COL 702 Lecture 17 Sept 24

Designing efficient implementation of Kruskal's algorithm

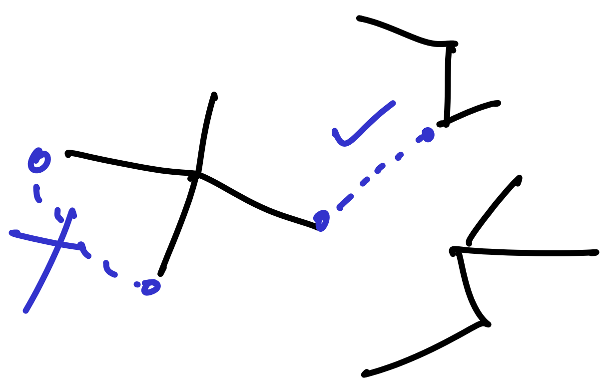
1. Assume elements are presorted

2. 2a Check if $T \cup \{e\}$ remains independent

if so $T \leftarrow T \cup \{e\}$ 2b

Cycle test

Union the edge to the forest



If the next edge e connects two trees, then no cycle else it induces a cycle

$e = (u,v)$

If u, v belong to different trees then add else discard

Find(v) : Return the tree T_i s.t. $v \in T$

if Find(u) $\neq$ Find(v) then add

Union$(e=(u,v))$: Since u and v belong to different trees, now we must join them into a single tree.

$$\boxed{\text{We need a suitable data structure that represents a forest, i.e. a set of trees and that supports Find and Union opns}}$$

$T_1 \quad T_2 \quad T_3 \quad T_4 \quad \cdots \quad T_k$

$V(T_1) \quad V(T_2) \quad V(T_3) \quad \cdots \quad V(T_k)$

subsets of vertices : disjoint

1. Find returns the label of the tree
2. Union joins T_i, T_j to produce a new tree T_k

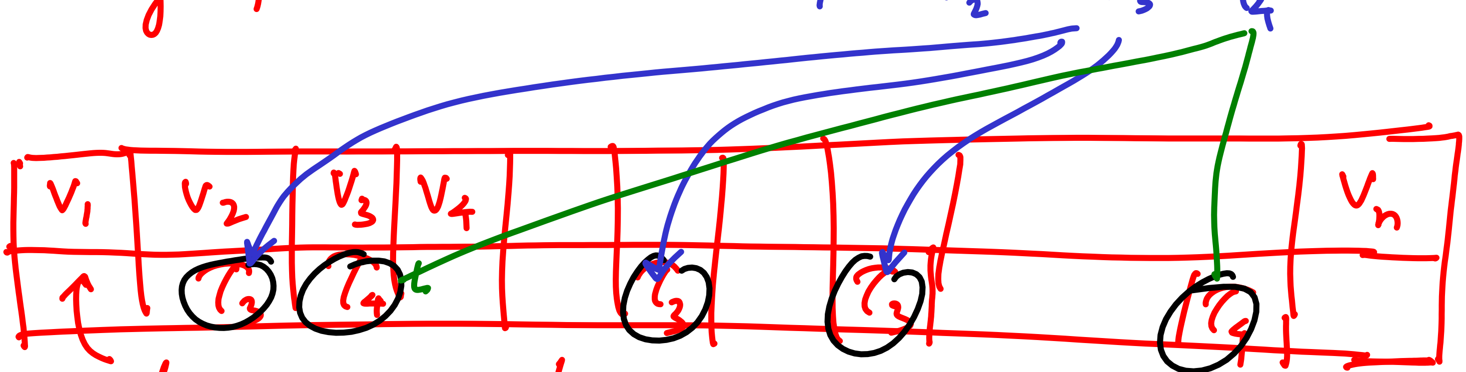
In Kruskal's algorithm, we will perform

1. $\leq 2m$ FIND opns

2. $\underline{n-1}$ UNION opns, starting with n singleton sets

Total Running : $O(m \cdot \text{cost of FIND}) + O(n \cdot \text{cost of Union})$

Array for vertices τ_1 τ_2 τ_3 τ_4 .



: FIND takes $O(1)$-time

: UNION $(\tau_3, \tau_4 \to \tau_x)$ We need to change the labels of the vertices that belong to τ_3, τ_4

How many labels would change?
That is the cost of UNION

Let us change the labels of only one of the trees, say $T_4 \leftarrow T_3 \cup T_4$

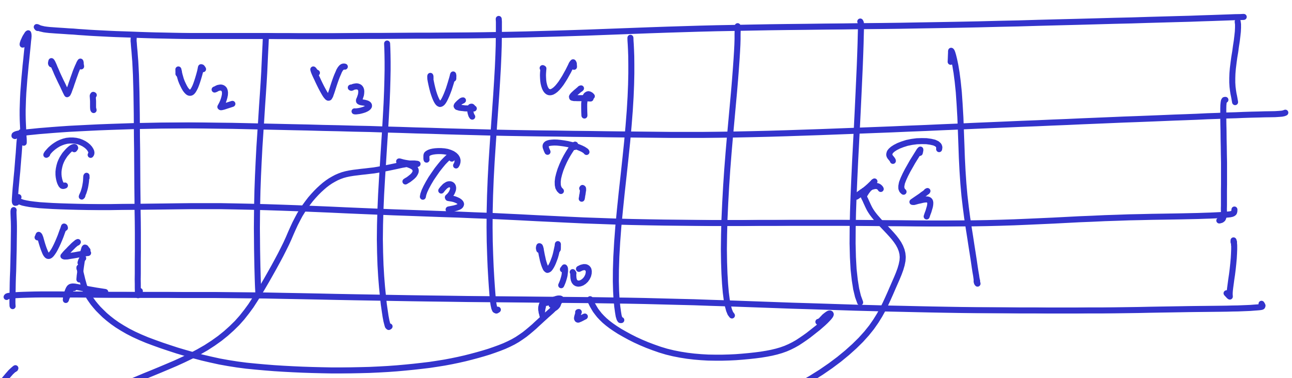
$T_3 \leftarrow T_3 \cup T_4$

Choose $\min\{|T_3|, |T_4|\}$

Still could be $\Omega(n)$

?? How do we know smaller T_3, T_4

How do we locate the vertices of same tree



Union(T_3, T_4)

size(T) should be maintained

Cost of union is proportional to the size of the smaller tree

Worst case calculations yield

$$O(m + n \cdot n) = O(m+n^2)$$

$$= O(|E| + |V|^2)$$

If graph is dense $|E| \sim |V|^2$

What about sparse graphs

"ideal perfm" $O(|E| + |V|)$

QUESTION: How many times does the label of a vertex change?

Say $n(v_i)$ = # times the label of v_i changes

Total cost of all UNIONS $= \sum_i n(v_i)$

Claim $n(v_i) \le O(\log n)$

$\Rightarrow$ Total time for all UNIONS

$= O(n \log n)$

$\Rightarrow$ Kruskal's algorithm takes $O(m + n \log n)$ using this data structure