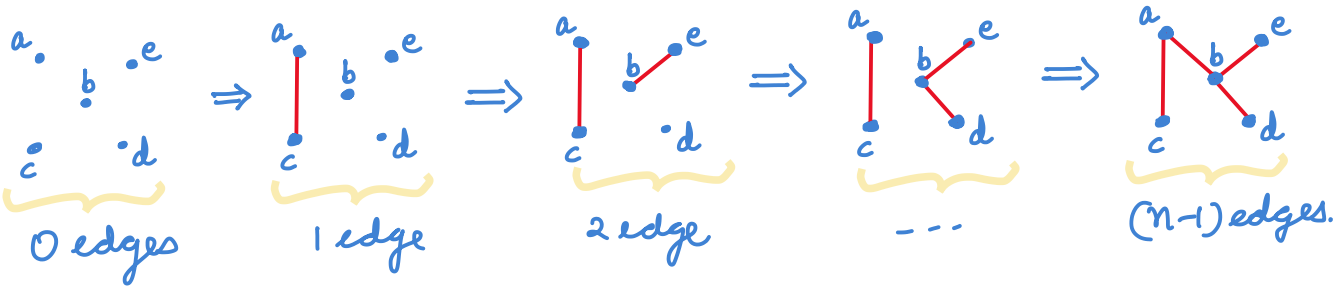
Operation – Join 2 trees
Query – check if x, y are in same tree

Union Find Problem

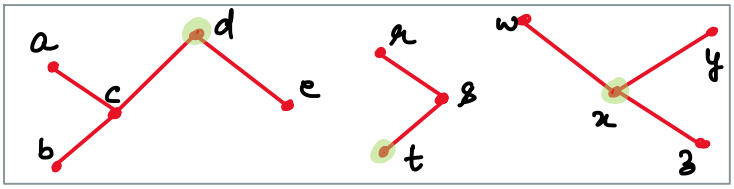
Suppose "F" is a forest growing with time.



Two operations

① Find (x) ← Points to one representative vertex in tree of (x).

Efficiently check if two vertices in same tree



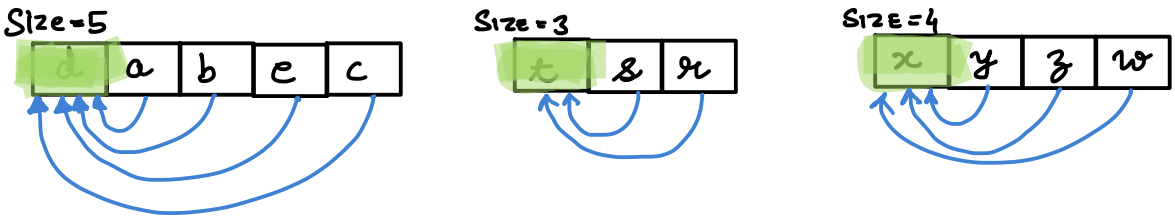
Find (a) = d Find (s) = t Find (w) = x

Observe: x & x' in different tree ⟺ Find (x) ≠ Find (x').

② Union (x, y) ← Merge tree of x, y (assuming Find (x) ≠ Find (y))

Union - Find - Data Structure (List Based)

Represent trees in forest "F" as link-lists.



* Each vertex x stores in Head (x): First element of the lis

⊛ Representative vertex 'x' stores in → size(x) = Size of Link list
→ last(x) = Pointer to last element of list.

Initialization

$\forall v \in V(G)$

Make List (v) :
Create a link list of size 1, & set
Head(v) = {v}
Last(v) = {v}
Size(v) = 1

Find(x)

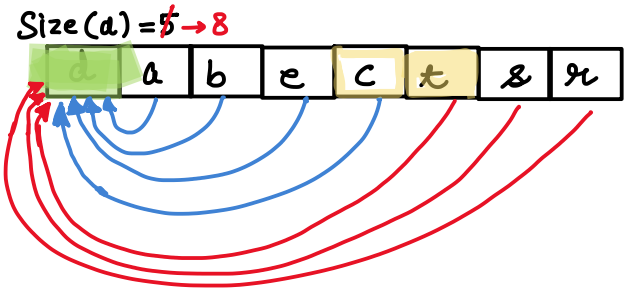
Report Head(x)

eg.

Find(x) = t

Union(x, y)

- Change Head(v), ∀ v in smaller
- Append (Merge) one list at end of
- Update size at Head
- Update last pointer.



Eg. Union(a, s)

① Compute d = Head(a), c = last(d), and t = Head(s).

② Set Next(c) = t.

③ Set Last(d) = Last(t), and Last(t) = Null.

