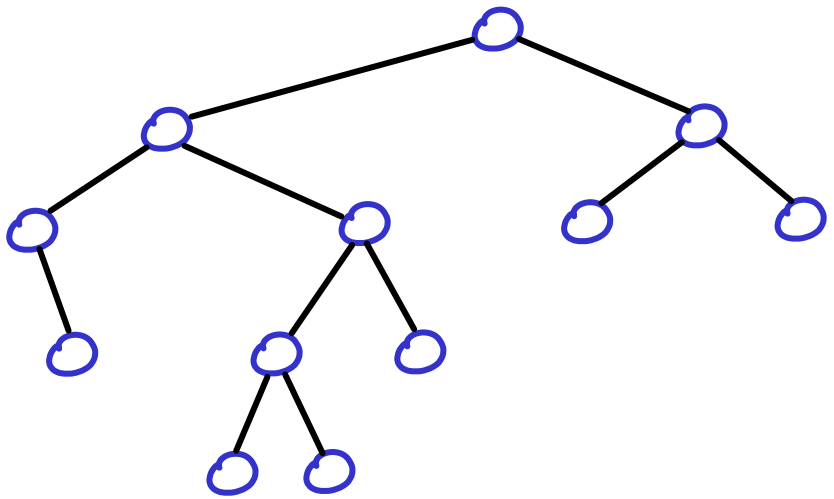
worst case		AVL	RB
	height	1.4 log n	2 log n
insert	# rotations	O(1)	O(1)
delete	# rotations	O(log n)	O(1)

WAVL - Weak AVL trees

(Haeupler, Sen, Tarjan - 2009)

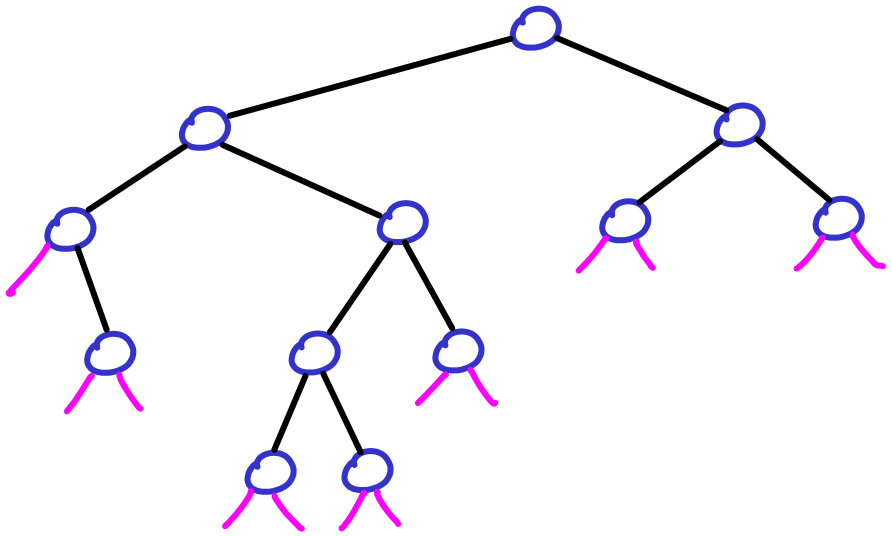
WAVL - Weak AVL trees

- Every node has a score that must be 1 or 2 larger than children's.



