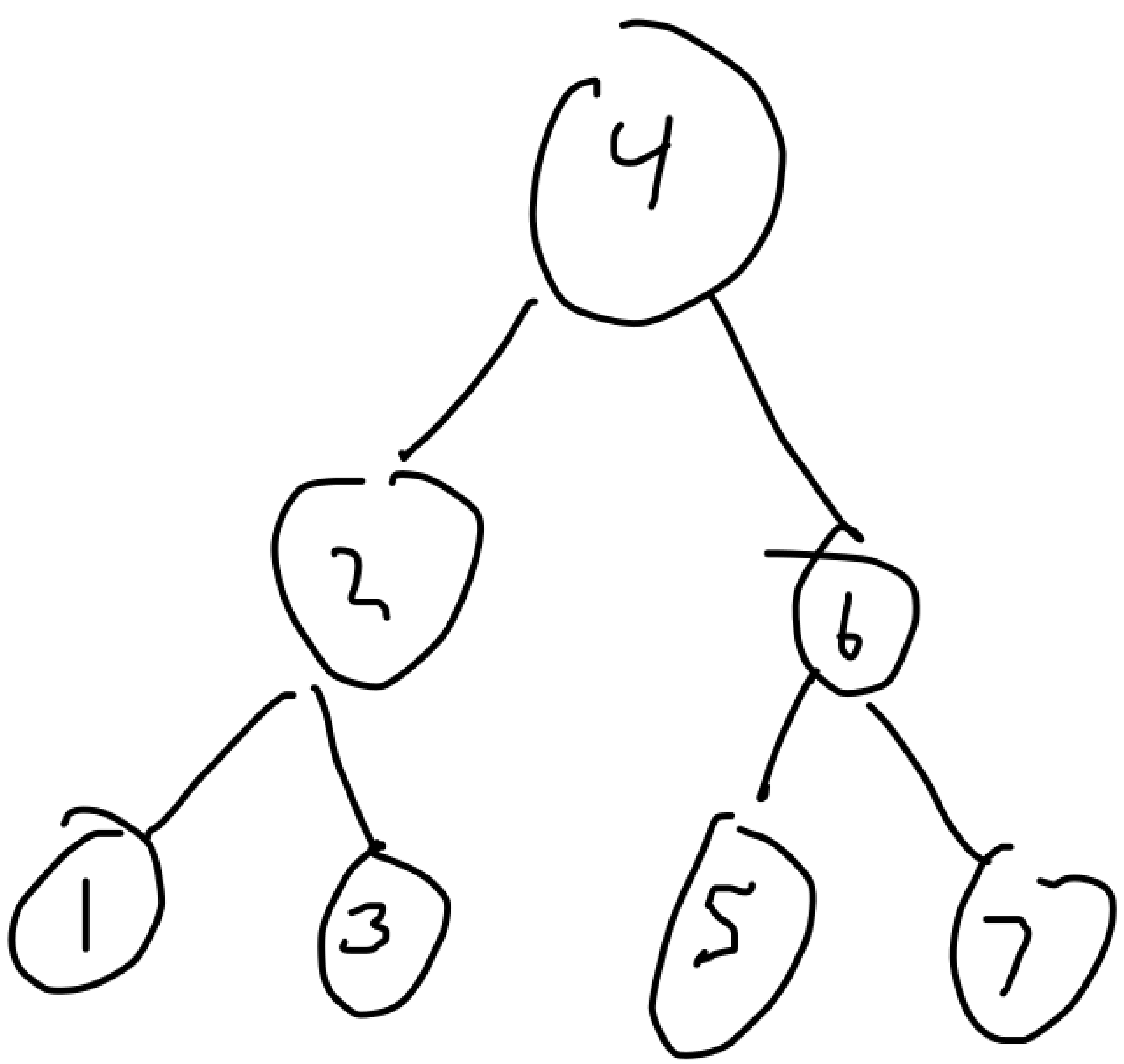
