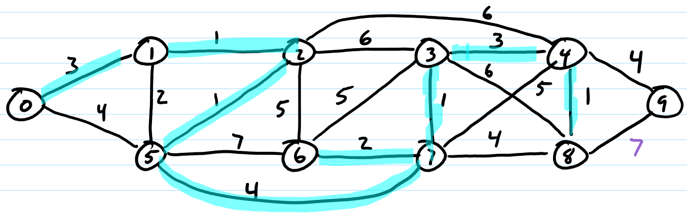
Disjoint Set Data Structure

ADD (u) : add {u} to partition

FIND-SET (u) : return representative of u's part of partition

UNION (u,v) : merge u, v's parts of partition



| v | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Comp[v] | ~~0~~ | 1 | ~~2~~ | 3 | ~~4~~ | ~~8~~ | ~~6~~ | ~~7~~ | ~~8~~ | 9 |
| Size[v] | ~~1~~ | ~~3~~4 | | 5 | | | | | | |
| UNION (1,2) | | | 1 | | | | | | | |
| UNION (2,5) | | | | | | 1 | | | | |
| UNION (3,7) | | | | | | | | 3 | | |
| UNION (4,8) | | | | | | | | | ~~4~~ | |
| UNION (6,7) | | | | | | | 3 | | | |
| UNION (3,4) | | | | | 3 | | | | 3 | |
| UNION (0,1) | 1 | | | | | | | | | |
| UNION (5,7) | | | | | | | | | | |

1) change reps of smaller connected comp

2) keep track of list of verts in each connected comp

UNION (u,v) ← reps of comps to merge

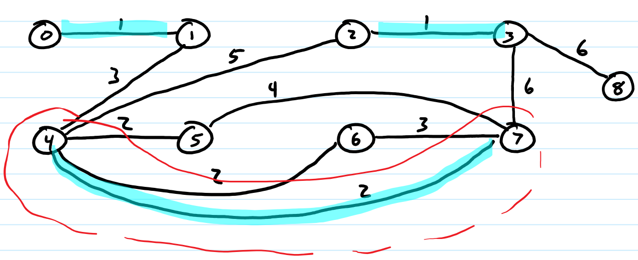
if size[u] ≤ size[v]
  for each x ∈ list[u]
    comp[x] = v
    add x to list[v]
else
  ⋮

O(n) worst-case

but — for a single x, it is iterated over at most log n

b/c on each add to list, size of comp x is in doubles (at least)

so total iterations is O(n log n)



UNION(0,1)

UNION(2,3)

UNION(4,5)

UNION(4,6)

UNION(4,7)

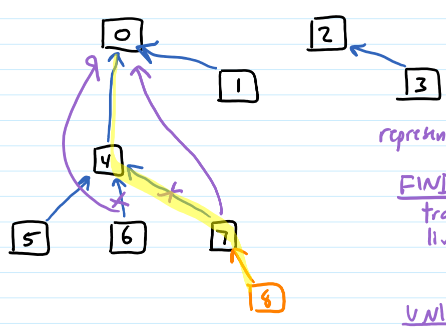
UNION(1,4)

FIND(6), FIND(7)

FIND(5), FIND(7)

FIND(4), FIND(2)

UNIONS




representative is root

FIND(v) — path compression
traverse tree from v to root
link all verts on that path directly to root

UNION(u,v) — roots — union by rank (approximation of height)
if u "smaller than" v — O(1)
  link u to v
else
  ⋮

if rank(u) = rank(v)
  link u to v
  rank(v) ← rank(v)+1
else if rank(u) < rank(v)
  link u to v

link u to v O(1)

else

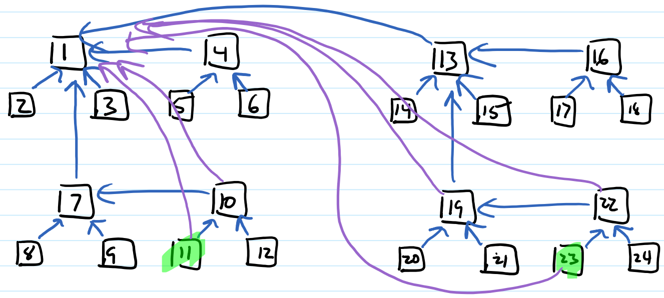
⋮

link u to v
rank(v) ← rank(v)+1
else if rank(u) < rank(v)
link u to v
else
link v to u

**UNIONS**

1 2
1 3
4 5
4 6
1 4
7 8
7 9
10 11
10 12
7 10
1 7
13 14
13 15
16 17
16 18
13 16
19 20
19 21
22 23
22 24
19 22
13 19
1 13

**FINDS**
11, 23




total time over m operations on n items

$$O(m\,\alpha(n))$$

inverse Ackerman's fxn

really slow-growing – slower than $\log^*$

$\alpha(n) \leq 4$ for any reasonable

we did a lot of easy work to set up 1 FIND that takes a few extra steps – spread that extra work over all prev operations – amortized time is low

# Kruskal's Algorithm

MST-KRUSKAL(G) precondition: $G$ is connected, no neg-weight edges

sort edges in order of nondecreasing weight $O(m \log n)$

$A \leftarrow \emptyset$

for each $v \in V$

  P.add(v)

for each edge (u,v) in order of sort

  $u' = P.find(u)$

  $v' = P.find(v)$  O(1)

  if $u' \neq v'$ then

    $P.union(u', v')$  $O(n \log n)$ total (list-based)

    $A \leftarrow A \cup \{(u,v)\}$

$O(m \log n)$ overall

$O(m\alpha(n))$ total (tree w/ path comp & union-by-rank)

INVARIANT: a) for all (x,y) before current, x,y connected in A

b) P is connected comps of A

c) A is a proto-MST

**Basis:** a) vacuously T  b) done by init of P  c) $A = \emptyset$

**Induction:** Suppose a, b, c T before loop

a) we never remove edges, so anything connected before remains connected, and if u,v not connected (in different connected components), we connect them

b) whenever we connect u,v's components, we call UNION(u,v)

c) Define cut by S = component containing u

A respects (S, V−S) b/c cut defined by A's connected comps - any edge in A is between 2 verts in same component

(u,v) crosses cut b/c $u \in S$, $v \notin S$

Consider any (x,y) that crosses cut

Suppose $\ell(x,y) < \ell(u,v)$. Then (x,y) iterated over, so by INV a, x,y connected, so (x,y) doesn't cross cut $\Rightarrow\Leftarrow$

So $\ell(u,v) \leq \ell(x,y)$ — (u,v) is as light as any other edge across the cut

Now apply Light Edge Thm to get $A \cup \{(u,v)\}$ is a proto-MST

**Termination:** At termination, all edges in G have been examined.

For any pair of vertices $u, v \in V$, u,v are connected in G there is a path $\underline{u = x_1, \dots, x_k = v}$ in $G$

$$u = x_1 \xrightarrow{G} x_2 \xrightarrow{G} x_3 \xrightarrow{G} \cdots \xrightarrow{G} x_k = V$$

$A \quad A \quad A \quad A$

for all $i$, $\underline{x_i, x_{i+1} \text{ are connected in } A}$

$(u,v)$ are connected in $A$

$\therefore$ $A$ spans $G$

$A$ is a proto-MST     INV c

$A$ is an MST     $A$ is acyclic and connects $G \to A$ is a tree

$\to$ only tree $A$ can be $\subseteq$ of is itself

$\to$ only MST $A$ can be $\subseteq$ of is itself