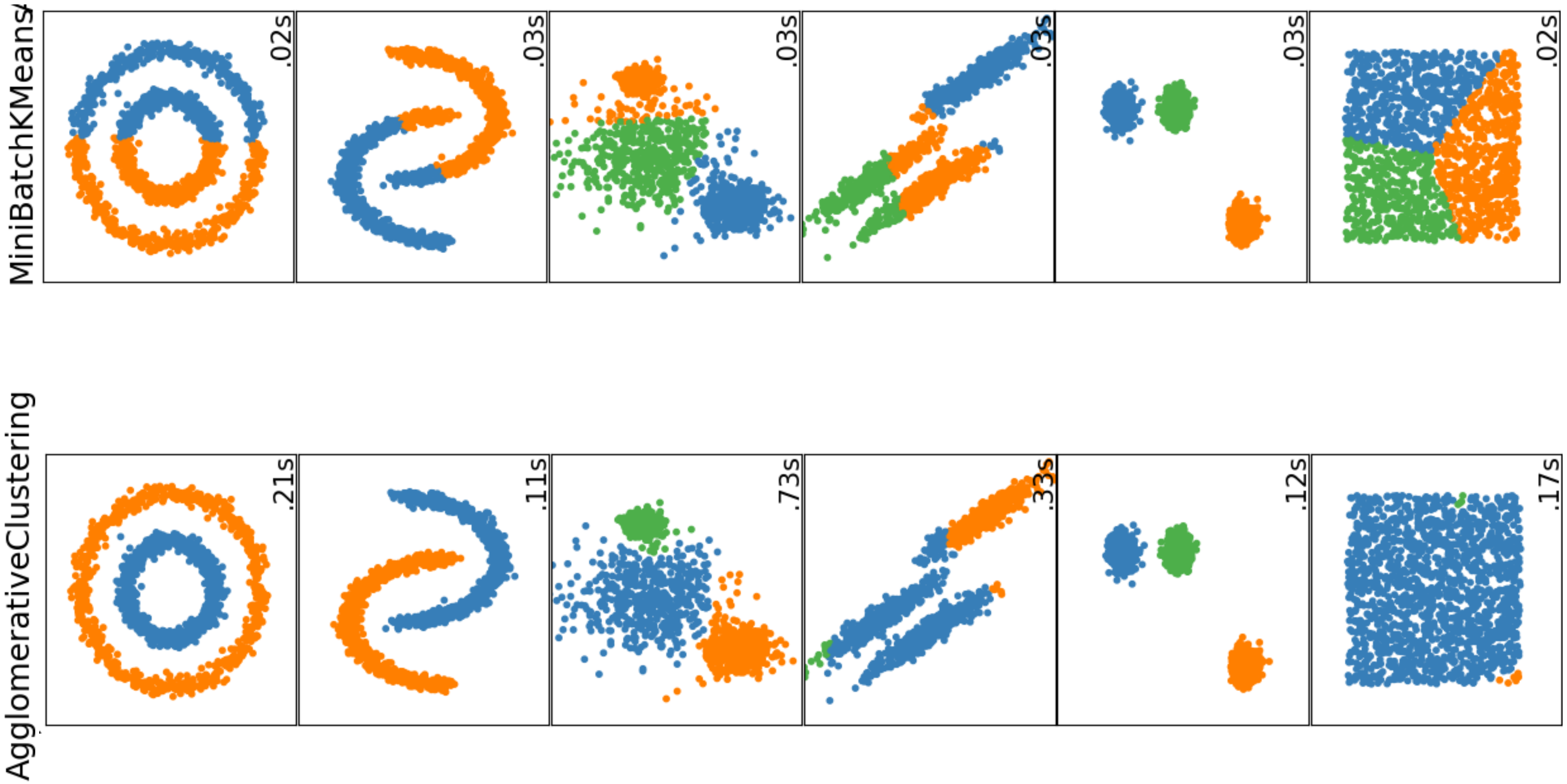


Cutting level of Dendrogram

- Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.



Agglomerative Clustering



Run Time

- Time complexity : $O(n^3)$
- Memory : $O(n^2)$
- *With heap* : $O(n^2 \log n)$

Distance Metric

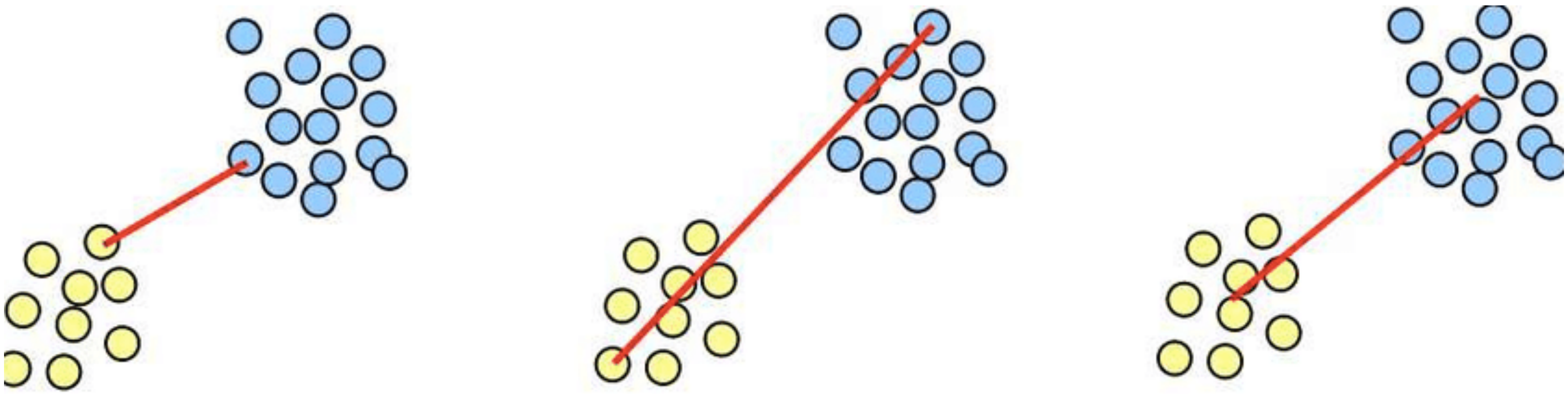
Names	Formula
Euclidean distance	$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\|a-b\|_1 = \sum_i \|a_i - b_i\|$
maximum distance	$\|a-b\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis distance	$\sqrt{(a-b)^\top S^{-1} (a-b)}$ where S is the Covariance matrix

