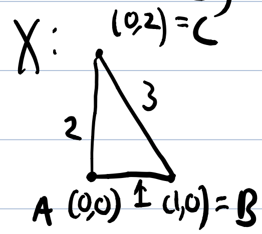
# AMAT 583 Lecture 23

<u>Today</u>: Single linkage review
Single linkage for real-valued metrics.
Algorithm for computing single linkage dendrograms
Average linkage.

## Single linkage ($\mathbb{N}$-valued metrics)

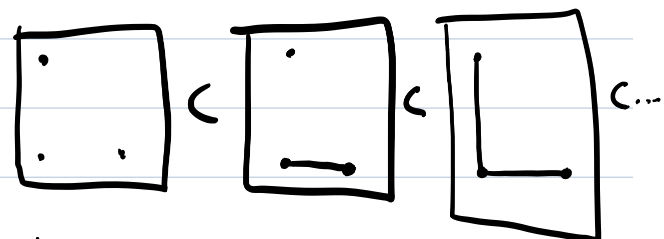
$\boxed{\text{Finite Metric Space } X}$

$X$:




$\downarrow$

$\boxed{\text{Neighborhood Graphs } N_0(X) \subset N_1(X) \subset N_3(X) \subset \ldots}$



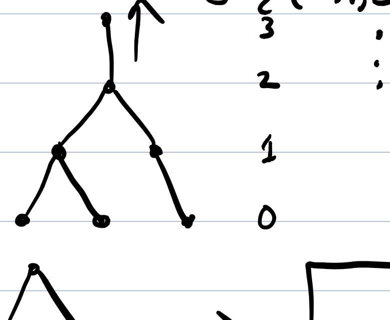
$\downarrow$ Consider connected components

$\boxed{\text{Hierarchical Partition } \{SL(X)_z\}_{z \in \mathbb{N}} \quad SL(X)_0 \quad SL(X)_1 \quad SL(X)_2}$

$SL_0 = \{\{A\}, \{B\}, \{C\}\}$
$SL_1 = \{\{A, B\}, \{C\}\}$
$SL_2 \{ A, B, C \}\}$
$\vdots$

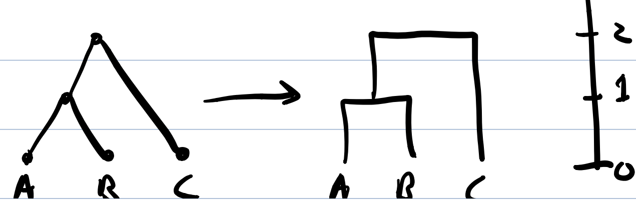
$\downarrow$

$\boxed{\text{(Untrimmed) Dendrogram}}$




$\downarrow$

$\boxed{\text{Trimmed dendrogram}}$



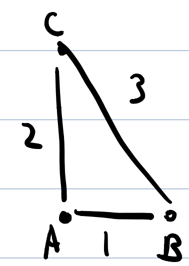
Algorithm for construction of the (trimmed) single linkage dendrogram.

Input: A finite metric space $(X, d)$
Output: Trimmed single linkage dendrogram $D$.

Psuedocode:

- Order pairs of points in $X$ according to increasing distance. [break ties arbitwrily] $E = ([A,B], [A,C], [B,C])$
- Initially, put each element of $X$ into its own cluster. [As we build we will merge the clusters.]
- Add a vertex to $D$ at level $0$ for each cluster.
- For each edge $e = [v, w]$ in increasing order:
  If $v$ and $w$ belong to different clusters $S_v$ and $S_w$
    - merge the clusters $S_v$ and $S_w$ to form a new cluster $S$.
    - Add the vertex $S$ to $D$ at level $d(v, w)$
    - Add the edges $[S_v, S]$ and $[S_w, S]$ to $D$.

Example: X:



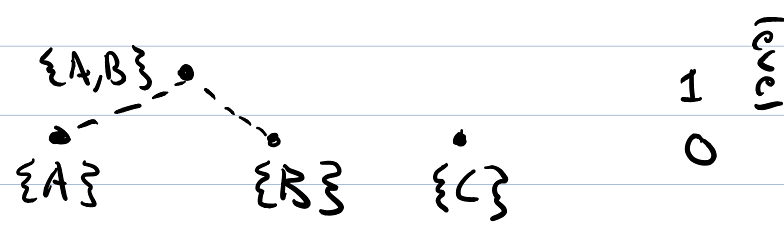

E=([A,B],[A,C],[B,C])

Initial clusters: {A}, {B}, {C}.

Consider first edge [A,B]. $\overset{v}{\overset{\parallel}{A}}$ and $\overset{w}{\overset{\parallel}{B}}$ belong to different clusters $\Longrightarrow$ merge these. (Merged cluster is {a,b}.)

now the clusters are $\underset{\underset{S}{\parallel}}{\{A,B\}}$, {C}.

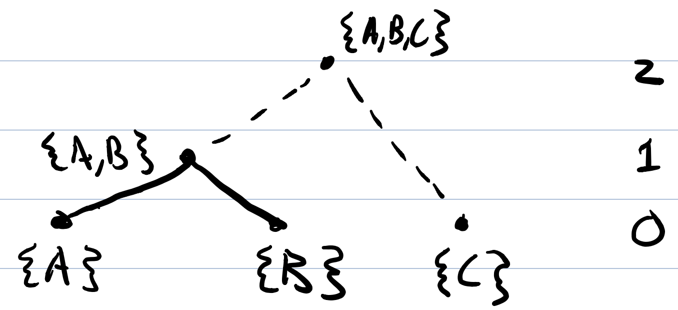
- Add a new vertex labelled {A,B} at level d(A,B)=1,
- Add in edges from the unmerged clusters to the merged cluster




Consider next edge [A,C]. A and C belong to different clusters $\Rightarrow$ merge these. Merged cluster is {A,B,C}.

Now the clusters are {A,B,C} (just one cluster)

- Add a new vertex labelled $\{A,B,C\}$ at level $d(A,C)=1$
- Add in edges from the unmerged clusters to the merged cluster




Consider next edge $[B,C]$ $B$ and $C$ already belong to the same cluster so we are done.

Missing details: When implimenting this on a computer, how do we store the clusters in a way that allows us to quickly check whether two points belong to the same cluster and quickly merge the clusters?

Answer: Union-find data structure (classical data structure from computer science.)

Reference: Edelsbrunner / Harer Computational Topology
CLRS Chapter 21.