Discussion Section 3 (CS145)

2015-10-16

Week 03

Outline

- Homework 2 will be posted on Monday (Oct 19)
  - Due Oct 26 at noon (beginning of the class)
- Review:
  - Clustering Algorithms
    - Hierarchical Clustering
    - B+Tree & BIRCH
    - DBSCAN

Why clustering

- Discover some hidden interesting patterns.
  - Example:
    - (1) abbreviation: UCLA, University of California Los Angeles
    - (2) cognates (cross-language): Vienna theater, Vienna theatre
    - (3) similar expression: Florida fine cars, Florida fine auto
    - (4) Arabic numerals vs English words: 24 hours, twenty-four hours

What is a good clustering

- (1) Keep similar objects together and dissimilar objects apart.
- (2) In other words, high ***intra-class*** similarity and low ***inter-class*** similarity.

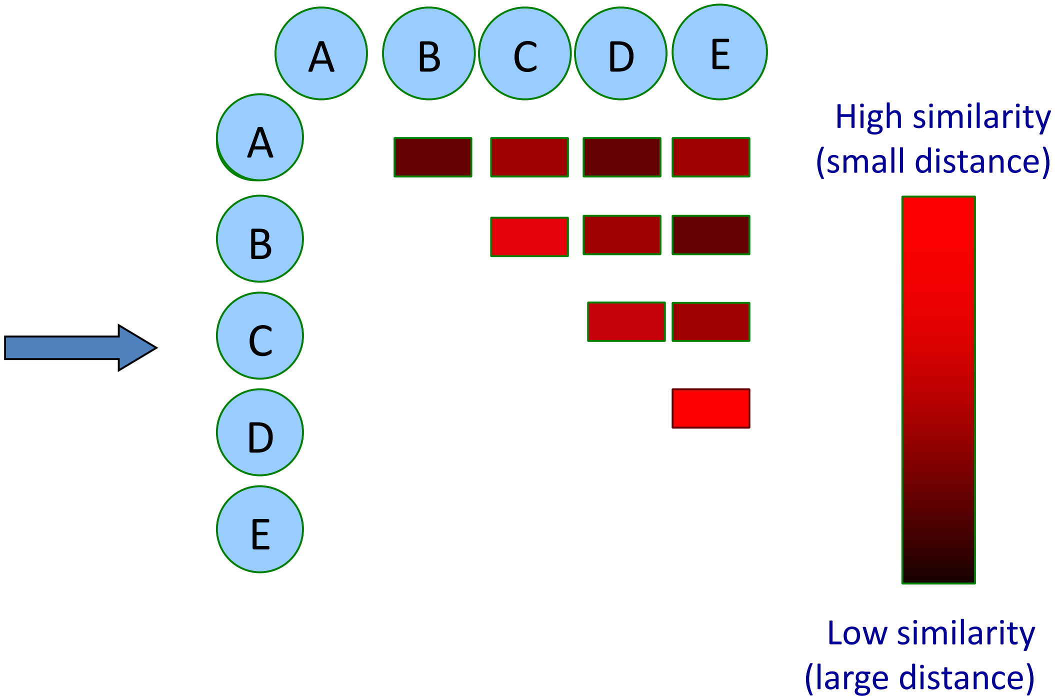
Hierarchical Clustering

- Goal: Group data objects into a tree of clusters
- Approaches:
  - Agglomerative
    - A “bottom up” approach
    - Each object starts as its own cluster
    - A pair of clusters are merged at each iteration until all objects form a single big cluster
  - Divisive
    - A “top down” approach
    - All objects start as one big cluster
    - Clusters are split at each iteration until each cluster contains only one object

Hierarchical Clustering

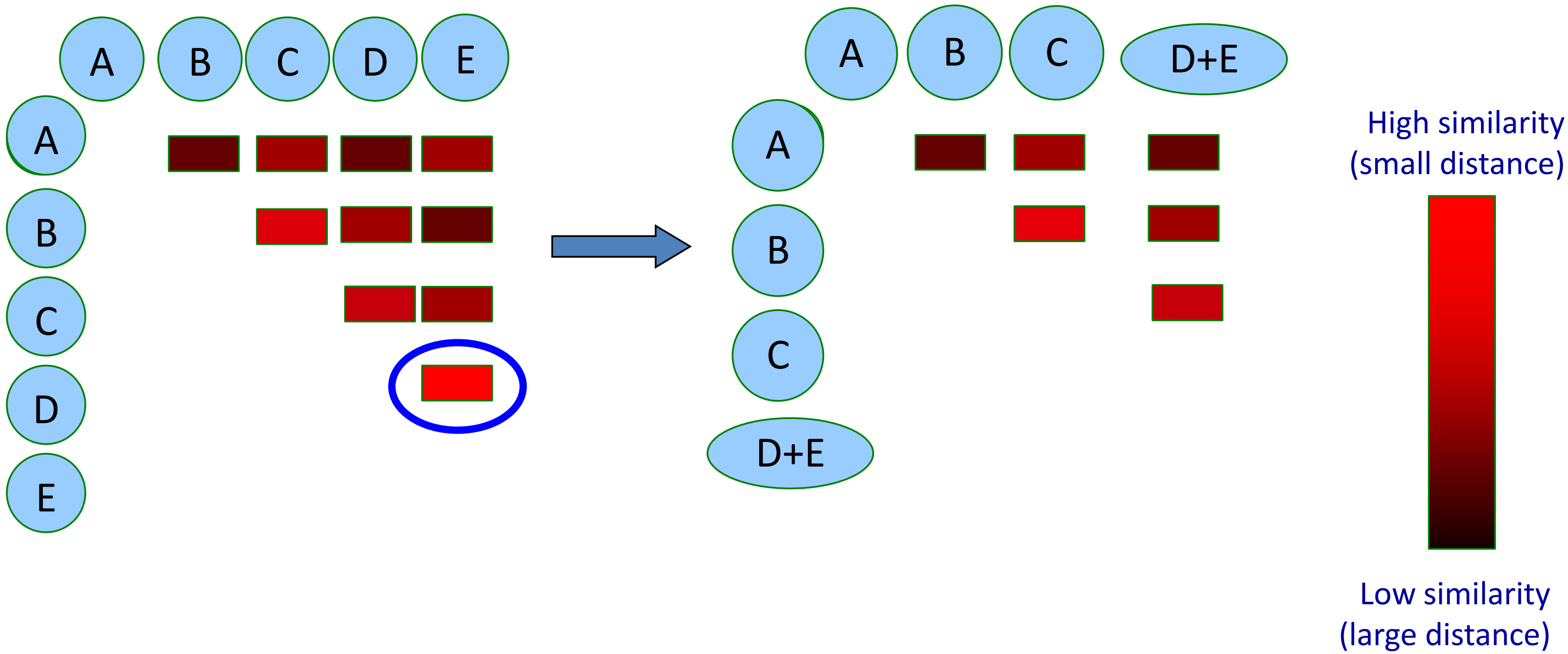
- Agglomerative Approach
  - Step 1: Calculate the distance matrix
  - Step 2: Join two members with the closest distance and recalculate the distance matrix
  - Step 3: repeat step 2 until all members form a single cluster

	Feature X	Feature Y	Feature Z
A	X_A	Y_A	Z_A
B	X_B	Y_B	Z_B
C	X_C	Y_C	Z_C
D	X_D	Y_D	Z_D
E	X_E	Y_E	Z_E



Hierarchical Clustering

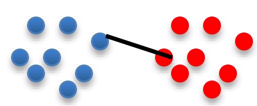
- Agglomerative Approach
  - Step 1: Calculate the distance matrix
  - Step 2: Join two members with the closest distance and recalculate the distance matrix
  - Step 3: repeat step 2 until all members form a single cluster



