Memory Errors

invalid read
invalid write
seg fault
memory leak

invalid read

try to access memory we don't have access to

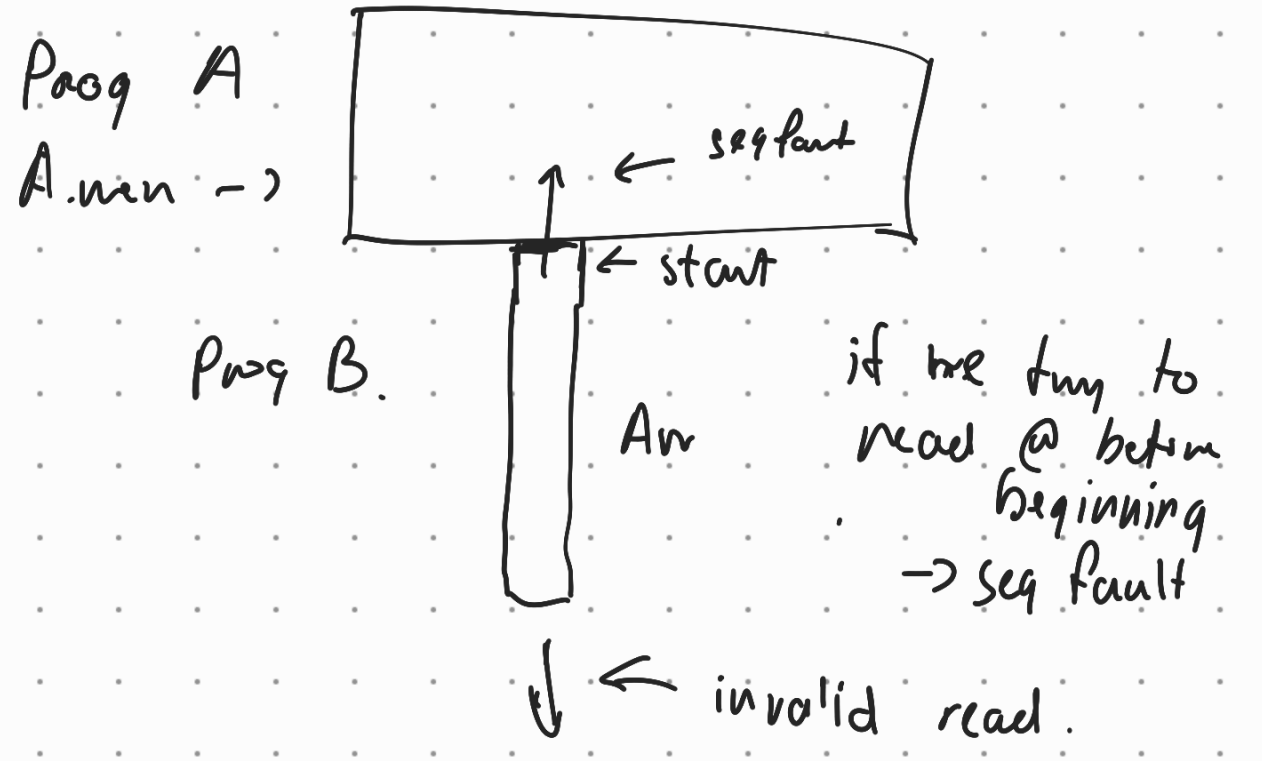
```
int arr[3] = {1,2,3};   ← only alloc 3 int worth
printf("%d", arr[4]);   // invalid read.
                  ↑
            this one don't "own"
int* ptr = malloc(sizeof(int) * 3);
free(ptr);    // we tell OS that we give
                  up the ownership.
int i = *ptr;   // invalid read
ptr = malloc(...)   // ours still one ptr.
   -> ptr = NULL;
```

deref. ↗ *ptr ← always crash : cannot deref null ptr.

invalid write: exact same as invalid read.

