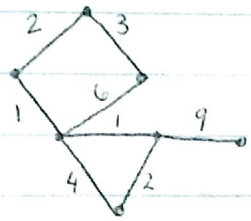
2/2/16

midterm next Tuesday (1½ hr)
no hw assigned this week

minimum spanning tree



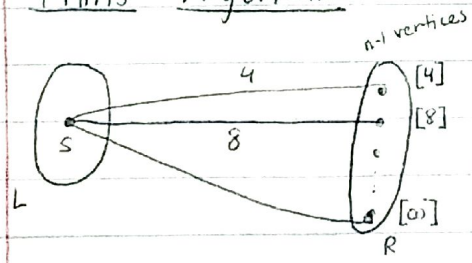

depends on topology as well as weight
connected
no cycles

partition: look at all edges that go between left and right (must have at least one edge otherwise not connected)

at least one MST w/ emin in it

contradiction: assume there is an MST does not have emin in it ⇒ if so, implies a cycle since MST has n-1 edges and now has n edges

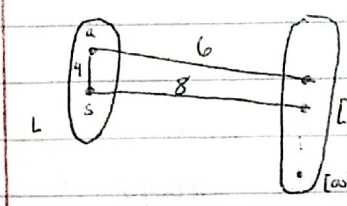
Prims Algorithm




any vertex connected s, must initialize it to the weight otherwise infinity if not directly connected
s is a part of the MST

runtime analysis

find the minimum and move it into L
must reconsider the neighbors of the vertex that was moved
+ if the new vertex and s are connected to the same vertex, take the minimum
+ only change the neighbors of the vertex that was moved!




you will continue moving vertexes n-1 times
⇒ since MST has n-1 edges

extract constant time
maintain minimum log n $\sum deg_i$
update vertices

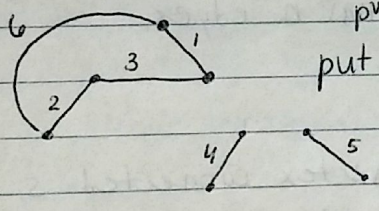
runtime analysis of Prims $O(e \log n)$

extracting a value from heap log n
inputting a value to heap log n
extract minimum (constant time) but updating/balancing heap is log n

Kruskal's Algorithm

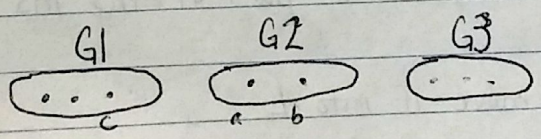
edge centric

sort edges based on weight
look at the minimum, include in MST
put it in one side of the partition and put everything else on the other side




6 can never be in MST

ex: given a set of groups w/ elements
prepare a data structure that answers the question if the elements are in the same group

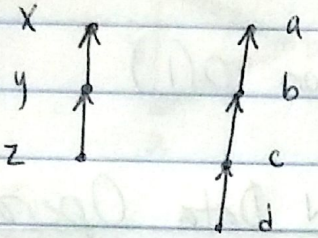



Data Find Operation

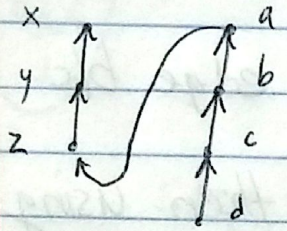
Union → merge G1 and G2 and name it G12
now a and c are in the same group
not possible to break up groups

Union Find Algorithm

decide to use a rooted tree as data structure

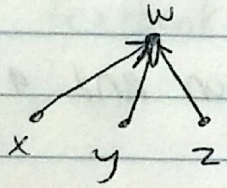



follow the elements all the way to the root, if the roots are the same then they are in the same group find (O(n))




Union: connect root of one tree to another tree
union: O(1)

want union/find to be balanced so change the shape of the tree

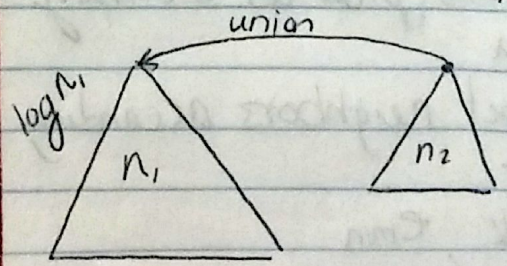



now find is O(1)
but union is O(n)

combine the two ideas so that the time complexity is between O(1) and O(n)

for each tree the height will be logarithmic

if height = h then take the smaller height tree and point to the larger tree




→ height of new tree does not change though vertices increase

$height_{new} = max(h_{left}, h_{right})$ except

if trees have equal size height goes up by 1

but number of vertices doubles

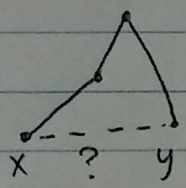
runtime: find $O(\log n)$, union $O(1)$

## Kruskal's Algorithm w/ New Data Operations

ask if the vertices belong to the same group

if they do, do not add that edge b/c that creates a cycle

continue adding edges and then using unions to merge groups

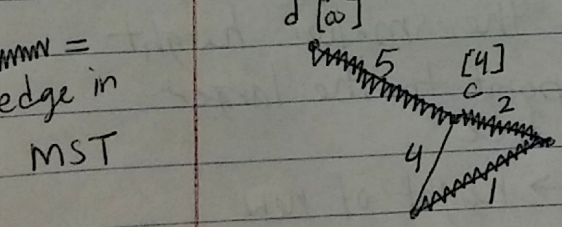



begin w/ n groups and finish w/ 1 group

for every edge must do a union and a find

sorting $O(e \log e)$
algorithm $O(\log n \cdot e)$
total: $O(e \log e + e \log n)$

ex: consider this graph




wwww = edge in MST

in Prims, pick an arbitrary vertex like a

label neighbors according to weight

pick $e_{min}$

look at neighbors of b (only c)
change c's 4 to a 2
d is unchanged