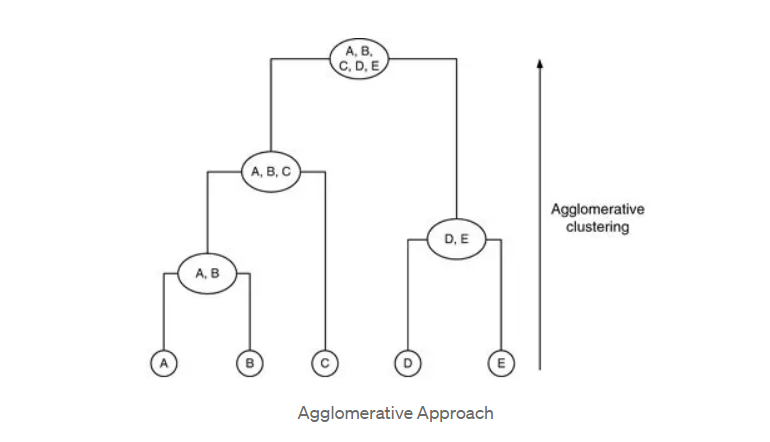
**Birch Algorithm.**

In Hierarchical clustering, as you might have already guessed, a hierarchy or a tree structure is created to represent the clustering process. A tree structure can be created either in top-down (root to leaf nodes) or in bottom-up (leaf nodes to root) fashion. So, accordingly, the Hierarchical clustering is either called Divisive (which follows top-down approach) or Agglomerative (which follows bottom-up approach) clustering.

In Agglomerative (or bottom-up) approach, we start with each object in its own cluster (leaf nodes) and we go on merging them into larger and larger clusters to form higher level nodes until all the objects are merged into a single node (root) or given termination criterion met.

****

The Divisive approach is difficult to implement, mainly because of two challenges — first, how to decide splitting criterion, and second, having a set of clusters at particular level in tree, which cluster to split. For both these challenges, the answer is select a cluster to split and split it such that error is minimized.

**BIRCH is a connectivity based algorithm, assumes that nearby objects (data points) are more related than far away objects. Birch is a more advanced algorithm, a better version of** [**Hierarchical Agglomerative Clustering (HAC)**](https://towardsdatascience.com/hac-hierarchical-agglomerative-clustering-is-it-better-than-k-means-4ff6f459e390)**.**

BIRCH stands for Balanced Iterative Reducing & Clustering using Hierarchy. It is multi-phase hierarchical clustering based on Clustering Features (CFs). It is designed for clustering large amount of numeric data by integrating hierarchical clustering in initial phase, called micro-clustering and other clustering methods in later phase, called macro-clustering.

**The Clustering Features (CFs)**

The most important characteristic of this algorithm is that it uses CF to summarize a cluster and CF tree to represent the clustering hierarchy. So, let’s first see what Clustering Feature (CF) is.

The Clustering Feature (CF) of a cluster is a 3-D vector summarizing information about clusters of objects. It is defines as,

CF = (n, LS, SS)

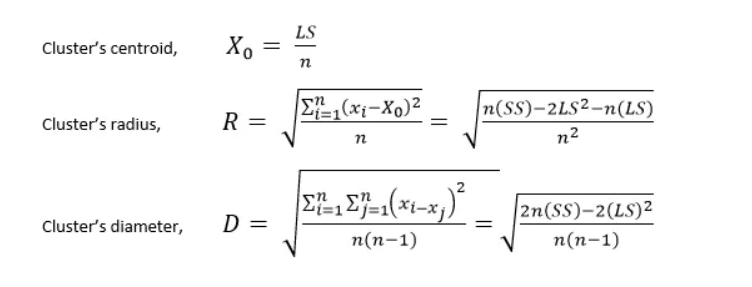
where n is the number of objects in the cluster, LS is the linear sum of the objects and SS is the squared sum of the objects.



For example, consider a cluster C1={9,12,10,8,11} then CF(C1)=(5,50,510) where n=5, LS=9+12+10+8+11=50 and SS=9²+12²+10²+8²+11²=510

Another example with 2-D objects, C2={(1,1),(2,1),(3,2)} then CF(C2)=(3,(6,4),(14,6)) where n=3,LL=(1+2+3,1+1+2)=(6,4) and SS=(1²+2²+3², 1²+1²+2²)=(14,6)

Using CF, we can derive many other useful statistics of a cluster as given below:



The algorithm improves space efficiency as it needs to store only CFs of the clusters instead of detailed information about the individual objects within the clusters.