seg fault

- try to access memory that you explicitly don't own



-> free ptr

alter free: ptr still points to same addr

-> forget to set to NULL

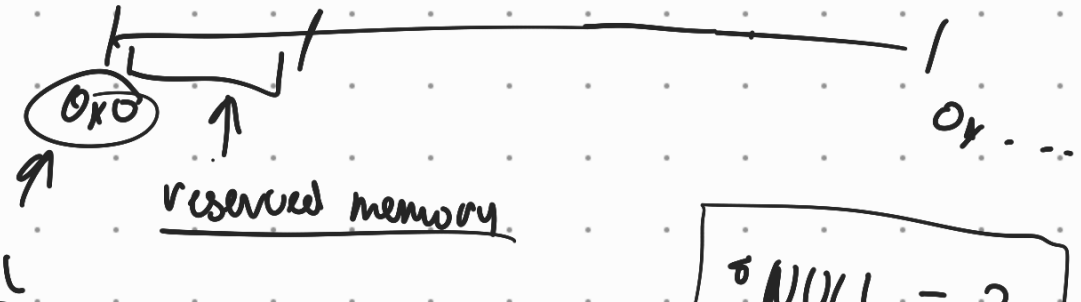
*ptr ← seg fault.

why explicitly invalid?

→ points @ memory another program is using

→ 0x30 → 0x1150

trying to write to read-only memory.



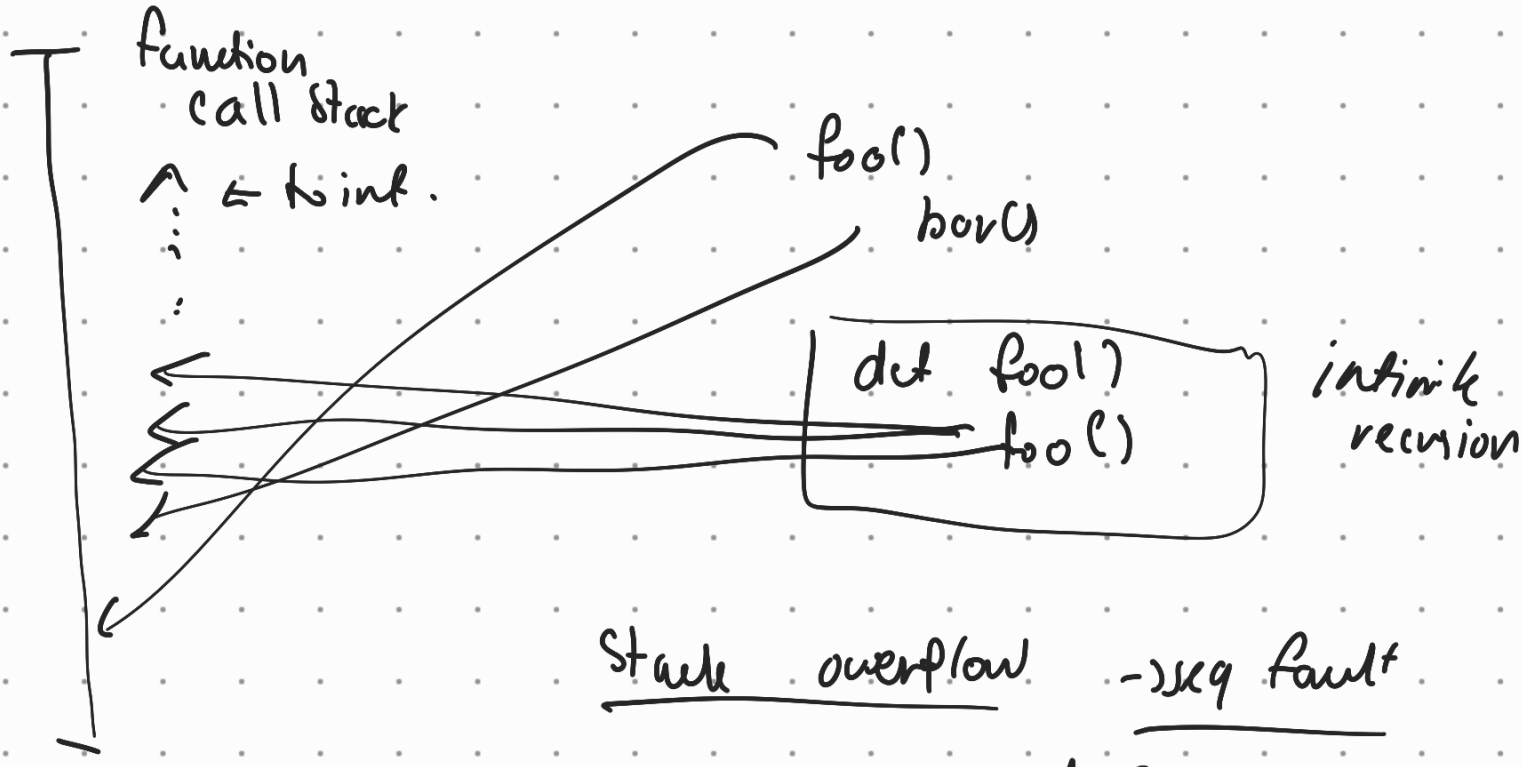
NULL

*NULL → segfault

*NULL = 2

↑ invalid write

any memory addr we don't have access to.



