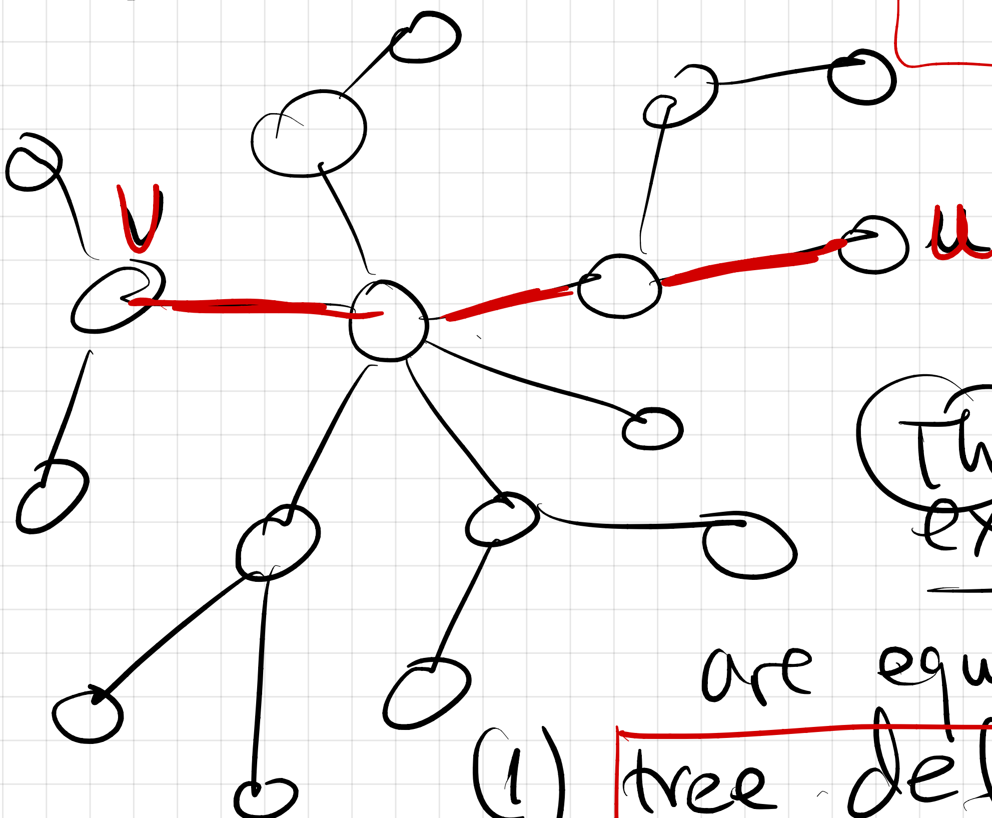
Tree $T = G(V, E)$ = tree $\begin{cases} \text{connected} \\ \text{no cycles} \end{cases}$ &



exercise 1: $|E|$ in a tree is precisely $|V| - 1$

Th **exercise 2** following statements are equivalent:

(1) $\boxed{\text{tree def}}$: connected + no_cycles

(2) any two vertices (u, v) connected with unique path

(3) T minimal connected (remove any edge $\Rightarrow$ disconnect)

(4) T $\boxed{\text{max}}$ acyclic (add any missing edge $\Rightarrow$ cycle)

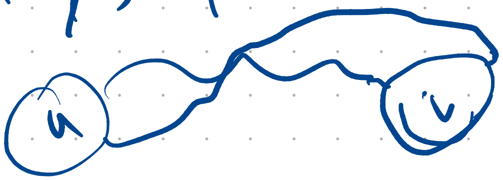
(5) connected & $|E| = |V| - 1$ || (6) acyclic & $|E| = |V| - 1$

proof $1 \Leftrightarrow 2 \Leftrightarrow 3 \Leftrightarrow 4$

(1⇒2) tree def ⇒ $\forall u,v \in V$ $\exists$ path(u,v) unique $u \rightsquigarrow v$

tree def ⇒ connected ⇒ $\exists$ path(u,v)

assume(hyp) path $u \rightsquigarrow v$ is not unique ⇒ $\exists$ 2 dif paths $u \to v$



2 dif paths (not directional)

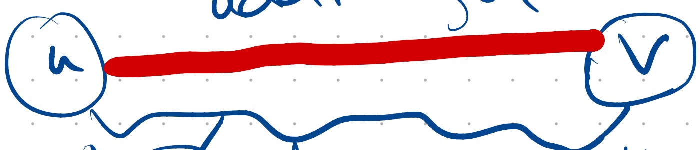
found cycle $u \xrightarrow{\text{path1}} v \xrightarrow{\text{path2}} u$

contradiction!