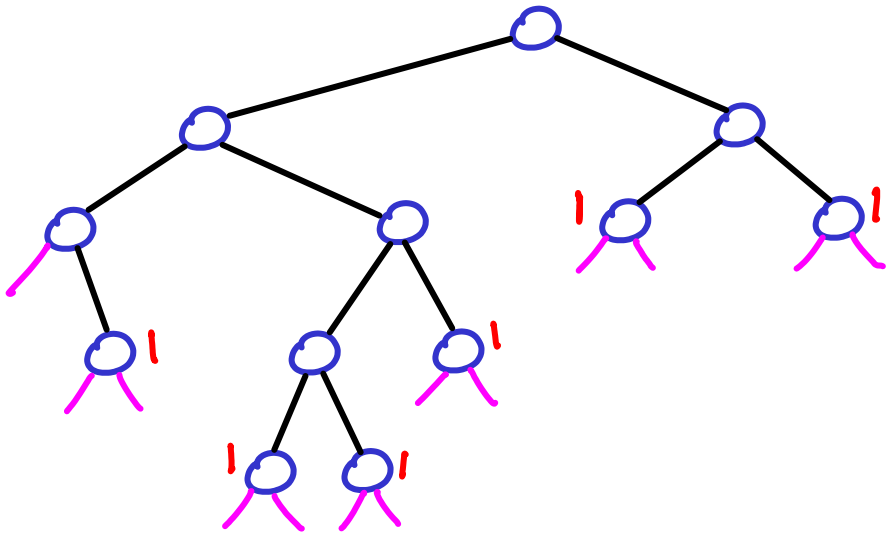
WAVL - Weak AVL trees

- Every node has a score that must be 1 or 2 larger than children's.
- For definition, use fake leaves with score = 0



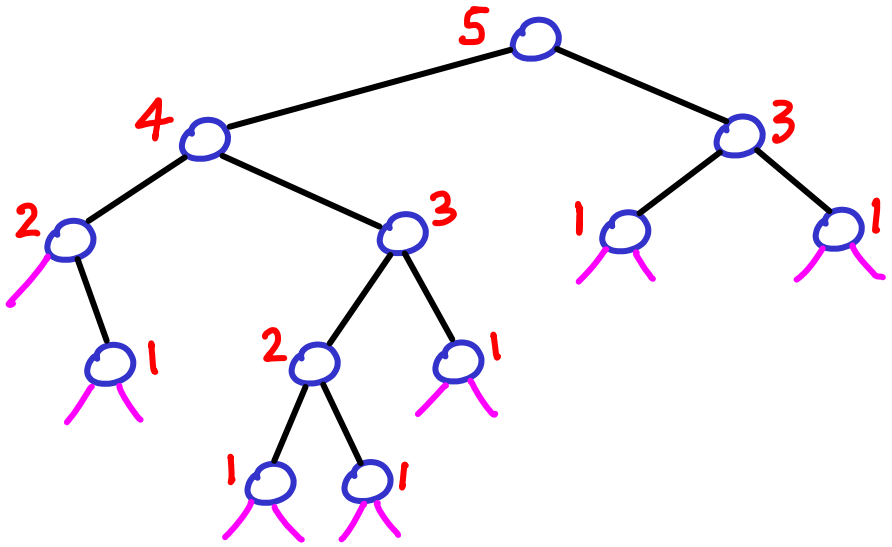
WAVL - Weak AVL trees

- Every node has a score that must be 1 or 2 larger than children's.
- For definition, use fake leaves with score = 0
- Actual leaves (parents of 2 fake) must have score = 1



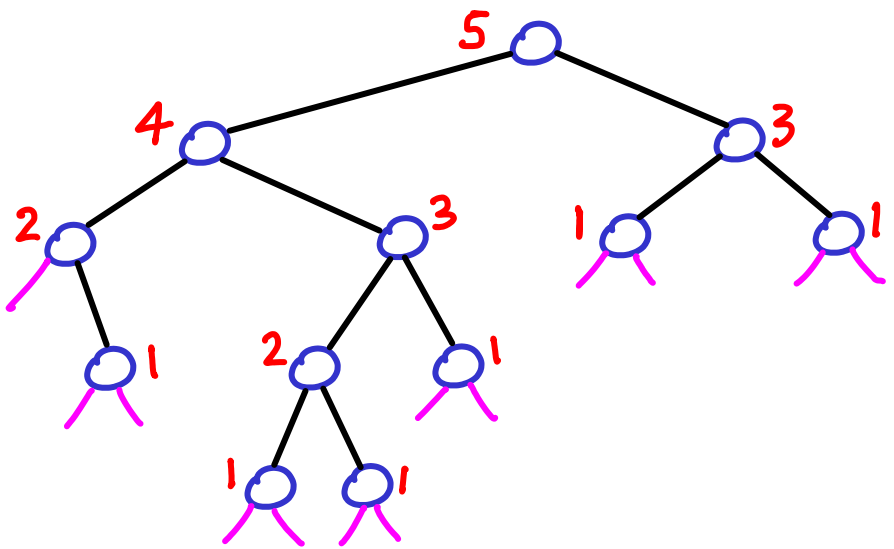
WAVL - Weak AVL trees

- Every node has a score that must be 1 or 2 larger than children's.
- For definition, use fake leaves with score = 0
- Actual leaves (parents of 2 fake) must have score = 1