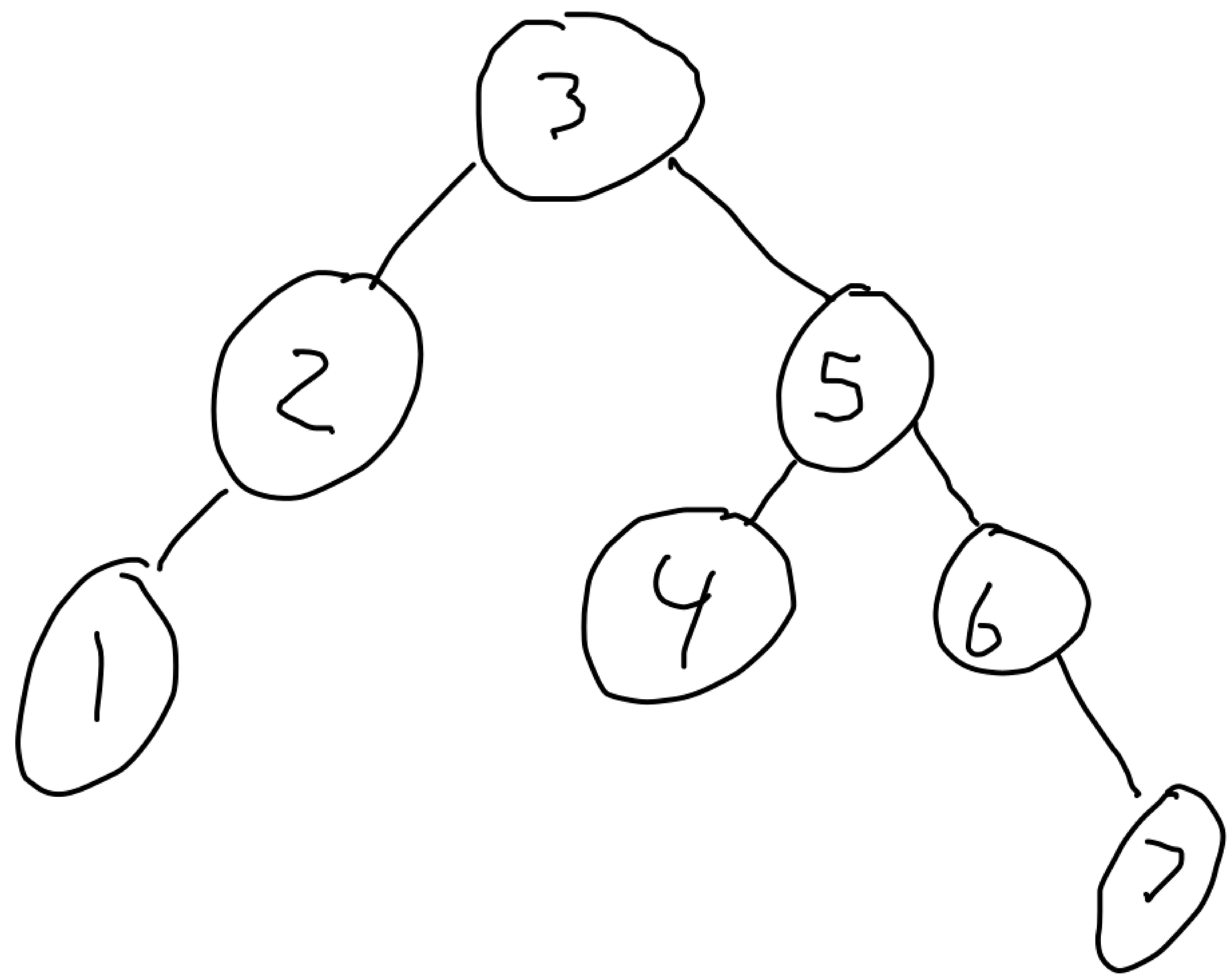
Splay Tree Justification