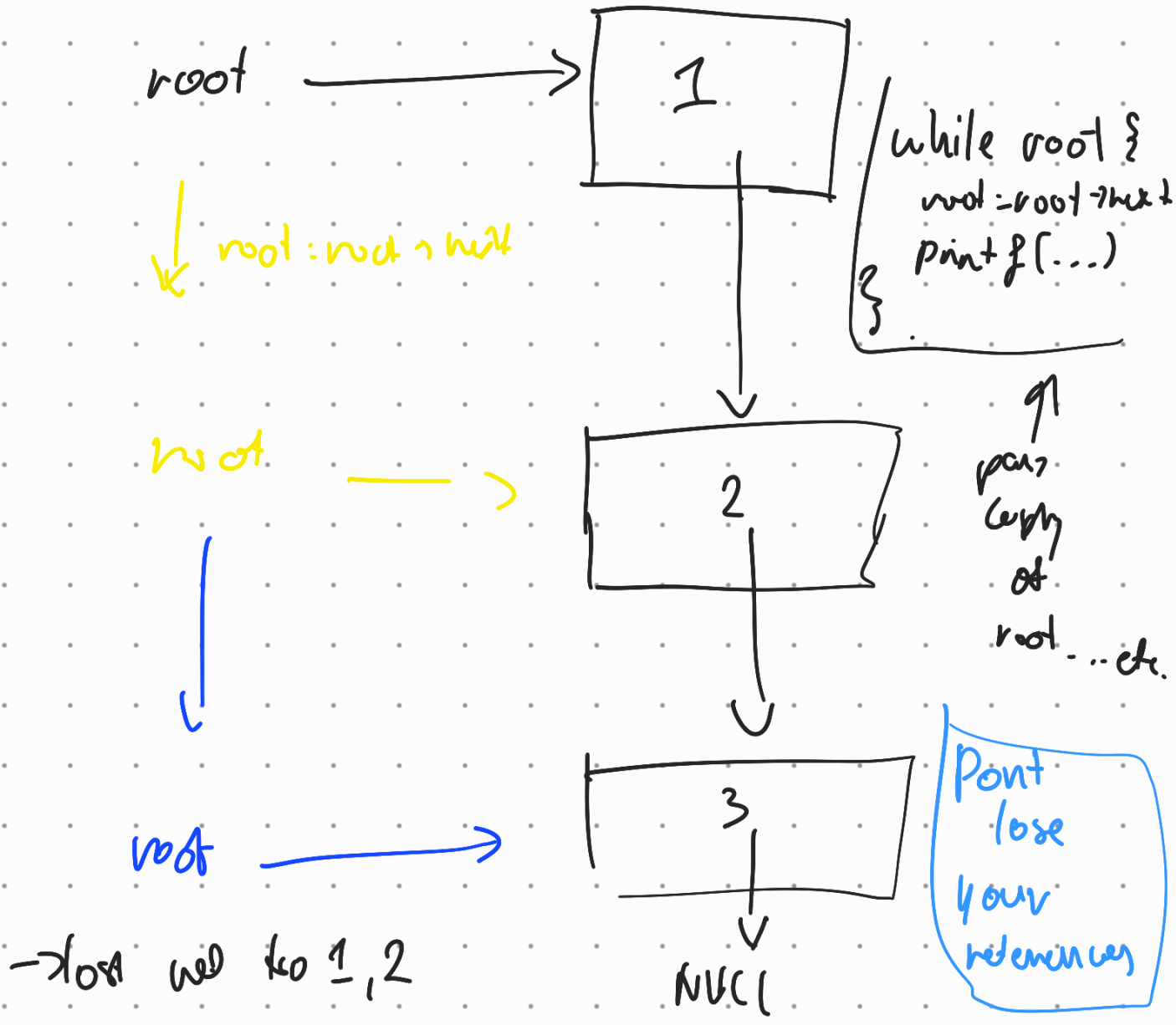
Stack overflow → seg fault

why?

going outside of stack size bounds.

memory leaks

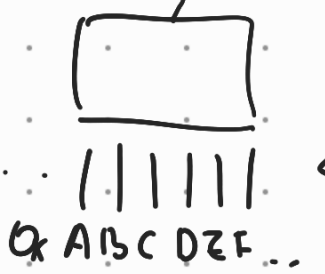
free every malloc or calloc



→ lost ref to 1, 2

double frees

— don't free same ptr twice.