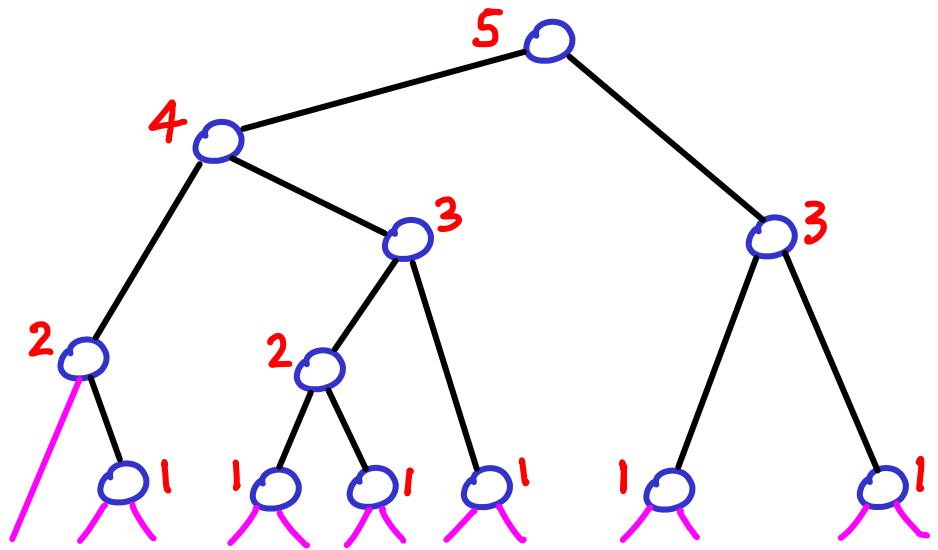


WAVL - Weak AVL trees

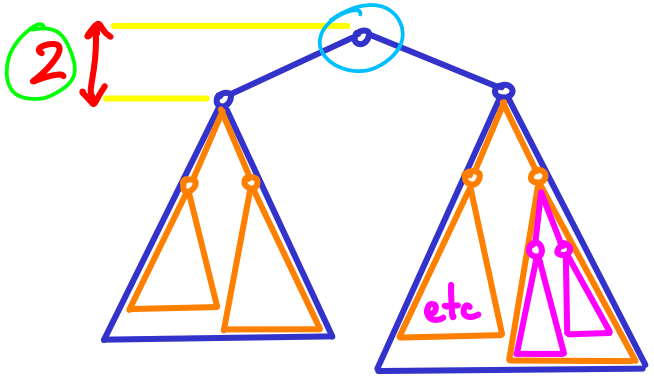
- Every node has a score that must be 1 or 2 larger than children's.
- For definition, use fake leaves with score = 0
- Actual leaves (parents of 2 fake) must have score = 1



not AVL

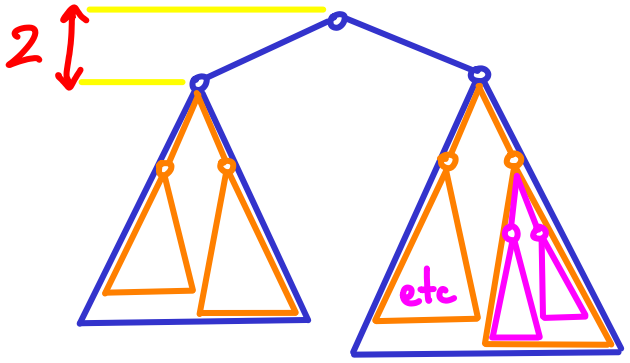


$N(r)$ = min #nodes to achieve score r.



$$N(r) = 1 + 2N(r-2) \quad // \quad N(1) = 1 \qquad N(2) = 2$$

$N(r)$ = min #nodes to achieve score r.

