

Lecture 22: Friday March 7, 2003

today

- disjoint set union-find problem
 - implementation: forest of rooted trees
 - improvement: union by rank
 - improvement: compressed find

announcements

recall: disjoint sets union-find problem [CLRS Ch. 21]

- the DSUF ADT
- analysis: sequence of operations
- implementation: array of representatives
- application: graph components
- implementation: forest of rooted trees
 - basic imp'n
 - improvement: union by rank
 - improvement: compressed find

disjoint sets: abstract data type

- maintain: pairwise disjoint sets
- one element of each set is *representative*
- operations
 - `MakeSet(x)` $S_x \leftarrow \{x\}$
 - `Find(x)` return representative of set containing x
 - `Union(x,y)` $S_z \leftarrow S_{f(x)} \cup S_{f(y)}$

simplest DS implementation: array of representatives

- all n Makesets take $\Theta(n)$ time
- each Find(x) takes $\Theta(1)$ time
- each Union(x,y) takes $\Theta(n)$ time



better(?) DS implementation: forest of rooted trees

- elements of set \leftrightarrow nodes of rooted tree
- representative of set \leftrightarrow root of tree
- each node needs only parent \Rightarrow implement forest via array

```

proc allMakeSets(n)  { initialize all parents
  for x <- 1 to n do
    P[x] <- x      { parent of root is root

proc find(x) { return element at root
  while P[x]<>x do
    x <- P[x]    { parent of x
  return x

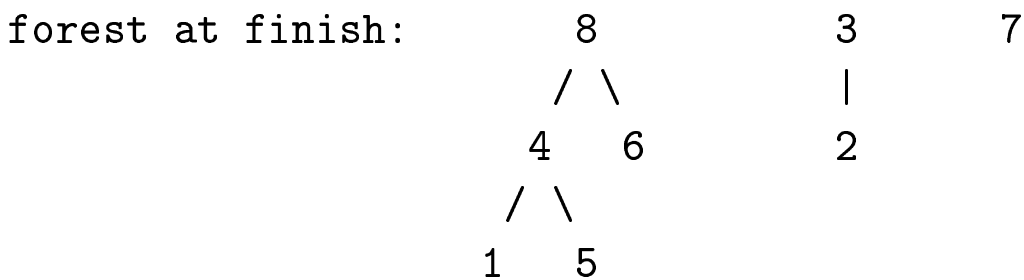
proc union(x,y) { make root of x's tree child of root of y's
  rx <- find(x)
  ry <- find(y)
  P[rx] <- ry

```


same example *****
 sets at start {1} {2} {3} {4} {5} {6} {7} {8}

indices	1	2	3	4	5	6	7	8
P	[1	2	3	4	5	6	7	8]
U(1,4)	[4	2	3	4	5	6	7	8]
F(1)								
F(4)								
U(2,3)	[4	3	3	4	5	6	7	8]
U(5,1)	[4	3	3	4	4	6	7	8]
U(1,8)	[4	3	3	8	4	6	7	8]
U(6,5)	[4	3	3	8	4	8	7	8]
F(6)								
F(3)								

sets at finish: {1,4,5,6,8} {2,3} {7}



DS rooted forest implementation: analysis

- recall: n MakeSets, then $m - n$ Union/Finds
- each MakeSet: $\Theta(1)$ time
- each Find(x): $\Theta(\text{depth}(x)) \subseteq O(n)$ time
- each Union(x,y): $\Theta(\text{depth}(x)+\text{depth}(y)) \subseteq O(n)$ time
- Union(1,2), Union(1,3), ..., Union(1, n), then Find(1), Find(1), ...
- total, this sequence: $\Theta\left(n + \sum_{j=2}^n (j-1) + (m-n-(n-1))(n-1)\right)$
 $\Theta(mn)$ if $m \gg n$
- for $m \gg n$, amortized cost per operation still $\Theta(n)$ 

better(!) DS imp'n: rooted forest with union by rank

- if rUnion used with find height = rank
- if rUnion used with cFind height \leq rank

```
proc rUnion(x,y) { make smaller rank root child of other root
  rx <- find(x)  { root of x's tree
  ry <- find(y)  { root of y's tree
  if rank[rx]>rank[ry] {compare ranks; initially 0
    then P[ry] <- rx
  else { rank[rx] less/equal rank[ry]
    P[rx] <- ry
    if rank[rx]=rank[ry]
      then INC[rank[ry]]
```

DS, union by rank, ordinary find: analysis

let $T(x)$: subtree rooted at x let $|T(x)|$: number of nodes in $T(x)$

- claim: $\text{rank}[x] = \text{height of } T(x)$
- claim: $|T(x)| \geq 2^{\text{rank}[x]}$
- ...so $\text{rank}[x] \leq \lg |T(x)| \leq \lg n$
- proofs? induction on $|T(x)|$

- time for rUnion/find in $O(\text{maximum depth})$,
- so time for rUnion/find in $O(\text{maximum rank})$,
- so worst case time in $O(n + (m - n) \lg n)$
- can show worst case time in $\Theta(n + (m - n) \lg n)$
- so for $m \gg n$, amortized time/operation now $O(\lg n)$