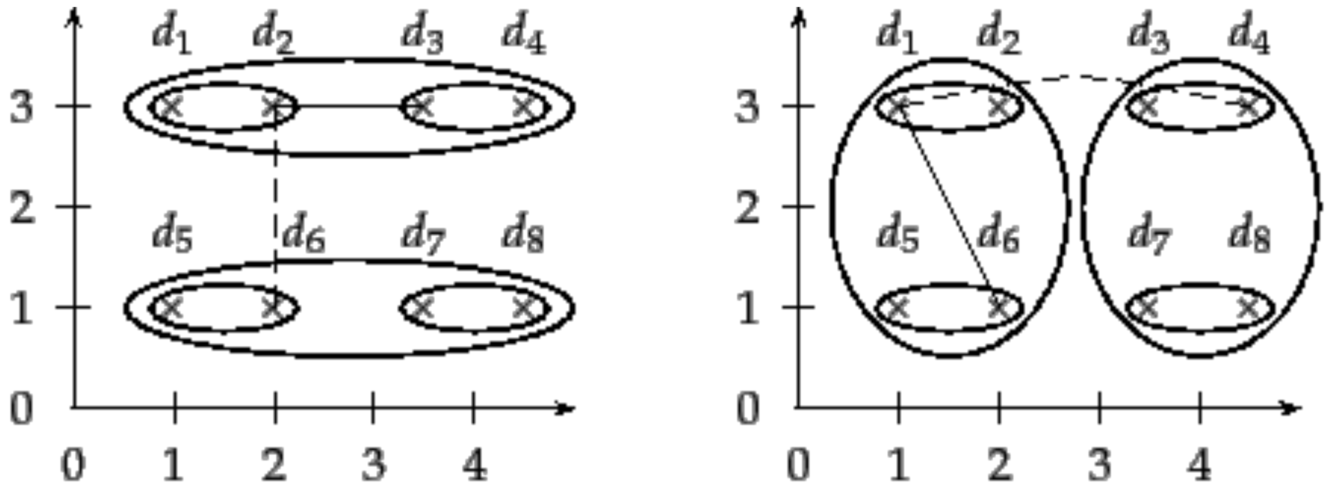
Linkage

Names	Formula
Maximum or complete-linkage clustering	$\max\{\, d(a,b) : a \in A,\, b \in B \,\}.$
Minimum or single-linkage clustering	$\min\{\, d(a,b) : a \in A,\, b \in B \,\}.$
Mean or average linkage clustering, or UPGMA	$\frac{1}{\lvert A\rvert \lvert B\rvert} \sum_{a \in A} \sum_{b \in B} d(a,b).$
Centroid linkage clustering, or UPGMC	$\lVert c_s - c_t \rVert$ where c_s and c_t are the centroids of clusters *s* and *t*, respectively.
Minimum energy clustering	$\frac{2}{nm} \sum_{i,j=1}^{n,m} \lVert a_i - b_j \rVert_2 - \frac{1}{n^2} \sum_{i,j=1}^{n} \lVert a_i - a_j \rVert_2 - \frac{1}{m^2} \sum_{i,j=1}^{m} \lVert b_i - b_j \rVert_2$