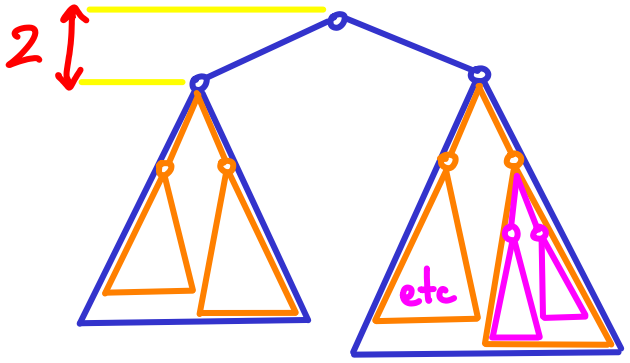


$N(r) = 1 + 2N(r-2)$ // $N(1) = 1$ $N(2) = 2$

$N(r) = \Omega(2^{r/2})$

$r \leq 2\log_2(N+1)$

$N(r)$ = min #nodes to achieve score r.

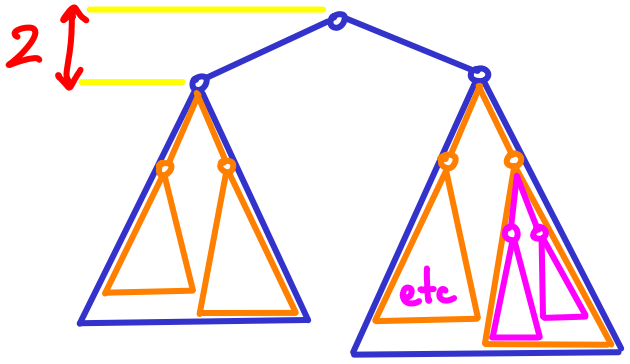


$N(r) = 1 + 2N(r-2)$ // $N(1) = 1$ $N(2) = 2$

$N(r) = \Omega(2^{r/2})$ $r \leq 2\log_2(N+1)$

*trivially, every AVL is also WAVL

$N(r)$ = min #nodes to achieve score r.



$N(r) = 1 + 2N(r-2)$ // $N(1) = 1$ $N(2) = 2$

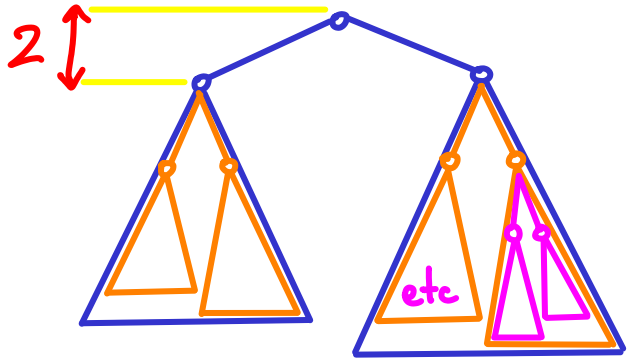
$N(r) = \Omega(2^{r/2})$ $r \leq 2\log_2(N+1)$

* trivially, every AVL is also WAVL
* every WAVL can be colored red-black

} AVL trees are RB (but not vice versa)

⋮

$N(r)$ = min #nodes to achieve score r.



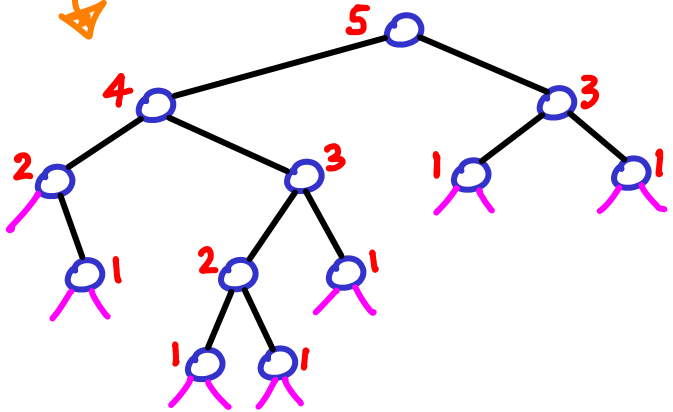
$N(r) = 1 + 2N(r-2)$ // $N(1) = 1$ $N(2) = 2$

