Union-Find

Given n sets, each of size one:

$$A_1 = \{1\},\ A_2 = \{2\}, \ldots, A_n = \{n\}.$$

Process a sequence of operations, where each operation is one of:

Union (A, B, C): Set $C = A \cup B$; $A = \phi$; $B = \phi$

Find (x): return the name of the set that contains x.

The sequence consists of

$n-1$ Union operations

m Find operations

(in any order)

We are interested in the total time to process any such sequence.

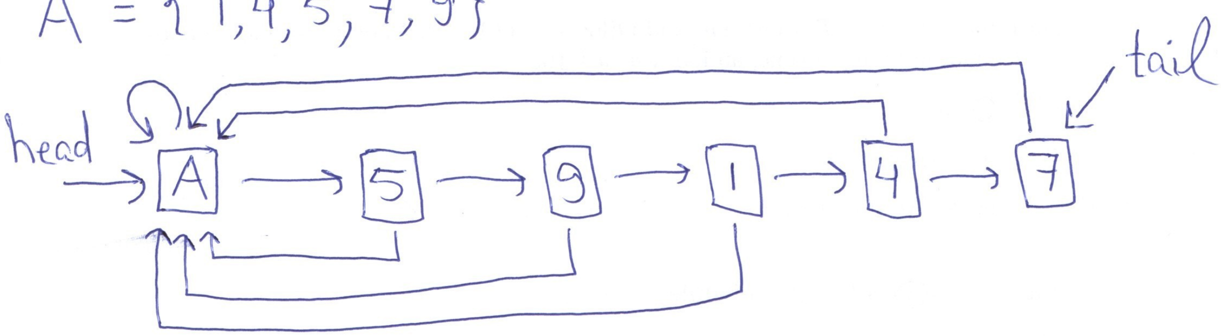
Store each set in a list:

* first node stores the name of the set.
* each other node stores one element of the set
* each node u stores two pointers:

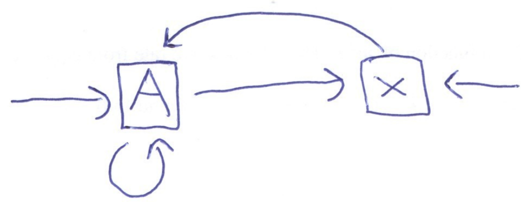
  $next(u)$: next node in the list

  $back(u)$: first node in the list

$A = \{1, 4, 5, 7, 9\}$

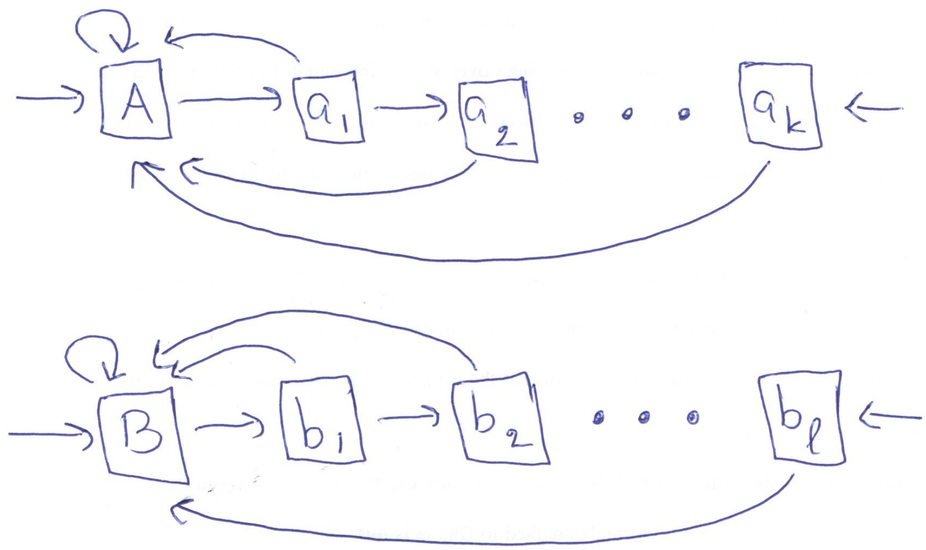


Start: for each set $A = \{x\}$:

