→ "HIERARCHICAL CLUSTERING":

Another type of clustering algorithms is 'Hierarchical Clustering"
They are of two types :→

1> "Agglomerative Hierarchical Clustering"
2> "Divisive Hierarchical Clustering"

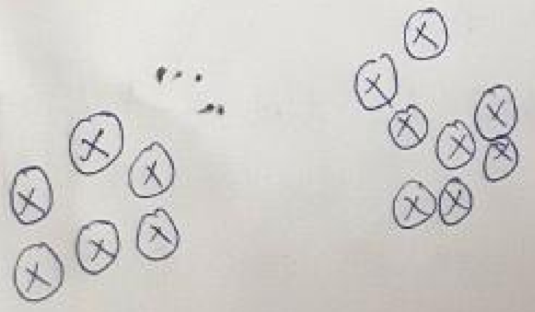
Typically Agglomerative is more popular.

Let's understand hierarchical clustering with the help of an example :→

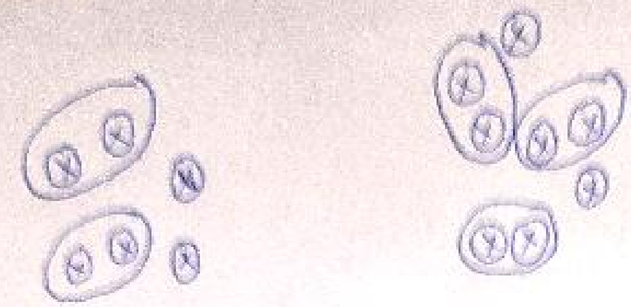
Let's assume we have a bunch of points as shown below:-

Now, if we want to cluster these points, the way agglomerative clustering works is as follows :-

It initially assumes that, each point is a cluster on to itself.

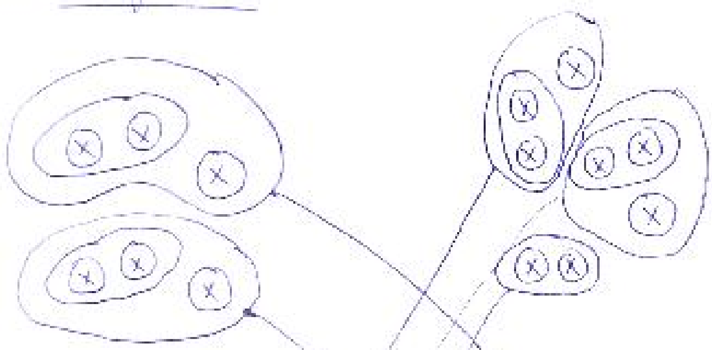


Then, it takes two clusters & combines them into one as shown.



So, initially we have "14 points" & each point became a cluster. And after grouping we have "9 clusters" after the end of first stage, of which some clusters have "two points" & some have only 'one-point'

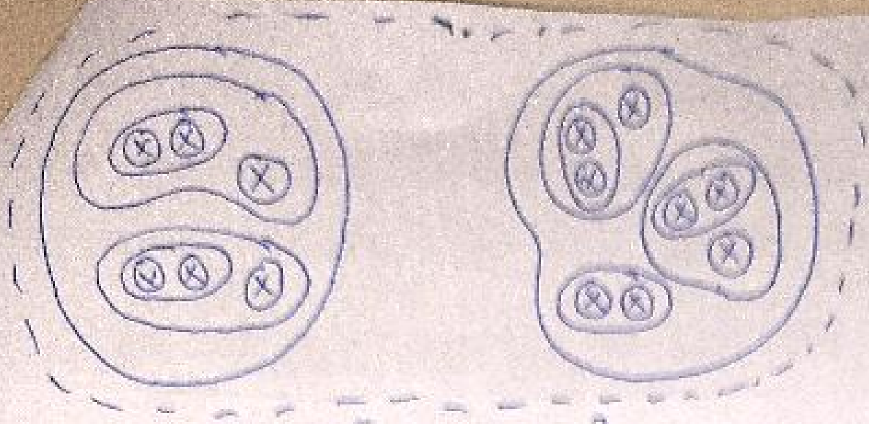
Now the algorithm keeps on continuing to form the clusters as shown further.



Now we form more clusters, so what happened is from ["9-clusters"] we have reached ["5-clusters"]

So, the algo. keeps on incrementally combining clusters which are close to each other.

Now in the next stage it will say that these two clusters are close to each other, let's combine them together. & these three clusters are close to each other, let's combine them together...



Now from ["5 cluster"] we have reached ["2 clusters"].

At the boundary case, since we have only two clusters, Now these two clusters will be grouped into one large cluster at the end.

