Topological Space (Point set topology)

Let X : point set and U : set of subsets of X.

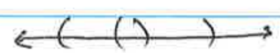
Def 1 U is a topology of X if : (a) $X, \phi \in U$
(b) Any union of sets in U is also in U.
(c) A finite intersection of sets in U is in U.

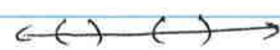
Def : if U is a topology of X then (X, U) is called a topological space.

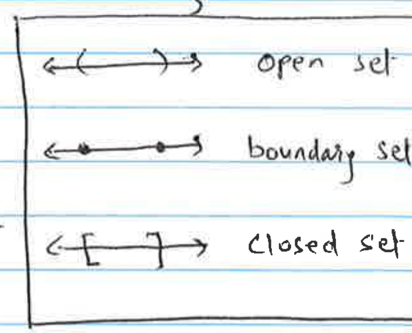
Example 1 : $X = \{1, 2, 3\}$. $U = \{\phi, \{1, 2, 3\}\}$
U satisfies conditions (a) (b) & (c) $\therefore$ U is topology on X
$\rightarrow$ U is trivial topology on X.

Example 2 : $X = \{1, 2, 3\}$, U : power set of X $\rightarrow \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}$

Example 3 : $\mathbb{R}^1$ with B : set of all open intervals
$\hookrightarrow$ * set of all open sets.

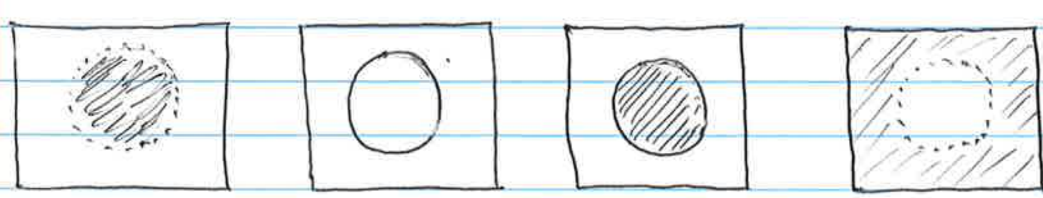

intersection and union are both open


intersection is ϕ
union is still an open set



Def : A subset u of $\mathbb{R}^n$ is is called open if given any point $x \in u$, there exists a real number $\epsilon > 0$ such that for all points $y \in \mathbb{R}^n$ such that $dist(x,y) < \epsilon$ $y \in u$

Def : A closed set is a set whose complement is an open set



Def: A function $f: \mathbb{X} \to \mathbb{Y}$ is continuous if the pre-image of every open set is open

→ For all open sets $V \subseteq \mathbb{Y}$, $f^{-1}(V) = \{x \in \mathbb{X} \mid f(x) \in V\}$ ~~is~~ then $f^{-1}(V)$ is ~~an~~ open set in $\mathbb{X} \Rightarrow f$ is continuous

Example: $f: \mathbb{R} \to \mathbb{R}$ $\quad f(x) = \begin{cases} 0 & x \in (-\infty, 0) \\ 1 & x \in (0, \infty) \end{cases}$

for any open interval $(-\epsilon, \epsilon)$, $f^{-1}((-\epsilon, \epsilon)) \to$ Not open in $\mathbb{R}$

(*) if we allow infinite intersection, by definition of topology, it will have to be open → a single point on real line would be open set which would mean every function is continuous.

Def: A path is a continuous function $\gamma: [0,1] \to \mathbb{X}$

→ A topological space is path connected if every pair of points is connected by a path.

$\boxed{\text{Union - find}}$ also called disjoint set data structure,

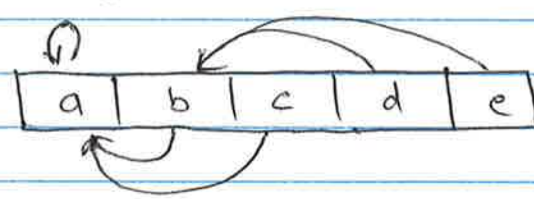
→ with algorithm to test connectedness.

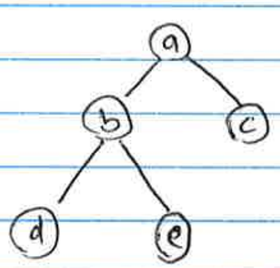
→ Represent each ~~element~~ set as a tree of elements.

→ Maintain a collection of sets under operation of:

1. MakeSet(x): Create a set containing single element x.
2. Find(x): Return the root of the tree containing x.

Example: $\{a, b, c, d, e\} \Rightarrow$

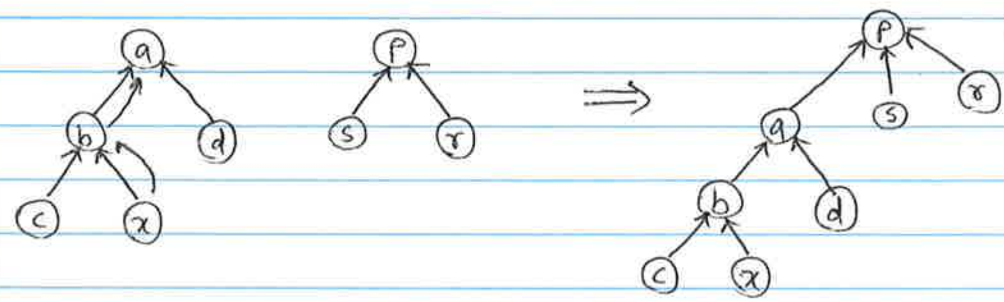




3. Union(x, y): make the root of tree containing x to also be the root of tree containing y

Reversed tree data structure

1. Make Set (x) : make a singleton pointing to itself
2. Find (x) : traverse from x to root, return the root.
3. Union (a, p) :



Find (a) will return a.

Union (a, p) will give the tree

→ issue with union : long, skinny trees will increase running time of Find (e) ~ $O(n)$

→ Union - find running times

when roots are already known.

	Make Set	Union	Find
• worst case	$O(1)$	$O(1)$	$O(\log n)$
• Amortized	$O(1)$	$O(\alpha(n))$	$O(\alpha(n))$

where $\alpha(n)$ is a very slow growing function (almost constant)

→ Requires 2 hacks :

(a) Union by rank : Always hang the smaller tree on the larger tree [Need to store rank / depth]

(b) Path - Compression : In the Find operation, having all the nodes on the path directing to the root