$N(r) = \Omega(2^{r/2})$ $r \leq 2\log_2(N+1)$

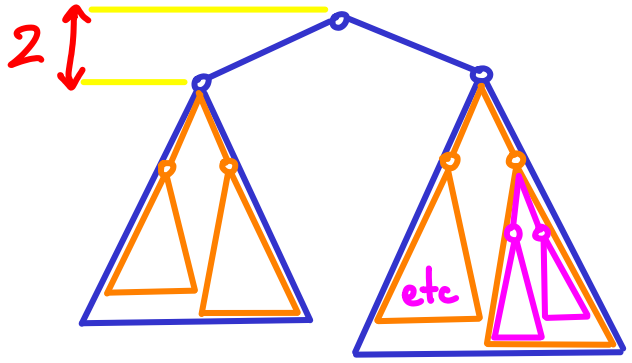
* trivially, every AVL is also WAVL

* every WAVL can be colored red-black } AVL trees are RB (but not vice versa)

e.g.



$N(r)$ = min # nodes to achieve score r.



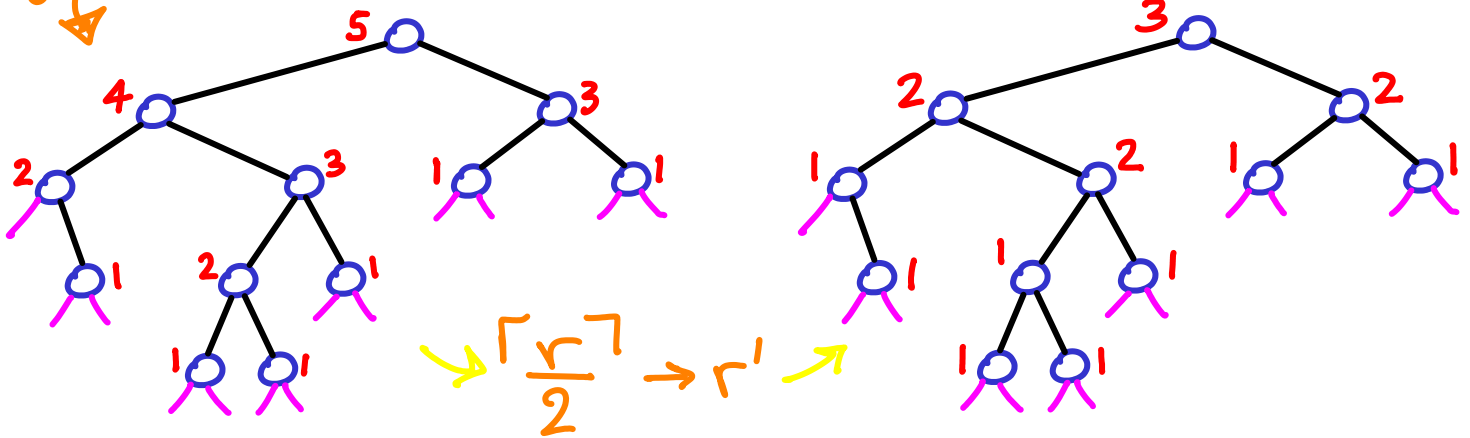
$N(r) = 1 + 2N(r-2)$ // $N(1) = 1$ $N(2) = 2$

$N(r) = \Omega(2^{r/2})$ $r \leq 2\log_2(N+1)$

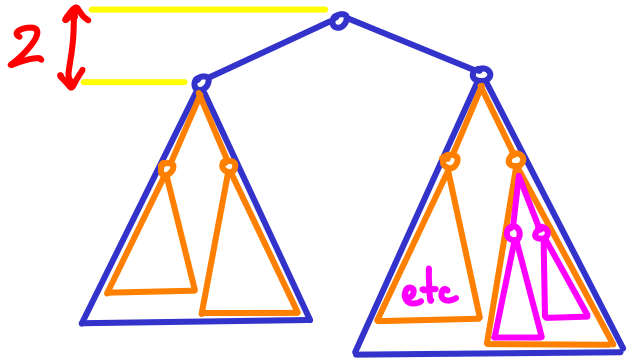
* trivially, every AVL is also WAVL

* every WAVL can be colored red-black } AVL trees are RB (but not vice versa)

e.g.



