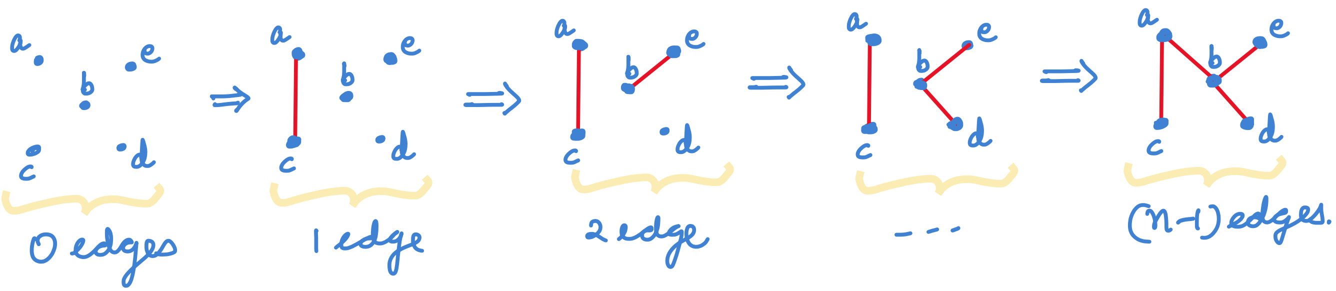
Lecture 05

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

Union Find Problem

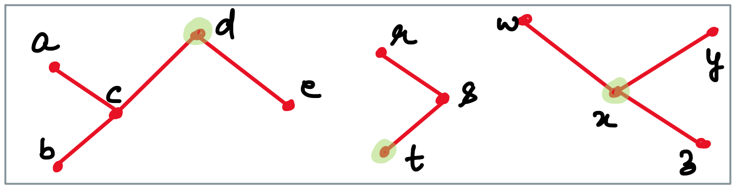
Suppose "F" is a forest growing with time.



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

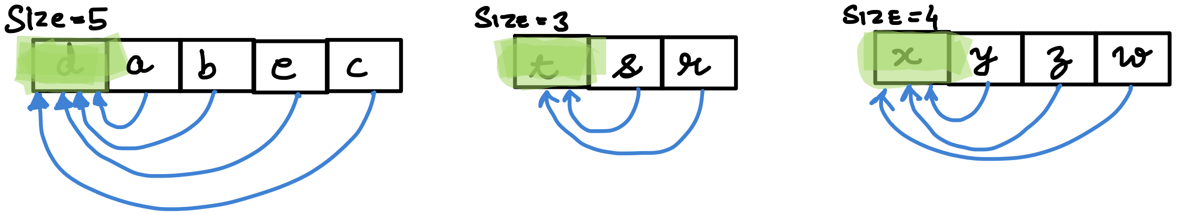
Observe: x & x′ in different tree ⟺ Find (x) ≠ Find (x′).

② Union (x, y) ← Merge t... (assuming Find (x) ≠ Find (y))

② Union(x,y) ← Merge tree of x,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 list.

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

Initialization

∀ v ∈ 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)

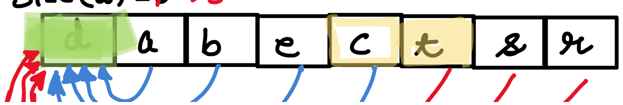
eg.

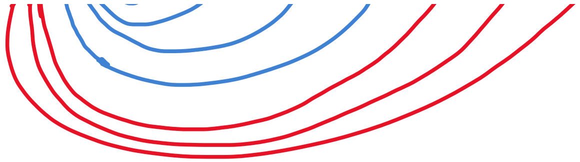
Find(r) = t

Union(x,y)

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

Size(d) = $\cancel{5}$ → 8





Eg. Union (a, s)

① Compute $d = \text{Head}(a)$, $c = \text{Last}(d)$, and $t = \text{Head}(s)$.

② Set $\text{Next}(c) = t$.

③ Set $\text{Last}(d) = \text{Last}(t)$, and $\text{Last}(t) = \text{Null}$.

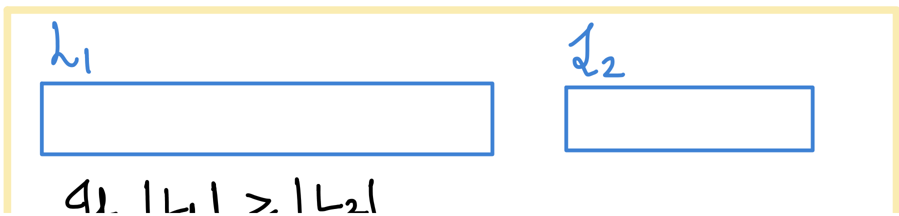
④ Set $\text{Size}(d) = \text{Size}(d) + \text{Size}(t)$, and set $\text{Size}(t) = 0$

⑤ Update $\text{Head}(w)$, for each w in appended list to d.

$$\boxed{\text{Time complexity} = O(\text{size of appended list})}$$

$\Rightarrow$ So we append smaller list.

$\Rightarrow$ Time Complexity of one "Union" $= O(\text{size of smaller list})$



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?

<u>Ans</u> $(\log_2 n)$

Whenever Head(v) changes, then $|list(v)|$ doubles.