---

2) $\forall u,v$ $\exists$path(u,v) unique ⇒ 3) Min connected $T + uv$ (edge add'l) cycle.

add1 edge (missing in T)



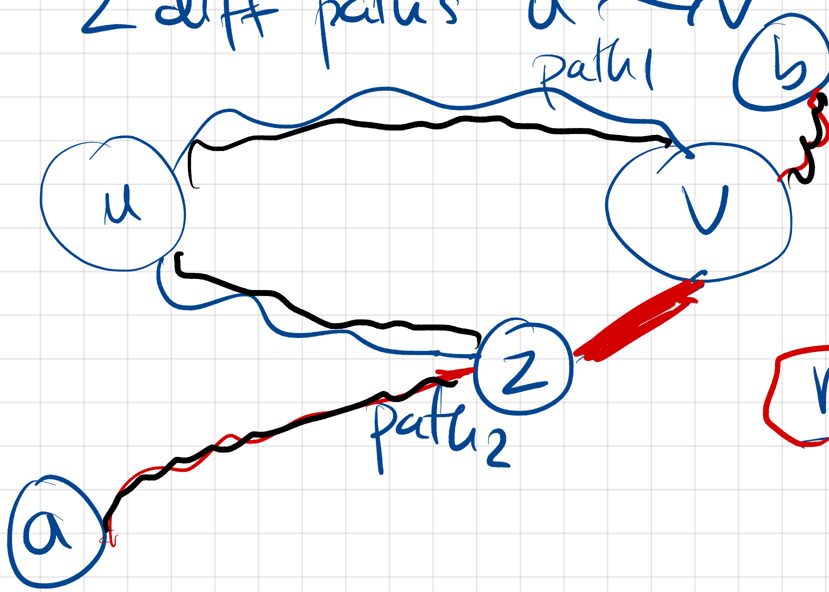
2) $\exists$path(u,v) $\nexists$ edge uv path

⇒ cycle $u \xrightarrow{\text{path}} v \xrightarrow{\text{add edge}} u$

③ Min connected T $\longrightarrow$ ② $\forall u,v$ $\underline{\exists\ \text{path}(u,v)}$ unique

u, v T connected $\Rightarrow$ $\exists$ path(u,v). We want $\underline{\text{uniqueness}}$

assume 2 diff paths $u \to v$



Z = last node on path2 before v

path2(a,v) = path(a,z) + edge zv

$\boxed{\text{remove}}$ edge zv from T

$\Rightarrow$ T still connected

a, b $\in$ Vertices look at path(a,b)

case1 • path(a,b) $\not\ni$ $\boxed{\text{edge zv}}$ removed $\Rightarrow$ like before path(a,b)

case2 • path(a,b) $\ni$ $\boxed{\text{edge zv}}$ removed $\Rightarrow$ new path(a,b)

