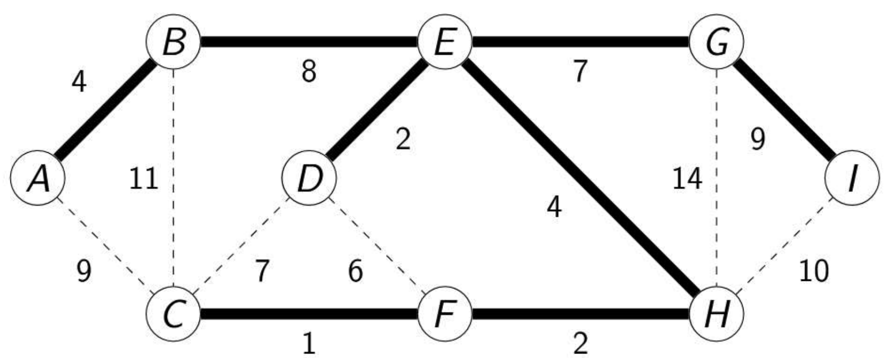
CompSci 161
Spring 2021 Lecture 25:
Greedy Algorithms:
Kruskal's Algorithm, Union-Find
Data Structure

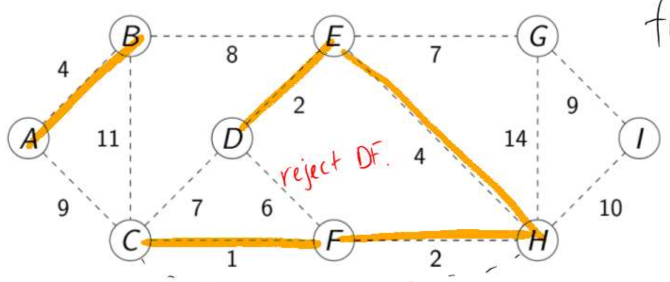
What is a Minimum Spanning Tree?



3

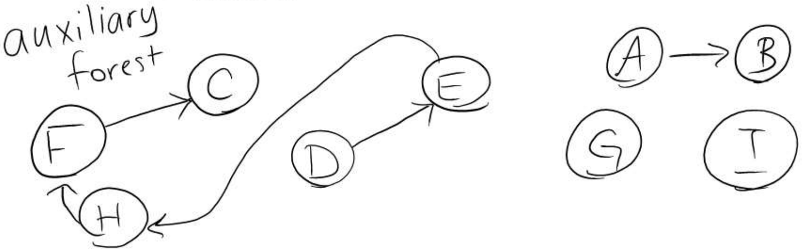
Finding a MST (Kruskal)

find(v):



- Sort edges in order (break ties arbitrarily)
- Consider each edge $e = (u, v)$ in order.
  - If u and v not in same connected component yet ← find(u) ?= f(v)
    - add e to T.

auxiliary forest



4

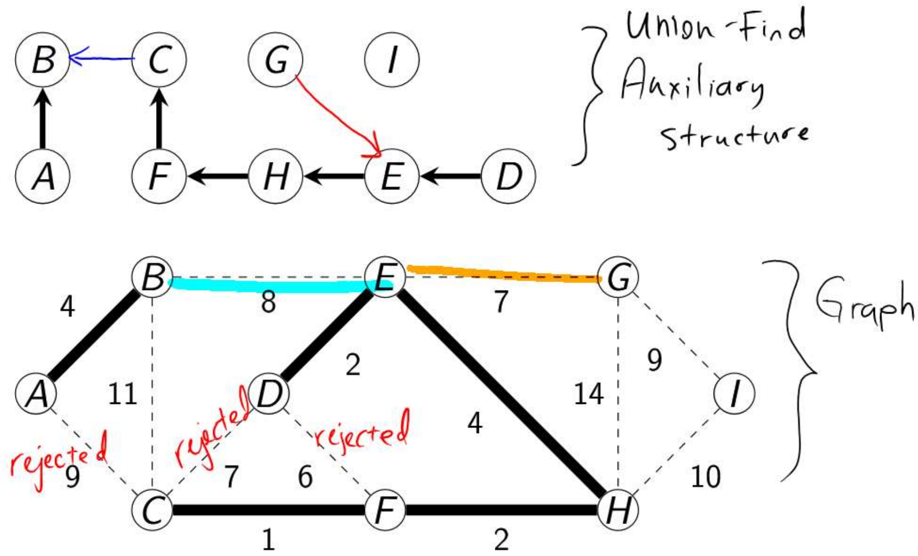
Union-Find Data Structure

- Data Structure informed by demand
- How do we know if we add an edge?
- We need to support the following:
  - Construct: there are n disjoint sets
  - Ask: "are these in the same set?"
  - Tell: "these two now are in the same set"
- Let's discuss TELL first.
- The first 5 edges added by Kruskal are:

(C,F)	(E,D)	(F,H)
(E,H)	(A,B)	

