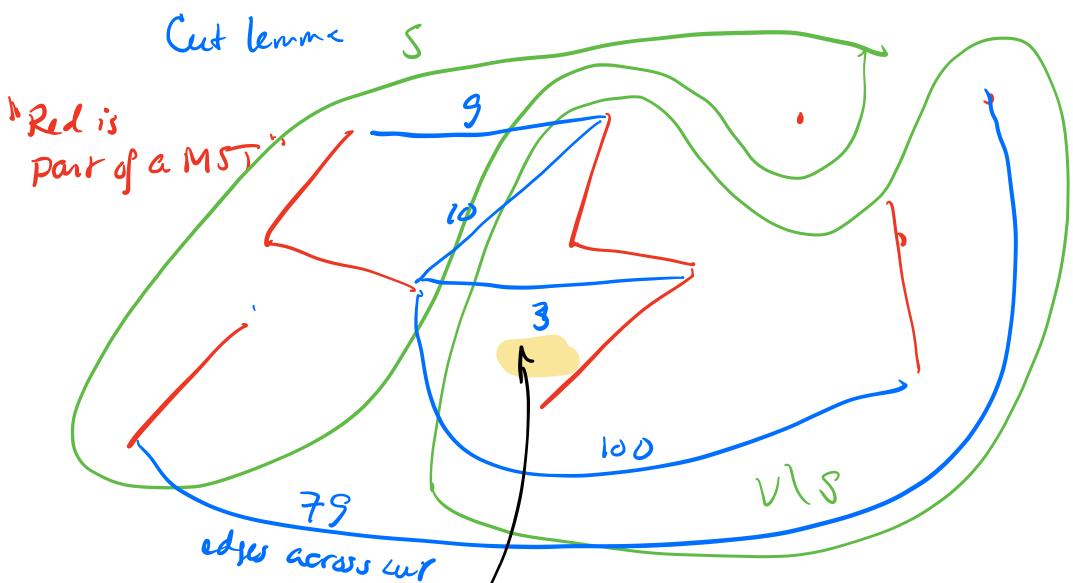
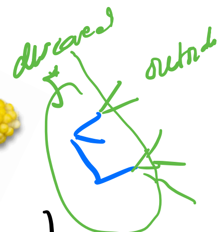
Minimum Spanning Trees (cont)



3 can be safely added to red edges + thus is still part of some MST

best connected comp then from v



Kruskal's algorithm (G, w)

Building a forest, each tree has a root

$A \leftarrow \emptyset$

for each $v \in V(G)$

Makeset (v) ←

Sort edges

Add lightest to heaviest if we can

for each edge $(u,v) \in E$ (in order)

do if Find $(u) \neq$ Find (v) ←

then $A \leftarrow$ add $[(u,v)]$

Union (u,v) ←

Prim's Algorithm (G, w, v)

for each $u \in V$

do key $(u) \leftarrow \infty$ (unseen)

$\pi(u) = \emptyset$

key $(v) \leftarrow 0$

while $Q \neq \emptyset$

do $u \leftarrow$ del min (Q) (cheapest)

for each $v \in V$ adjacent to u, do

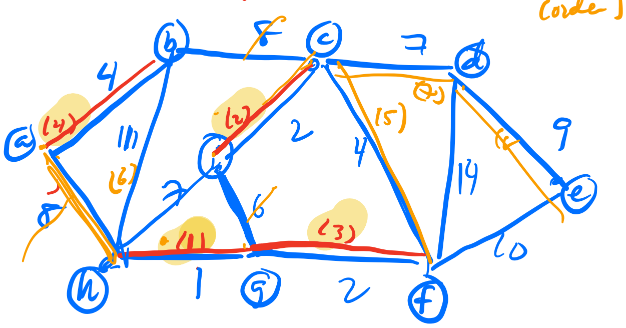
if $w(u,v) <$ key (v)

then $\pi(v) \leftarrow u$

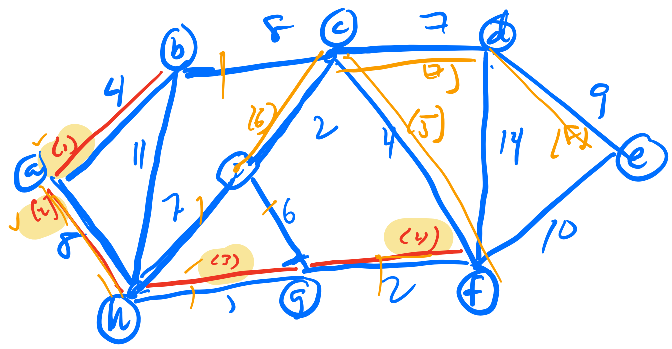
key $(v) \leftarrow w(u,v)$

} return A

First 4 edges of Kruskal : continue (code)



First 4 edges of Prim :



Data Structures

Prim : Priority queue.