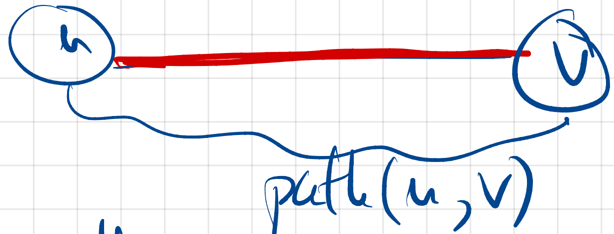
a $\to$ z $\to$ u $\to$ v $\to$ b

contradicts ③ hypothesis $\longrightarrow$ path(u,v) unique

① tree def connected + no cycles $\Longrightarrow$ ④ max acyclic

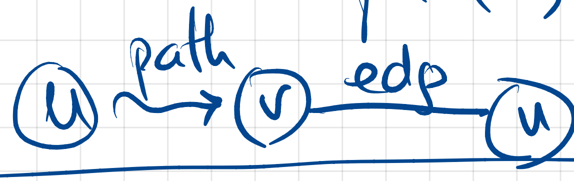
T + edge cycle

add missing edge



$\Downarrow$

cycle

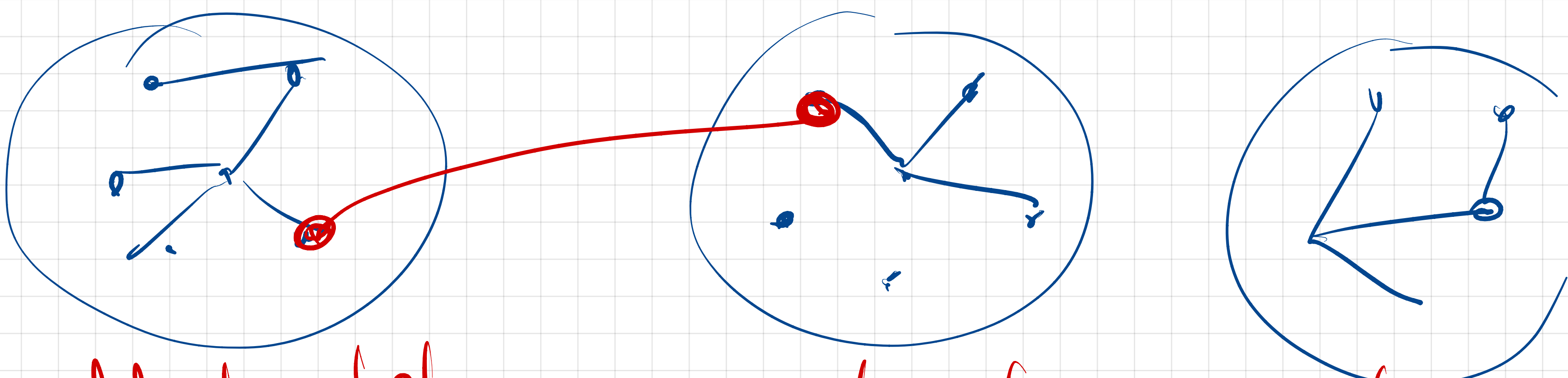


---

④ max acyclic $\Longrightarrow$ ① tree def connected + no cycles

T + new edge $\Rightarrow$ cycles

- connected: Assume not $\Rightarrow$ at least 2 disconnected compon.

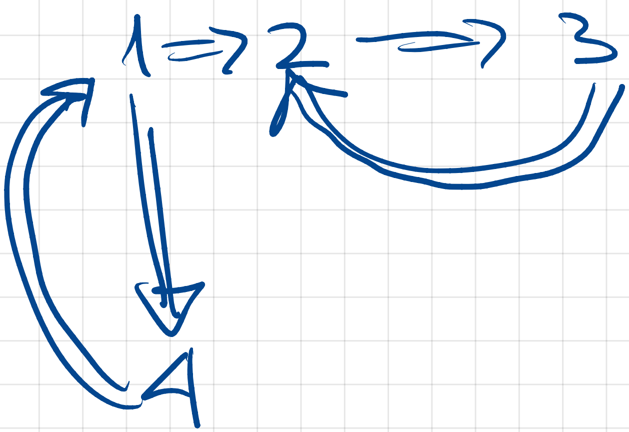


add edge between components $\rightarrow$ forms no cycle

contradicts hyp $\Longrightarrow$ connected

• no cycles

~~max~~ $\boxed{\text{acyclic}} \Rightarrow$ no cycles



Need one of those: $2 \to 1$
$2 \to 4$
$3 \to 1$
$3 \to 4$

(2) $\forall u,v\ \exists$ path(u,v) unique $\Longrightarrow$ (1) tree def: connected + no cycles