"13-clusters" → "9-clusters" → "5-clusters" → "2-clusters" → "1-cluster"

So, in agglomerative clustering we are starting with each point being an individual cluster & we are grouping these clusters based on some sense of similarity or distance.

This is the core idea of ["Agglomerative clustering"]

Note :→ Here we are grouping together "points or clusters", to form "larger clusters".

In case of "Divisive", we will do exact opposite of 'Agglomerative' :→

Divisive starts by saying that everything is one big cluster. (It starts with one big cluster, which consists of all the points).

In the first iteration, divisive will group all the points into one big cluster. Then in the next iteration, divisive will see can we break up this big cluster into two smaller clusters

So, it tries to divide the ["large cluster"] into [smaller clusters]

Iter.1 :- 1-Cluster (all the points)

↓

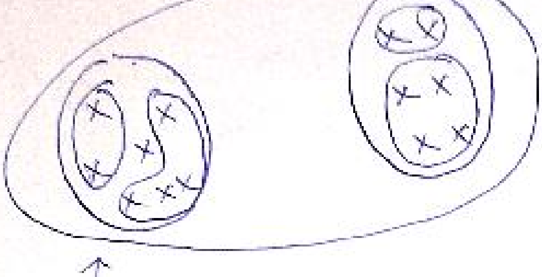
Iter.2 :- 2-Clusters

↓

Iter 3 :- 4-Cluster

⋮

Iter n :- 13-Cluster (each single point)



In third iteration, it will probably break it into smaller clusters like.

So on so forth, it will keep on breaking the larger clusters into smaller ones till each cluster becomes effectively a single point.

So divisive starts off with one big cluster comprising of all the points & it reaches "13 clusters" => one cluster for each point.

Agglomerative does the other way round.

Note:> One problem with divisive clustering is, how to divide a larger cluster into smaller ones

Agglomerative clustering is often used becoz trying to group things on the basis of similarity or distance is easy ~~than~~ trying to break things. But that doesnot mean divisive clustering is hard or impossible to implement.

PTO

Now let us understand why it is known as ["Hierarchical clustering"]

If we think about it, there is basically a kind of hierarchy it follows like

$$\left[\begin{array}{l} 13 \text{ pts} \longrightarrow 13 \text{ clusters} \\ \downarrow \\ 9 \text{ clusters} \\ \downarrow \\ 5 \text{ clusters} \\ \downarrow \\ 2 \text{ clusters} \\ \downarrow \\ 1 \text{ clusters} \end{array}\right] \rightarrow \text{"hierarchy"}$$

The key ingredient for Agglomerative clustering to work is basically the similarity measure or a distance measure, between not just points but also between clusters.

We are grouping nearby clusters, or similar clusters. So, we need to have a sense of how to measure similarity or distance between clusters.

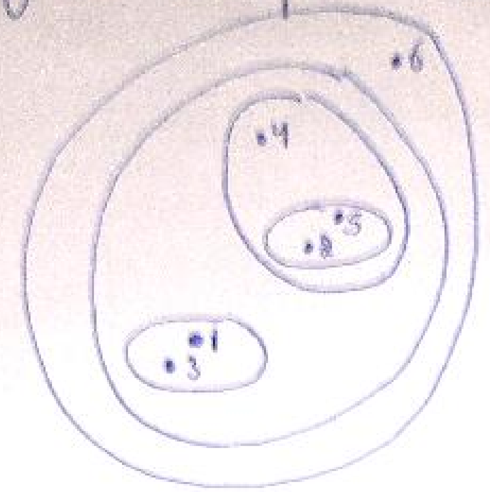
So once we have this, then the algorithm is very very straight forward.

There is another way to visualize or understand hierarchical clustering :->

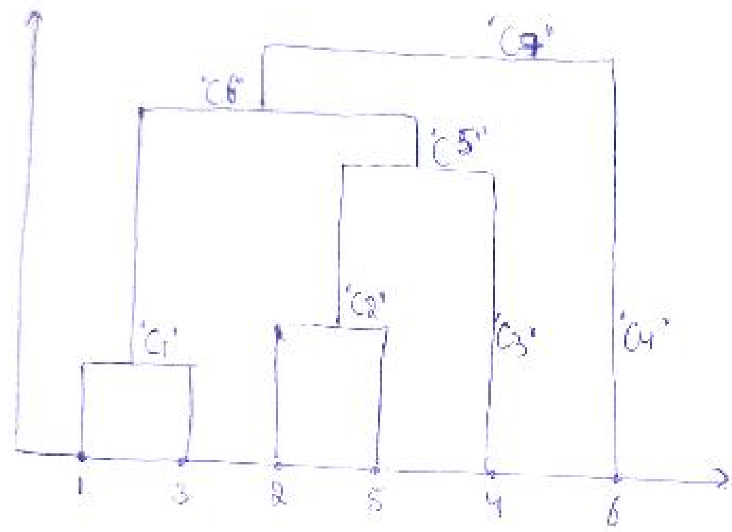
↳ Produces a set of nested clusters organized as a hierarchical tree.

let's understand it from another perspective

let's say we have points as shown :



So, we have 6 points in total.