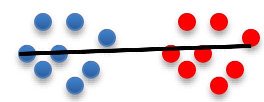
Hierarchical Clustering

- Agglomerative Approach
  - Step 1: Calculate the distance matrix
  - Step 2: Join two members with the closest distance and recalculate the distance matrix
  - Step 3: repeat step 2 until all members form a single cluster

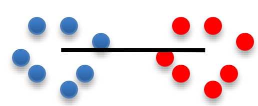
<u>Distance Measure Between Clusters</u>



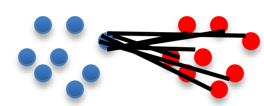
1. Minimum Distance (Single Linkage)



2. Maximum Distance (Complete Linkage)



3. Mean Distance

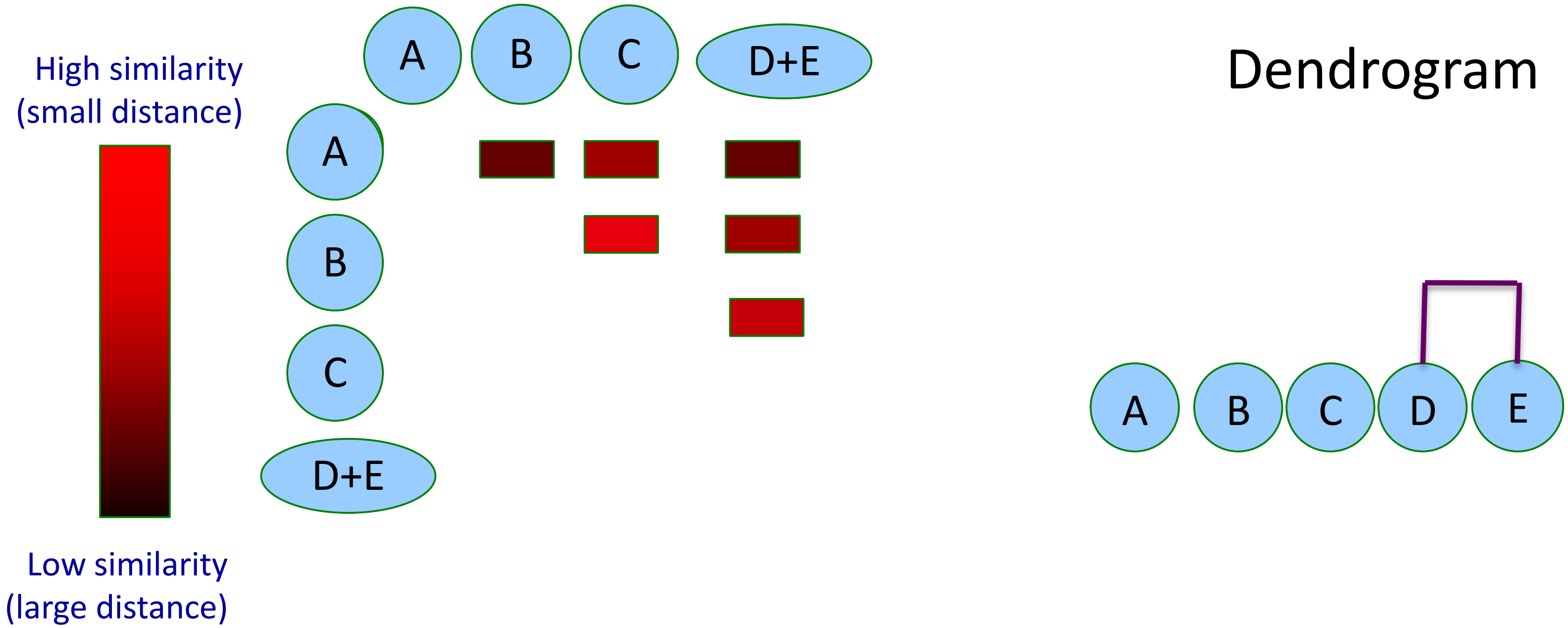


4. Average Distance

(average of all pairs)

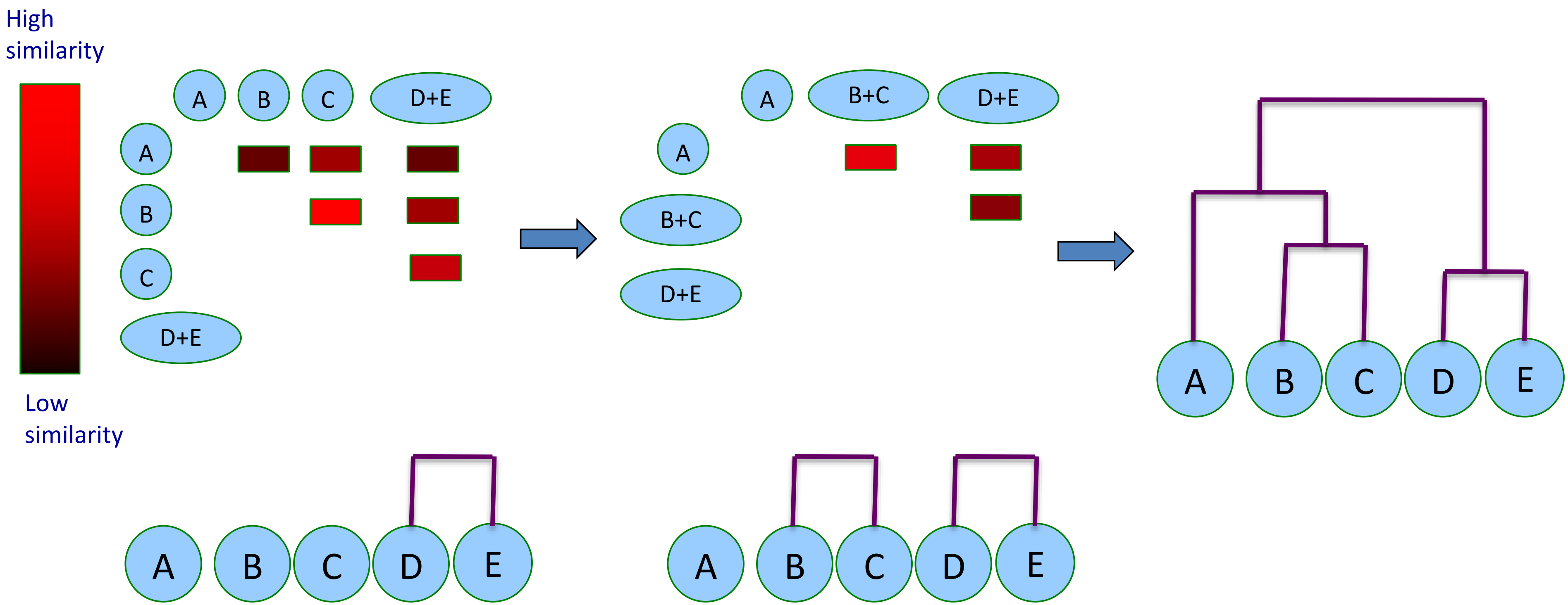
Hierarchical Clustering

- Agglomerative Approach
  - Step 1: Calculate the distance matrix
  - Step 2: Join two members with the closest distance and recalculate the distance matrix
  - Step 3: repeat step 2 until all members form a single cluster



Hierarchical Clustering

- Agglomerative Approach
  - Step 1: Calculate the distance matrix
  - Step 2: Join two members with the closest distance and recalculate the distance matrix
  - Step 3: repeat step 2 until all members form a single cluster