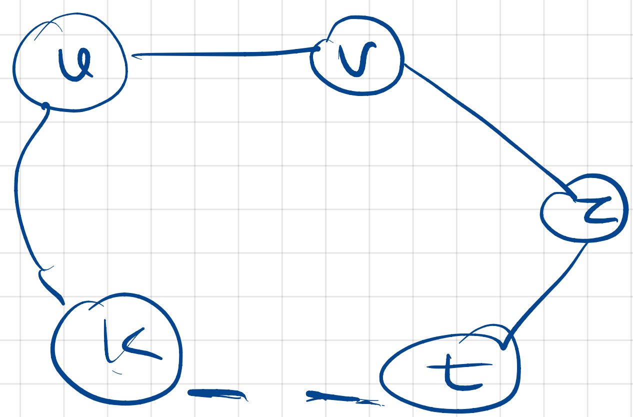
• connected: done path(u,v) exists

• Assume (hyp) $\exists$ cycle

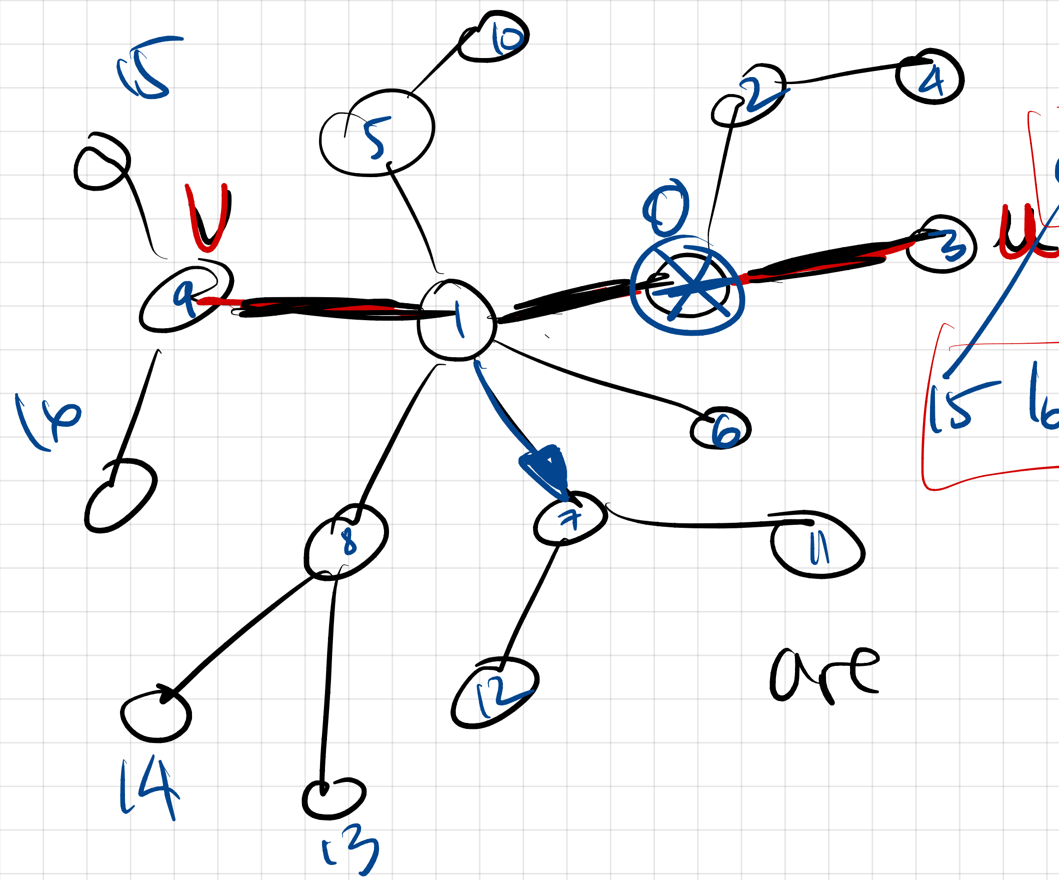
2 dif paths u,v:

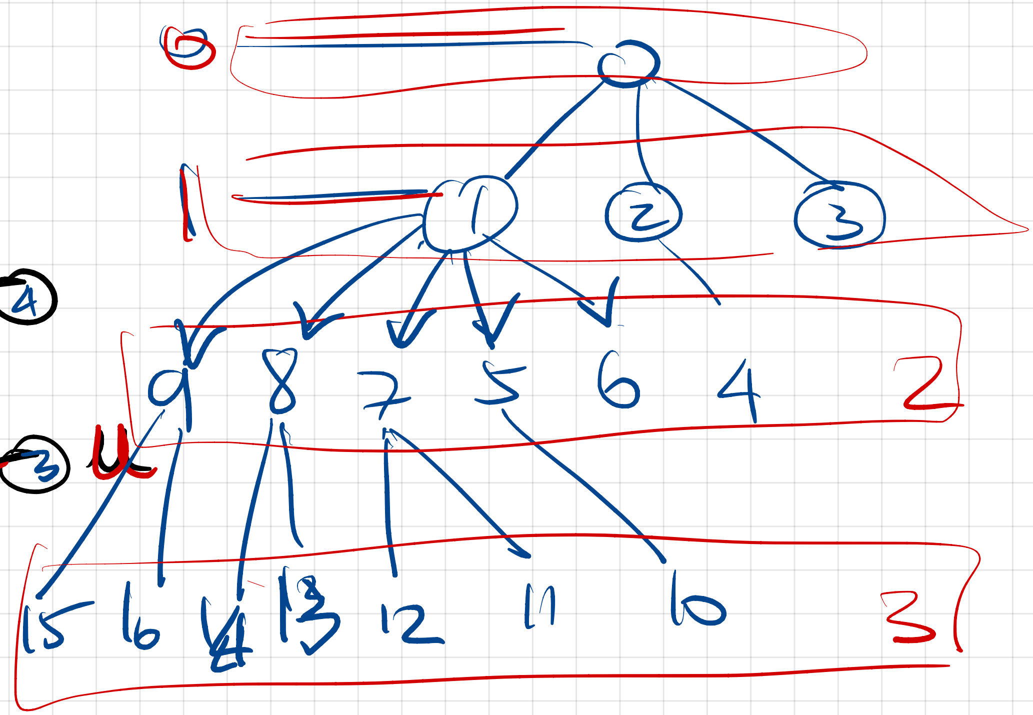
$u - v$ edge (path 1)

