Union Find

# Dynamic Connectivity

It automatically establishes ad-hoc network connections between nodes on a network.

Given two nodes, p, and q, in a network, the two nodes have an equivalence relation.

* 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

Connect through other nodes. If not connected, do not connect directly.

## Examples

p and q represent two brain regions, and the pair represents a network connection between these two brain regions connected.

## Terminology

Objects (nodes) are called **sites**, node pairs are **connections,** and the equivalence classes are the connected components.

Components can be represented by one of the sites (nodes)

## Implementation

### UnionFind

In the “Algorithm” book, they considered three different implementations for UnionFind, all based on the array data structure. The book abstracts the implementation of Union and Find routines in a base class called UnionFind and defines it in three different implementations. In our Python implementation, we mimicked this behavior by declaring a base class *UnionFind* with abstract methods find and union. These methods are implemented in child classes. The difference between our implementation and the book implementation, Is that we separated the implementation from the interface. Therefore, we added the function connect, which takes an array as input, iterates over the array, and performs the union operation.   
Data Structure  
Arrays: where the index of an array represents a node, and the value on that index is the connected component.

# References

## Reading

Algorithm Sedgewick et al. l, Chapter 1. Section 1.5

## Pseudocode

There is no pseudocode per the formal definition but @page 222; there is a text that explains the implementation of Quick-find, @page 224 for Quick-Union, and @page 227 for Weighted Quick-Union.

## Source Code

UnionFind.py

## Demo

#### Toy demo

runUnionFind.py

#### Real demo