

## CS61B Lecture #38

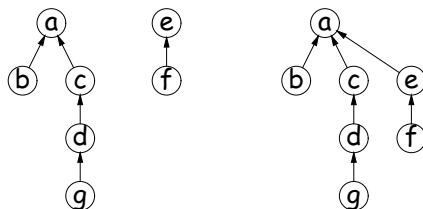
- Preliminary grading run will be next Monday.
- **Today:** Union-find and some style comments.
- **Next topic:** Goodrich and Tamassia, Chapter 11.

## Union Find

- Kruskal's algorithm required that we have a set of sets of nodes with two operations:
  - *Find* which of the sets a given node belongs to.
  - Replace two sets with their *union*, reassigning all the nodes in the two original sets to this union.
- Obvious thing to do is to store a set number in each node, making finds fast.
- Union requires changing the set number in one of the two sets being merged; the smaller is better choice.
- This means an individual union can take  $\Theta(N)$  time.
- Can union be fast?

## A Clever Trick

- Let's choose to represent a set of nodes by *one* arbitrary representative node in that set.
- Let every node contain a pointer to another node in the same set.
- Arrange for each pointer to represent the *parent* of a node in a tree that has the representative node as its root.
- To find what set a node is in, follow parent pointers.
- To union two such trees, make one root point to the other (choose the root of the higher tree as the union representative).



## Path Compression

- This makes unioning really fast, but the find operation potentially slow ( $\Omega(\lg N)$ ).
- So use the following trick: whenever we do a *find* operation, *compress* the path to the root, so that subsequent finds will be faster.
- That is, make each of the nodes in the path point directly to the root.
- Now union is very fast, and sequence of unions and finds each have essentially constant amortized time.
- Example: find 'g' in last tree (result of compression on right):

