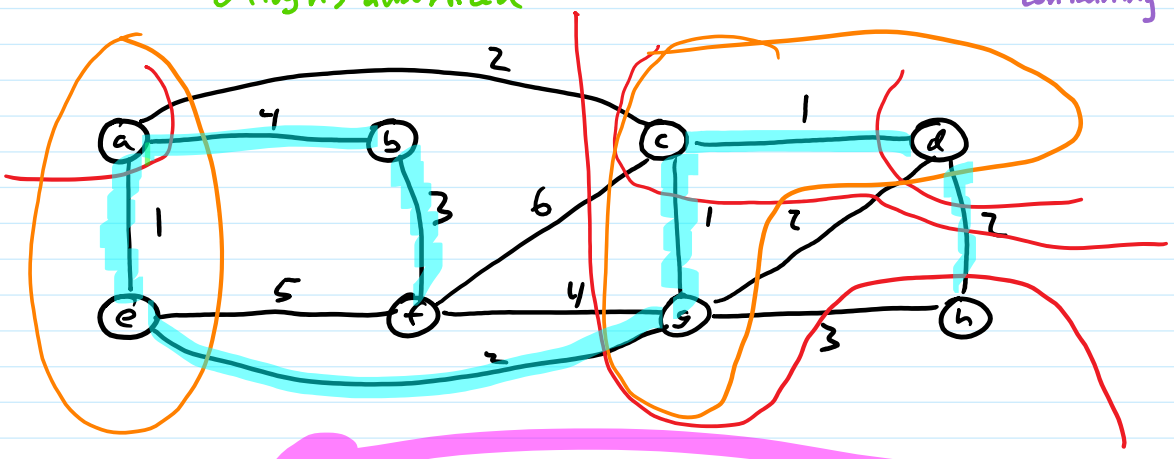
Disjoint Set Data Structure

ADD (u) : adds set {u} to partition (one elt in partition)
$\Theta(1)$

FIND-SET (u) : returns id of part of partition containing u
$\Theta(1)$ same for everything in the same part

UNION (u,v) : merges part of partition containing u with that containing v
$O(\log n)$ amortized

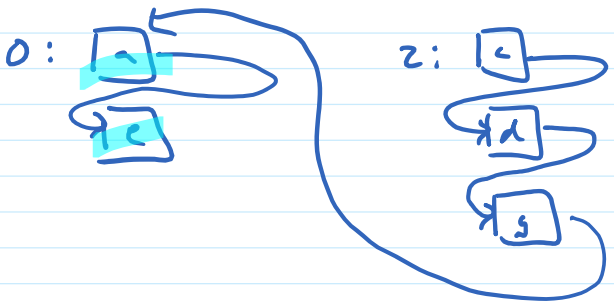
one conn. comp. using selected edges)



a	b	c	d	e	f	g	h
~~0~~	1	2	~~3~~	~~4~~	5	~~6~~	7

UNION (a, e)
UNION (c, d)
UNION (b, g)
UNION (e, g)

0 : a e
1 : b
2 : c
3 : d
4 : e
5 : f
6 : g
7 : h



Let $R_{v,i} = \begin{cases} 1 & \text{if vertex } v \text{ relabelled during union \# } i \\ 0 & \text{otherwise} \end{cases}$

total # relabellings $= \sum_{v} \sum_{i=1}^{n-1} R_{v,i}$

$R_{a,1} + R_{b,1} + R_{c,1} + \cdots + R_{h,1} \leq \frac{n}{2}$
$R_{a,2} + R_{b,2} + \cdots + R_{h,2} \leq \frac{n}{4}$

total # relabellings $= \sum_{v} \sum_{i=1}^{} R_{v,i}$

$R_{a,1} + R_{b,1} + R_{c,1} + \cdots + R_{h,1} \leq \frac{n}{2}$

$R_{a,2} + R_{b,2} + \cdots + R_{h,2} \leq \frac{n}{4}$

$R_{a,3} + R_{b,3} + R_{c,3} + \cdots + R_{h,3}$

$\vdots$

$R_{a,m-1} + R_{b,m-1} + \cdots + R_{h,m-1} \leq \frac{n}{2}$

$\leq \log_2 n \quad \leq \log_2 n \quad \leq (h-1) \cdot \frac{n}{2}$

total $\leq n \cdot \log_2 n$

UNION operation amortized $\log n$

[a] [b] [c] [d] [e] [f] [g] [h]