← memory controller

take an addr
-> return value @ addrss.

0x1ABCDEF

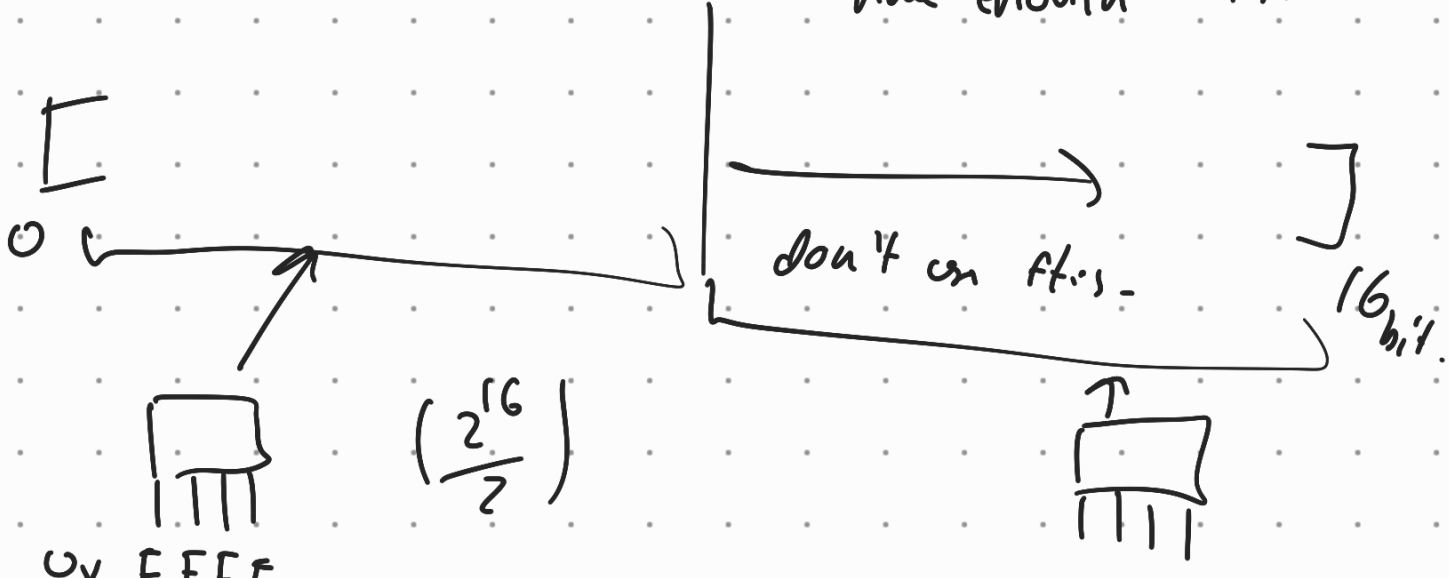
7

↑

28 bit addr

but we only have 24 bits.

Bus Error:
we physically don't have enough bits.



$\left(\frac{2^{16}}{2}\right)$

0x FFFF

→ outside of our address space ∴ error.