Another important property of the CFs is that they are additive. That is, two disjoint clusters C1 and C2 with CFs CF1=(n1,LS2,SS1) and CF2=(n2,LS2,SS2) respectively, the CF of the cluster formed by merging C1 and C2 is given as, CF1+CF2=(n1+n2,LS1+LS2,SS1+SS2)

For example, C1={(2,5),(3,2),(4,3)} and C2={(1,1),(2,1),(3,1)} then

CF1=(3,(2+3+4,5+2+3),(2²+3²+4²,5²+2²+3²))=(3,(9,10),(29,38)) and CF2=(3,(1+2+3,1+1+1),(1²+2²+3²,1²+1²+1²))=(3,(6,3), (14,3)) now, if C3=C1UC2 then CF3=CF1+CF2=(6,(15,13),(43,41))

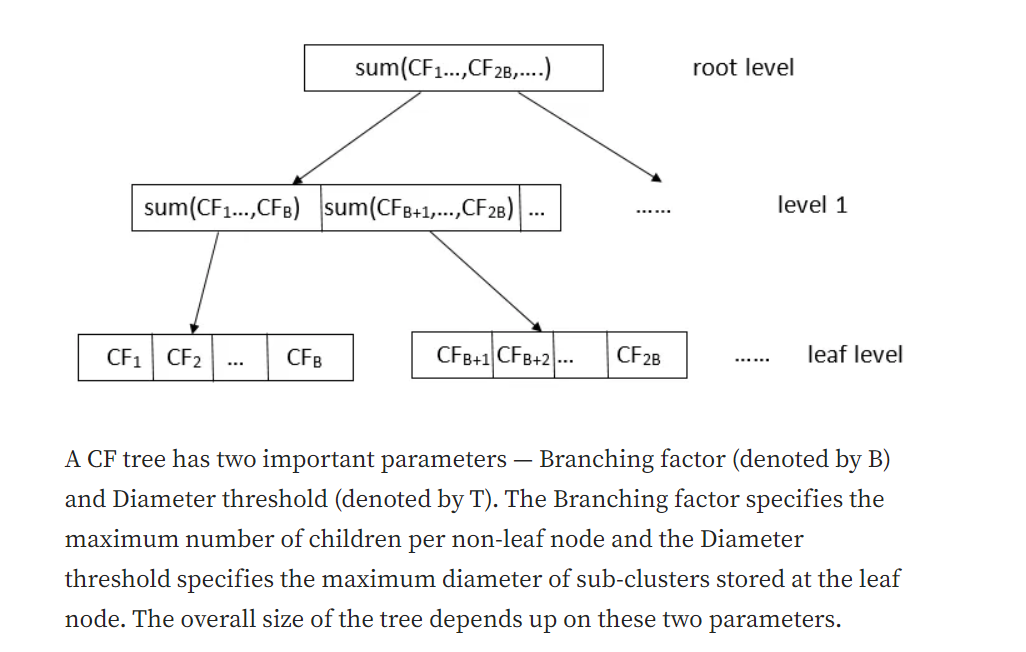
\*Remember: CF = (n, LS, SS)

where n is the number of objects in the cluster, LS is the linear sum of the objects and SS is the squared sum of the objects.

**The CF tree**

A CF tree is used to store CFs for hierarchical clustering. Let’s see how it is created and maintained.

Each leaf node of CF tree stores CFs for a fixed number of clusters. Each non-leaf node stores sum of CFs of its child nodes.



**The Phases**

There are two primary phases of the algorithm:

Phase 1 — The algorithm scans the objects and constructs an initial in-memory CF tree, which can be viewed as multilevel compression of the data that tries to preserve the inherent clustering structure of the data.

Phase 2 — The algorithm uses a selected clustering method to cluster the leaf nodes of the CF tree.

During Phase 1, objects are dynamically inserted to build the CF tree. An object is inserted into the closest leaf entry. If the diameter of the sub-cluster stored at the leaf node, after insertion, is greater than the specified threshold T then the leaf node is split. After inserting the current object successfully, the information about the inserted object is passed towards the root of the tree. If the memory required to store the CF tree is larger than the available memory then the diameter threshold T is updated (with larger value) and the tree is rebuilt.

Once the CF tree is built, any cluster method, such as partitioning method, can be used with CF tree in Phase 2, to obtain the final clusters.