Hierarchical Clustering

- Challenges
  - Hard to choose merge/split points
  - Never undo merging/splitting
  - Do not scale well
  - Data may not fit in memory

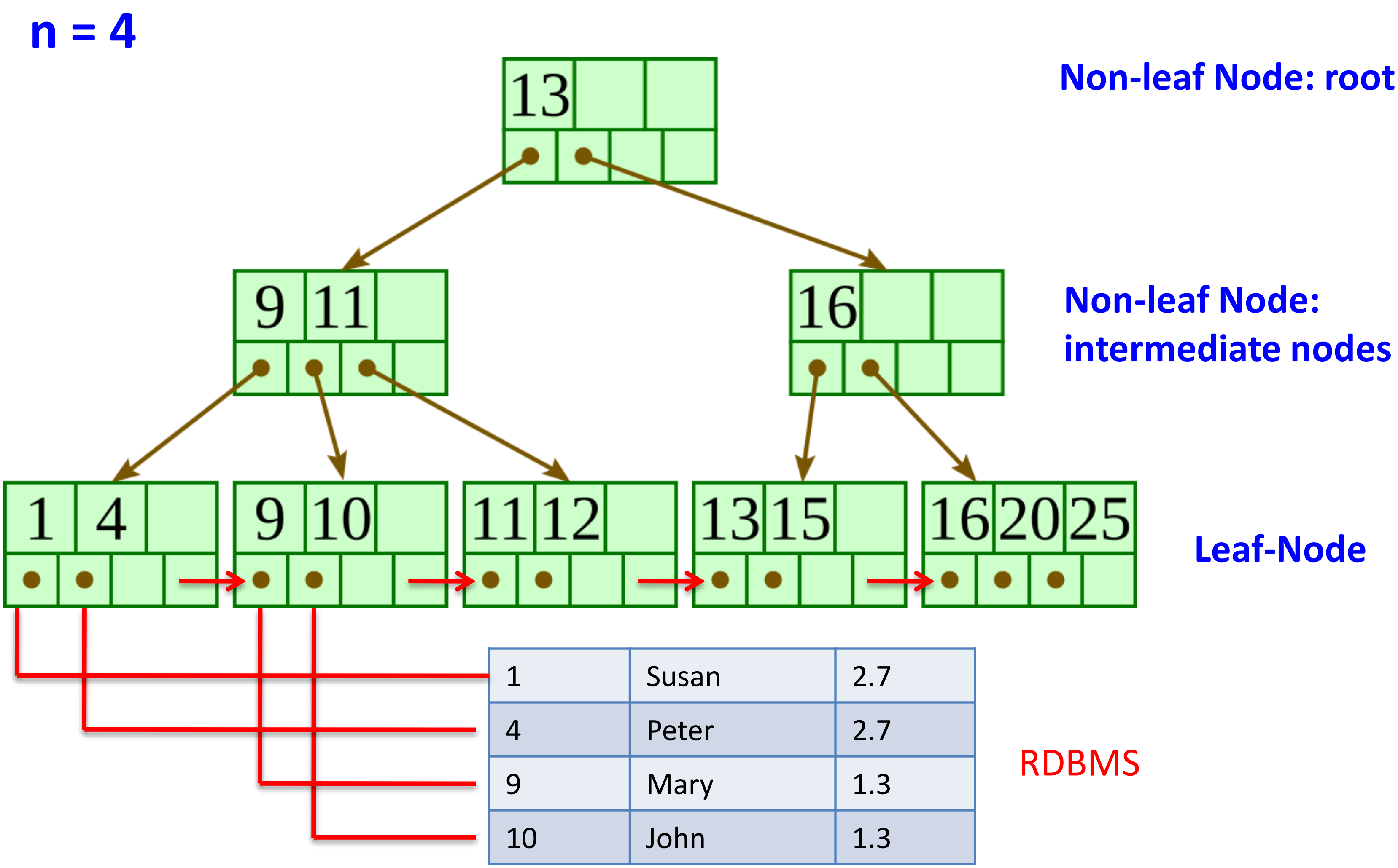
BIRCH

- **B**alanced **I**terative **R**educing and **C**lustering Using **H**ierarchies
- Why BIRCH?
  - Overcome the bottleneck of datasets not being able to fit in main memory
- Organize the clustering features in CF tree, which is structurally similar to B+ tree

B+ Tree

- A popular index structure in Relational Database Management System
- Advantage
  - Suitable for dynamic updates
  - Balanced
  - Minimum space usage guarantee (50%)