UNION (e, f)

UNION (c, h)

UNION (c, a)

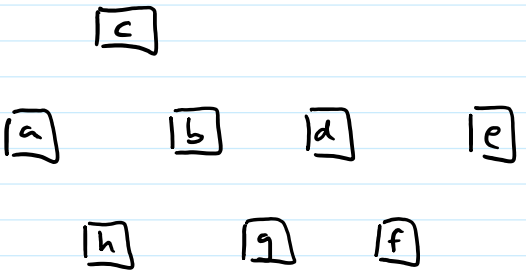
UNION (d, g)

UNION (c, e)

UNION (b, f)

UNION (e, g)

[a] [b] [c] [d] [e] [f] [g] [h]



Kruskal's Algorithm

n = # vertices
m = # edges

Kruskal's Algorithm: consider edges in order of ↑ weight — $O(m \log n)$
add edge if connects two different components of proto-MST

proto-MST: subset of some MST

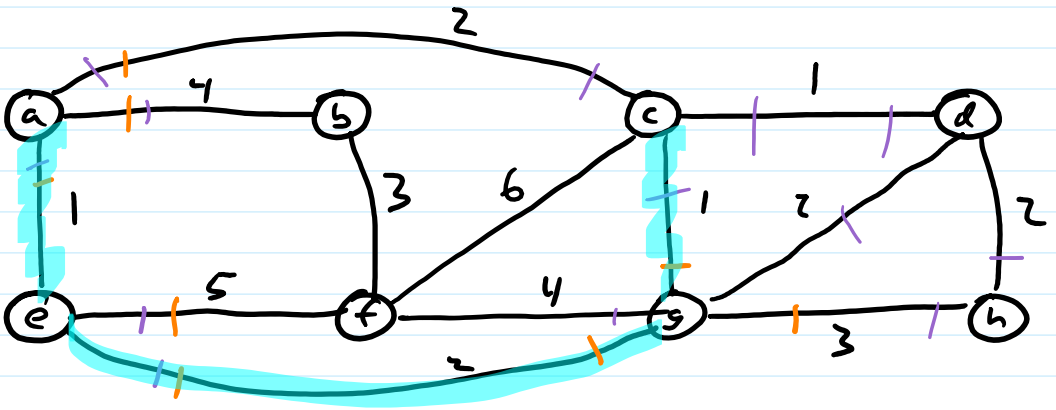
DFS? $O(n+m)$
for every edge, so total $O(n \cdot m + m^2)$ need to do better

$O(1)$

$T \leftarrow \emptyset$
sort edges in order of increasing weight — $O(m \log n)$
for each edge (u,v) — $O(m \cdot 1)$
  if u,v in different connected components — $O(n \cdot \log n)$ — FIND-SET$(u) \neq$ FIND-SET(v)
    $T \leftarrow T \cup \{(u,v)\}$ — UNION(u,v)

$O(m \log n + m + n \log n)$

Prim's Algorithm



a b c d e f g h

4/a ~~2/h~~ 1 5/e 2 3/g

1/g

Priority Queue

Priority Queue Implementations	build	$O(n)$ calls extract-min	$O(m)$ calls decrease-key	Prim TOTAL	$m \in \Theta(n)$ sparse	$m \in \Theta(n^2)$ dense
unsorted array	$\Theta(n)$	$\Theta(n)$ search entire array	$\Theta(1)$	$O(n^2 + mn)$	$O(n^2)$	$O(n^2)$
sorted array	$O(n \log n)$	$\Theta(1)$ wrap-around style remove 1st element	$O(n)$ find elt to change swap it to new loc.	$O(n + n \cdot m)$		
binary heap or balanced BST	$\Theta(n)$ $O(n \log n)$	$O(\log n)$	$O(\log n)$	$O(m \log n)$	$O(n \log n)$	$O(n^2 \log n)$
fib heap	$\Theta(n)$	$O(\log n)$	$\Theta(1)$ amortized	$O(n \log n + m)$	$O(n \log n)$	$O(n^2)$