Unweighted **P**air **G**roup **M**ethod with **A**rithmetic Mean

Mean is difficult to compute for categorical data

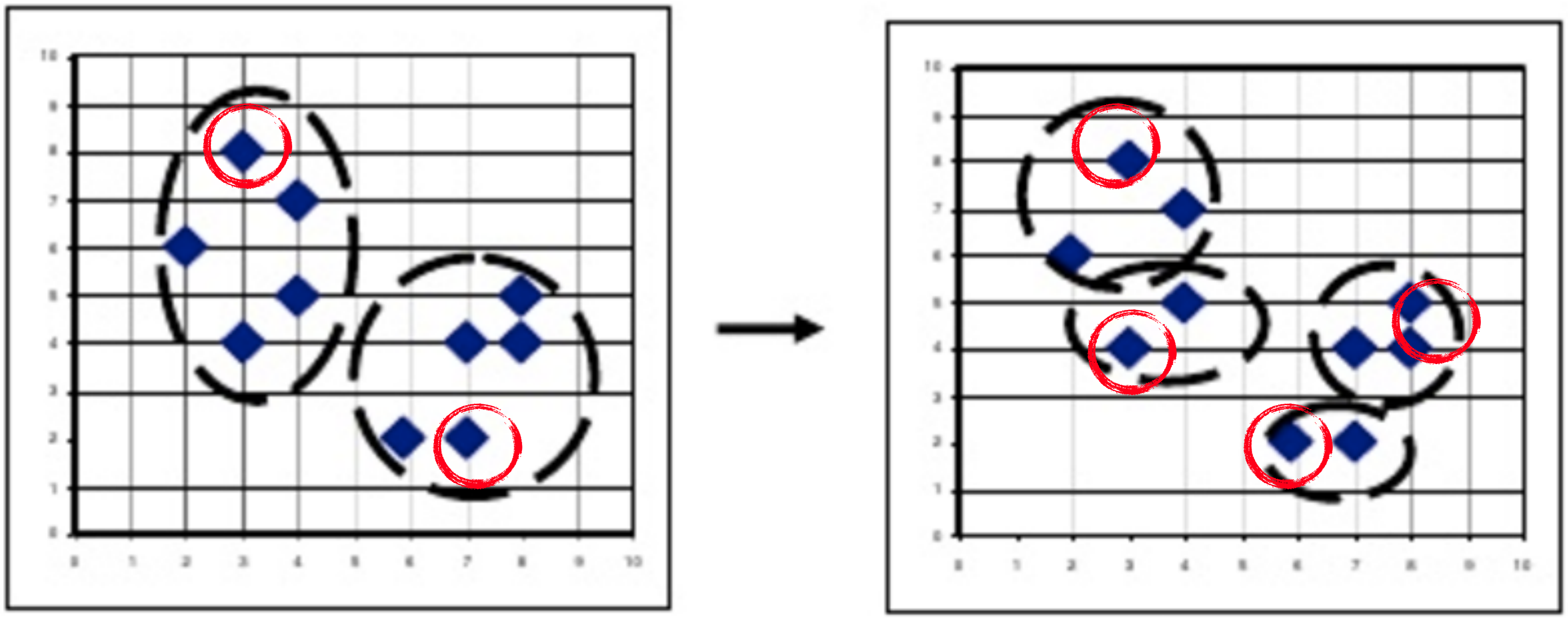




► **Figure 17.2** A single-link (left) and complete-link (right) clustering of eight documents. The ellipses correspond to successive clustering stages. Left: The single-link similarity of the two upper two-point clusters is the similarity of d_2 and d_3 (solid line), which is greater than the single-link similarity of the two left two-point clusters (dashed line). Right: The complete-link similarity of the two upper two-point clusters is the similarity of d_1 and d_4 (dashed line), which is smaller than the complete-link similarity of the two left two-point clusters (solid line).

DIANA (DIvisive ANAlysis Clustering)

Find object which have maximum distance between them
Cluster all object depending upon their closeness to these two selected objects



Issues

- **Lack of a Global Objective Function:** agglomerative hierarchical clustering techniques perform clustering on a local level and as such there is no global objective function like in the K-Means algorithm.

- **No Backtracking:** a particular merge or split turns out to be poor choice, it cannot be corrected.

- **Deciding level of clustering :** subjective decision

- **Complexity** : Time Complexity: $O(n^2 \log n)$

CS4801 : BIRCH

Sahely Bhadra

27/9/2017

BIRCH

- BIRCH: Balanced Iterative Reducing and Clustering Using Hierarchies

- Agglomerative Clustering designed for clustering a large amount of numerical data

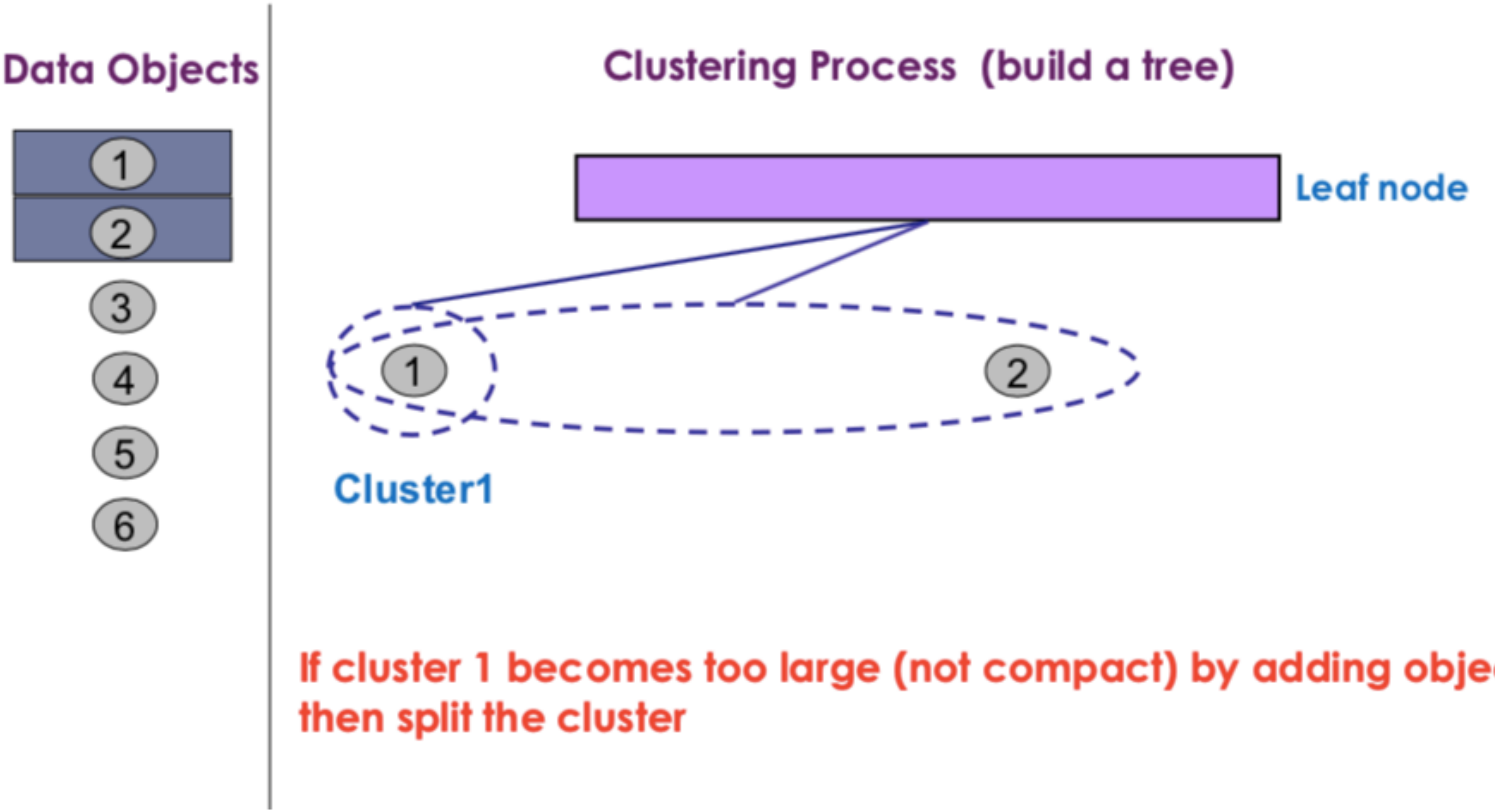
- What Birch algorithm tries to solve?