B+ Tree

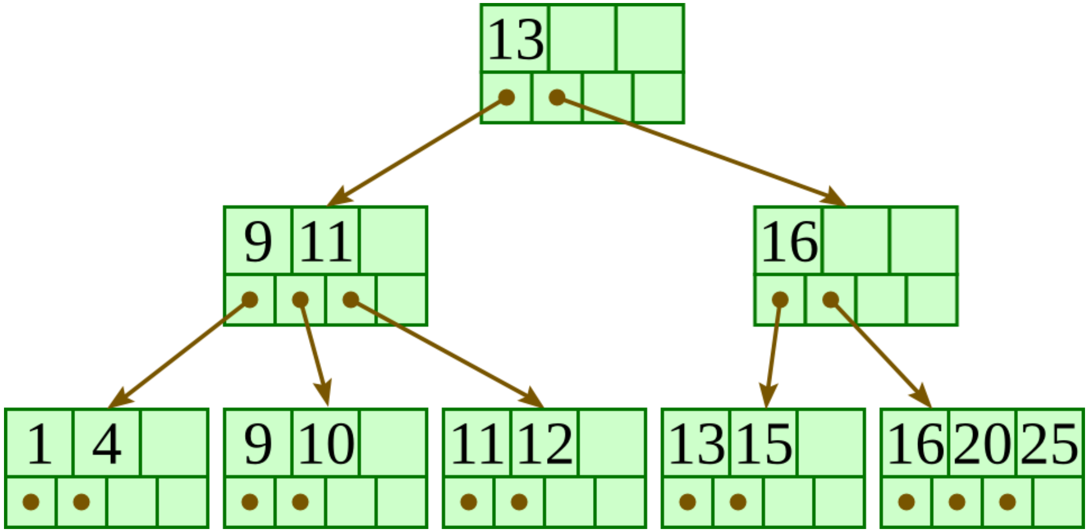


1	Susan	2.7
4	Peter	2.7
9	Mary	1.3
10	John	1.3

http://www.cburch.com/cs/340/reading/btree/index.html

B+ Tree

n = 4



Non-leaf Node: root

Non-leaf Node: intermediate nodes

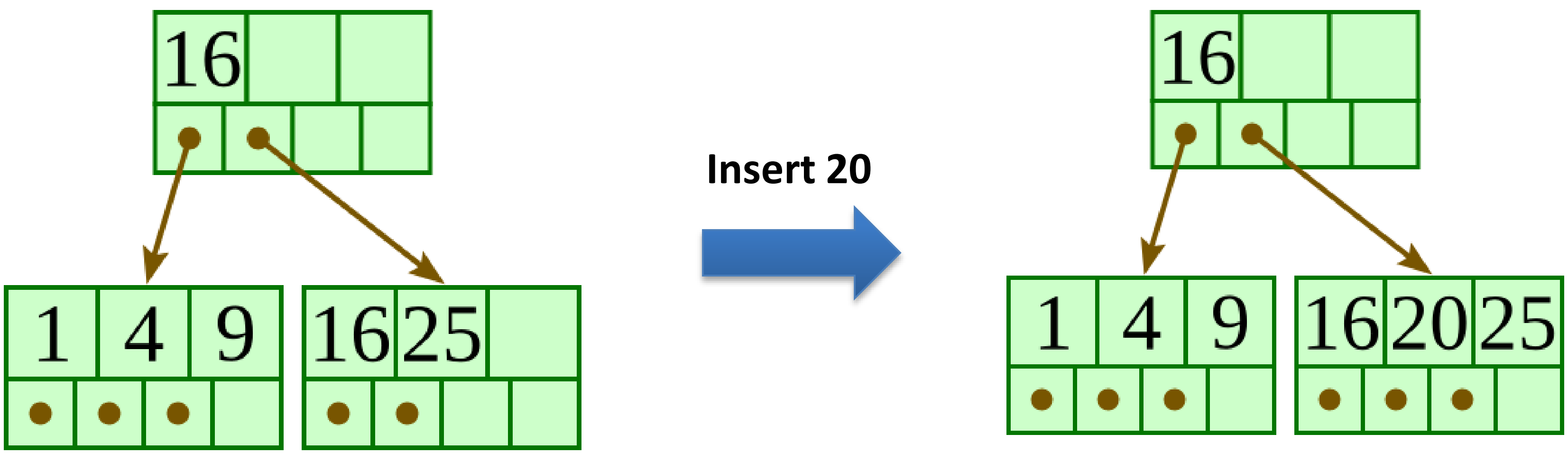
Leaf-Node

- B+ Tree is Balanced
  - The depth is the same for each path from the root to a leaf
  - Every node except the root must be at least half full
    - In this case, each intermediate node and leaf node must have 2 pointers

http://www.cburch.com/cs/340/reading/btree/index.html

B+ Tree: Insertion

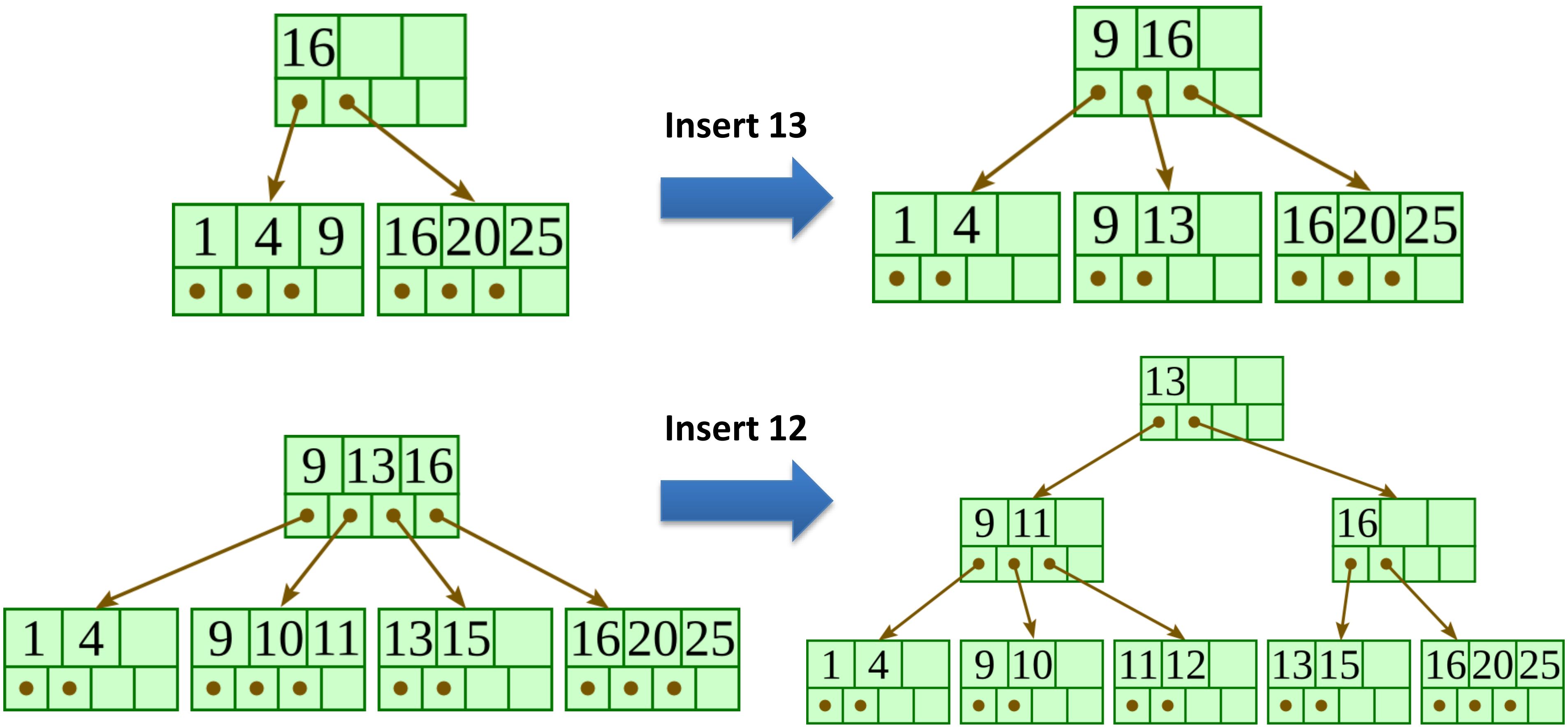
- Case 1: the node has empty space



http://www.cburch.com/cs/340/reading/btree/index.html

B+ Tree: Insertion

- Case 2: the node is full



http://www.cburch.com/cs/340/reading/btree/index.html

B+ Tree

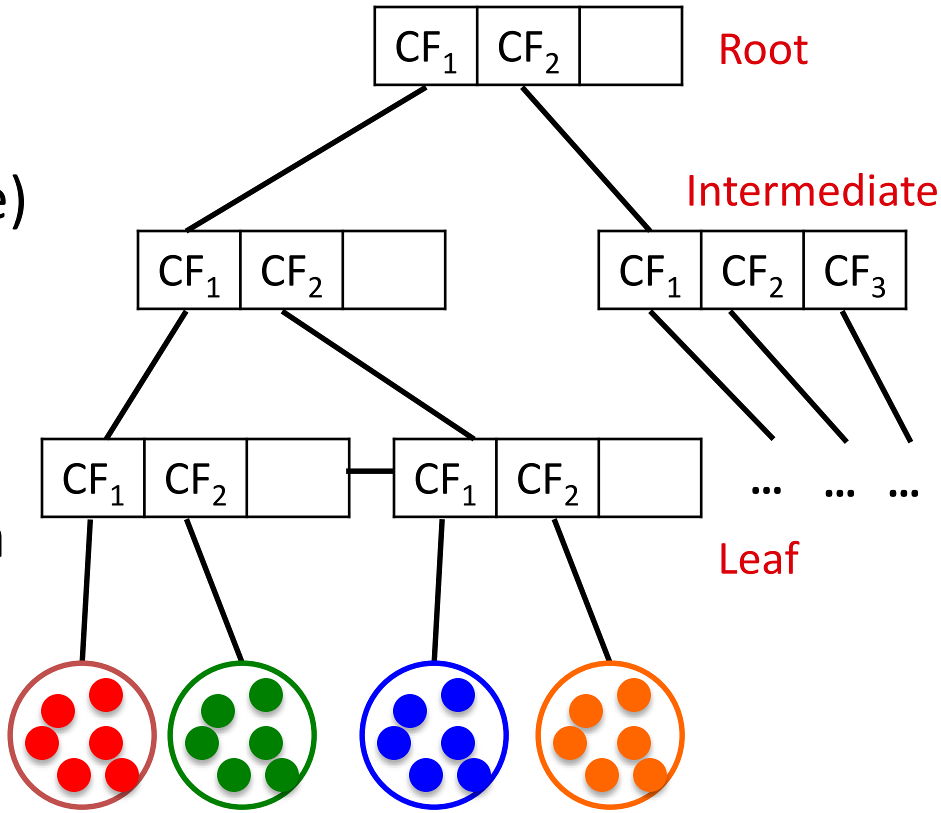
- More about B+ Tree
  - Visualization Toolhttps://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