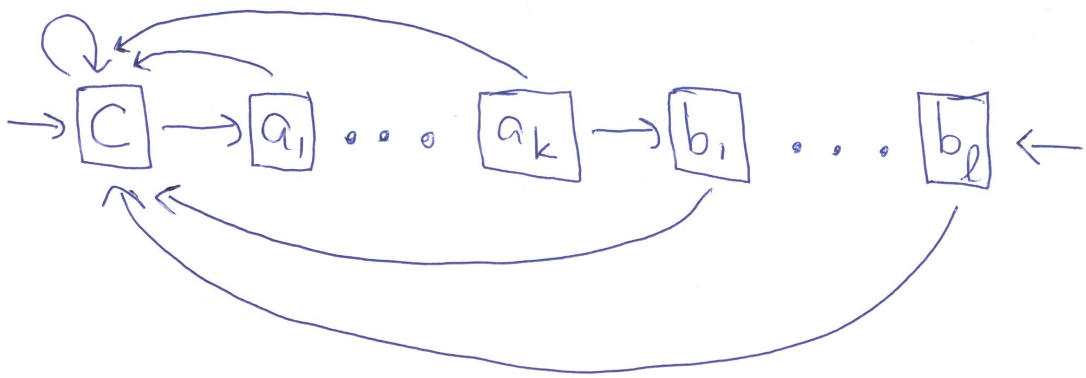


Total time $= O(n)$

Union (A, B, C) :



Combine the lists for A and B, give each element in the B-list a pointer to the head of the new list, change the name in the head of the new list from A to C :



Time $= O(1+\ell) = O(\ell) = O(\text{size of } B)$.

Find(x): follow the back pointer from the node storing x to the head of the list; return the name stored at the head.

Time $= O(1)$.

Example:

Union	Time
$\{1\}, \{2\}$	1
$\{3\}, \{1,2\}$	2
$\{4\}, \{1,2,3\}$	3
$\vdots$	$\vdots$
$\{n\}, \{1,2,\ldots,n-1\}$	$n-1$
	+
	$\Theta(n^2)$

Better solution:

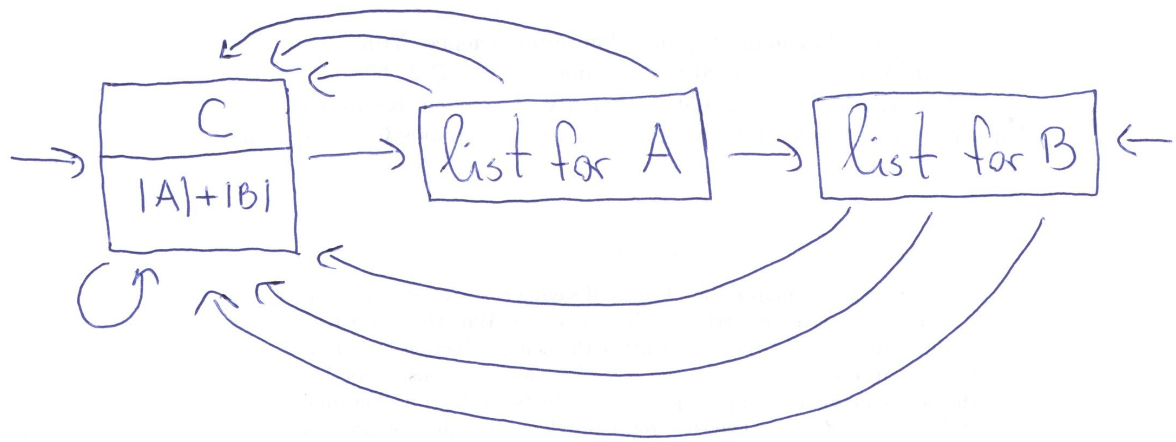
for each list: head stores → name of the set