  - Most of the existing algorithms DO NOT consider the case that datasets can be **too large to fit in main memory**

  - They DO NOT concentrate on **minimizing the number of scans of the dataset** I/O costs are very high

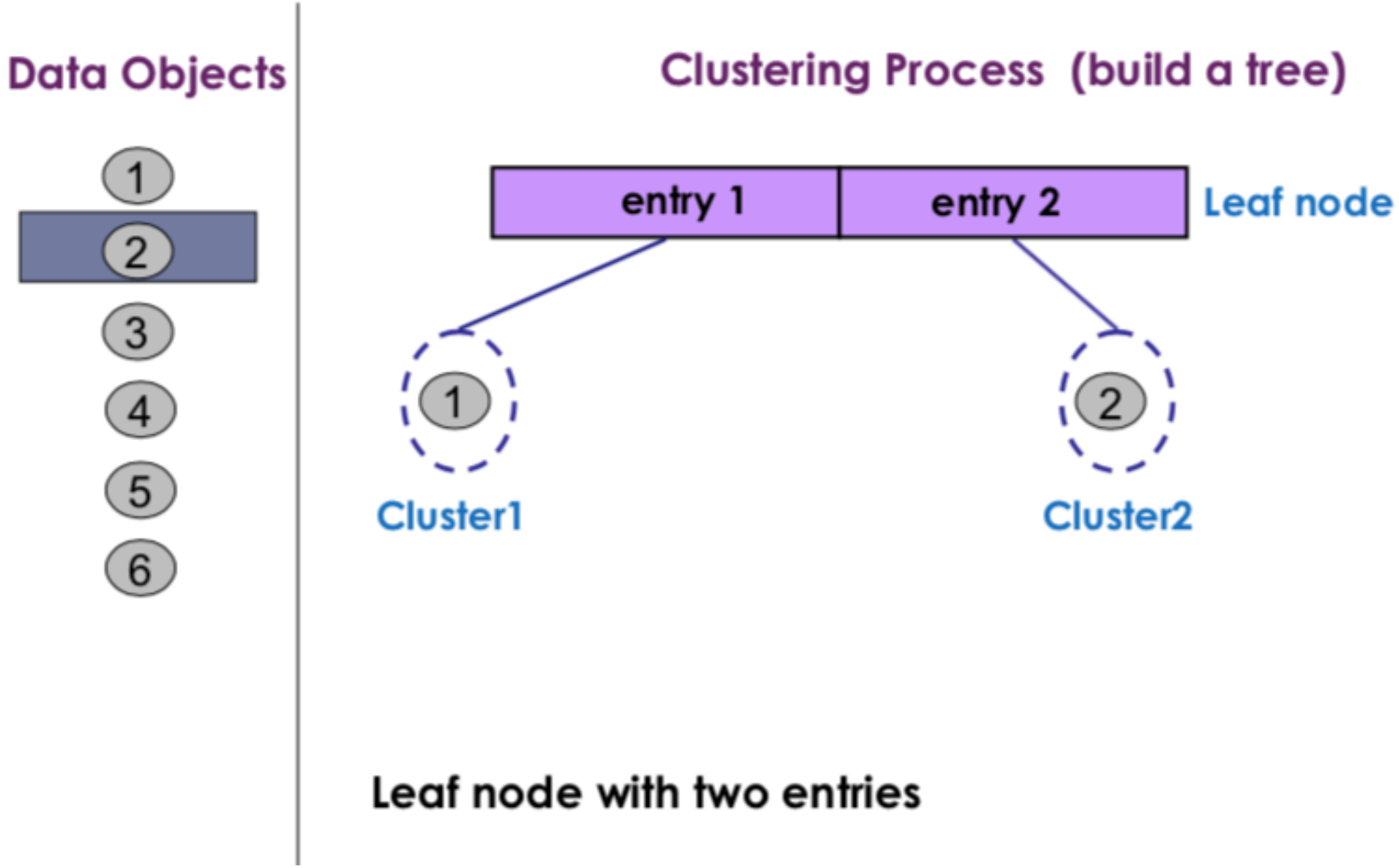
  - The complexity of BIRCH is **O(n)** where n is the number of objects to be clustered.

BIRCH



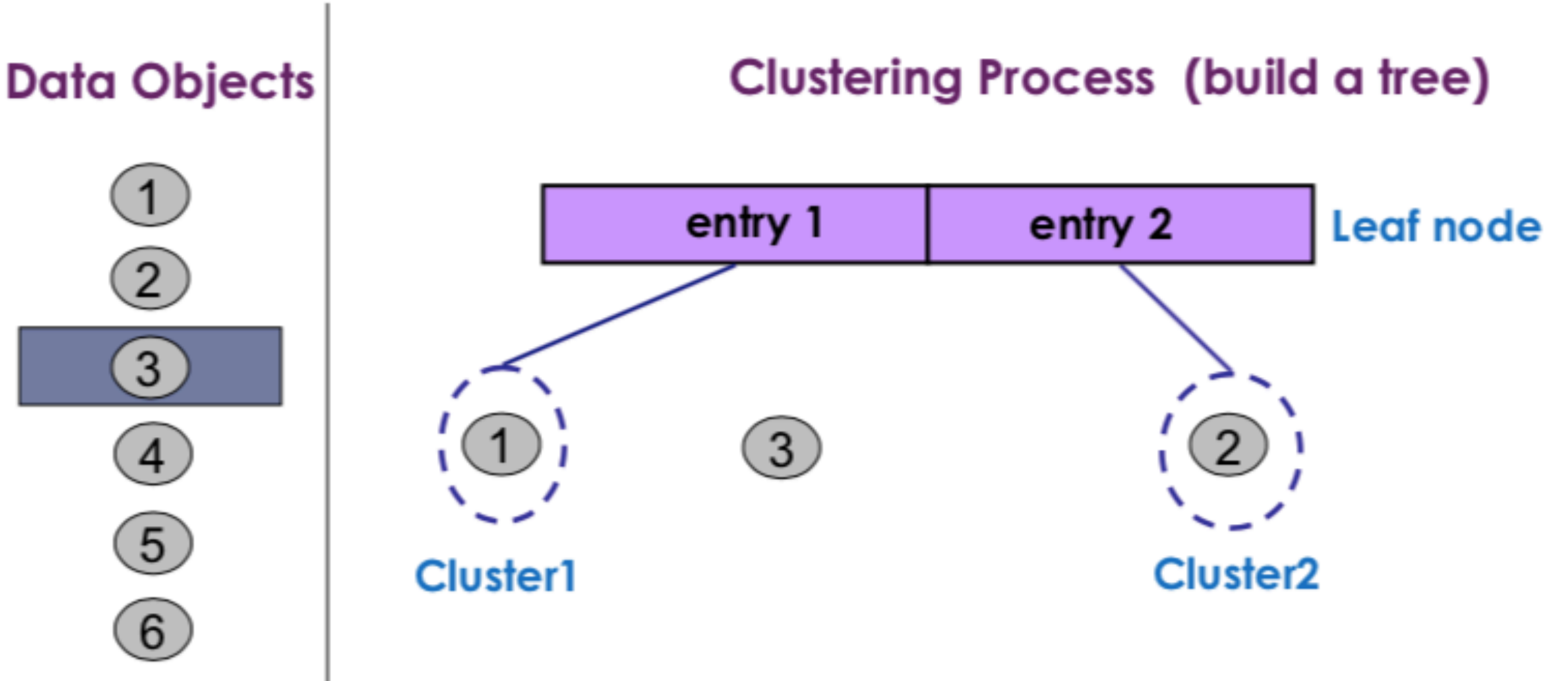
If cluster 1 becomes too large (not compact) by adding object 2, then split the cluster