BIRCH

- Clustering Features: A summary of statistics of the cluster
  - 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 $\sum_{i=1}^{N} X_i^2$
  - A CF entry has sufficient information to calculate the centroid, radius, diameter, and other distance measures

BIRCH

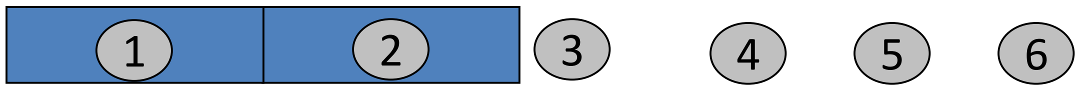
- CF Tree
  - Parameters
    - **B** = branching factor (max children in a non-leaf node)
    - **L** = number of entries in leaf node
    - **T** = threshold for diameter or radius of the cluster in a leaf
  - CF entry in parent = sum of CF entries of a child of that entry



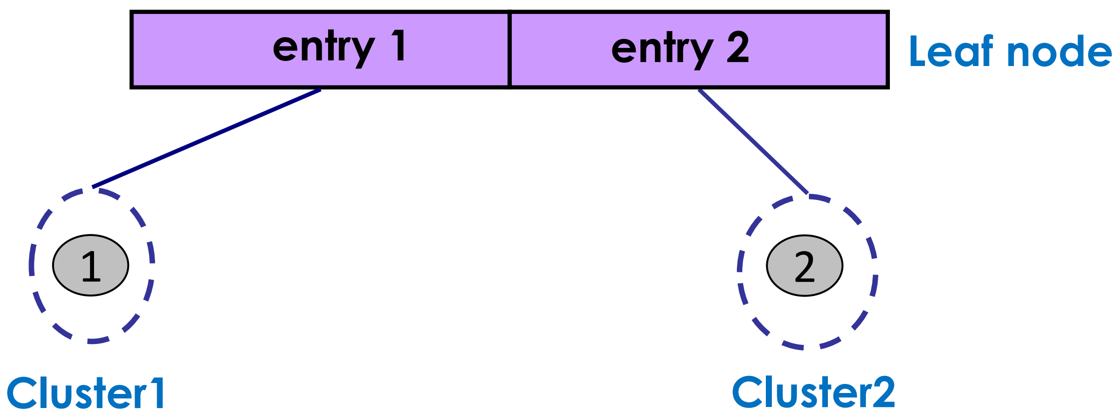
BIRCH

- Building the CF tree

Data Objects



Branching factor = 2
Number of entries in leaf node = 3
Cluster tightness threshold = T

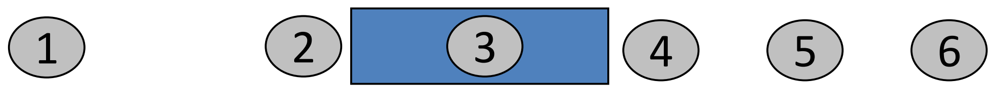


Leaf node with two entries

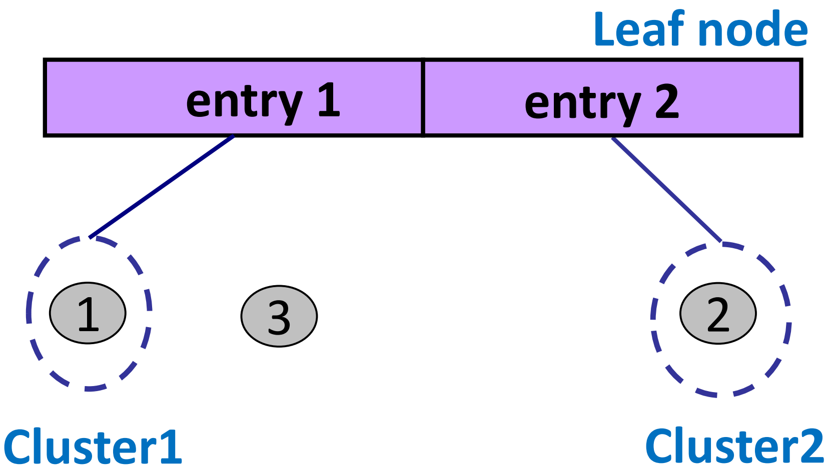
BIRCH

- Building the CF tree

Data Objects



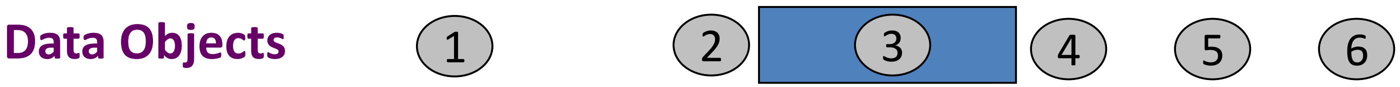
Branching factor = 2
Number of entries in leaf node = 3
Cluster tightness threshold = T



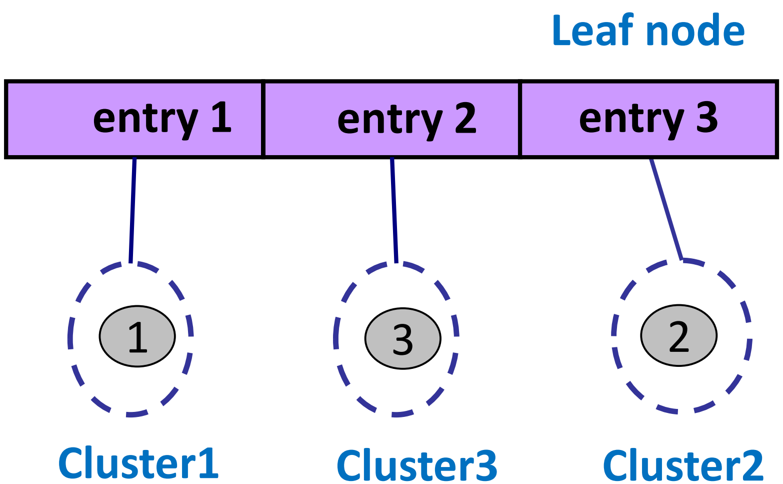
- Object 3 is closer to entry 1
- However, adding object 3 exceed Cluster 1 threshold

BIRCH

- Building the CF tree



Branching factor = 2
Number of entries in leaf node = 3
Cluster Tightness Threshold = T