$N(r)$ = min # nodes to achieve score r.



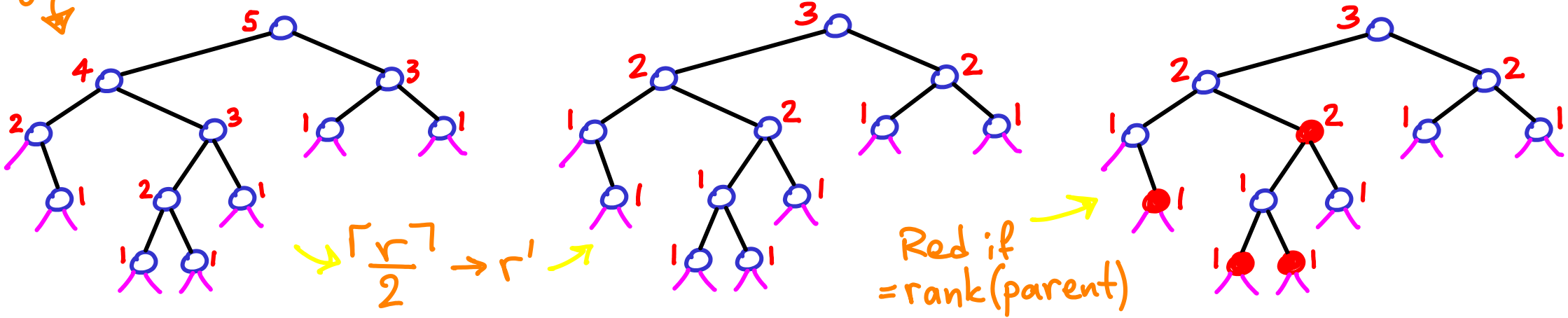
$N(r) = 1 + 2N(r-2)$ // $N(1) = 1$ $N(2) = 2$

$N(r) = \Omega(2^{r/2})$ $r \leq 2\log_2(N+1)$

* trivially, every AVL is also WAVL
* every WAVL can be colored red-black

} AVL trees are RB (but not vice versa)

e.g.



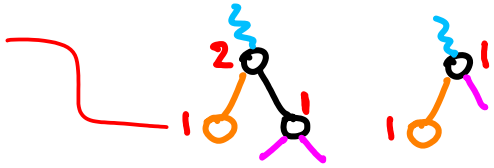
INSERTION IN WAVL

(if only inserting we actually get = AVL thus $1.4 \log n$ height)

INSERTION IN WAVL

(if only inserting we actually get = AVL thus 1.4 log n height)

- inserted node becomes (non-fake) leaf with score = 1

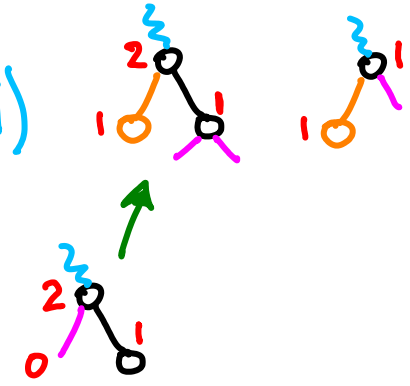


INSERTION IN WAVL

(if only inserting we actually get = AVL thus 1.4 log n height)

- inserted node becomes (non-fake) leaf with score = 1

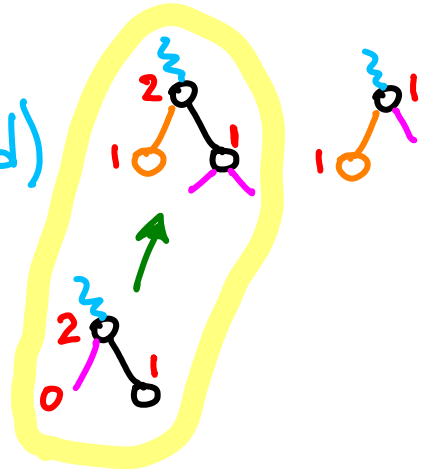
  (replaces fake leaf, so score was incremented)