BIRCH



Data Objects
1
2
3
4
5
6
Clustering Process (build a tree)
entry 1
entry 2
Leaf node
1
2
Cluster1
Cluster2
Leaf node with two entries

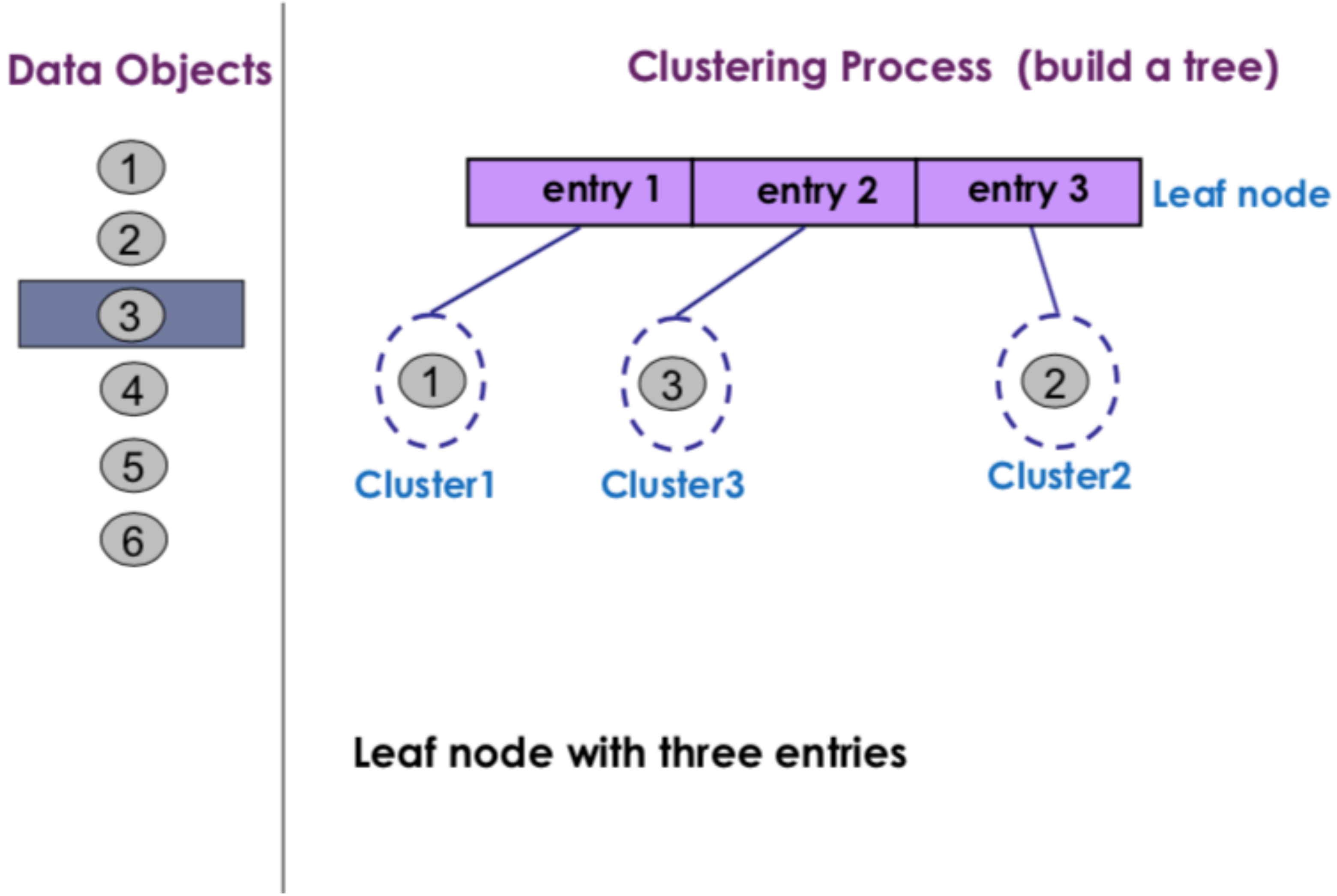
BIRCH



entry1 is the closest to object 3

If cluster 1 becomes too large by adding object 3, then split the cluster

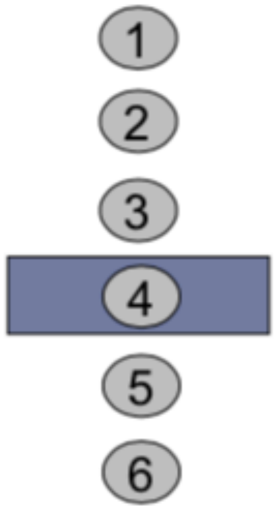
BIRCH



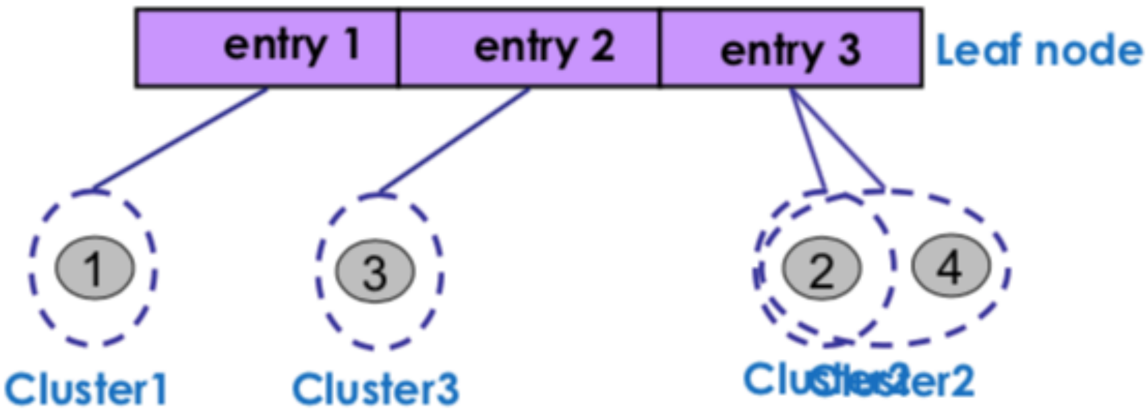
Leaf node with three entries

BIRCH

Data Objects



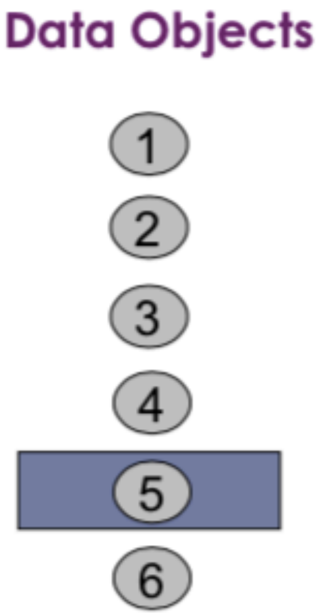
Clustering Process (build a tree)