$u - k - \dots - t - z - v$ (path 2)



Contradiction $\Rightarrow$ tree. (connected, no cycles) pick root ⊛ → level 0





≃ BFS alg

are

Th tree $\exists$ 2 vertices of degree $= 1$

\# incident edges

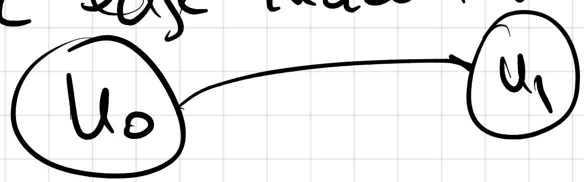
tree $\Rightarrow \deg(x) \geq 1$

Proof | by contradiction: Assume $|\{u \in V \mid \deg(u) = 1\}| \le 1$

• either none such vertex, or there is one

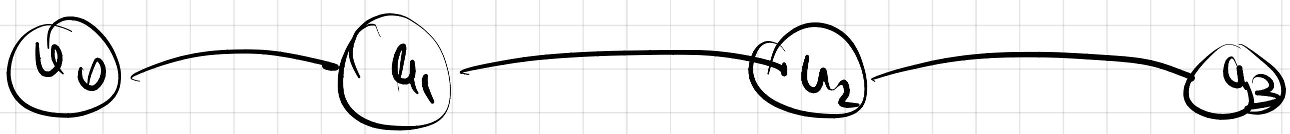
Start build a path in that one if exists, or anywhere is no node $\deg = 1$ exists. Start $= u_0$, $\forall$ other node $u \neq u_0$ $\deg(u) \geq 2$

pick edge incident in u_0 — u_1



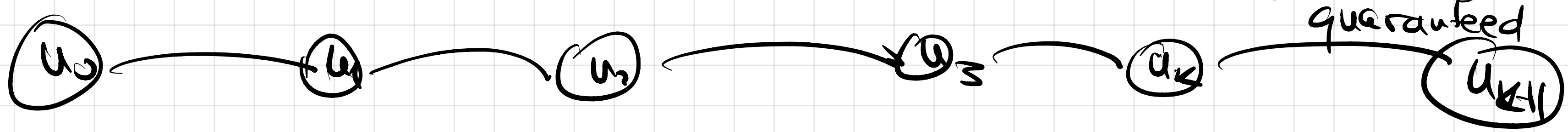
$\deg(u_1) > 1 \Rightarrow u_1$ must have another edge to u_2

$\deg(u_2) > 1 \Rightarrow u_2$ must have another edge (say u_3)



$\deg \geq 2$

new edge guaranteed



This cannot go on forever!

- $\deg(u_k) = 1$ no further edge $\Rightarrow$ done found 2^{nd} vertex $\deg = 1$

OR

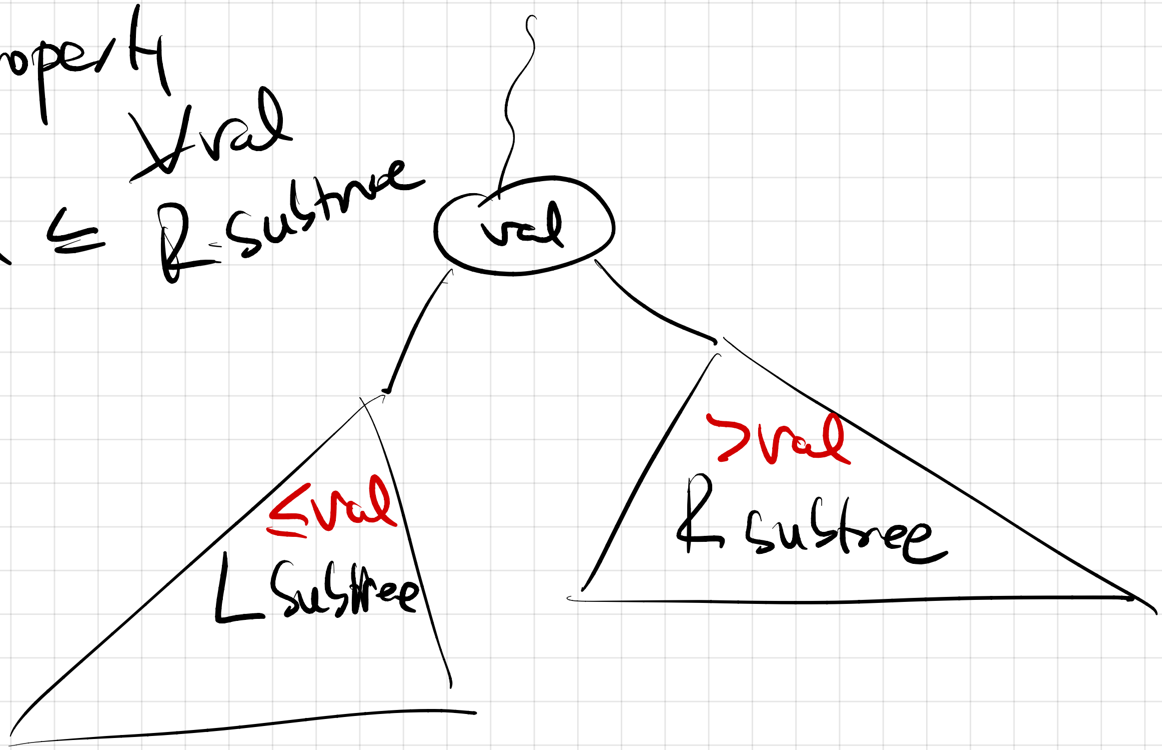
- u_k repeats $u_k =$ previous u_t $(t < k)$
  $\Rightarrow$ cycle contradiction