**The most important advantage of this algorithm is that its time complexity is only O(n), where n is the number of objects.**

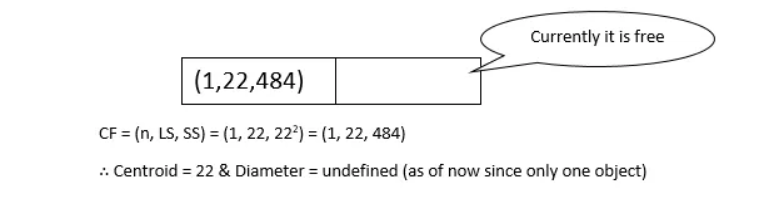
**Its disadvantage is that it works well only for spherical shape, like k-means, clusters and numeric attributes.**

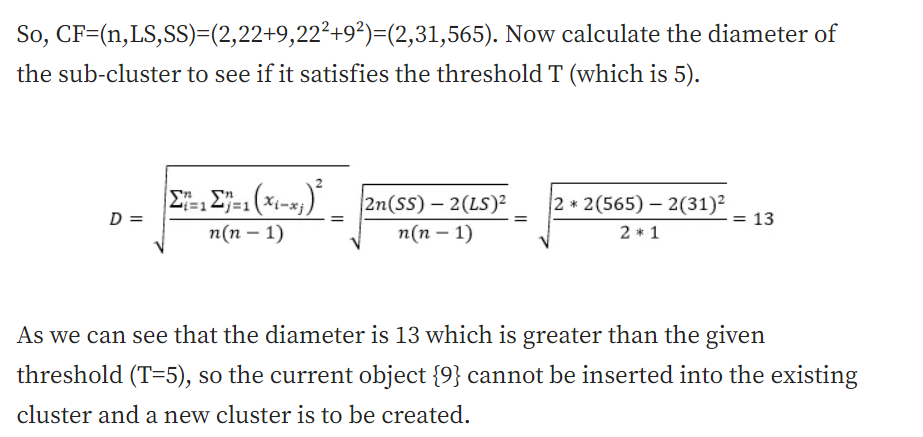
Example:

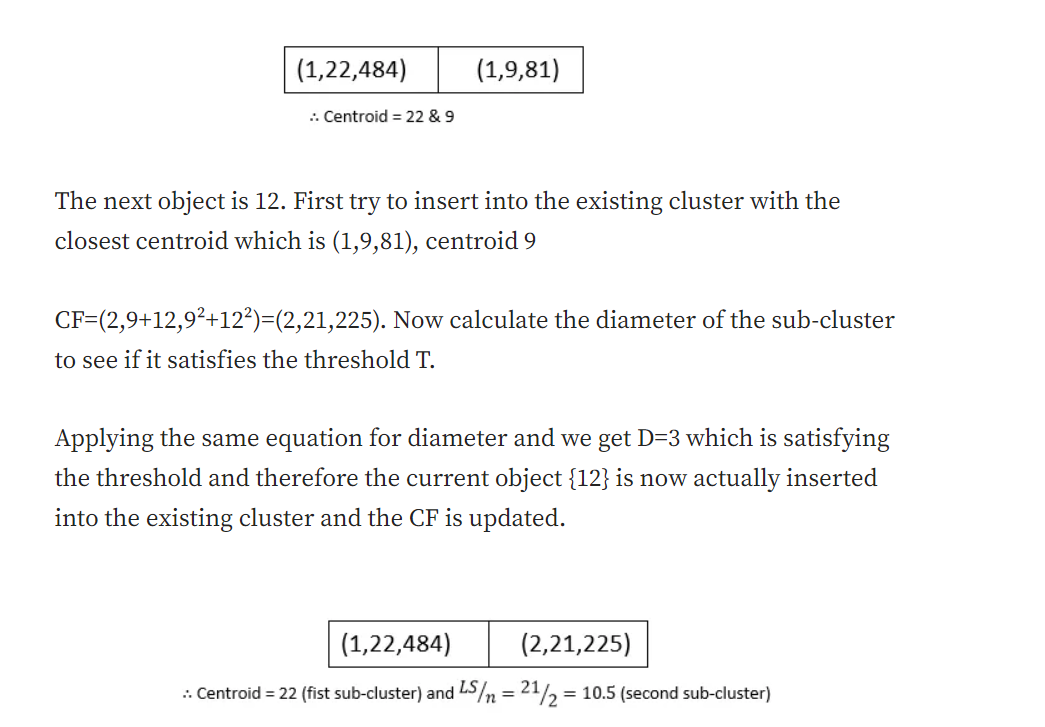
So, consider a set of 1-D objects {22,9,12,15,18,27,11,36,10,3,14,32}. Let the Branching factor B=2 and the Diameter threshold T=5.

Read the objects in the given order, and keep inserting them into the tree in terms of CFs.

