Graphs

V vertex set

E edge set

$$\sum_{v \in V} \deg(v) = 2|E|$$

Simple graphs : no self loops

only 1 edge bet a pair of vertices

Directed graphs :  "Still simple"

Path : $v_1\ v_2\ \cdots\ v_k$ cannot visit same vertex twice

cycle : $v_1\ v_2\ \cdots\ v_k\ v_1$ path with only initial, final vertices same

Tree (undirected graph) Proof ?

connected graph on n vertices $\Leftrightarrow$ graph is a tree

$+\ |E| = n-1$ $\Leftrightarrow$ def

connected + no cycles

Directed graph with no directed cycle

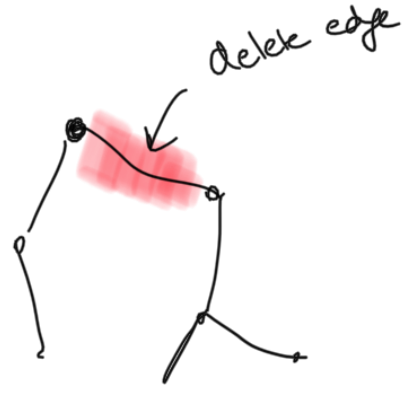
: "directed acyclic graph" (dag)

Proof Graph is a tree $\Rightarrow$ $|E| = n-1$

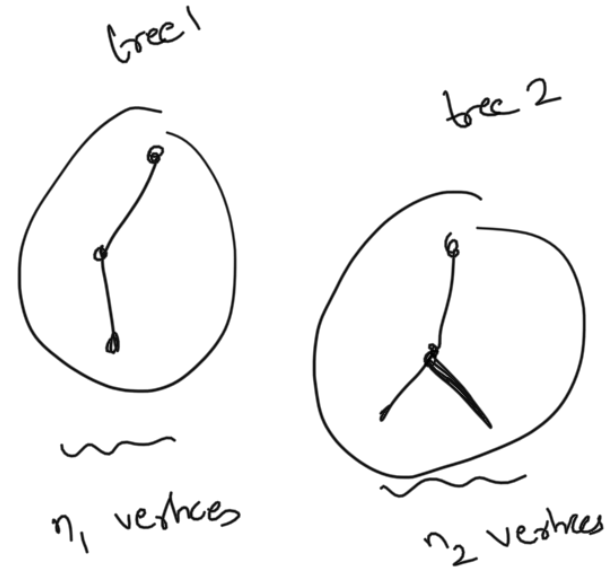
$|V| = n$

Induction on $|V|$

G



$|V| = n$



$n_1 + n_2 = n$

$n_1, n_2 < n$

$n_1 - 1$ edges

$n_2 - 1$ edges

$\therefore$ overall

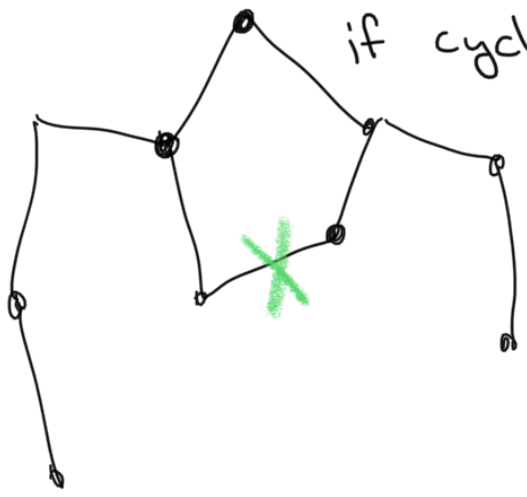
$n_1 + n_2 - 2 +$ [redacted]

$= n_1 + n_2 - 1$

$= n - 1$ edges

Graph connected, $|V| = n$, $|E| = n-1$ $\Rightarrow$ graph is a tree

Pf



if cycle → Remove edge

:

till graph is tree

$|V| = n$

so $n-1$ edges

but $|E| = n-1$ to start with

so couldn't have deleted edges

(ie) no cycles to start with

Spanning tree of graph G

touches all vertices of G using subset of edges

Goal Find min. cost spanning tree

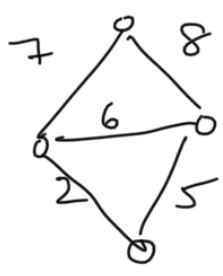
(where edges of G have weights)