Binary Search Tree

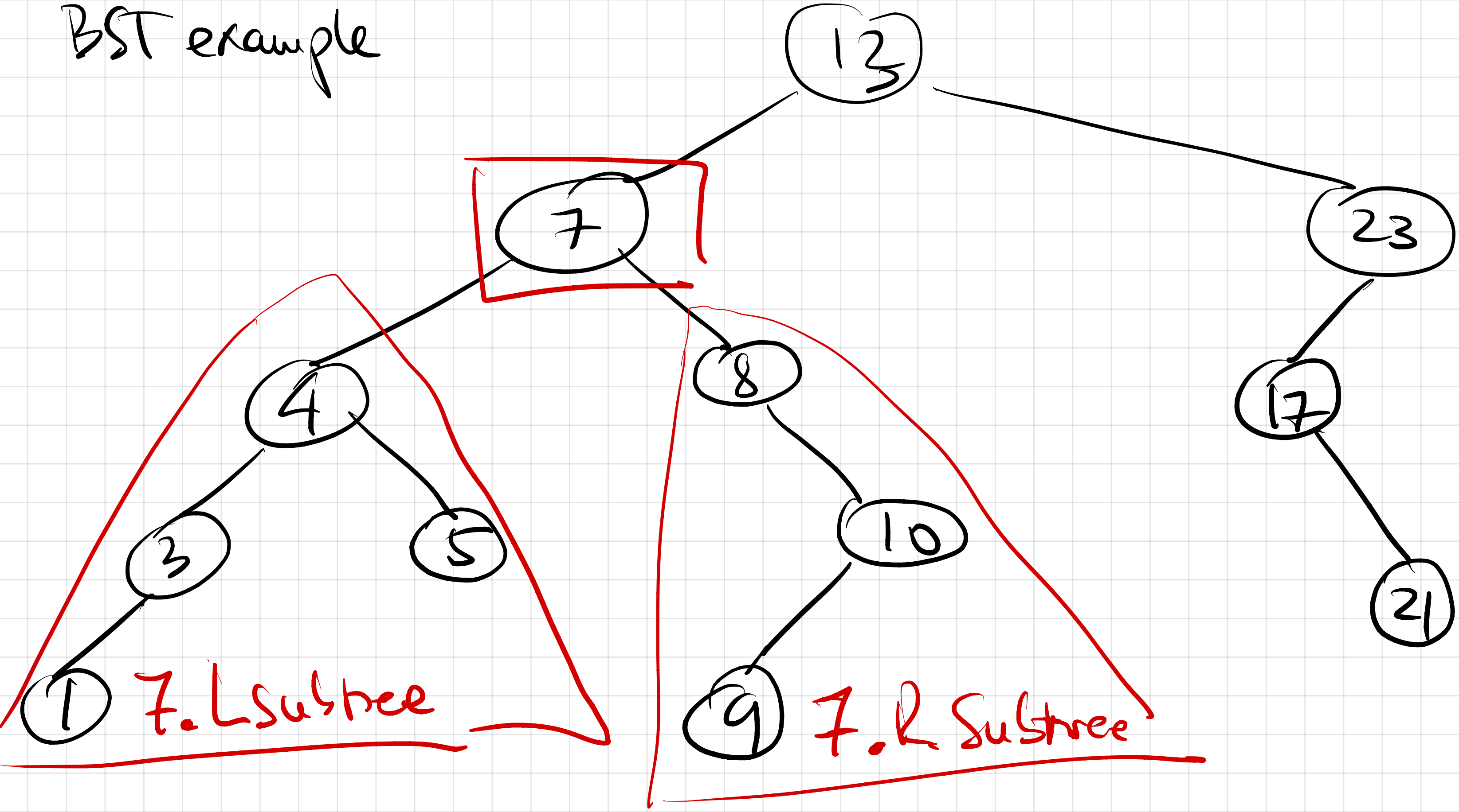
- store values in nodes (vertices)
- root
- binary: at most 2 children.