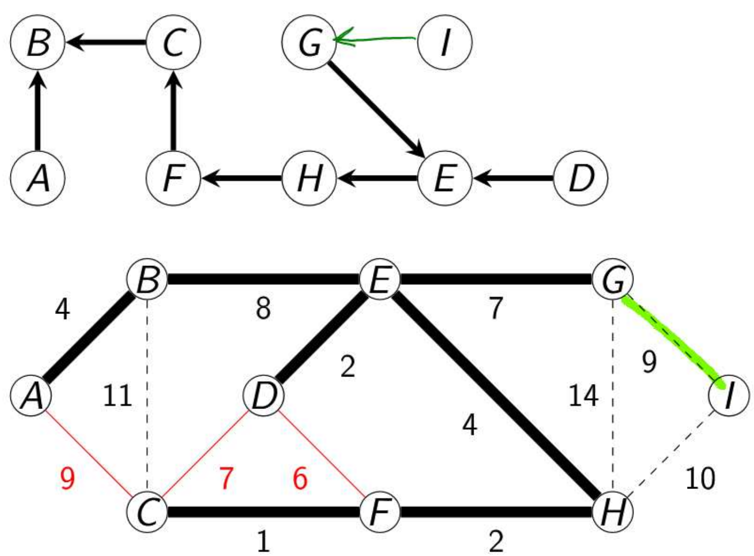
5

Continuing the structure



6

Continuing the structure



7 ## Running time for Operations

Union(A,B)
  $X \leftarrow$ find(A)
  $Y \leftarrow$ find(B)
  if $X \neq Y$ **then**
    X.parent $\leftarrow Y$
O(whatever find is)

find(A)
  if A.parent $\neq$ nullptr
  then
    return find(A.parent)
  return A
O(n) ~~potentially~~

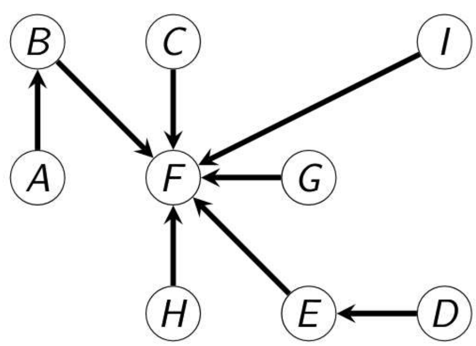
- If there are n elements, times?
- Can we improve the **worst-case** for one?

Track: how many nodes have this as the root?

"Union by rank"

(one w/ fewer reports).parent $\leftarrow$ (one w/ more reports)

8 ## Example

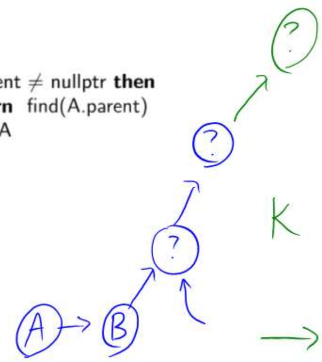


- This is the result of *Union by Rank*
- Ties are broken alphabetically
  Earlier letter $\rightarrow$ later letter (when tied)

9 ## Running time for Operations

Union(A,B)
  $X \leftarrow$ find(A)
  $Y \leftarrow$ find(B)
  if $X \neq Y$ **then**
    if X.count $>$ Y.count **then**
      Swap X and Y
    X.parent $\leftarrow Y$
    Y.count $+=$ X.count

find(A)
  if A.parent $\neq$ nullptr **then**
    return find(A.parent)
  return A



K hops to root

→ at least

2^k nodes in grouping

- If there are n elements, find is $\mathcal{O}(\log n)$
- Union takes time $2\times$ find $+ \mathcal{O}(1)$

10 ## Improving Find

- Can we improve **find** further?

find(A)
  if A.parent $\neq$ nullptr **then**
    A.parent $\leftarrow$ find(A.parent)
    return A.parent

  return A

D ~~→~~ E → F

Path compression

11
Path Compression

- Path compression change worst-case of find?

- Does path compression improve over time?

- Suppose we have m union and f find operations

$K = 2m + f$ Inverse Ackermann

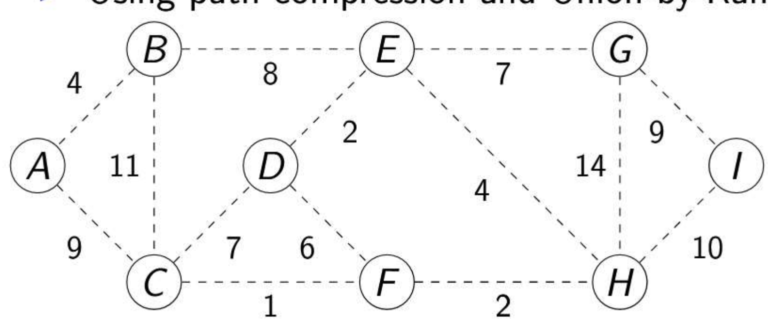
Total $O(K\alpha(K))$ total time

$\approx O(1)$ per find

12
Kruskal with U-F

$m = \#$ edges
$n = \#$ vertices

- Using path compression and Union-by-Rank



- Sort edges in order (break ties arbitrarily) $\Big\}\ \theta(m \log n)$
- Consider each edge $e = (u, v)$ in order.
  - If u and v not in same connected component yet
    - add e to T.

$\Big\}\ \approx \theta(m)$