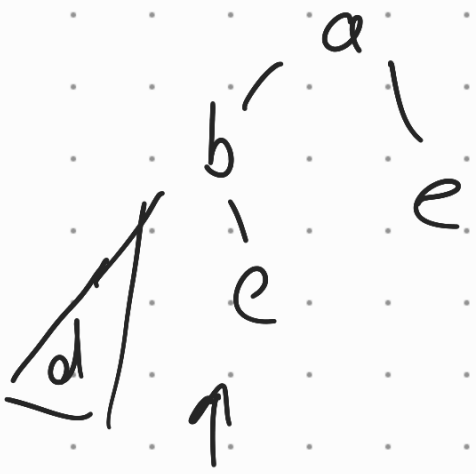
AVL Trees

Balanced BST

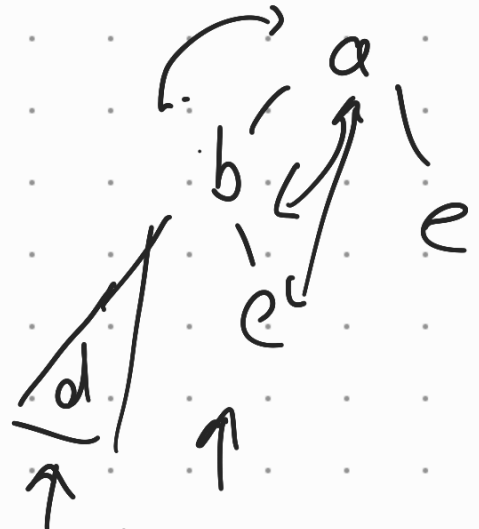
Balance condition: height diff between children is at most 1

- want balanced BST to keep ops @ $\log n$.

