Topological Space (Point set topology)

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

Def: $U$ is a topology of $X$ if:
1. $\emptyset, \Omega \in U$
2. Any union of sets in $U$ is also in $U$.
3. 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: $X = \{1, 2, 3\}$, $U = \emptyset, \Omega, \{1, 2, 3\}$

$U$ satisfies conditions $\circ \circ \circ \circ$ : $U$ is topology on $X$

$U$ is trivial topology on $X$.

Example 2: $X = \{1, 2, 3\}$, $U$ : power set of $X$ - $\emptyset, \Omega, \{1, 2, 3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1\}, \{2\}, \{3\}$

Example 3: $\mathbb{R}$ with $B$ : set of all open intervals, $\times$ set of all open sets.

Intersection and union are both open.

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

Def: A closed set is a set whose complement is an open set.
Def: A function \( f: X \to Y \) is continuous if the pre-image of every open set is open.

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

Example: \( f: \mathbb{R} \to \mathbb{R} \)
\[
f(x) = \begin{cases} 
0 & \text{if } x \in (-\infty, 0) \\
1 & \text{if } x \in (0, \infty) 
\end{cases}
\]

for any open interval \( (-\varepsilon, \varepsilon) \), \( f^{-1}((-\varepsilon, \varepsilon)) = \) Not open in \( \mathbb{R} \)

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

Def: A path is a continuous function \( Y: [0, 1] \to X \)

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

Union - find

also called disjoint set data structure,

- with algorithm to test connectedness.
- Represent each set as a tree of elements.
- Maintain a collection of sets under operation \( \text{union} \)!
- Make-Set \((x)\): Create a set containing single element \( x \).
- Find \((x)\): Return the root of the tree containing \( x \).

Example: \( \{a, b, c, d, e\} \)

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** \((x)\) will return \(a\).
   - **Union** \((a, p)\) will give the tree.

   ![Diagram showing the union operation]

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

   → **Union - Find running times** when roots are already known:

   - Worst case \(O(1)\)
   - 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