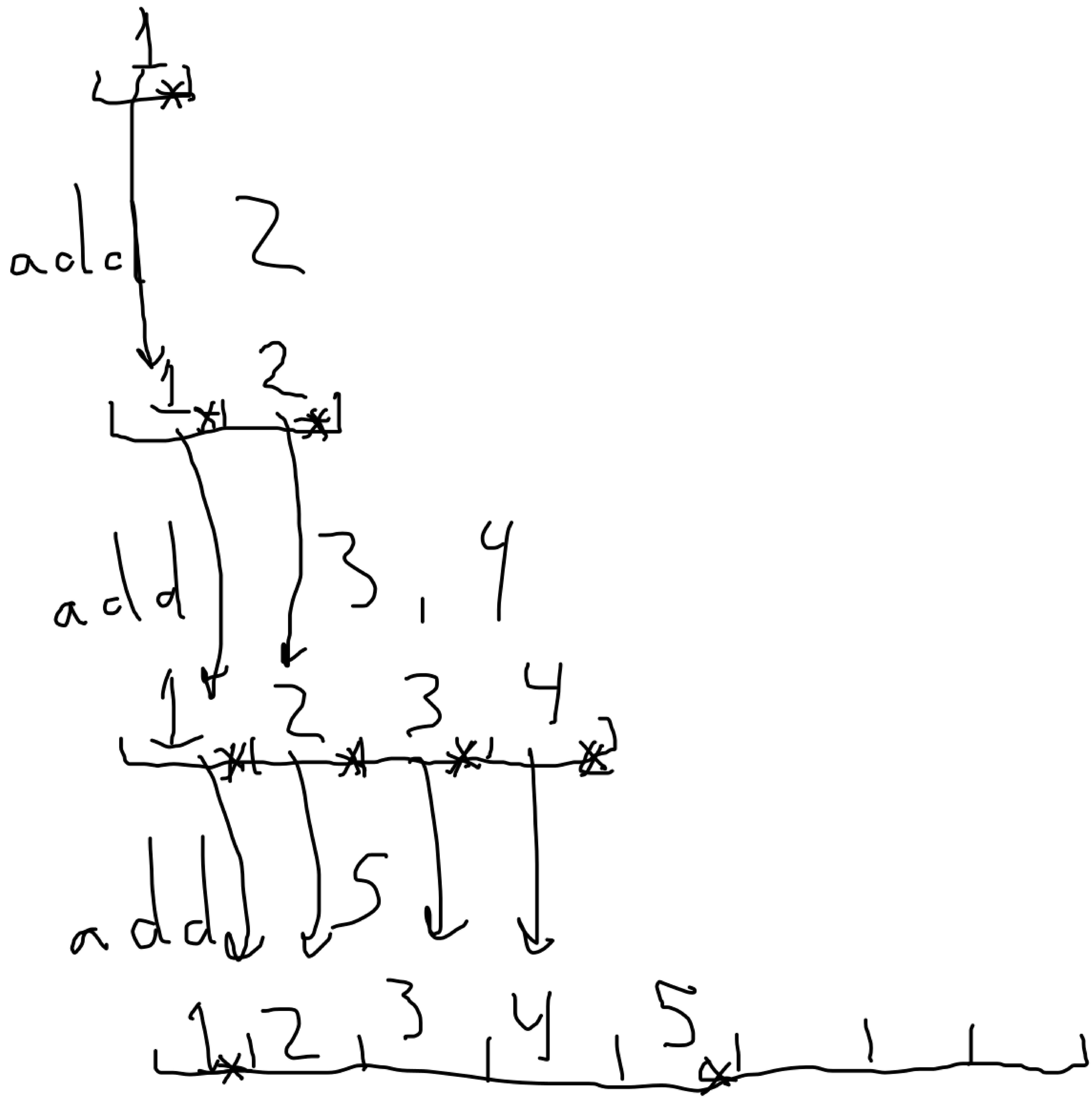
Amortized Time for Array Resize

- when we don't resize, we 'save' credit for the resize
- always use 3 credits to add
  - each credit is for 1 data move
  - when we resize, we have the credits to copy the array
- start with no credits
- we use one credit to add the value and the other two are placed on the added value (i) and $i - \frac{capacity}{2}$