→ size of the set

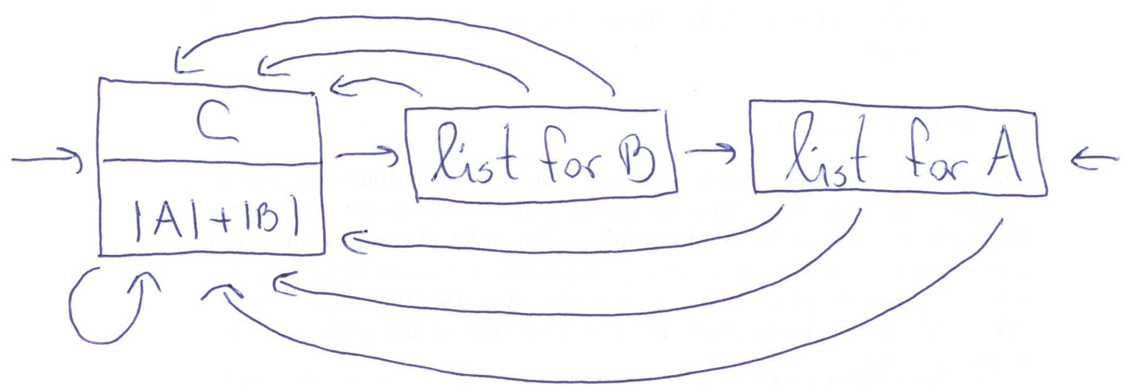
Find(x) takes O(1) time, as before.

Union(A, B, C):

if $|A| \geq |B|$:



if $|A| < |B|$:



$$\text{Time} = O(\min(|A|, |B|))$$

$$= O(\text{number of back-pointers that are changed}).$$

What is the total time for a sequence of n-1 Union operations:

$$\text{total time} = \text{total number of back-pointer changes}$$
$$= \sum_{x=1}^{n} \text{total number of times back}(x) \text{ is changed.}$$

Consider an element x. How many times do we change back(x):

Start: x is in a set of size 1.

First time back(x) is changed:

the set containing x is merged with a set of size ≥ 1.

$\therefore$ new set containing x has size ≥ 2.

Second time back(x) is changed:

the set containing x is merged with a set of size ≥ 2.

$\therefore$ new set containing x has size ≥ 4.

Third time back(x) is changed:

the set containing x is merged with a set of size ≥ 4

$\therefore$ new set containing x has size ≥ 8.

etc., etc.

Since there are n elements: back(x) is changed

$\leq \log n$ times.

$\therefore$ total time for n-1 Union operations

$= O(n \log n)$.

<u>Conclusion</u>: Any sequence of n-1 Union and

m Find operations:

$O(m + n \log n)$ time.

Minimum Spanning Trees

$G = (V, E)$, undirected, connected graph,
each edge $\{u, v\}$ in E has weight $wt(u, v)$.

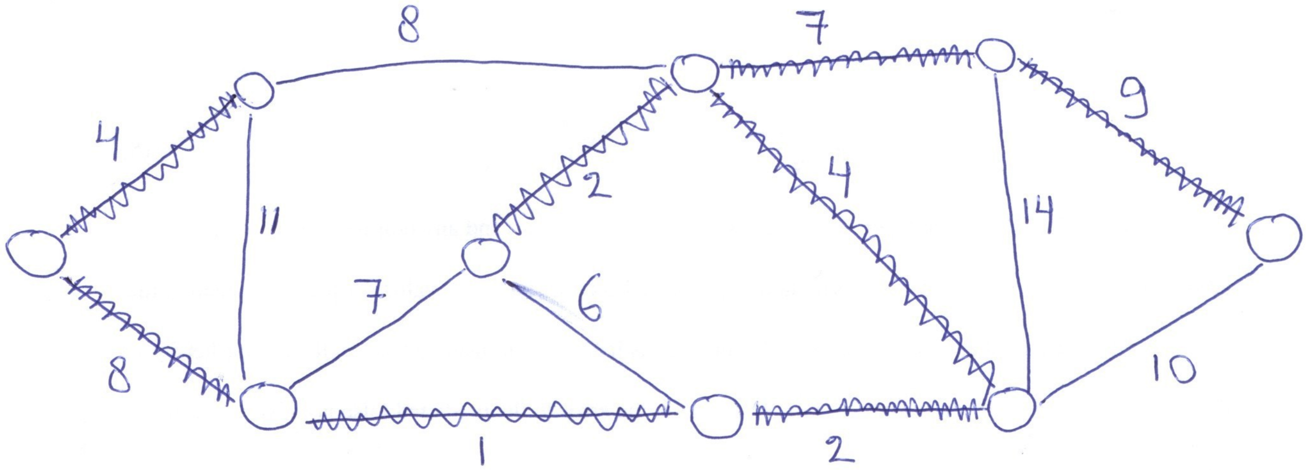
Compute a subgraph G' such that

* the vertex set of G' is V,
* G' is connected, and
* $\text{weight}(G')$ = sum of the weights of the edges in G'

is minimum.

