L18 10/30
L19 11/01

CS-25 Algorithms 10/30/85

Last time (chap 20, 6)
- Mergeable heaps
- probabilistic

Today (chap 22)
- Disjoint set DS: Union-Find

Handouts
- HW5 solution

Discuss course outline: — Finish Advanced DS

— Graph Algs Chaps 23-26

— NP-completeness Chap 36

— Special topics

Discuss midterm — no exp algs

— can use techniques or ideas from class/HW to solve every prob

CS25-X94 Lecture 16 8/1/94

Disjoint-set DS

Maintain collection $\mathcal{S} = \{S_1, \ldots, S_k\}$ of disjoint dynamic sets.
Each set identified by a representative in the set.

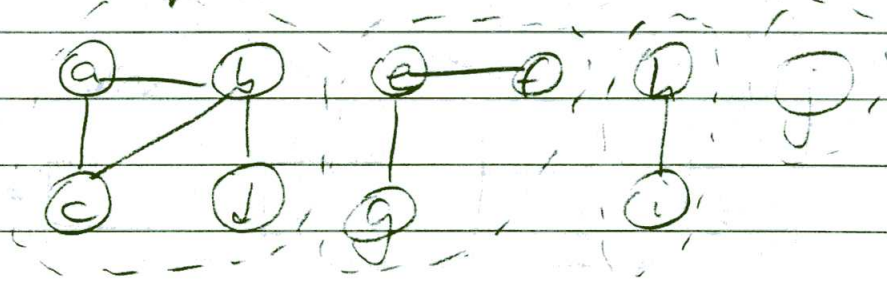
Operations:

- Make-Set(x): create new set $S_x = \{x\}$.
- Union(x, y): if $x \in S_x$, $y \in S_y$, create new set $S = S_x \cup S_y$ whose rep is either rep of S_x or rep of S_y. Destroy S_x, S_y.
- Find-Set(x): return rep of set containing x.

Connected components

An application of disjoint set DS.

Connected comps of undirected $G = (V, E)$ is a partition of V into $V_1, \ldots, V_k$ s.t. $u, v \in V_i$ iff $\exists$ path $u \rightsquigarrow v$.

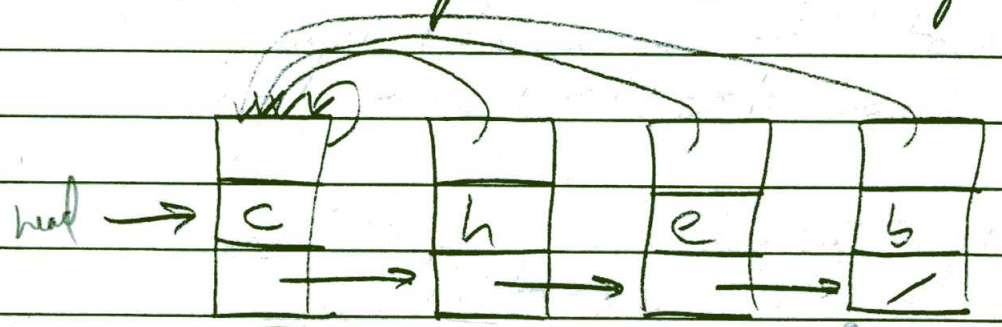


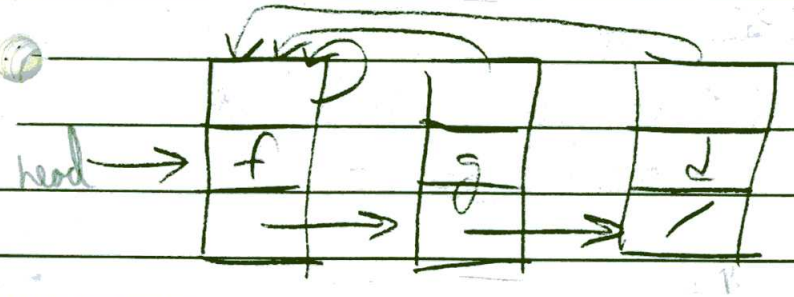
```
CC(V,E)
for each vertex v ∈ V
    do Make-Set(v)
for each edge (u,v) ∈ E
    do if Find-Set(u) ≠ Find-Set(v)
        then Union(u,v)
```

```
Same-CC(u,v)
if Find-Set(u) = Find-Set(v)
    then return TRUE
    else return FALSE
```

Go through example.