Amortized Time for Array Resize

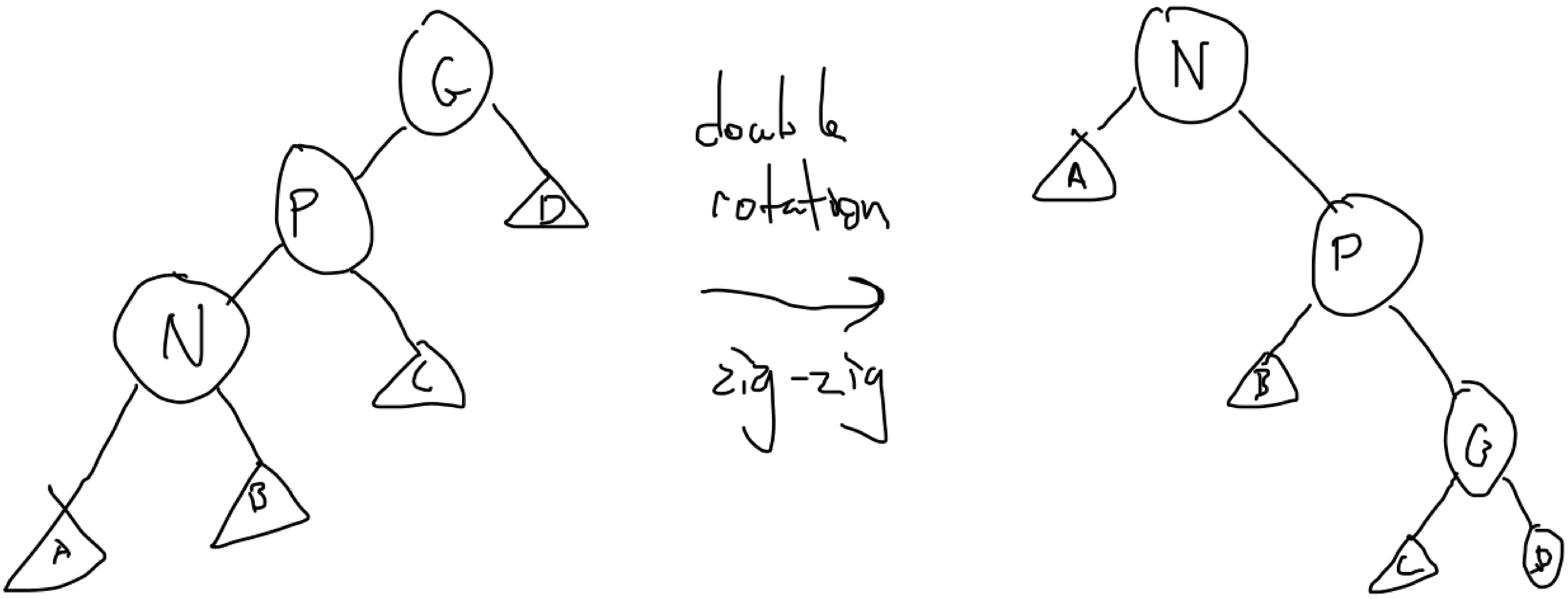
size $\emptyset$

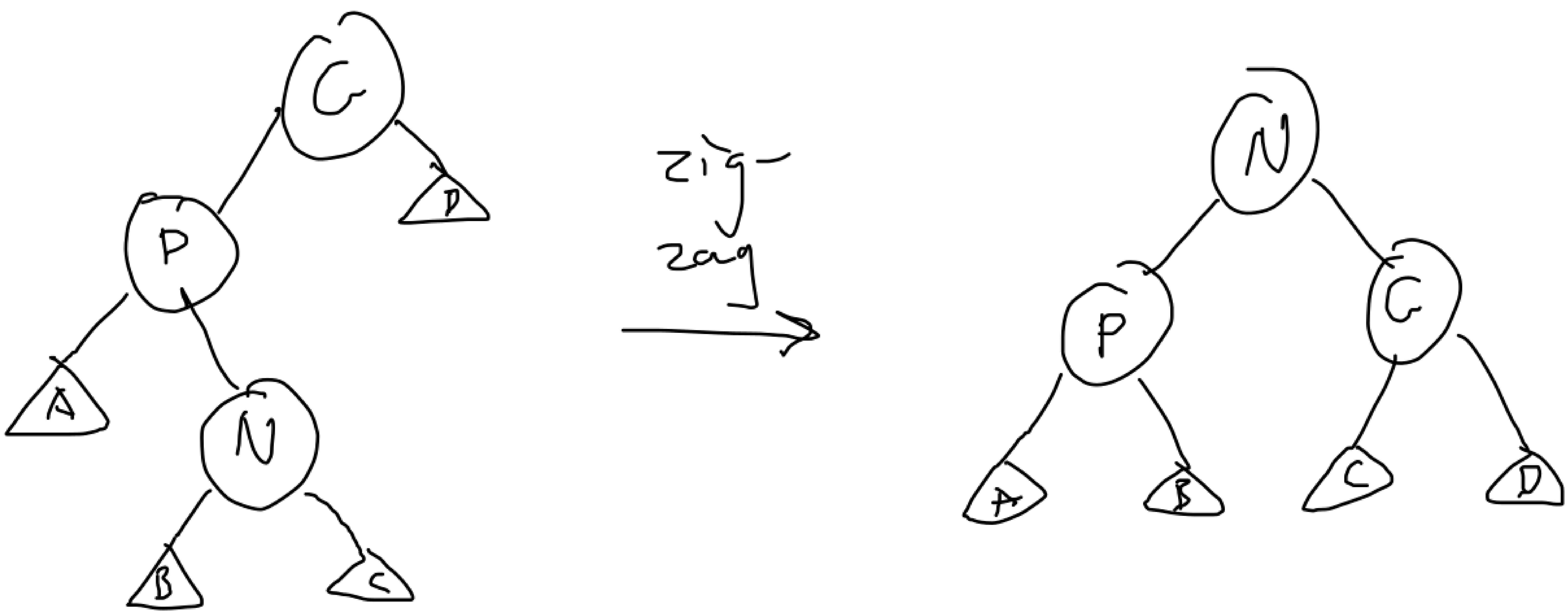
add $\underline{1}$

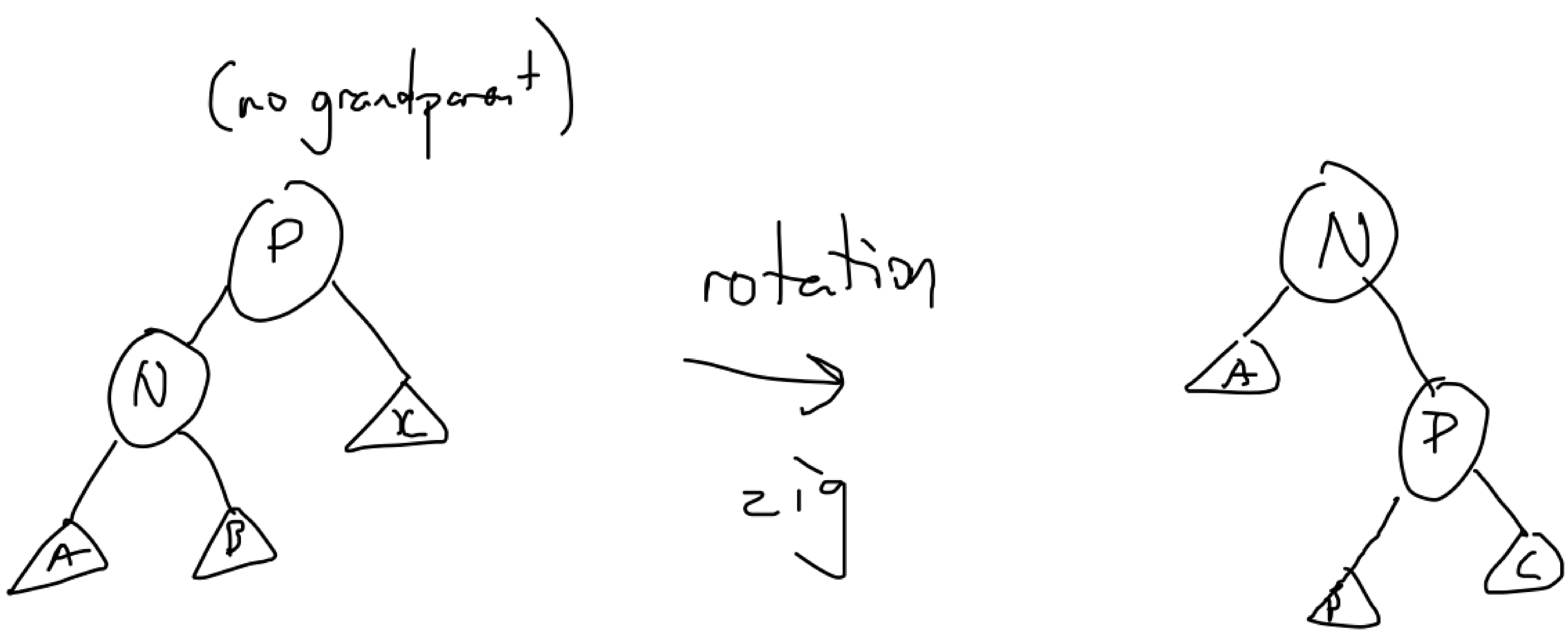


Each step "costs" $3 \in O(1)$

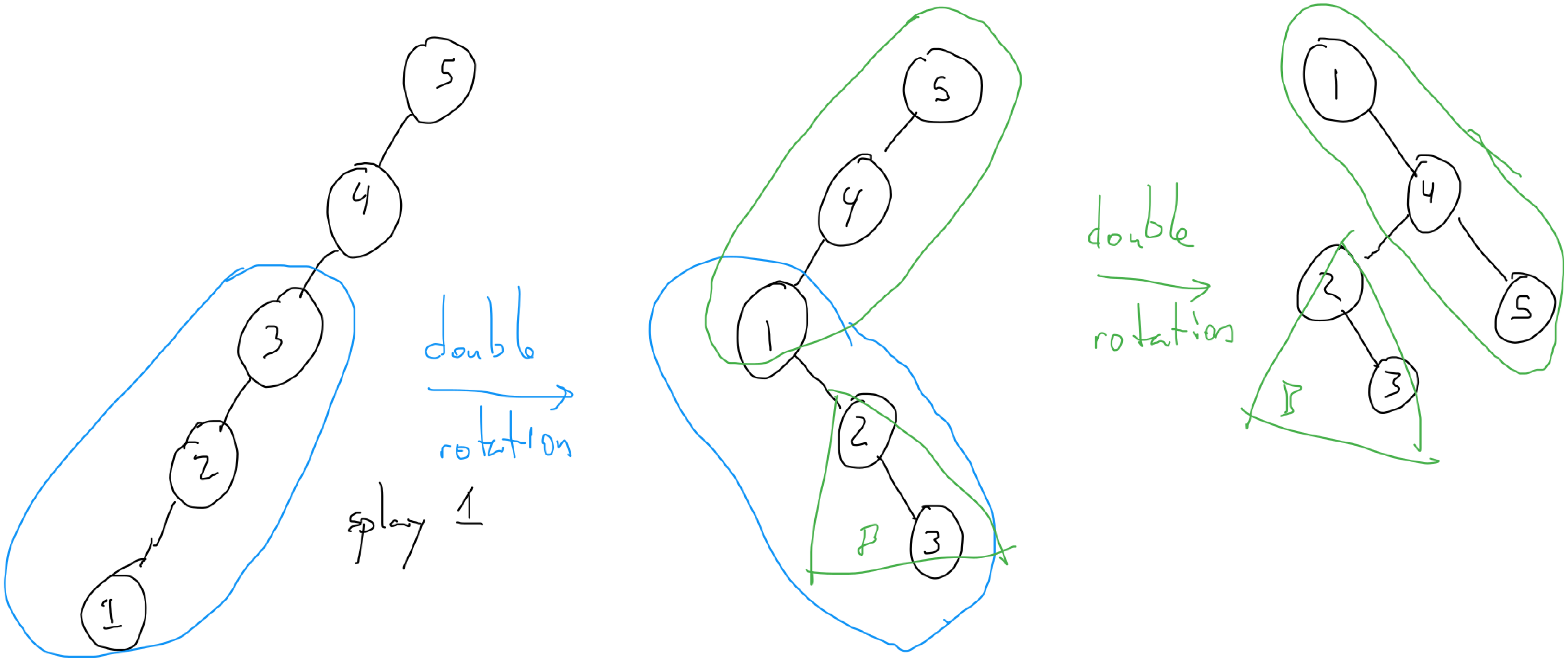
Splay Rotations