<u>Claim</u>: G' must be a tree (i.e., connected and no cycles).

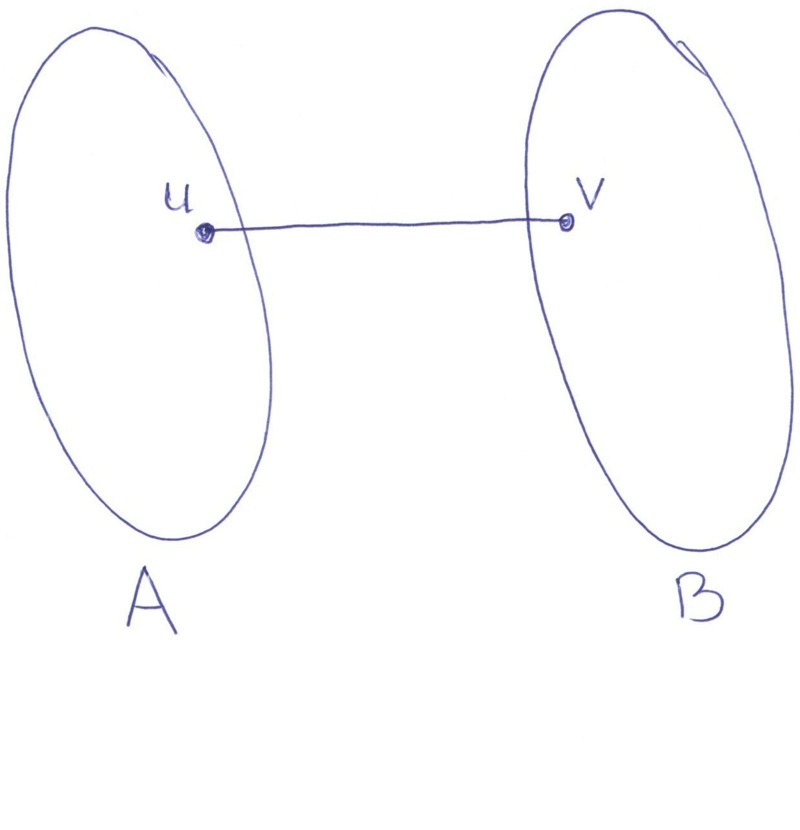
G' is called a <u>minimum spanning tree</u> of G.



Lemma: Split V into A and B.

$\{u,v\}$: shortest edge connecting A and B.

Then: there is an MST of G that contains the edge $\{u,v\}$.

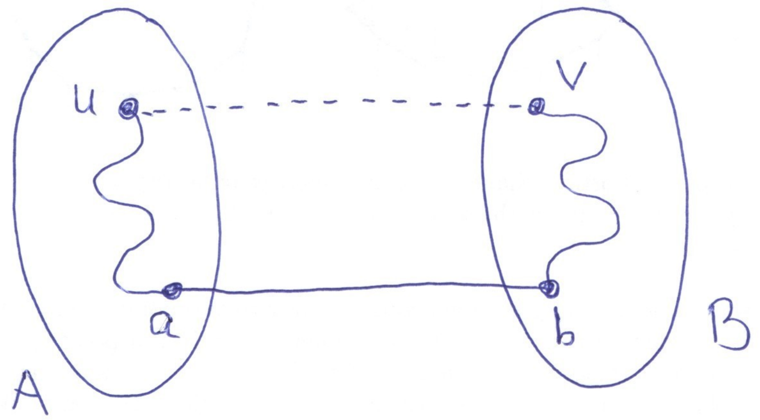


Proof: Let T be an MST of G.

If $\{u,v\}$ is an edge in T: done.

Assume $\{u,v\}$ is not an edge in T.

T is connected $\therefore$ there is a path in T between u and v. This path contains an edge $\{a,b\}$ with $a \in A$ and $b \in B$.



Define $T' = T$ minus $\{a,b\}$ plus $\{u,v\}$

Observe: T' is a tree.

$$\text{weight}(T) \leq \text{weight}(T') \quad // \, T \text{ is MST}$$

$$= \text{weight}(T) - wt(a,b) + \underbrace{wt(u,v)}_{\leq wt(a,b)}$$

$$\leq \text{weight}(T).$$

$\therefore$ $\text{weight}(T') = \text{weight}(T)$

$\therefore$ T' is an MST that contains the edge $\{u,v\}$.