The first object is 22. It is directly inserted into a new node:

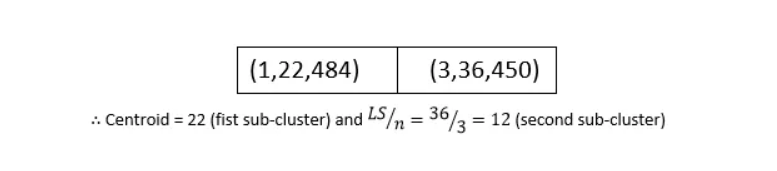






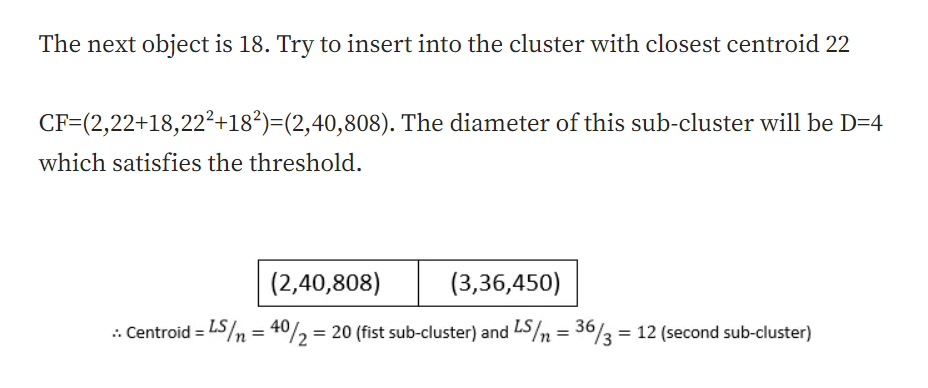
The next object is 15. First try to insert into the existing cluster with the closest centroid which is (2,21,225), centroid 10.5

CF=(3,21+15,225+15²)=(3,36,450). The diameter of this sub-cluster will be, using the same equation, D=4.24 which is satisfying the threshold. So, {15} is inserted into this existing cluster and the CF is updated.



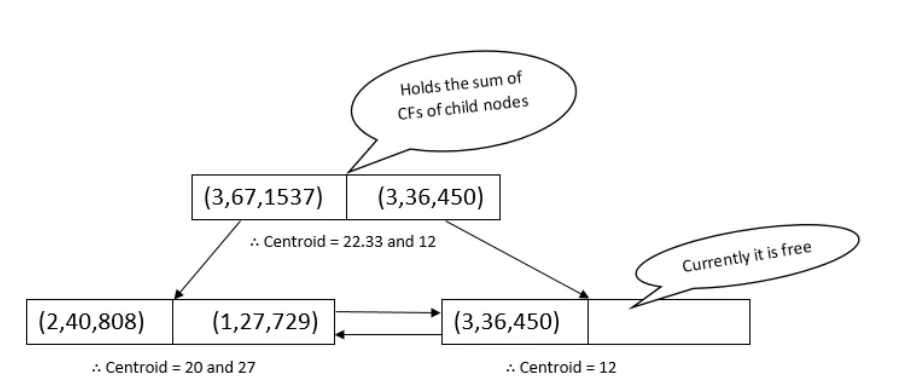
The next object is 18. Try to insert into the cluster with closest centroid 22

CF=(2,22+18,22²+18²)=(2,40,808). The diameter of this sub-cluster will be D=4 which satisfies the threshold.