INSERTION IN WAVL

(if only inserting we actually get = AVL thus 1.4 log n height)

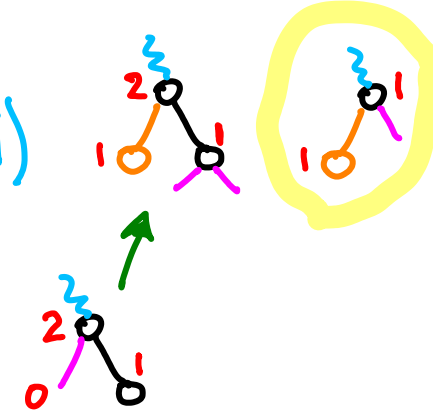
- inserted node becomes (non-fake) leaf with score = 1
  (replaces fake leaf, so score was incremented)
- if parent score is still greater, DONE



INSERTION IN WAVL

(if only inserting we actually get = AVL thus 1.4 log n height)

- inserted node becomes (non-fake) leaf with score = 1
  (replaces fake leaf, so score was incremented)
- if parent score is still greater, DONE

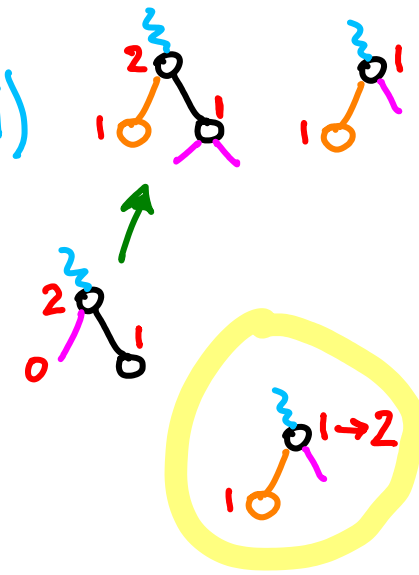


else - if parent score is +1 of sibling ... ?

INSERTION IN WAVL

(if only inserting we actually get = AVL thus 1.4 log n height)

- inserted node becomes (non-fake) leaf with score = 1
  (replaces fake leaf, so score was incremented)
- if parent score is still greater, DONE

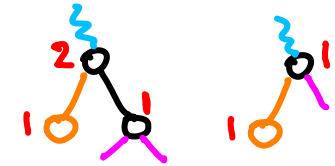


else – if parent score is +1 of sibling
increment parent score (fixed locally)

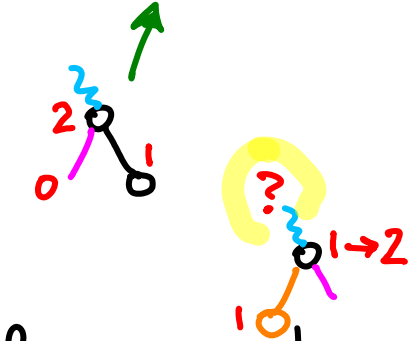
INSERTION IN WAVL

(if only inserting we actually get =AVL thus 1.4 log n height)

- inserted node becomes (non-fake) leaf with score = 1
  (replaces fake leaf, so score was incremented)



* • if parent score is still greater, DONE

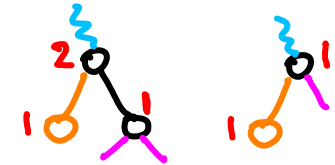


else – if parent score is +1 of sibling
increment parent score (fixed locally) & fix upward *

INSERTION IN WAVL

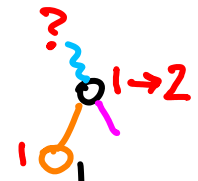
(if only inserting we actually get =AVL thus 1.4 log n height)

- inserted node becomes (non-fake) leaf with score = 1
  (replaces fake leaf, so score was incremented)