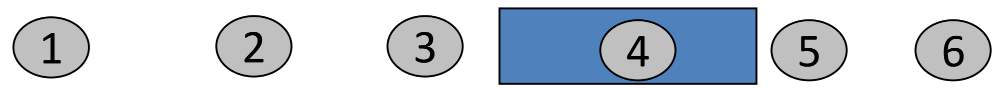


Leaf node with three entries

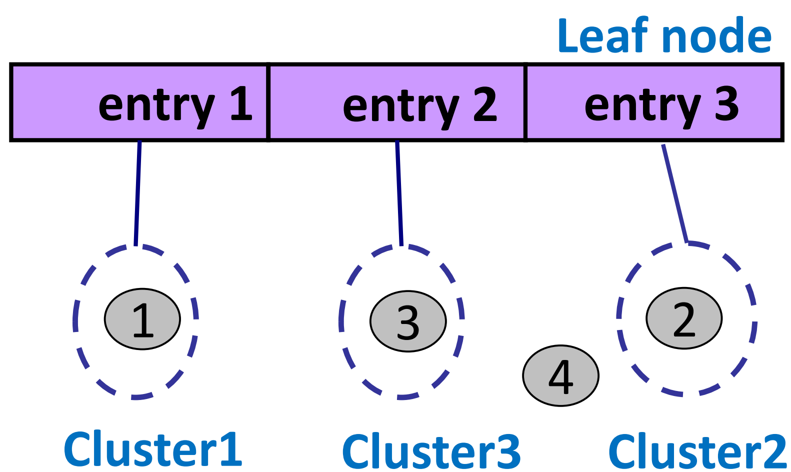
BIRCH

- Building the CF tree

Data Objects



Branching factor = 2
Number of entries in leaf node = 3
Cluster tightness threshold = T

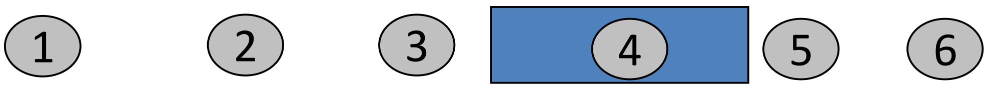


- Object 4 is closer to entry 3
- Cluster 2 remains compact after adding object 4

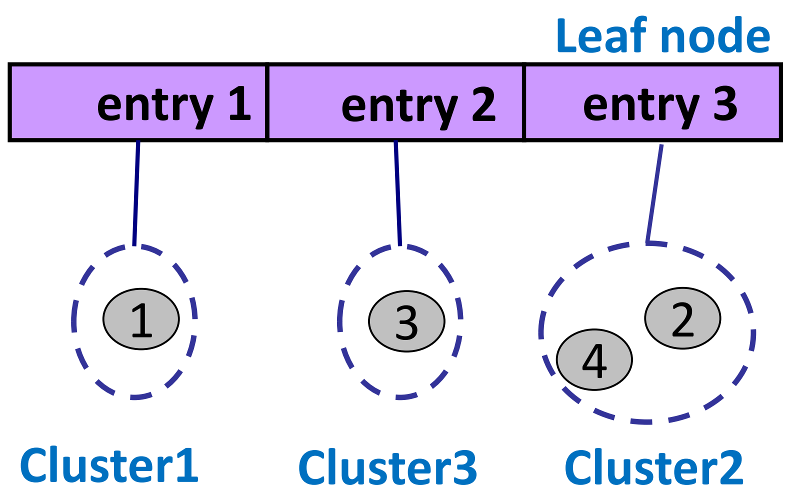
BIRCH

- Building the CF tree

Data Objects



Branching factor = 2
Number of entries in leaf node = 3
Cluster Tightness Threshold = T

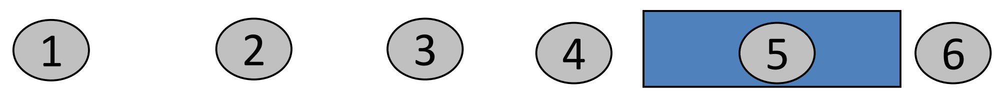


Add to cluster 2; Update CF entry

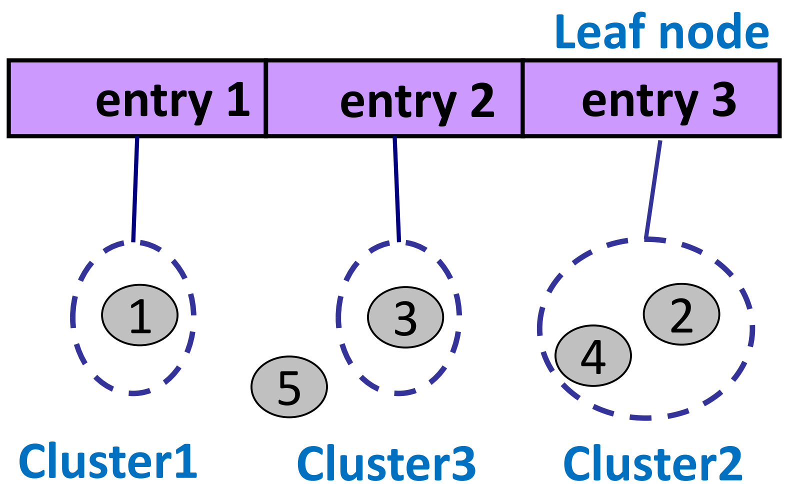
BIRCH

- Building the CF tree

Data Objects



Branching factor = 2
Number of entries in leaf node = 3
Cluster Tightness Threshold = T

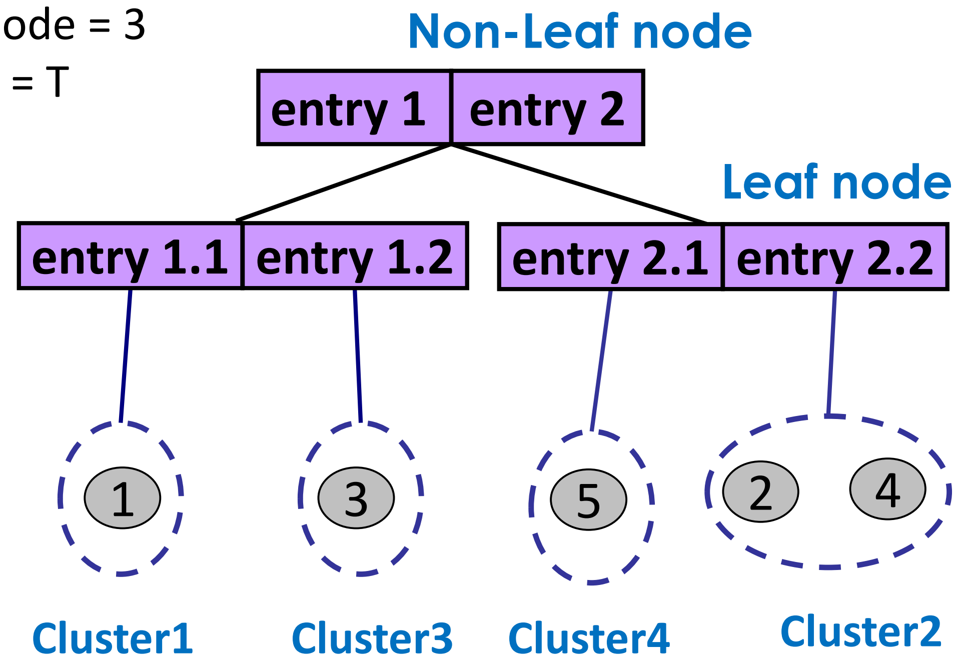


- Object 5 is closer to entry 2
- However, adding object 5 exceed Cluster 3 threshold
- Exceeds the limit for number of entries in leaf node

BIRCH

- Building the CF tree

Branching factor = 2
Number of entries in leaf node = 3
Cluster tightness threshold = T