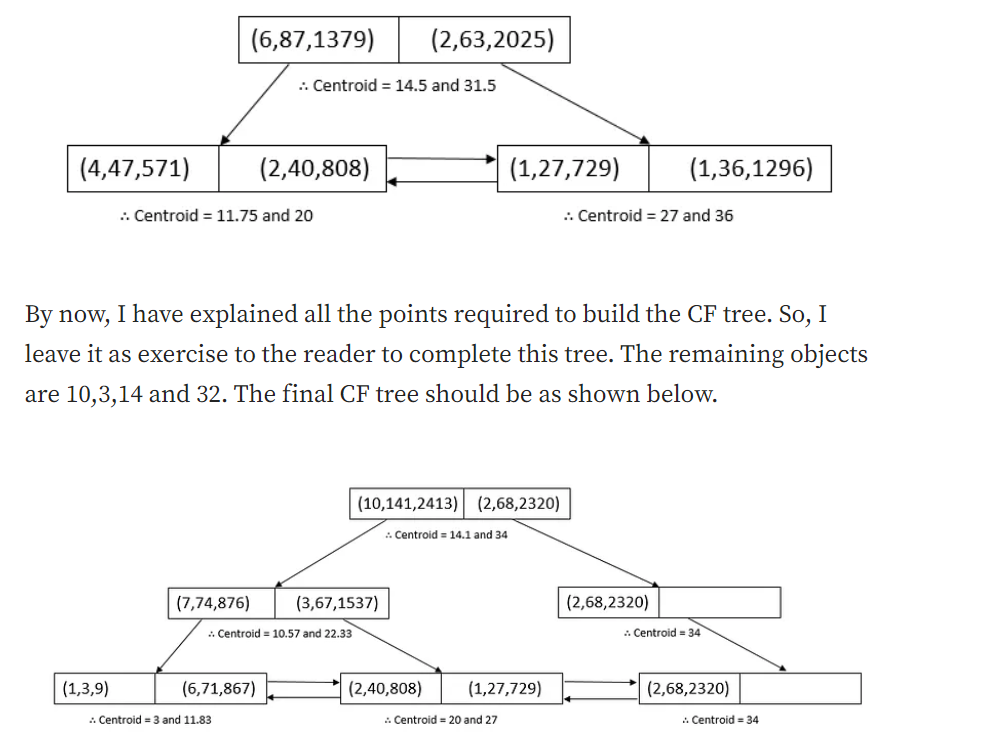


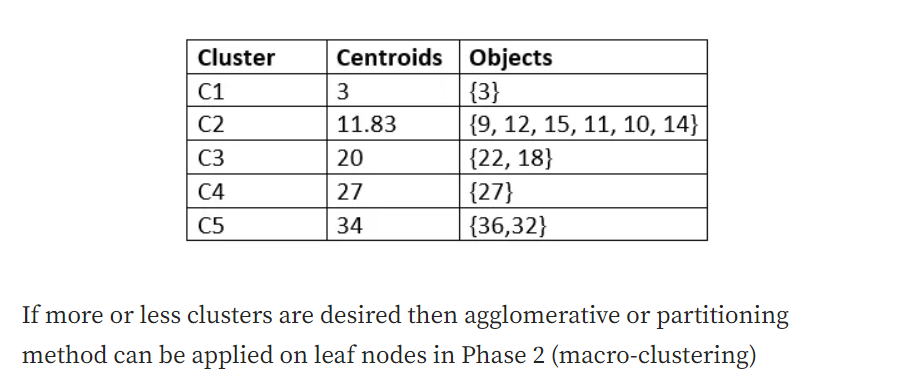
The next object is 27 and the closest centroid is 20.

CF=(3,40+27,808+27²)=(3,67,1537) & we get D=6.37 which doesn’t satisfy the threshold. Therefore a new cluster to be created, but there is no space in the leaf node as it can hold only two CFs (recall that the Branching factor B=2). So, the leaf node is split and a new level is added to the CF tree.



The next object is 36. The closest centroid in the root is 22.33, follow the branch and the closest centroid in leaf is 27. Up on calculating diameter, we find D=9. It is not satisfying the threshold, so there is a need to create new cluster, but there is no space in the leaf node, so there is a need to split it. But we can see that there is a space in another node in the leaf level. Therefore, to keep the tree height balanced, the CFs at the leaf level are redistributed, the current object is inserted in a new cluster and the corresponding CFs at root level are updated as follows,





**Parameters:**

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html>

**threshold*float, default=0.5***

The radius of the subcluster obtained by merging a new sample and the closest subcluster should be lesser than the threshold. Otherwise a new subcluster is started. Setting this value to be very low promotes splitting and vice-versa.

**branching\_factor*int, default=50***

Maximum number of CF subclusters in each node. If a new samples enters such that the number of subclusters exceed the branching\_factor then that node is split into two nodes with the subclusters redistributed in each. The parent subcluster of that node is removed and two new subclusters are added as parents of the 2 split nodes.

**n\_clusters*int, instance of sklearn.cluster model or None, default=3***

Number of clusters after the final clustering step, which treats the subclusters from the leaves as new samples.

* None : the final clustering step is not performed and the subclusters are returned as they are.
* [**sklearn.cluster**](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster) Estimator : If a model is provided, the model is fit treating the subclusters as new samples and the initial data is mapped to the label of the closest subcluster.
* int : the model fit is **[AgglomerativeClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html" \l "sklearn.cluster.AgglomerativeClustering" \o "sklearn.cluster.AgglomerativeClustering)** with n\_clusters set to be equal to the int.

**compute\_labels*bool, default=True***

Whether or not to compute labels for each fit.

**copy*bool, default=True***

Whether or not to make a copy of the given data. If set to False, the initial data will be overwritten.