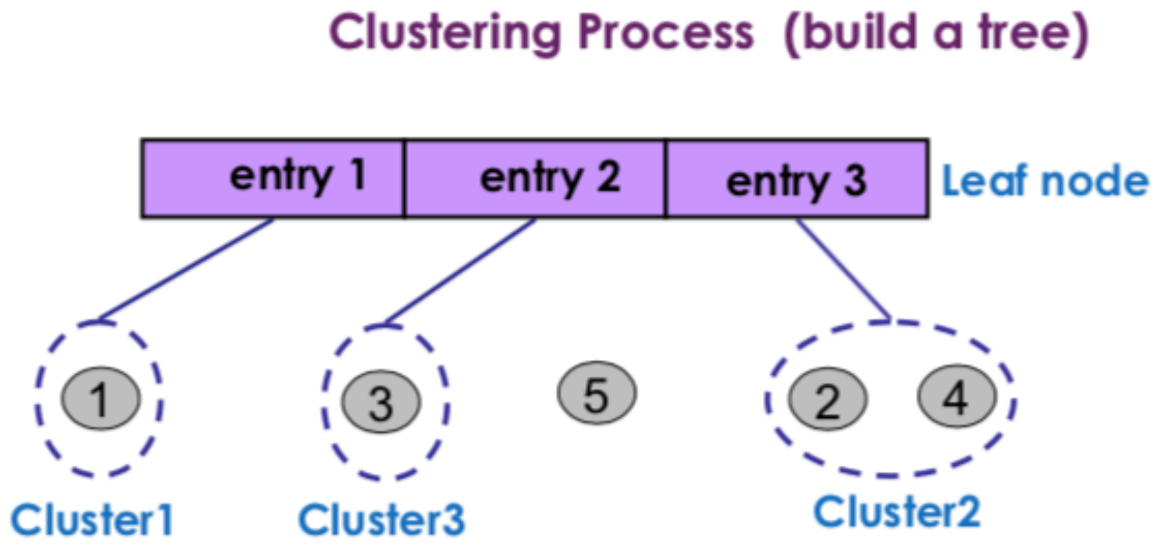


entry3 is the closest to object 4

Cluster 2 remains compact when adding object 4
then add object 4 to cluster 2

BIRCH



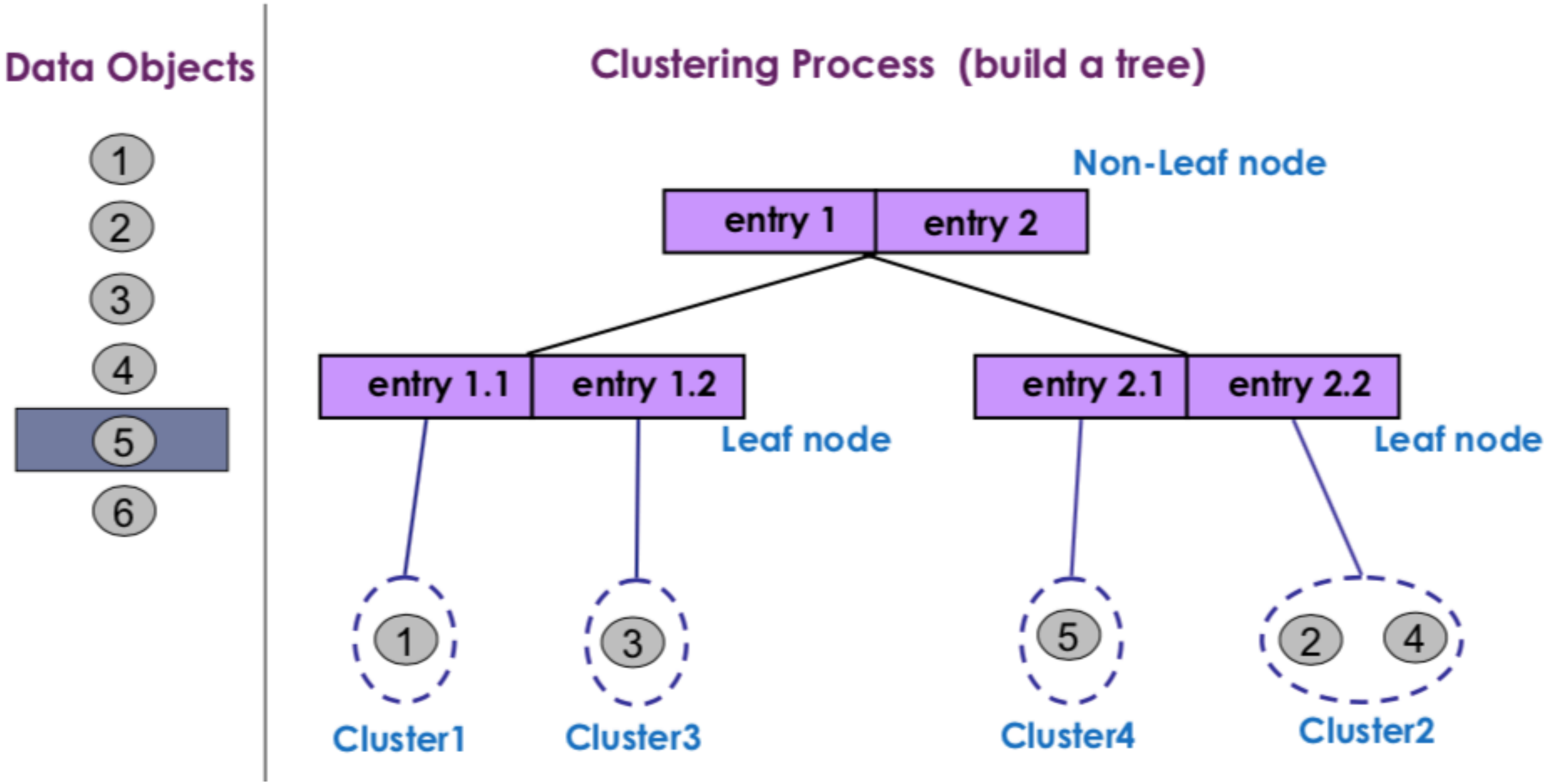