Kruskal's Algorithm

(I) Sort edges by weigh

2 5 6 7 8

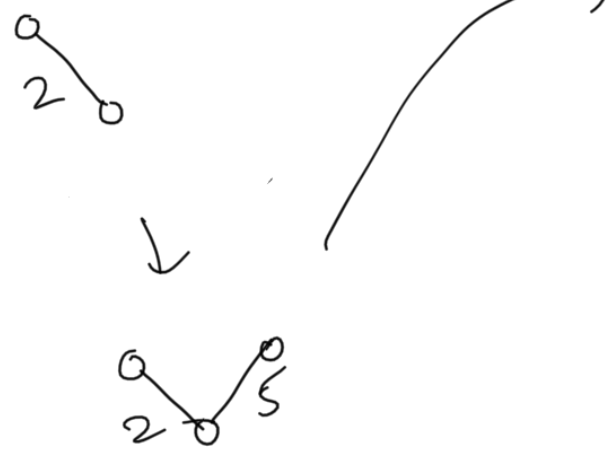
G

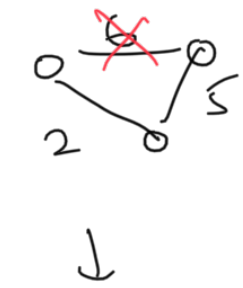


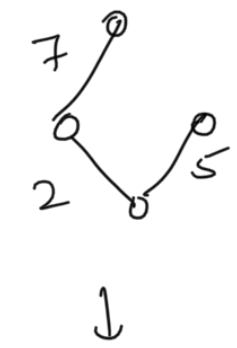
(II) Insert edges in ascending order

→ if you create cycle, don't add that edge

(ex)

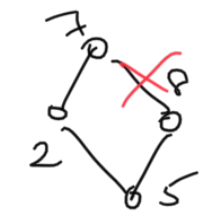




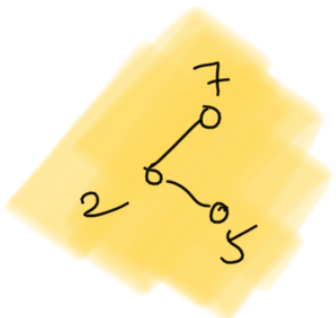


To detect cycles

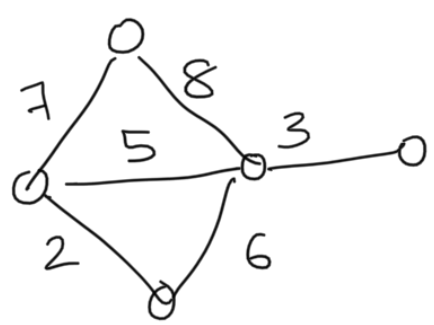
- mark vertices that have been added.
- if new edge is let marked vertices, it will create cycle



