Lecture 23: Monday March 10, 2003

today

- disjoint set union-find problem
  - improvement: compressed find
- graphs
  - basic definitions
  - traversal: breadth first search, depth first search

announcements

```
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]]
```

recall: weight/height/depth/rank

- node weight: number of nodes in subtree with that root
- node depth: distance (number of parent links) to root
- tree depth: max node depth
- node height: max distance to a leaf
- tree height: max node height = tree depth
- `rUnion` with ordinary `find` height(x)=rank[x]
- `rUnion` with compressed `cFind` height(x) $\leq$ rank[x]
- node with rank r, weight w: $2^r \leq w \leq n$ easy induction

how to prove height(x)=rank[x] with `rUnion`, ordinary `find`

- proof by induction: on what? maximum rank, or number of **`rU`** calls
- e.g.: induction on max rank
- base case? max rank 0, so …?
- …so no **`rU`** calls, so
  - $r[x] = 0$ for all nodes why?
  - $h(x) = 0$ for all nodes why?
- inductive case: assumptions?
  - max rank $= t \geq 0$
  - ind. hyp. holds h(x)=r[x] for all nodes
  - now suppose a **`rU`** call increases the max rank
- this implies
  - one node's rank is changed to $t + 1$ why?
  - rx and ry must have been t why?
  - ry is now $t + 1$ why?
  - rx and ry each had height t why?
  - ry's new height? $t + 1$ (why?)
  - observe: for all other nodes, h and r unchanged why?
  - so have h(x)=r[x] for all nodes ☺

- recall: union/find WC $\Theta(n + (m - n)n)$
- recall: rUnion/find WC $\Theta(n + (m - n)\lg n)$

even better(!!): union by rank, compressed find

- cFind: on node-root path, change parent of each node to root

```
non-recursive                         * recursive
                                      *
                                      *
proc cFind(x)                         *
  t <- x                              * proc cFind(x)
  while P[t]<>t do { find root        *   if x<>P[x] then
    t <- P[t]                         *     P[x] <- cFind(P[x])
  root <- t                           *   return P[x]
  t <- x                              *
  while P[t]<>t do { compress path    *
    x <- t                            *
    t <- P[t]                         *
    P[x] <- root                      *
  return root                         *
```

analysis: rUnion and cFind

- what's the complexity?
- $\lg^* n$: smallest t so that $2^{2^{2^{\cdots}}} \geq n$, where t is number of 2's
- $\lg^* n \leq 5$ for $n \leq 2^{65536}$ virtually constant
- $\alpha(n)$ grows even more slowly than $\lg^* n$
- $\Theta(1) \subset o(\alpha(n))$ and $\alpha(n) \in o(\lg^* n)$

```
  n     2  ...4  ...16  ...65536  ...2^{65536}
lg*n    1    2     3      4             5
```

DSUF conclusion

- U/F WC $\Theta(n + (m-n)n)$
- rU/F WC $\Theta(n + (m-n)\lg n)$
- rU/cF (proof not too hard) WC $O(n + (m-n)\lg^* n)$
- rU/cF (proof harder; see text) WC $O(n + (m-n)\alpha(n))$