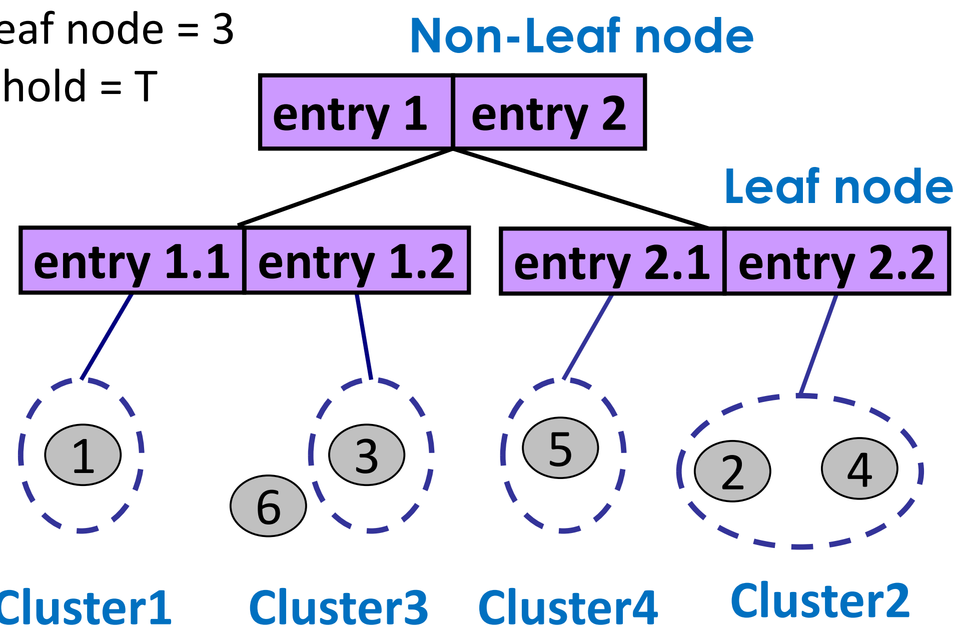


Split the leaf node; Propagate CF entries one level up

BIRCH

- Building the CF tree

Branching factor = 2
Number of entries in leaf node = 3
Cluster tightness threshold = T

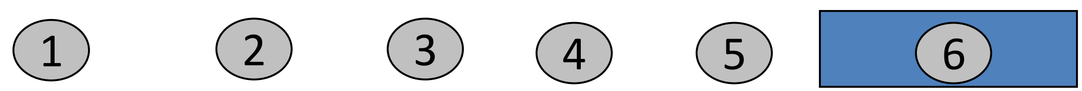


- Object 6 is closer to entry 1.2
- Cluster 3 remains compact

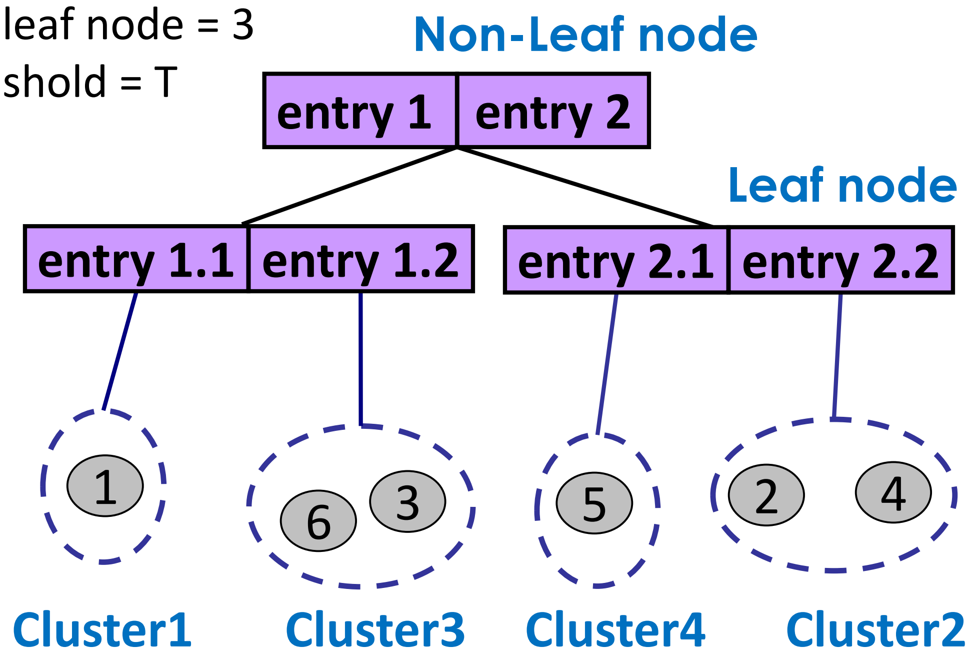
BIRCH

- Building the CF tree

Data Objects



Branching factor = 2
Number of entries in leaf node = 3
Cluster tightness threshold = T

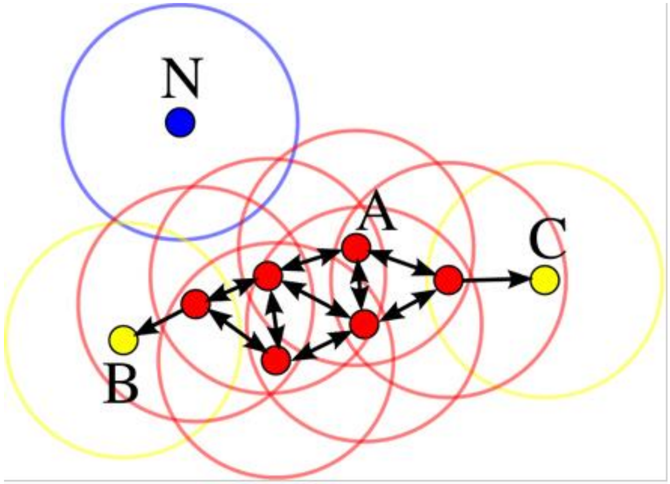


Add to cluster 3, Update CF entry

Density-based clustering

- Features:
  - Discover clusters of arbitrary shape.
  - Handle noise.
  - One scan.

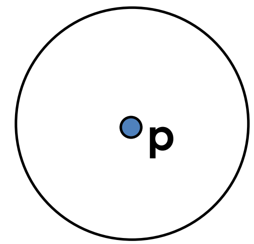
DBSCAN



- Inputs:
  - (1) Radius: 1 cm
  - (2) Minimum # of neighbors: 3
- Identify Three kinds of objects:
  - (1) Core object (red)
  - (2) Outlier (blue)
  - (3) Border object (yellow)

Basic Concepts: ε-neighborhood & core objects

- The neighborhood within a radius ε of a given object is called the **ε-neighborhood** of the object



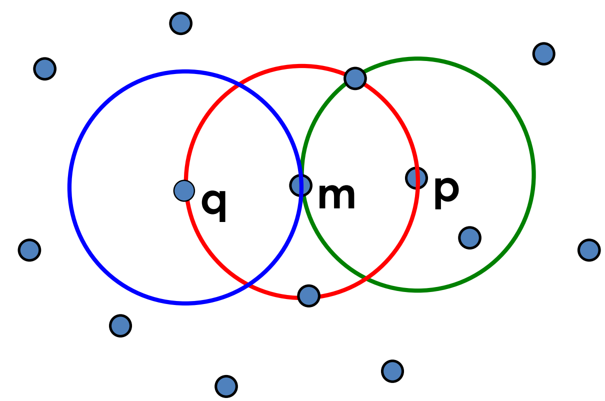
ε = 1 cm

- If the ε-neighborhood of an object contains at least a minimum number, **MinPts**, of objects then the object is called a **core object**
  - → **Example:** ε = 1 cm, MinPts=3

m and **p** are core objcets because

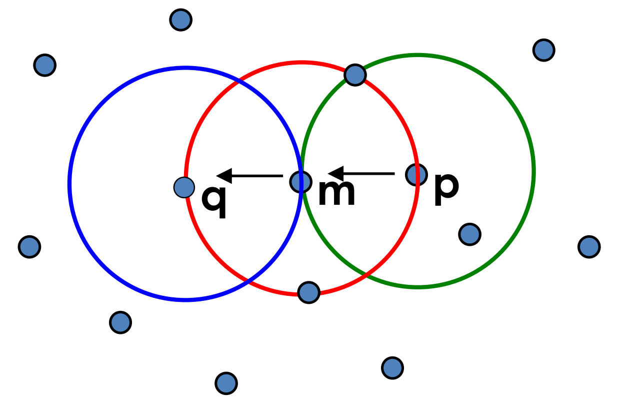
their ε-neighborhoods

contain at least 3 points



Directly density-Reachable Objects

- An object **p** is **directly density-reachable** from object **q** if **p** is within the ε-neighborhood of **q** and ***q*** is a core object