In "Agglomerative clustering" since '1' & '3' are close together, they will be grouped into one cluster. Similarly '2' & '5' are also close, hence they will be grouped together into one cluster.

So we are trying to build a tree-like structure.

In this we are taking points, which are close to each other & we are grouping them into a cluster

In first iteration, we are grouping $\boxed{1 \& 3}$ & $\boxed{2 \& 5}$

At the end of first iteration, we have four clusters.
$C_1 \rightarrow \{1, 3\}$, $C_2 \rightarrow \{2, 5\}$, $C_3 \rightarrow \{4\}$ & $C_4 \rightarrow \{6\}$

In second iteration, we notice that cluster C_2 & C_3 are close together, so we will group them into one cluster & name it as 'C5'.

$$C_5 \rightarrow \{C_2, C_4\} \rightarrow \{2, 5, 6\}$$

In third iteration, we notice that cluster C1 & cluster C5 are close together so, we will group them into one cluster & name it as "C6"

$$C_6 \rightarrow \{C_1, C_5\} \rightarrow \{1, 3, 2, 5, 6\}$$

& finally we will group together "C6 & C4" to form C7

$$C_7 \rightarrow \{C_4, C_6\} \rightarrow \{6, 1, 3, 2, 5, 6\}$$

So, we can construct a tree or a hierarchy. When we have a tree, we have a hierarchical structure of groupings. So what the tree literally records, is the sequence of merges or splits

"merges in Agglomerative clustering"
&
"splits in Divisive clustering"

It means the order in which we merge or splits these points

The tree is often referred to as "Dendrogram" in clustering.

Dendrogram :- It is defined as a tree which records the sequence of merges in case of "agglomerative - clustering" or sequence of splits in case of "divisive - clustering".

One thing to notice here is, when we do Agglomerative or Divisive clustering, we never mentioned the no of clusters, once we build this hierarchy,

13 clusters
↓
9 clusters
↓
5 clusters
↓
2 clusters
↓
1 cluster.

if we want 5-clusters we get 5-clusters & we stop at the point where we obtained that, similarly if we want 2-clusters we can get it, if we want 9-clusters again we can obtain it.

We can stop, where-ever we want. We donot give no. of clusters as hyperparameters.

PTO

Note :> In K-Means we gave no. of ~~hyperparameters~~ "clusters" as hyperparameter. (9)

for "agglomerative-clustering", there is no need to give the no. of clusters as a ["hyperparameter"]

Because once we have clustered & let's say we are using agglomerative. first let's say we want 2 clusters we can look at 2 clusters & we say, oh these two clusters are not very good. let's go finer, let's go into more detailed structure, we can get 5 clusters if we further want to go into finer structure we can get 9 clusters.

This thing is not possible with "K-Means"

With K-means, if we have trained our model for K=3 & if we get "3-clusters"

& now if we want "5-clusters" or "7-clusters", we just have to retrain or redo K-means with K=5 or K=7 to get 5 & 7 clusters respectively.

We just have to redo K-means, there is no way we can go from ["3 to 5 clusters"] or ["3 to 7 clusters"]

P.T.O

→ "Agglomerative Clustering Algorithm"

Step 1:- Compute the proximity matrix.

Step 2:- Let each data point be a cluster

Step 3:- Repeat.

Step 4:- Merge the two closest clusters.

Step 5:- Update the proximity matrix.

Step 6:- Until only a single cluster remains.

Note:- Key operation is the computation of the proximity of two clusters.

Diff. approaches to defining the distance between clusters distinguish the different algorithms.

Numerical Example :-

Given 5 points & the distance matrix Draw a dendrogram using agglomerative clustering.

Q.

