**Module 2: Union Find**

Connecting objects

* Is connect to
  + Reflexive = p is connected to p
  + Symmetric = is 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
* Find Query, checks if two objects are in the same component
* Union command, replace components containing two objects with their union
* **Question**
  + How many connected components result after performing the following sequence of *union* operations on a set of 1010 items?
  + 1–2    3–4    5–6    7–8    7–9    2–8    0–5    1–91–23–45–67–87–92–80–51–9
  + **Answer 3** The connected components are {0,5,6}, {3,4} and {1,2,7,8,9}

Quick Find

* + You can tell if two objects are in the same component if they have the same ID
  + Quick find involves changing the id for every object union
  + Quick find is slow because it must go through the entire array
  + Takes N^2, quadratic time, which is too slow, these do not scale well. As computers get bigger these algorithms get slower
* Quadratic algorithms do not scale
  + 10^9 operations per second and word of main memory
* Question
  + What is the maximum number of id[]id[] array entries that can change (from one value to a different value) during one call to *union* when using the quick-find data structure on n elements?
  + Answer: n

Quick Union

* Lazy approach, avoid doing work until we must
* This approach uses a forest instead, elements have parents, they are connected by a root which represents the connected component
* Question
  + Suppose that in a quick-union data structure on 10 elements that the id[]id[] array is
    - i 0 1 2 3 4 5 6 7 8 9
    - -------------------------------------
    - id[i] 0 9 6 5 4 2 6 1 0 5
* **Answer** 6 and 6
* To union these you take the id of Qs root and merge this to the ID of Ps root,
  + This is faster because it only involves changing one entry in the array
* Difference between a quick union and a quick find is that unions on quick find are completed by changing the root of every index and quick union will just match the roots
* Quick union also can be too slow. If the root becomes too tall you still must loop through every single index.
* Question: Consider the maximum number of array accesses during a *find* operation when using the quick-union data structure on nn elements. How does this quantity grow as a function of nn?
* Answer: Linear

Quick Union Improvements

* Weighting
  + When implementing the quick union, take steps to avoid having a tall tree
  + Avoid putting large trees below the smaller trees
  + We can do this by keeping count of the number of ids in a tree before performing a union
  + This allows there to be some guarantee that no single item is too far from the root, creating a very tall tree
  + The depth on any node is always at most log base 2 of N aka lg N
    - Depth is very small relative to N
  + This performance is very much acceptable and good to scale
  + This can also be improved further
* Path compression
  + After computing the root of p, set the id of each examined node to point to that root.
  + This creates a two-pass implementation
    - Adds a second loop to root () to set the id [] grandparent, thereby halving the path length
  + Only one extra line of code and will keep the trees almost completely flat
  + The running time is going to be linear usually less than 5
* All of this tied together is abbreviated as WQUPC, weighted path compression quick find
* There is no linear solution to the quick union issue, but it is very close
* WQUPC allows us to solve problems that could not be solved otherwise
  + Quick find, time complexity = M N
  + Quick union, time complexity = M N
  + Weighted QU, time complexity = N + M Log N
  + Weighted QU + path complexity = N + M lg\* N
* Algorithm design helps the solution to the problem, a faster computer does not beat a bad algorithm

Union Find Applications

* Percolation
  + Way to get from the top to the bottom of a grid, is percolated if true otherwise false
  + This is a model for electricity, water flowing through a porous substance. Social network. There is either a connection between people or not
* Monte carlo simulation
  + Initialize n by n whole grid to be blocked
  + Declare random sites open until the top connected to the bottom
  + The percolation threshold is about 0.592746