entry2 is the closest to object 5

Cluster 3 becomes too large by adding object 5
then split cluster 3?

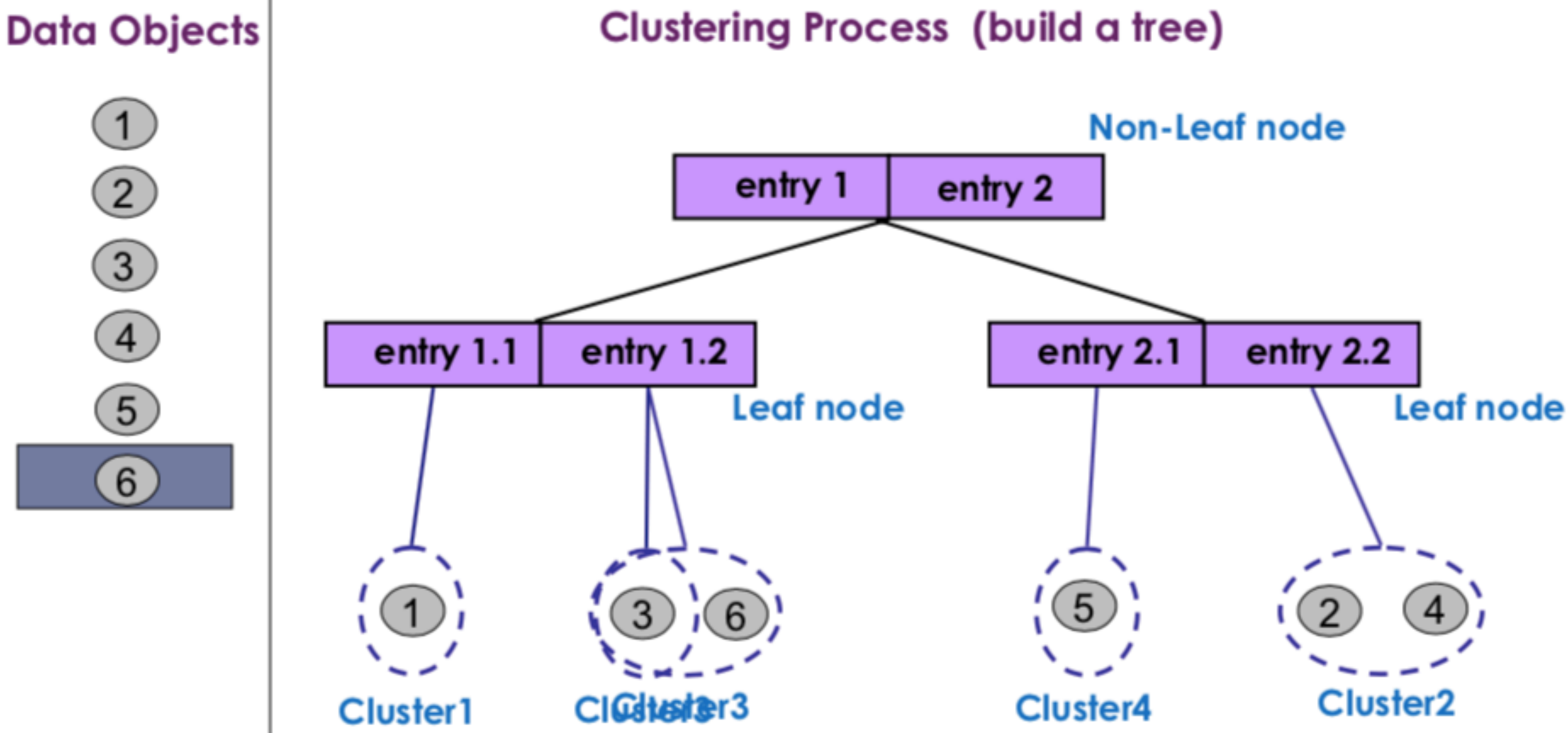
BUT there is a limit to the number of entries a node can have
Thus, split the node