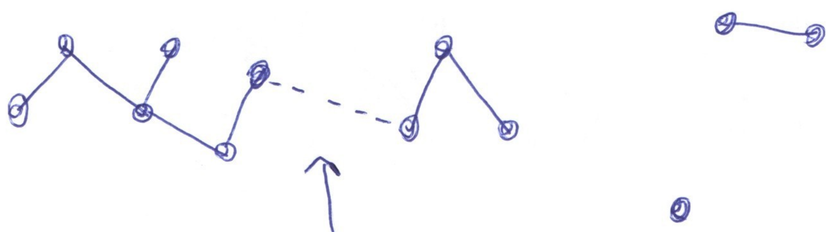
$\square$

Kruskal (1956):

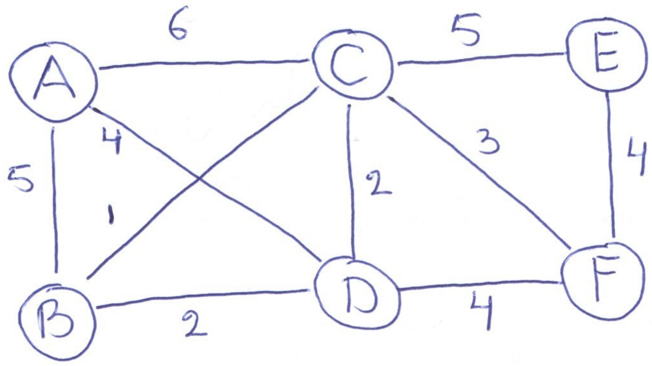
Approach: Maintain a forest. In each step, add an edge of minimum weight that does not create a cycle.

Start: forest = one tree for each vertex

One iteration:



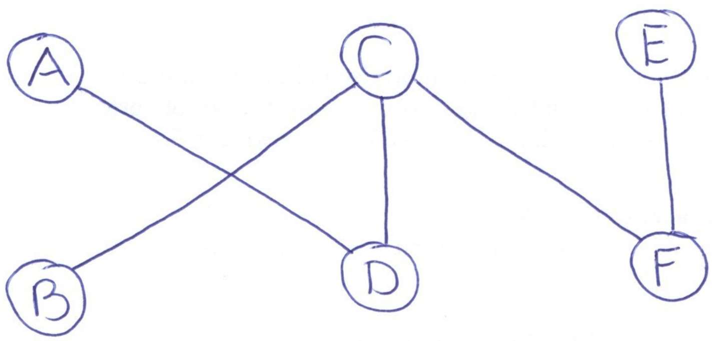
Combine 2 trees using an edge of minimum weight.



Sort edges by weight:

BC, CD, BD, CF, DF, EF, AD, AB, CE, AC

Illustrate the algorithm.



<u>Kruskal</u>: // $G=(V,E)$, $V=\{x_1,\dots,x_n\}$, $m=|E|$

Sort the edges of E by weight: $e_1, e_2, \dots, e_m$;

for $i=1$ to n: $V_i=\{x_i\}$; $T_i=\phi$;

for $k=1$ to m:

let u_k and v_k be the vertices of e_k;

i = index such that $u_k \in V_i$;

j = index such that $v_k \in V_j$;

if $i \neq j$: $V_i = V_i \cup V_j$; $V_j = \phi$;

$T_i = T_i \cup T_j \cup \{\{u_k, v_k\}\}$; $T_j = \phi$

Running time:

Sort: $O(m \log m) = O(m \log n)$, because $m \leq \binom{n}{2}$.

first for-loop: $O(n)$.

second for-loop:

store each T_i as a linked list: $O(n)$ total time to maintain these lists.

how to store the sets V_i: Union-Find structure.

$$\left.\begin{array}{ll} \text{Find: } & 2m \text{ times} \\ \text{Union: } & n-1 \text{ times} \end{array}\right\} \begin{array}{l} \text{total time} \\ = O(m + n\log n) \end{array}$$

Total time: $O(m\log n) + O(n) + O(m + n\log n)$

$= O(m\log n)$

↳ $m \geq n-1$ because G is connected.

<u>Conclusion</u>: Kruskal's algorithm computes MST in $O(m\log n)$ time.