Linked-list representation
Use a linked list for each set.
Head is rep. (head ptr)
All elts point back to rep.





Make-Set, Find-Set: $O(1)$ time.
Union(x,y): copy elts in x's set into y's set.
$O(|S_x|)$ time, where $x \in S_x$.

Problem: amortized time of Union can be $\Theta(n)$.
Make-Set for $x_1, \ldots, x_n \Rightarrow \{S_1\}, \ldots, \{S_n\}$.

Operation	# elts updated
Union(x_1, x_2)	1
x_2, x_3	2
x_3, x_4	3
⋮	⋮
x_{n-1}, x_n	$n-1$
Total:	$\Theta(n^2)$

Union(a,b)
point a's set to b's representative
- new representative = representative of b.

Improvement: copy smaller set into larger.
Need to store set size, & need to keep ptr to last elt in set.

A single Union can still take $\Theta(n)$ time (e.g., when both sets have $\frac{n}{2}$ elts).

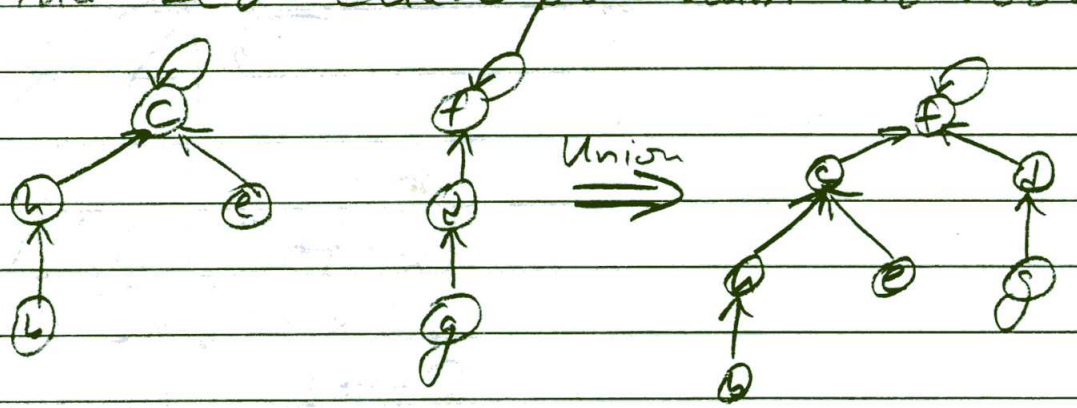
Thm: Copying smaller set into larger $\Rightarrow$ n Unions take $O(n \lg n)$ time.
i.e., a.c. $= \lg n$
Proof: Bound # of times each elt is copied.
Each time an elt is copied, it came from the smaller set. → parent pointer update