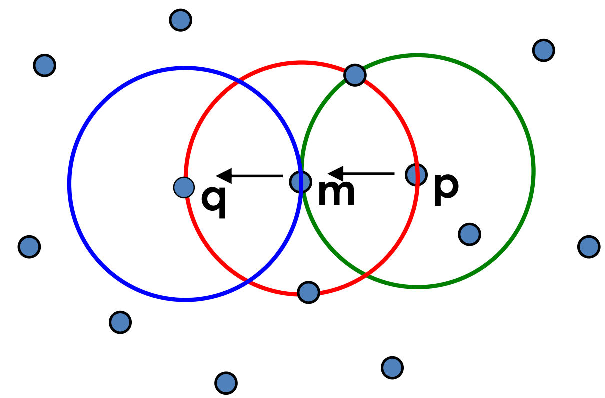


→ **Example:**

q is directly density-reachable from **m**
m is directly density-reachable from **p**
and vice versa

Density-Reachable Objects

- An object **p** is **density-reachable** from object **q** with respect to ε and **MinPts** if there is a chain of objects $\mathbf{p_1,\ldots p_n}$ where $\mathbf{p_1=q}$ and $\mathbf{p_n=p}$ such that $\mathbf{p_{i+1}}$ is directly reachable from $\mathbf{p_i}$ with respect to ε and MinPts



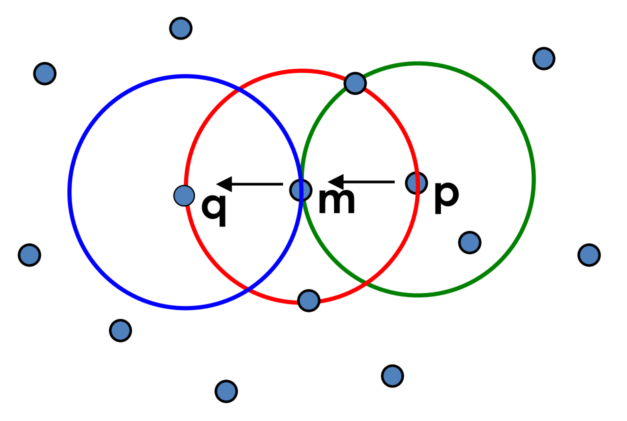
→ **Example:**

q is density-reachable from **p** because **q** is directly density-reachable from **m** and **m** is directly density-reachable from **p**

p is not density-reachable from **q** because **q** is not a core object

Density-Connectivity

- An object **p** is **density-connected** to object **q** with respect to ε and **MinPts** if there is an object **O** such as both **p** and **q** are density reachable from **O** with respect to ε and MinPts



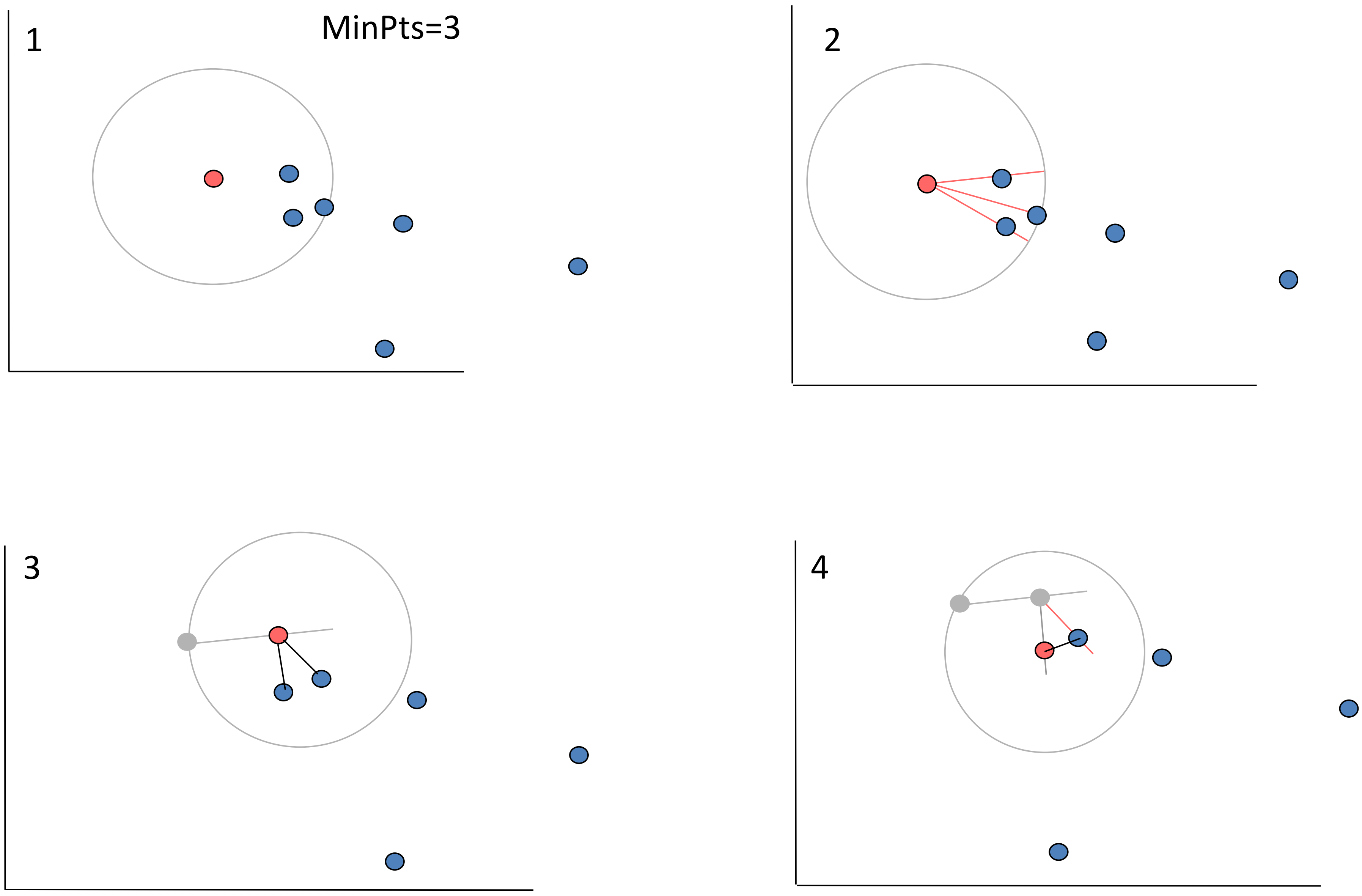
→ **Example:**

p,**q** and **m** are all density connected

DBSCAN algorithm

- (1) Arbitrary select a point p.
- (2) Retrieve all points density-reachable from p.
- (3) If p is a core point, a cluster is formed.
- (4) If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database.
- (5) Continue the process until all the points have been processed.

DBSCAN algorithm



DBSCAN algorithm