**Dynamic Connectivity Problem**

The **dynamic connectivity** problem involves determining if there is a path between two objects in a set of objects that can be connected or disconnected along time. This problem is frequently addressed by using algorithms that supports the following operations:

* **Union:** connects two objects;
* **Find:** verify if two objects are connected;

**REAL LIFE APPLICATIONS:** social networks, image pixels, electrics circuits,

The **connections** between objects must attend to natural and intuitive properties named as **equivalence relations:**

* **Reflexive:** *p* is connected to *p.*
* **Symmetric:** if *p* is connected to *q*, then *q* is connected to *p*.
* **Transitive:** if *p* is connected to *q* and *q* is connected to *r*, then *p* is connected to *r.*

**Connected components:** maximal set of objects that are mutually connected.

A diagram of a diagram of a diagram

AI-generated content may be incorrect.

**Image 1 – Connected components.**

**Quick Find Algorithm**

The **quick find algorithm** is one way to solve the **dynamic connectivity** problem with fast **find** operations but slow **union** operations.

**Data Structure:** uses an *array* ***id****[]*, where ***id[****i****]*** stores the identifier of the group which element *I* belongs to.

*Two elements are connected if they have the same* ***id****.*

**Operations:**

* **Find/Connected:** checks if ***id[****p****]*** *==* ***id[****q****]***. (constant time-complexity: **O(1)**).
* **Union:** to connect *p* and *q*, changes all elements with ***id[****p****]*** to ***id[****q****]***. (linear time-complexity: **O(N)**).

**Quick Union Algorithm**

The **quick union algorithm** is another way of solving the **dynamic connectivity** problem with fast **union** operations but slow **find** operations.

**Data Structure:** uses an *array* ***id[]***, but now ***id[****i****]*** stores the parent of *I* in a tree structure. The **root** of an element is found by scaling up to parents until it finds an element that is its own parent (***id[****i****]*** == *i*).

*Two elements are connected if they have the same* ***root****.*

**Operations:**

* **Find/Connected:** compares the **roots** of *p* and *q*. (tree height proportional time-complexity: **O(N)** worst-case scenario.)
* **Union:** connects the **root** of *p* to the root of *q*. (tree height proportional time-complexity: **O(N)** worst-case scenario.)