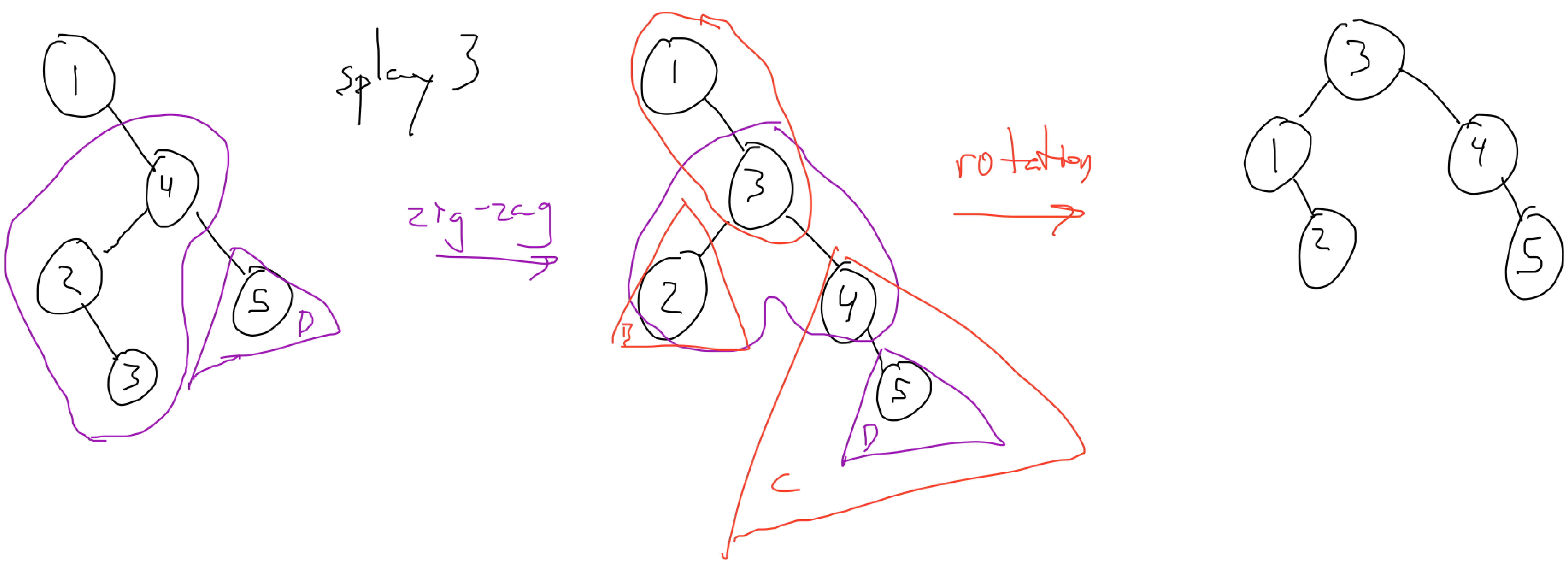






Splaying Examples





Splay Tree Operations

- Find
  - find the node
  - splay it
  - if not found, splay its parent if we had added it
- Add
  - add the value
  - splay the value (even if it was already there)
- Delete
  - delete using replacement
  - after deletion, splay the parent of the removed node

Splay Tree Analysis

- finding a node to manipulate/retreive takes $O(d)$ steps (d is depth of the node)

- we can prove that

  $O(d)$ + splay at depth d has amortized cost of $O(\lg n)$

Sorting

- arranging data in non-descending order

- requires a total ordering of the data
  - requires that any two data points can be compared
  - requires transitivity

$A < B,\ B < C \longrightarrow A < C$

Stable vs. Unstable Sorts

- consider names

			H's	K's
1	B	H	1	
2	D	K		1
3	J	H	2	
4	N	H	3	
5	P	K		2

- stable sort

B H
J H
N H
D K
P K

- unstable sort

? H
? H
? H
? K
? K

12 possible orders