<u>Total Complexity</u> :

- changing of ALL head ← $O(n \log n)$.
- changing size
- changing last pointer

(changing size, changing last pointer) } $O(1)$ per union operation

<u>Algo (Kruskal's)</u>

(1) Sort edges in Non-decreasing order of weights } $O(m \log n)$

$wt(e_1) \leq \cdots \leq wt(e_m)$ }

② Set $T = (V, \emptyset)$

③ For each $v \in G$: Create a link list contang v of size 1

Head $(v) = v$

Last $(v) = v$

Size $(v) = 1$

} $O(n)$

④ For $i = 1$ to m:

Let x_i and y_i be endpoints of e_i

If $\text{Find}(x_i) \neq \text{Find}(y_i)$ } $O(m * 1)$ time

- Add e_i to T.
- $\text{Union}(x_i, y_i)$

} In total $\boxed{O(n \log n)}$ time

⑤ Return tree T.

$O(m \log n)$ to find MST.

Union Find on Wiki

Another algo

Find (x) — $O(\log^* n) \ll O(\underbrace{\log \ldots \log}_{c \text{ times}} n)$

Union(x, y) — $O(\log^* n)$

$$\boxed{\log(m) \leq \log(n^2) = 2\log n = O(\log n)}$$

Correctness:

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

Hypothesis $H(i)$: $\exists$ MST of G with edges $\bar{e}_1 \ldots \bar{e}_i$

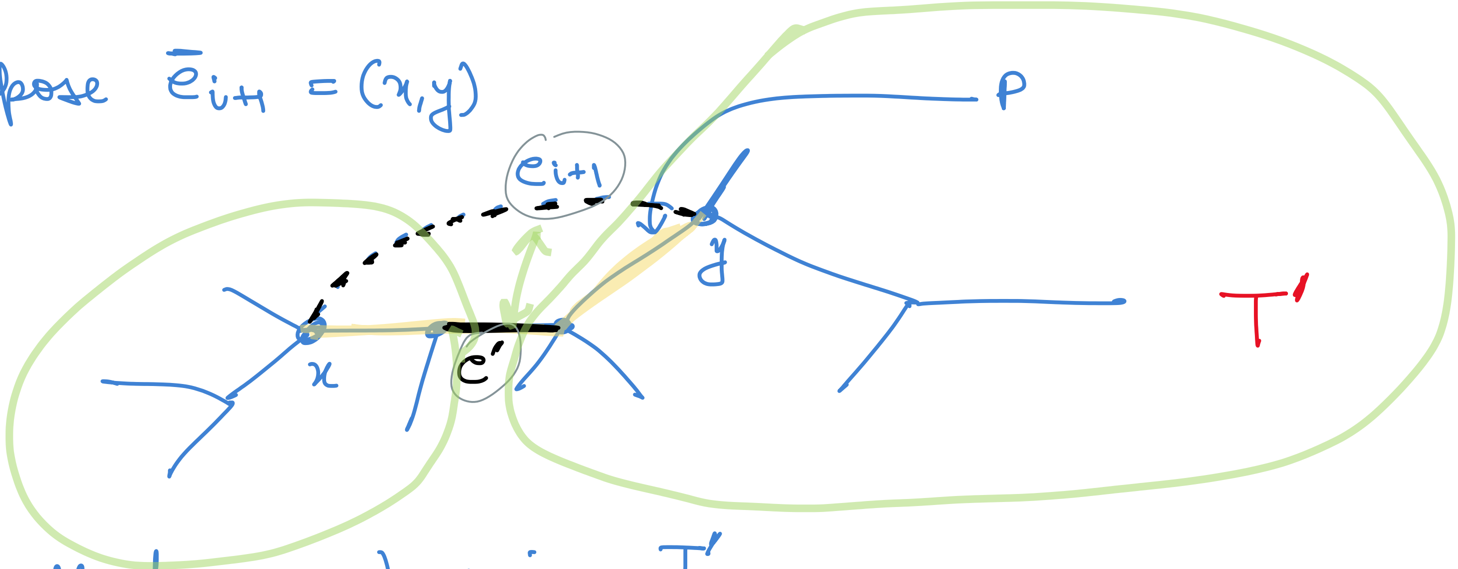
$H(i) \Rightarrow H(i+1)$

Take a MST T' of G with edges $\bar{e}_1 \ldots \bar{e}_i$

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

⊛ <u>If $\bar{e}_{i+1} \notin T'$</u>

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



P = path from x to y in T'

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

$\boxed{\text{Proof: } \bar{e}_1 \ldots \bar{e}_i\ \bar{e}_{i+1} \text{ is acyclic \& we know } P \cup e_{i+1} \text{ is a cycle}}$

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

$\{\bar{e}_1 \ldots \bar{e}_i\ \bar{e}_{i+1}\}$ — Do not form a cycle

Observe $\longrightarrow \{\bar{e}_1 \ldots \bar{e}_i \; e'\} \subseteq E(T')$ — Do not form cycle

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

$((T' \setminus e') \cup (e_{i+1})) \leftarrow$ a spang forest of weight at most $wt(T')$.

So, $T'' := ((T' \setminus e') \cup e_{i+1})$ is a MST of G, havg $\bar{e}_1 \ldots$

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