Examples:
- Given prime \( p \), \( a \equiv b \mod p \)
  
  \( \text{Eg } p=5 ; \text{Partition: } \{0,5,10,15\} \}
  \{1,6,11,16\} \}
  \{2,7,12,17\} \}

- Given graph \( G \), vertices \( u,v \),
  \( u \equiv v \) if in same connected component

Partition: \( \{2,3,10\} \{1,7,5,9\} \{4\} \{6,7\} \)

Union-Find:
Given set \( S=\{1,2,...,n\} \) maintain a partition supporting ops:
Init: Each element in its own set
\{1,2,3,...,\{n\}

\( \text{Union}(s,t) : \) Merge two sets \( s \cup t \)
and replace with their union

\( \text{Find}(x) : \) Return the set containing \( x \)

Example: Suppose: \( \{1,3\}, \{2,6,8\}, \{3,4,7\} \)
\( \{s_1, s_2, s_3\} \)

\( \text{Union}(s_1,s_3) \rightarrow \{1,3,4,5,7\} \)
\( \text{Find}(x) \rightarrow s_i \)
\( \text{Find}(8) \rightarrow s_2 \)
\( \text{Find}(x) = \text{Find}(y) \iff x \equiv y \)

Equivalence Relation:
Binary relation over set \( S \) such that \( \forall a,b,c \in S \):
- reflexive: \( a \equiv a \)
- symmetric: \( a \equiv b \Rightarrow b \equiv a \)
- transitive: \( a \equiv b \equiv c \Rightarrow a \equiv c \)

Any equivalence relation defines a partition over \( S \).

A simple approach to finding is to trace the path to the root:

\[ \text{Set Simple-Find(\text{Element } x) } \]
\[ \text{while (parent [x] } \neq \text{ null)} \]
\[ x \rightarrow \text{parent}[x] \]

return \( x \)

Disjoint Set Union-Find I

Inverted-Tree Approach:
- Store elements of each set in tree with links to parent
- Root node is set identifier

\( \text{Eg } \{1,7,10\} \{12\} \{2,5,6,7,11\} \{8,9\} \)

Array-Based Implementation:
Assume: \( S=\{1,2,...,n\} \)
parent \[ 1..n \] , where parent[\[ i \]] is is parent index or \( O=\null \) if root

\( \text{Example: } \{3,0,0,9,1,2,3,2,0,12,0\} \)
How to Union?
- Just link one tree under the other
- How to maintain low heights?
  - Rank: Based on height of tree. Link lower rank as child

Set Union \( \text{Set}_s, \text{Set}_t \) 
if \( \text{rank}[s] > \text{rank}[t] \)

Recall: These are just array indices of roots

\[ \text{rank}[t] \leftarrow \max (\text{rank}[t], 1 + \text{rank}[s]) \]
return \( t \)

Init: All ranks \( \leftarrow 0 \)
All parents \( \leftarrow \text{null (0)} \)

Example:
\[ \{1,3,7,10\} \leftarrow \{12\} \leftarrow \{2,5,6,8,11\} \leftarrow \{4,9\} \]

Union \((9,12)\) \[ \text{rank}[9] = \max (1, 0+1) = 1 \]

Union \((2,3)\) \[ \text{rank}[3] = \max (2, 2+1) = 3 \]

Lemma: Assuming rank-based merging, a tree of height \( h \) has at least \( n \geq 2^h \) nodes. [\( n \) elts \( \Rightarrow \) height \( + \log n \)]

Proof: By induction on num. of unions

Basis: Single node, \( h = 0 \), \( 2^0 = 1 \) node

Step: Consider the last of series of unions. Let \( T' + T'' \) be trees to merge:

- Heights: \( h' + h'' \)
- Sizes: \( n' + n'' = n \)

By induction: \( n' \geq 2^{h'} \), \( n'' \geq 2^{h''} \)

Cases: \( h' = h'' \):

Final tree height: \( h = h' + h'' + 1 \)
Final size: \( n' + n'' = 2^h + 2^{h'} = 2^h + 2^{h'} \)

Case 2: \( h' < h'' \):

Final height: \( h = h'' \)
Final size: \( n' = n'' \cdot 2^{h' + h''} = 2^{h''} \cdot 2^{h' + h''} = 2^{h''} \cdot 2^{h' + h'' - 1} \)

Case 3: \( h' > h'' \) (symmetrical)
Path Compression:
- Whenever we perform find, short-cut the links so they point directly to root.
- This does not increase running big more than constant, but can speed up later finds.

Simple Union-Find performs a sequence of $m$ Union-Finds on set of size $n$ in $O(m \log m)$ time.

Theorem: (Tarjan 1975) After init. any seq of $m$ Union-Finds (with path compression) takes total time $O(m \alpha(m,n))$.

- Amortized time (average per op) is $O(\log m)$.
- Not bad - But can we do better?

Example:

Disjoint Set Union-Find III:

Digression: Ackerman's Function (1926)

for $i,j \geq 0$

\[
A(i,j) = \begin{cases} 
  j+1 & \text{if } i = 0 \\
  A(i-1,1) & \text{if } i > 0, j = 0 \\
  A(i-1, A(i,j-1)) & \text{otherwise}
\end{cases}
\]

Looks innocent, but it's a monster!

Obs: $\alpha(m,n) \leq 4$ for any imaginable values of $m,n$ ($m \geq n$)

From super big to super small:

Inverse of Ackerman:

$$\alpha(m,n) = \min \{ i \geq 1 \mid A(i, \lceil m/n \rceil) > \log m \}$$
Thu, Sep 8:

- Office hours on class Web page
- Prog. Assign 0 - due next Tue at 11:59pm
  (Don't wait until last minute)
- Challenge problem:
  - Add comment at top of your file `ExpandingStack.java`

```java
// I did the challenge
// Here's how: ....
```