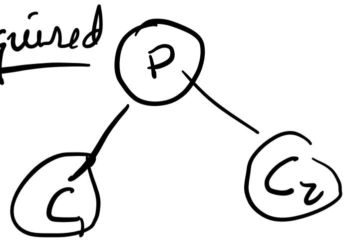
	insert	deletemin	remove	find min
unordered array 3 5 1 10 30 15	$O(1)$	$O(n)$	$O(1)$	$O(n)$
ordered array 1 3 5 10 15 30	$O(n)$	$O(1)$	$O(n)$	$O(1)$
* Binary heap	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(1)$

$O(m \log n)$ alg for Prim, Dijkstra

[$O(m + n\log n)$ using Fibonacci heap]

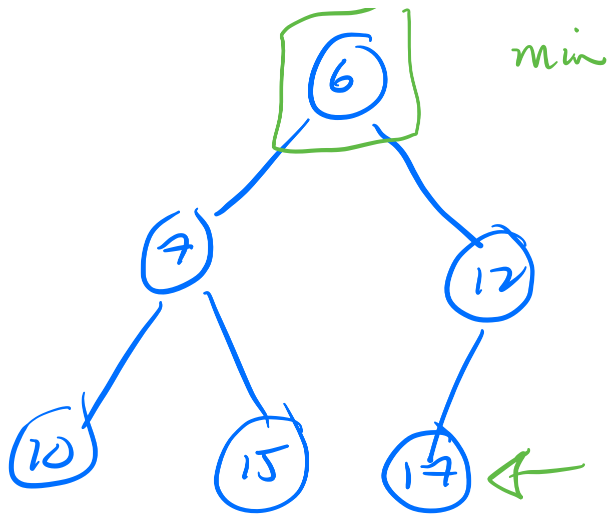
Minheap

Required



$p < c_1$ & $p < c_2$

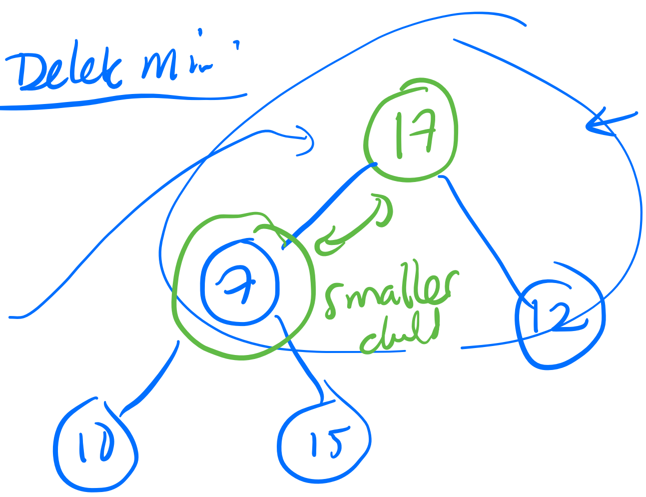
Invariant



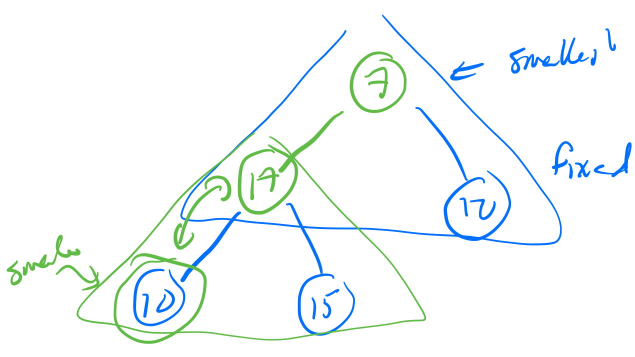
Delete min:

Shape invariant restored

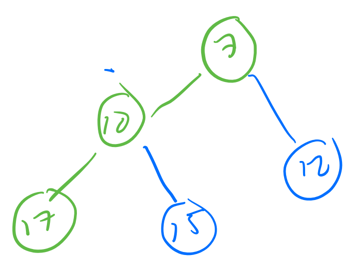
Value invariant



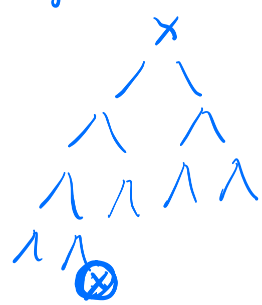
elevate final entry

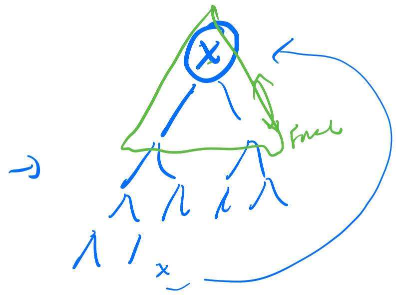


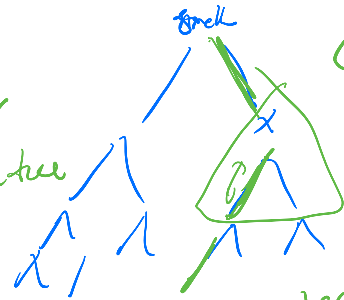
Restored value invariant



In general





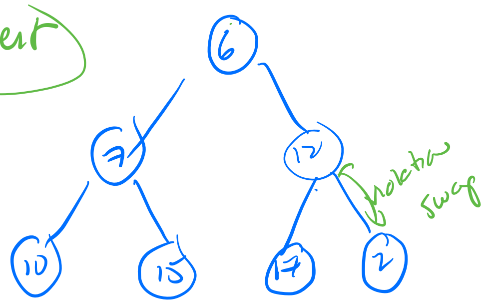


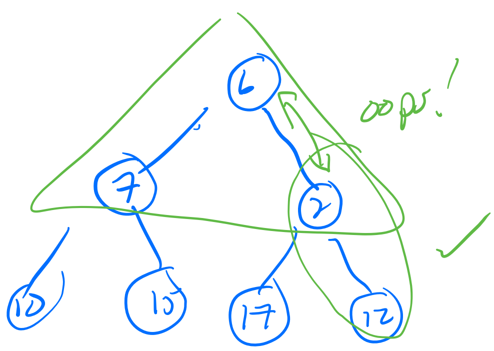
Do a swap at most once per level of the tree

$O(\log n)$

trickle down until invar. restored

Insert





Trickle up until invariant restored

$O(\log n)$ to insert

What about Kruskal?

(union-find data str)

sort: $|E| \log |E|$ time $m \lg m$

notice $\log m = O(\log n)$

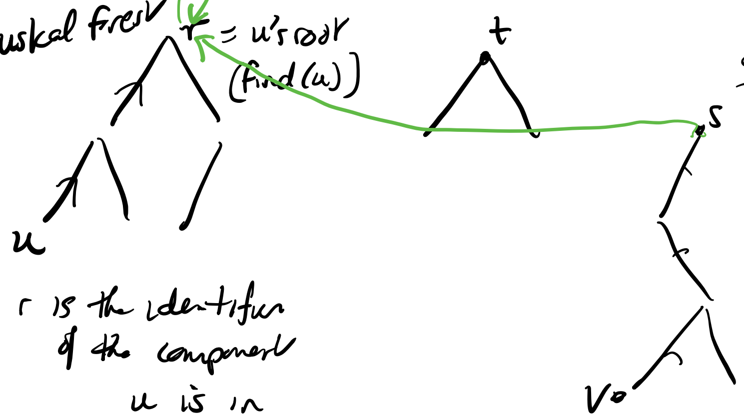
$= O(m \log n)$

maintain trees with roots (Kruskal forest

r = u's root (find(u))

same for Prim

s = find(v)



r is the identifier of the component u is in

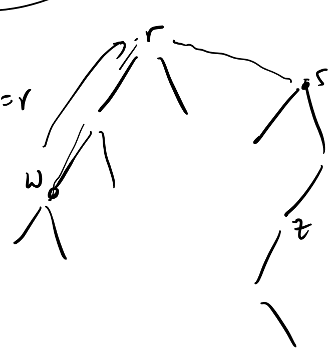
Def find(r) = r all roots point to themselves

Is find(u) $\overset{?}{=}$ find(v): find their roots

$r \overset{?}{=} s$ no

can add (u,v).

if find(u) = find(v) discard edge because it makes a cycle!



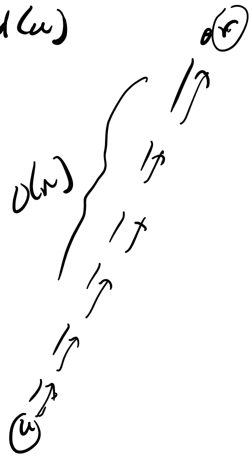
find(w) = r

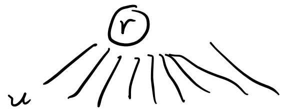
large tree will just r as the root,

find(z) = r

Problem:

find(u)