BST property

$\forall$ val in L.subtree $\leq$ val $\leq$ $\forall$ val R.subtree



BST example



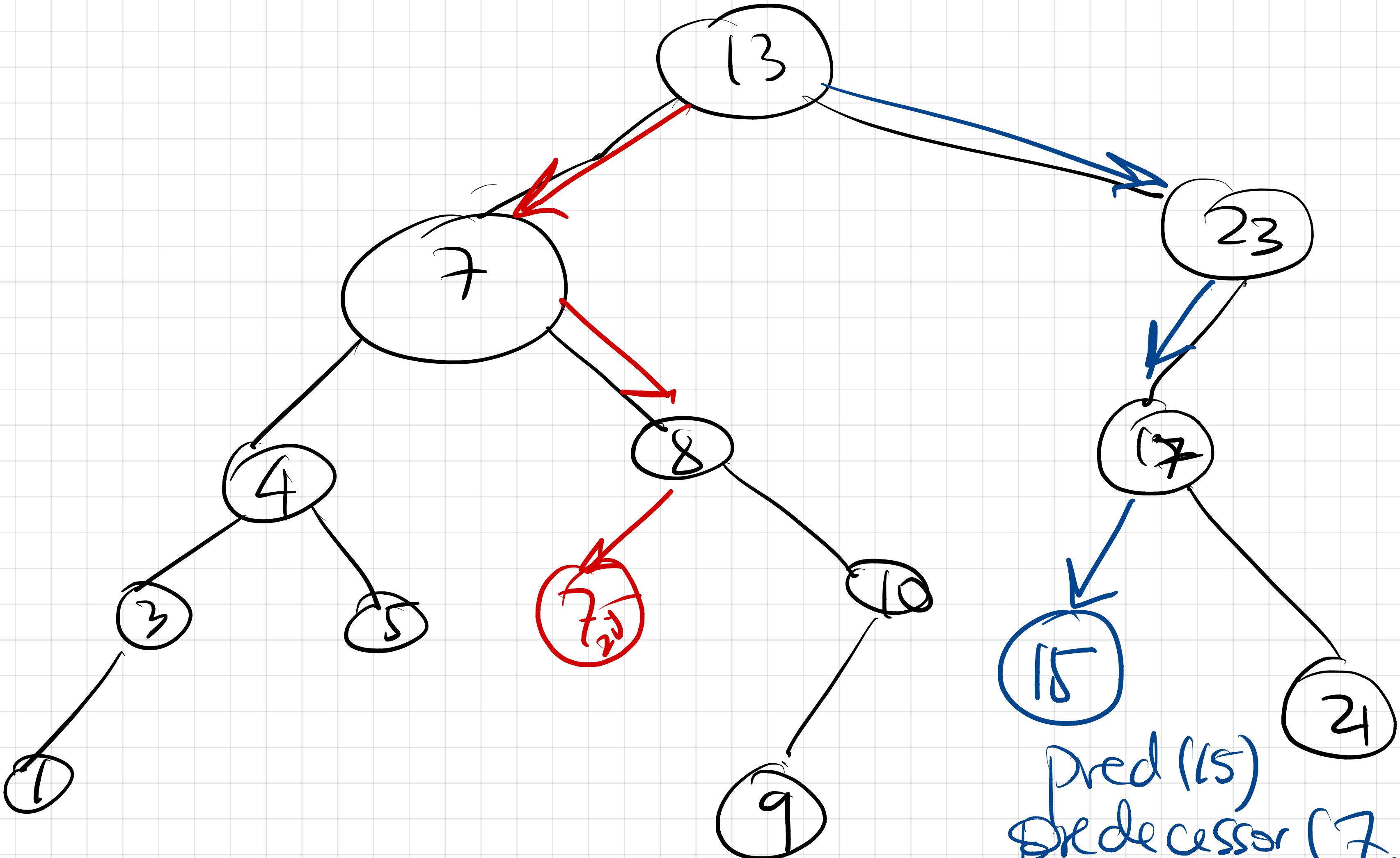
BST sort :- insert all elements in BST (maintain BST prop)
- read all values "in order" Left-val-Right

INORDER (node)

- inorder (node. L Subtree) if ∃ L. Subtree
- print "node. val"
- inorder (node R. Subtree) if ∃ R. Subtree.