	P_1	P_2	P_3	P_4	P_5
P_1	0				
P_2	9	0			
P_3	3	7	0		
P_4	6	5	9	0	
P_5	11	10	2	8	0

These values are nothing but the distances

Sol:- Leave the diagonal elements & select the value which is minimum among all the ["non-diagonal values"]

min. value is "8" & 8 lies at the crosssection of (P_3 & P_5) ∴ [P_3, P_5] can be combined into a new cluster.
Now we need to compute the proximity matrix again using [P_3, P_5] as a <u>new cluster</u>

	P_1	P_2	[P_3, P_5]	P_4
P_1	0			
P_2	9	0		
[P_3, P_5]	3	7	0	
P_4	6	5	8	0

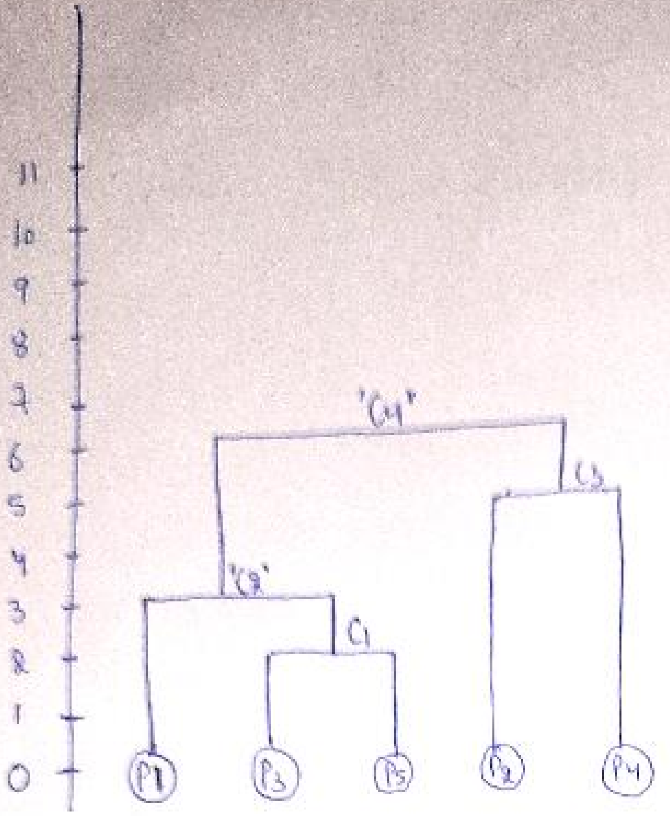
$\Rightarrow d(P_1, [P_3, P_5])$
$\Rightarrow \min(d(P_1, P_3), d(P_1, P_5))$
$\Rightarrow \min(3, 11) \Rightarrow 3$

$\Rightarrow d(P_2, [P_3, P_5])$
$\Rightarrow \min(d(P_2, P_3), d(P_2, P_5))$
$\Rightarrow \min(7, 10) \Rightarrow 7$

$\Rightarrow d(P_4, [P_3, P_5])$
$\Rightarrow \min(d(P_4, P_3), d(P_4, P_5))$
$\Rightarrow \min(9, 8) \Rightarrow 8$

Now from the new proximity or distance matrix, we need to again choose the minimum off diagonal value.

P.T.O



Now the minimum distance is '3' among all the off diagonal elements. & '3' lies at the crossection of [P1] & [P3, P5]. so the new cluster will be [P1, P3, P5]

Now let's again compute the proximity matrix :->

	[P1, P3, P5]	P2	P4
[P1, P3, P5]	0		
P2	7	0	
P4	6	5	0

$\Rightarrow d(P2, [P1, P3, P5])$

$\Rightarrow \min(d(P2, P1), d(P2, P3), d(P2, P5))$

$\Rightarrow \min(9, 7, 10)$

$\Rightarrow 7$

P.T.O

$\Rightarrow d(P_4, [P_1, P_3, P_5])$

$\Rightarrow \min(d(P_4, P_1), d(P_4, P_3), d(P_4, P_5))$

$\Rightarrow \min(6, 9, 8) \Rightarrow 6$

Now, select the min. off diagonal element. So, '5' is the min value. & it lies on the cross section of [P4]. & [P2] ∴ [P4, P2] will form a new cluster.

Now we need to upgrade the dendrogram & find the new proximity matrix.

	[P1. P3 P5]	[P2 P4]
[P1.P3 P5]	0	
[P2, P4]	6	0

$d([P_1, P_3, P_5], [P_2, P_4])$

$\Rightarrow \min(d(P_2, P_1), d(P_2, P_3), d(P_2, P_5), d(P_4, P_1), d(P_4, P_3), d(P_4, P_5))$

$\Rightarrow \min(9, 7, 10, 6, 9, 8)$

$\Rightarrow 6$

Now "6" lies at the cross section of [P2, P4] & [P1, P3 P5]

∴ It will be grouped as a cluster. [P1, P2, P3, P4, P5]

—o—