CSE373 Help Session
4/23/15

Union-Find ADT

•Given an unchanging set *S*, `create` an initial partition of a set

–Typically each item in its own subset: {a}, {b}, {c}, …

–Give each subset a “name” by choosing a *representative element*

•Operation `find` takes an element of *S* and returns the representative element of the subset it is in

•Operation `union` takes two subsets and (permanently) makes one larger subset

–A different partition with one fewer set

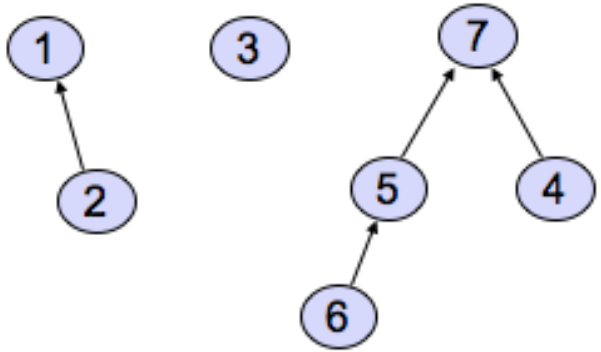
–Affects result of subsequent `find` operations

–Choice of representative element up to implementation

Up-tree data structure

- Tree with:
  - No limit on branching factor
  - References from children to parent
- Start with *forest* of 1-node trees

- Possible forest after several unions:
  - Will use roots for set names

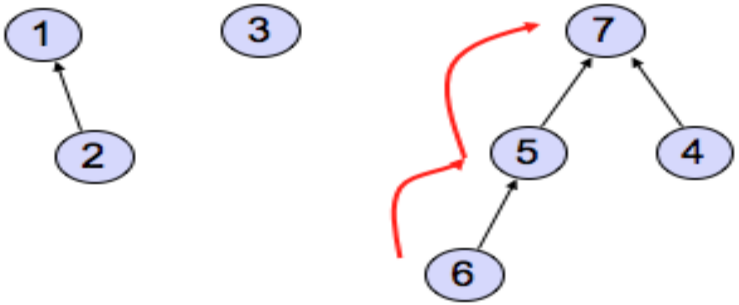


Find

`find(x)`:

- *Assume* we have *O*(1) access to each node
  - Will use an array where index `i` holds node `i`
- Start at **`x`** and follow parent pointers to root
- Return the root

`find`(6) = 7

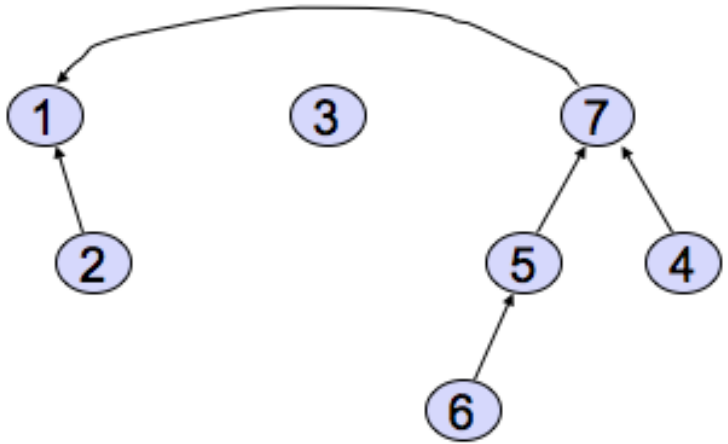


Union

`union(x,y)`:

- Assume **`x`** and **`y`** are roots
  - Else **`find`** the roots of their trees
- Assume distinct trees (else do nothing)
- Change root of one to have parent be the root of the other
  - Notice no limit on branching factor

`union`(1,7)



Path compression

- Simple idea: As part of a **`find`**, change each encountered node's parent to point directly to root
  - Faster future **`find`**s for everything on the path (and their descendants)