Inorder(root) $\Longrightarrow$ produce all values sorted.

INSERT : navigate to the spot, add node



Pred(15)

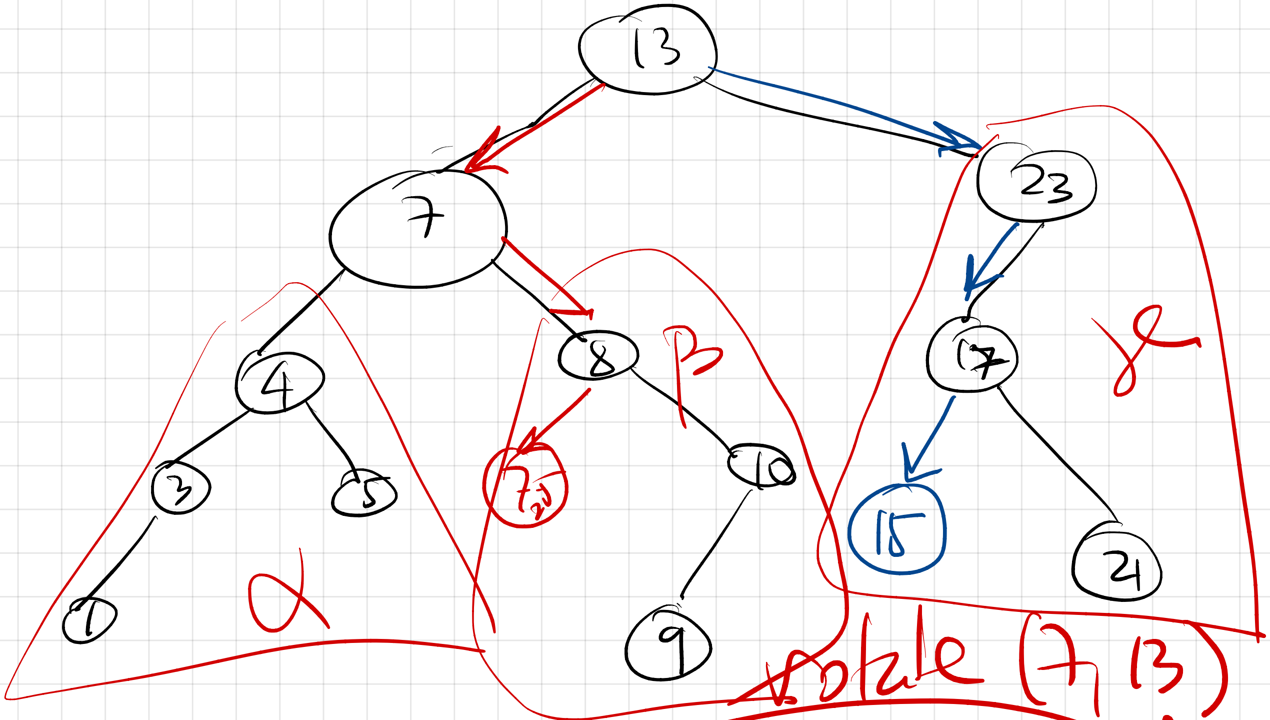
predecessor(7.5) = ?

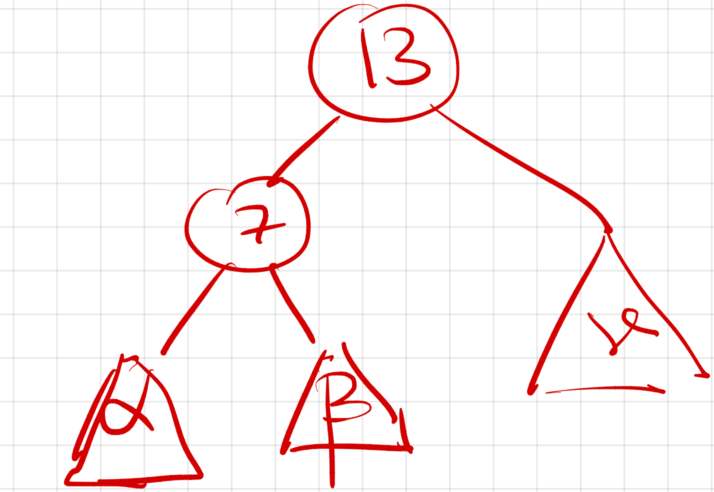
- min
- max

insert 7.5

insert 15

- delete
- rotate
- successor → next val in sorted order, predecessor





$\alpha \leq 7 \leq \beta \leq 13 \leq \gamma$

rotate (7, 13)

7 as root

STILL BST