Motivation

* Let’s say we are representing the groups in CS 61BL in our data structure. Three groups can come together to form a super-super-group. How to we connect a particular student to the other students in the super-super-group?

Disjoint Sets

* Easy way to represent connectivity
* Create an array, arr, whose length is the number of elements in your list
* arr[i] is the parent index of element i
  + If i doesn’t have a parent, arr[i] will represent the negative size of the set
* Operations:
  + Union(u, v): connect the set of u to the set of v
  + Find(u): find the root of the set containing v
* Optimizations:
  + Weighted quick union: union the root of the smaller set to the root of the larger set, will keep the data structure relatively shallow
  + Path compression: Once you call find(u), connect the whole path from u to the root of the tree to the root, next time you call find will be cheap!

Minimum Spanning Trees

* A spanning tree of the graph G is a tree T that touches all the vertices of G. When it is a minimum spanning tree, we are looking for a spanning tree whose total weight is the minimum number
* Algorithm:
  + Get all edges and sort by weight
  + For each edge (u, v):
    - If u is not connected to v, add the edge to the resulting graph
  + Terminate once the resulting graph has V-1 edges
* Cut property