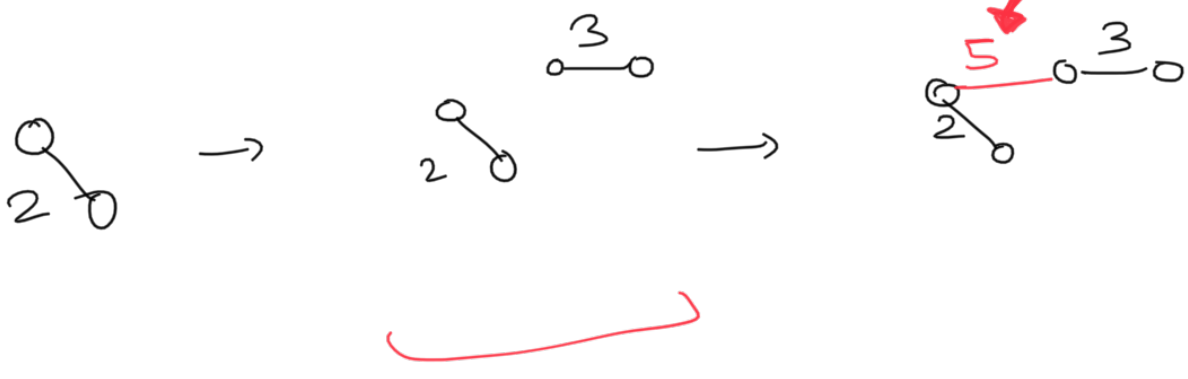
Solution



BUT



2, 3, 5, 6, 7, 8

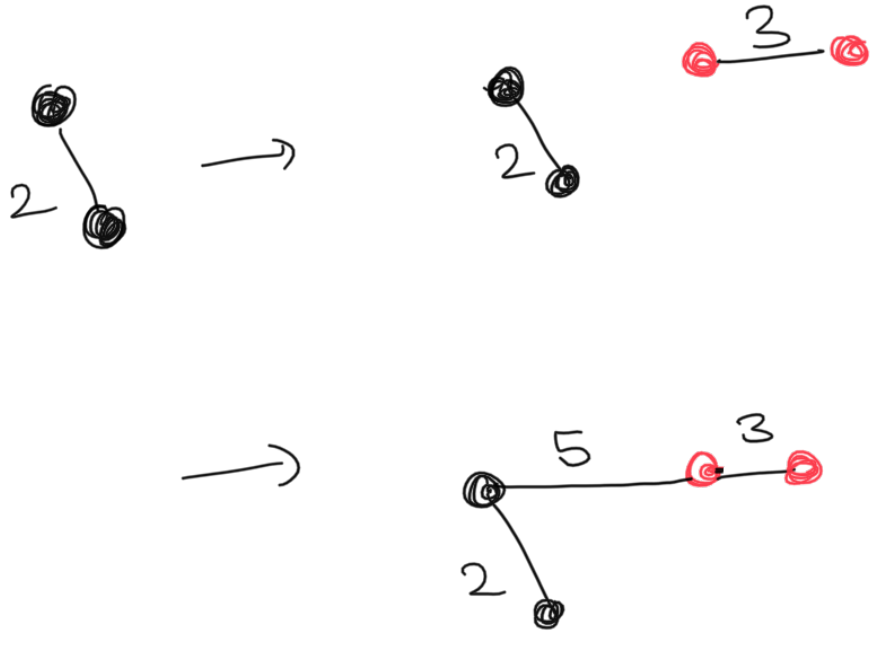


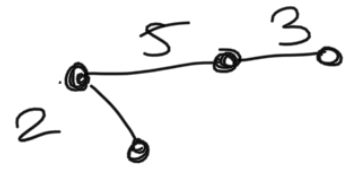
it doesn't create cycle

but our algo will mark it as creating cycle

Issue:
this is a forest, not a tree

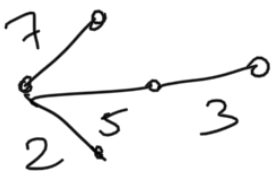
Fix:
- Use different colours to mark each component
- An edge that connects marked nodes of diffrt colour can be added
- Then merge the 2 colours into 1 (⇝ new single component)



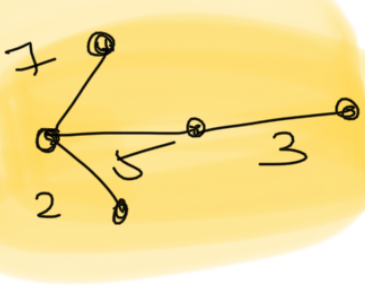


→ 6 can't be added, as cycle created

→



→ 8 can't be added as cycle created



What data structure to implement Kruskal's algorithm?

"Union find" : maintains a set partitioned into connected components

with 2 operations

Find: (1) given a vertex, efficiently finds which connected component it belongs to

Union: (2) given 2 connected components, merges them efficiently

next time