④ Set Size(d) = Size(d) + Size(t), and set Size(t) = 0

⑤ Update Head(w), for each w in appended list to d.

Time complexity = O(size of appended list)

⇒ So we append smaller list.

⇒ Time Complexity of one "Union" = O(size of smaller list)

If $|L_1| \geq |L_2|$

Then,

$|L_1 \circ L_2| = |L_1| + |L_2| \geq 2|L_2|$

Ques. How many times Head(v) can change

Ans $(\log_2 n)$

Whenever Head(v) changes, then |list(v)| dou

Total Complexity :

- changing of ALL head ← O(
- changing size
- changing last pointer

} O(1)

Algo (Kruskal's)

① Sort edges in non-decreasing order of weights

$wt(e_1) \leq \dots \leq wt(e_m)$

② Set $T = (V, \emptyset)$

③ For each $v \in G$: Create a link list containg v

Head(v) = v

Last(v) = v
Size(v) = 1

④ For $i = 1$ to m:

Let x_i and y_i be endpoints of e

If Find(x_i) ≠ Find(y_i)

- Add e_i to T.
- Union(x_i, y_i)

⑤ Return tree T.

$O(n$

Union Find on Wiki

Another algo

Find(x) — $O(\log^* n)$

Union(x, y) — $O(\log^* n$

$\log(m) \leq \log(n^2) = 2\log n =$

Correctness:

Let $\bar{e}_1 \cdots \bar{e}_{n-1}$ are edges in T.

... MST of G with e

Hypothesis H(i) : ∃ MST of G with ...

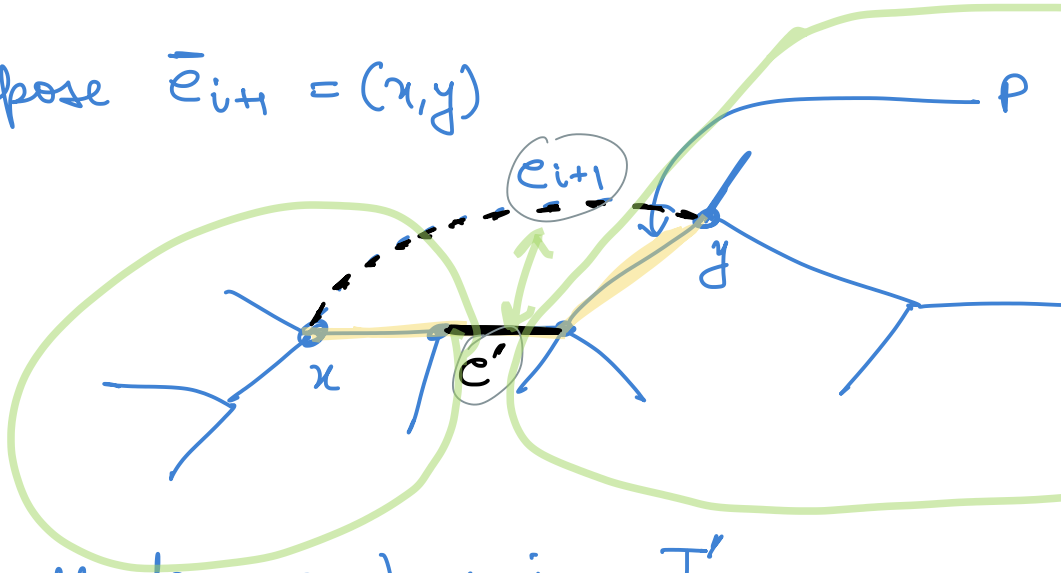
H(i) ⇒ H(i+1)

Take a MST T' of G with edges $\bar{e}$

(a) If $\bar{e}_{i+1} \in T'$ ⇒ H(i+1) holds

(b) If $\bar{e}_{i+1} \notin T'$

Suppose $\bar{e}_{i+1} = (x, y)$



P = path from x to y in T'

CLAIM: Edges(P) $\not\subseteq \{\bar{e}_1, \ldots, \bar{e}_i\}$

Let e' be edge in P not lying in $\{\bar{e}_1 \ldots \bar{e}_i\}$

Observe → $\{\bar{e}_1 \ldots \bar{e}_i \; \bar{e}_{i+1}\}$ ⊆ E

→ $\{\bar{e}_1 \ldots \bar{e}_i \; e'\}$ ⊆ E

By our MST algo, $wt(e_{i+1}) \leq wt(e')$.

$((T' \setminus e') \cup (e_{i+1}))$ ← a spang for
at most wt(

So, $T'' := ((T' \setminus e') \cup e_{i+1})$ is a MST

$wt(T'') \leq wt(T')$	T'' is MST