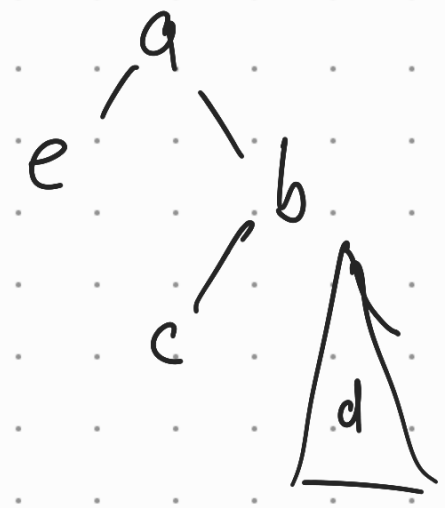
Outside



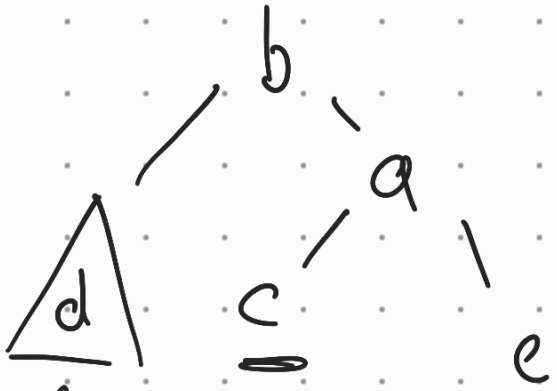
a.left = b.right
b.right = a



d is too tall
∴ drag it up

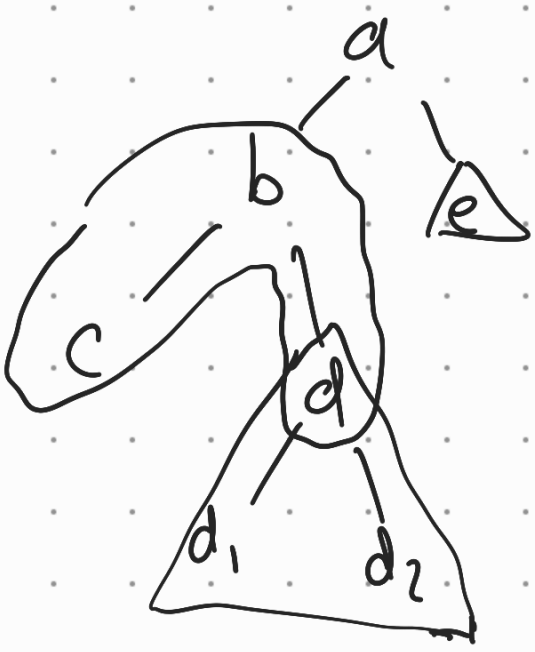


→ every subtree of a BST is also a valid BST

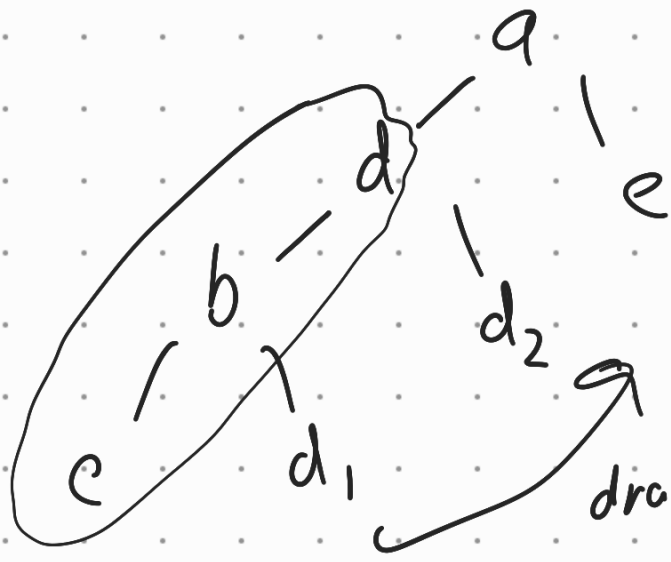


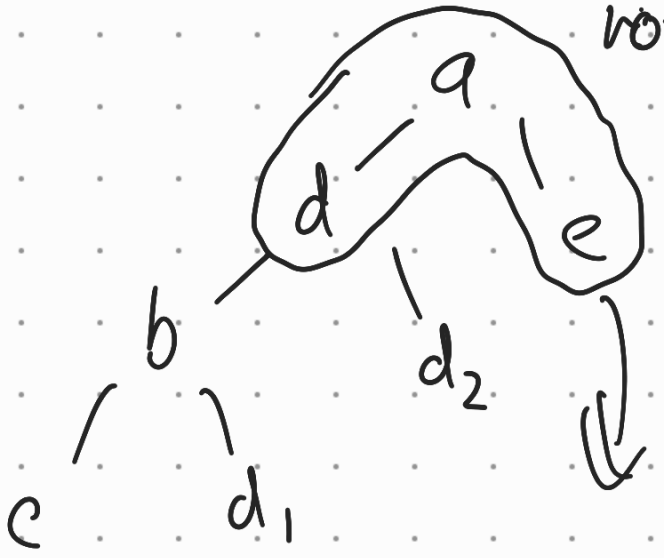
drag d upwards.

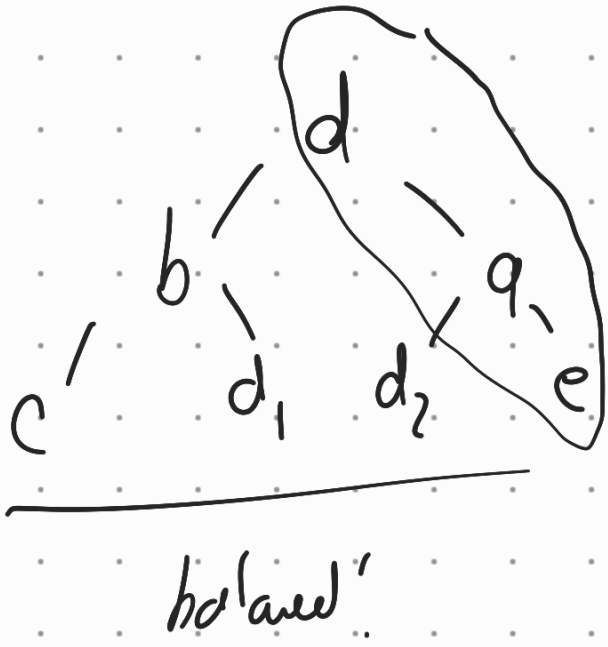
Inside imbalance.



1) rotate b left
2) rotate a right.







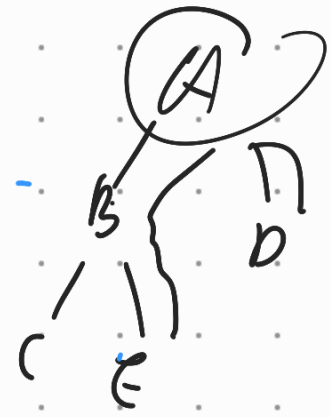
d.left = b
d.right = a
b.right = d_1
a.left = d_2

4 conds

Outer LL → N is Left child, L - imbalance.
Inner LR L child R - imbalance
Outer RR
Inner RL

LL → RL @ n ✓
LR → RL @ n, RR n.parent.
RR → RR @ n ✓
RL → RR @ n, RL @ n.parent.

→ Find LCA of 2 nodes.



LCA(E, D) → A
→ For BT
→ For BST

Some practice q

→ given a tree, check if it is a ...
heap, BST, AVL tree?

-> impl a set w/ a hash fun / table.

{1, 2, 5}. O(1) insert.

-> detect cycle in graph.