# times elt copied (parent pointer update)	size of resulting set
1	≥ 2
2	≥ 4
3	≥ 8
⋮	⋮
$\lg n$	n

∴ All elts copied $\leq n \lg n$ times. ☒

Disjoint-set forests

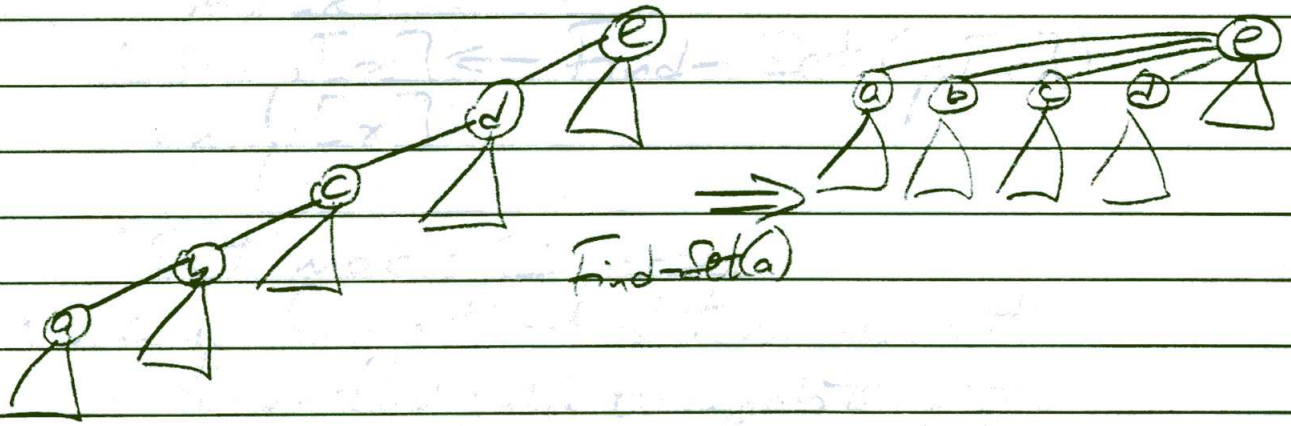
Rooted tree, rep at root.
Each elt points to its parent.
Root pts to itself.
Union: make root of one tree point to root of other
Find-Set: chase ptrs until hit root.



2 heuristics

- **Union by rank**
  Make root of smaller tree child of root of larger.
  Don't need to keep size – keep rank that approximates lg size.
  Make root w/ lower rank child of root w/ larger

- **Path compression**
  Make each node on find path point directly to root.




```
Make-Set(x)
p[x] ← x
rank[x] ← 0
```

x & y are representatives

```
Link(x,y)
  if rank[x] > rank[y]
    then p[y] ← x
    else p[x] ← y
         if rank[x] = rank[y]
           then rank[y]++
```

```
Union(x,y)
  Link(Find-Set(x), Find-Set(y))
```

```
Find-Set(x)
  if x ≠ p[x]
    then p[x] ← Find-Set(p[x])
  return p[x]
```

Find-Set makes 2 passes:
- one up find path to determine root
- one down find path to update ptrs.

∃ methods that make one pass & work as well

Performance:
Great. Constant factors are really low.

Thm: Sequence of m ops on n elts runs in $O(m \lg^* n)$ time in worst case.
Proof: Ugh.

$2^y = x$

$x = y/\log_{10} 2$

CS25-X94 L16 P7

$\log^{(i)} n = \log\log\ldots\log n$ (i)

$\log^* n$ = smallest i s.t. $\log^{(i)} n \leq 1$

So, not quite linear...

$\lg^* n$ = # times iterate lg until get to ≤ 1.

$\lg^* 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\}n = n+1.$

$\lg 2^{2^{2^2}} = 2^{65536} >> 10^{80} \approx$ # particles in universe. 19728.3 ≈ 10

$\therefore \lg^* n \leq 5$ for all practical purposes.

Can show an even better bound on same code: $O(m\,\alpha(m,n))$, where α is inverse of Ackermann's function.

$\alpha(m,n) \leq 4$ for all practical purposes.

$\lg^* 2^1 = \lg^* 2 = 1$

$\lg^* 2^2 = \lg^* 4 = 2$

$\lg^* 2^{2^2} = \lg^* 16 = 3$

$\lg^* 2^{2^{2^2}} = \lg^* 65536 = 4$

$\lg^* 2^{2^{2^{2^2}}} = \lg^* 2^{65536} \approx \lg^* 10^{19728.3} = 5$

$$A(m,n) = \begin{cases} n+1 & m=0 \\ A(m-1,1) & m>0,\ n=0 \\ A(m-1, A(m,n-1)) & m>0\ n>0 \end{cases}$$

$A(0,0) = 1$

$A(1,1) = 3$

$A(2,2) = 7$

$A(3,3) = 61$ 7/2

$A(4,4) = 2^{2^{2^{\cdot^{\cdot^{\cdot}}}}} - 3$

$= 2^{2^{65536}} - 3$

Lamm Bound $\Omega(m \cdot \alpha(m,n))$