BIRCH



Data Objects
1
2
3
4
5
6
Clustering Process (build a tree)
Non-Leaf node
entry 1
entry 2
entry 1.1
entry 1.2
entry 2.1
entry 2.2
Leaf node
Leaf node
1
3
5
2
4
Cluster1
Cluster3
Cluster4
Cluster2

BIRCH



entry1.2 is the closest to object 6

Cluster 3 remains compact when adding object 6
then add object 6 to cluster 3

BIRCH

Clustering Feature (CF)

- Summary of the statistics for a given cluster: the 0-th, 1st and 2nd
- moments of the cluster from the statistical point of view
- Used to compute centroids, and measure the compactness and distance of clusters

CF-Tree

- height-balanced tree
- two parameters:
  - number of entries in each node
  - The *diameter* of all entries in a leaf node
- Leaf nodes are connected via *prev* and *next* pointers

Clustering Features

Clustering features(CF) are organised in a ***CF tree***

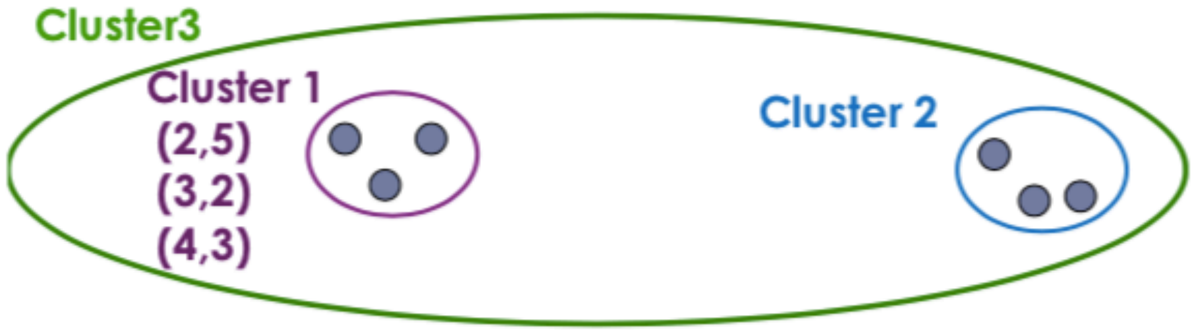
- Entries for each child : [CF_i, $Child_i$]
- **CF = (N, LS, SS)**

N: Number of data points

LS: linear sum of N points= $\sum_{i=1}^{N} X_i$

SS: square sum of N points= $\sum_{i=1}^{N} X_i^{\,2}$

CF3=CF1+CF2= ⟨3+3, (9+35, 10+36), (29+417 , 38+440)⟩ = ⟨6, (44,46), (446 ,478)⟩



CF_2= ⟨3, (35,36), (417 ,440)⟩

CF_1= ⟨3, (2+3+4 , 5+2+3), ($2^2+3^2+4^2$, $5^2+2^2+3^2$)⟩ = ⟨3, (9,10), (29 ,38)⟩

Clustering Features

Clustering features(CF) are organised in a ***CF tree***

- Entries for each child : [**CF_i, $Child_i$**]
- **CF = (N, LS, S**

N: Number of data points

LS: linear sum of N points= $\sum_{i=1}^{N} X_i$

SS: square sum of N points= $\sum_{i=1}^{N} X_i^{2}$

CF entry is a **summary** of statistics of the cluster

A **representation** of the cluster

A CF entry has **sufficient information** to calculate the centroid, radius, diameter and many other distance measures

Additively theorem allows us to **merge sub-clusters incrementally**

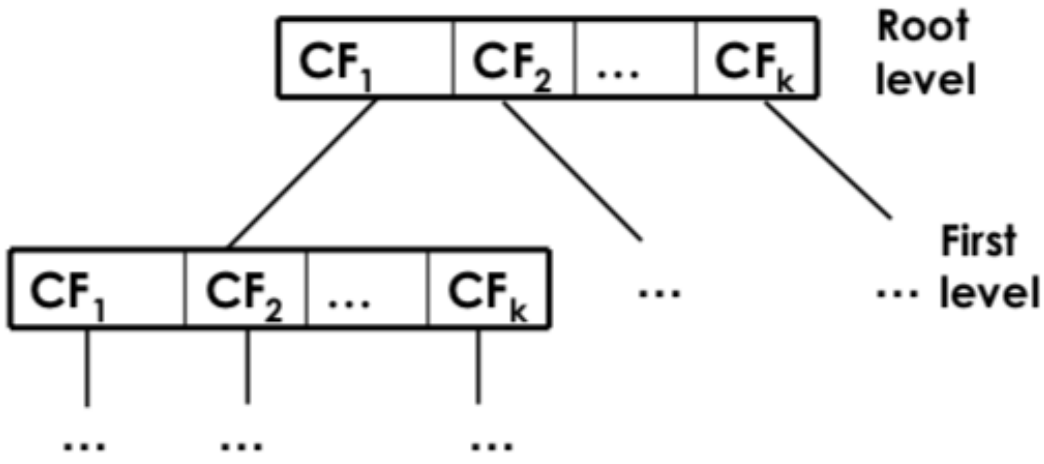
.

CF Tree

B = Branching Factor, maximum children in a non-leaf node

T = Threshold for diameter or radius of the cluster in a leaf

L = number of entries in a leaf



- CF entry in parent = sum of CF entries of a child of that entry
- In-memory, height-balanced tree
-

CF Tree Insertion

- Start with the root
- Find the CF entry in the root closest to the data point, move to that child and repeat the process until a closest leaf entry is found.
- At the leaf
    - If the point can be accommodated in the cluster, update the entry
    - If this addition violates the threshold T, split the entry,
        - if this violates the limit imposed by L, split the leaf.
            - If its parent node is full, split that and so on
- Update the CF entries from the leaf to the root to accommodate this point