Overall time is

$O(m \log n)$

This can be improved

$O(\underbrace{m \log n}_{\text{sorting}} + \text{"almost linear"})$

Amortized time

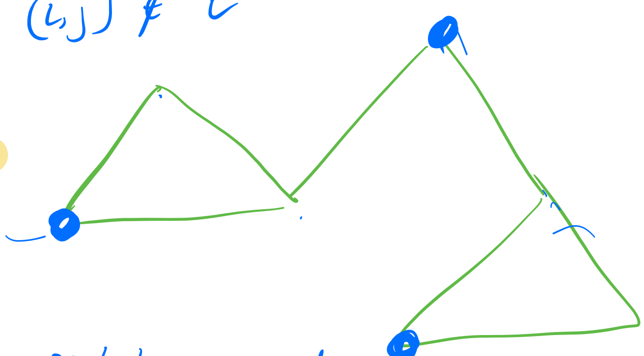
Independent sets

Given $G = (V, E)$

Def: $I \subseteq V$ s.t. $\forall\ i, j \in I$

$(i, j) \notin E$

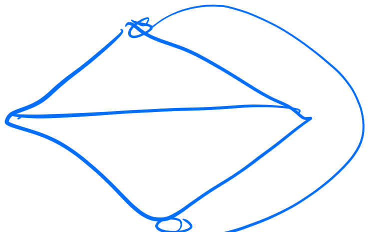
Maximal



Does every graph have an ind set?

$\emptyset$ is an I.S.

Single vtx is a I.S



Every graph has one, usually many

n vtcs

How many ind sets?

2^n

Maxim~~um~~al independent sets

We have an ind set & adding anything else will not be an ind set.

Every vtx is in I.S. or has a neighbor in I.S.

Maximum IS is the largest possible

Given G, is finding a maximal IS easy? Poly time?

Yes! Greedy alg works

Keep adding if you can until stuck

Maximum is not known. +

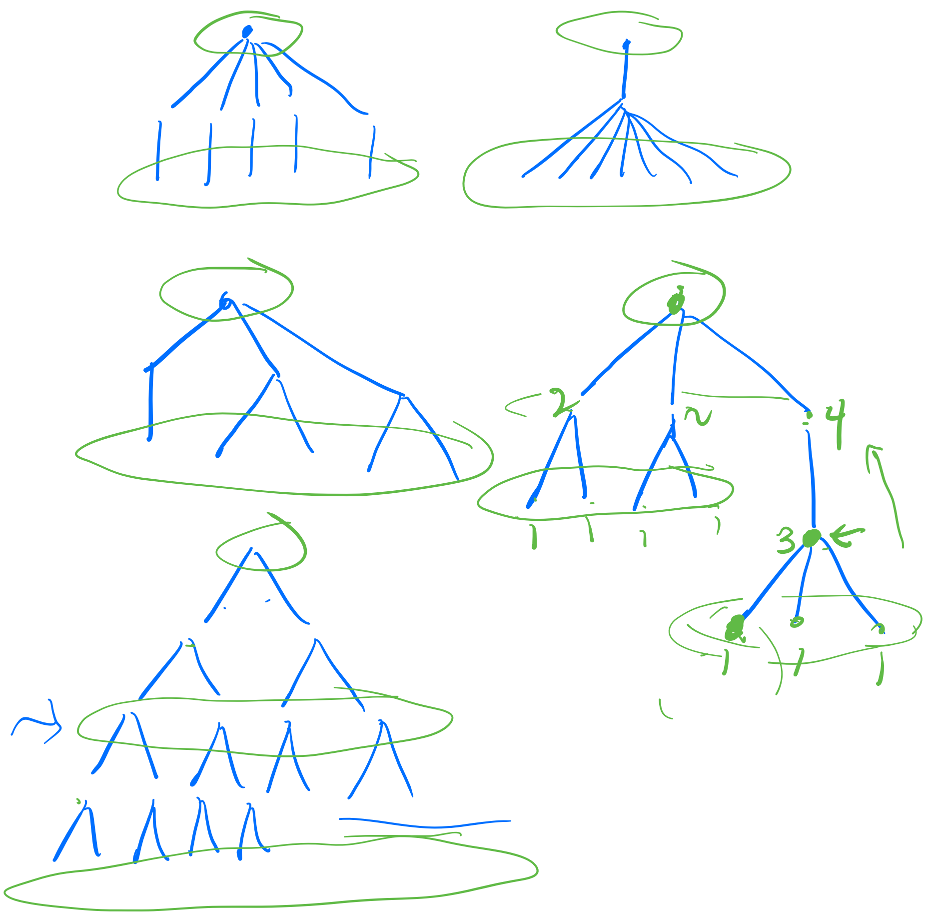
Is there an I.S. with size ≥ 100?

If we could solve this quickly

then we could find maximum using

binary search

If G is a tree then can solve I.S.



Start at leaves + decide max size IS of the subtree starting at v

Soln:

$$\max(v) = \max\left(1 + \sum_{u \text{ is a grandchild}} \max(u),\ \sum_{w \text{ is a child}} \max(w)\right)$$