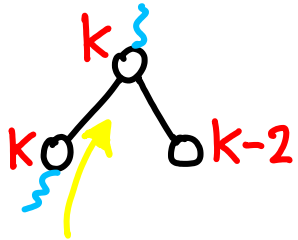
* • if parent score is still greater, DONE

else - if parent score is +1 of sibling



increment parent score (fixed locally) & fix upward *

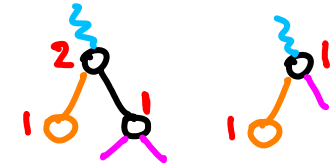
- else (we have same score as parent, which is +2 from sibling)



INSERTION IN WAVL

(if only inserting we actually get = AVL thus 1.4 log n height)

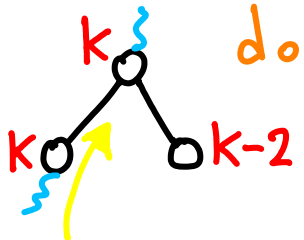
- inserted node becomes (non-fake) leaf with score = 1
  (replaces fake leaf, so score was incremented)



* • if parent score is still greater, DONE

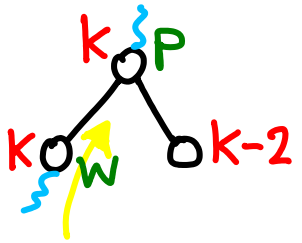
else – if parent score is +1 of sibling
increment parent score (fixed locally) & fix upward *

– else (we have same score as parent, which is +2 from sibling)
do some rotating and we're DONE



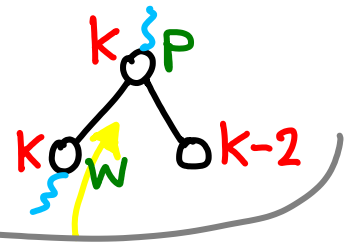
... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE

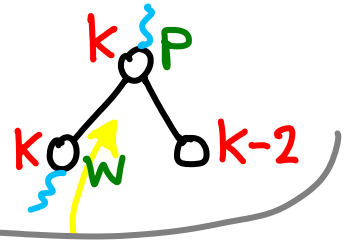


- Observation:

we have a child, C, with relative score -1

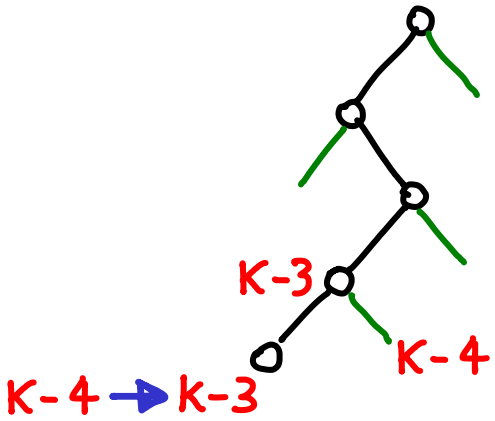
... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

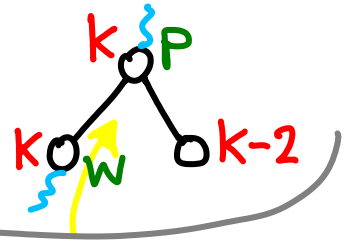


Recall:

if parent score is +1 of sibling → fix upward

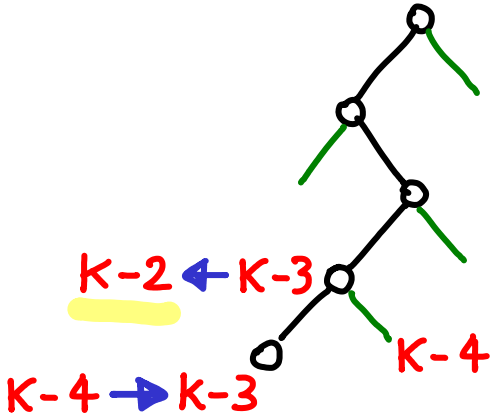
... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

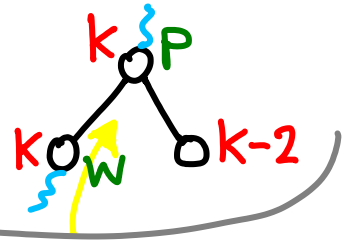


Recall:

if parent score is +1 of sibling → fix upward

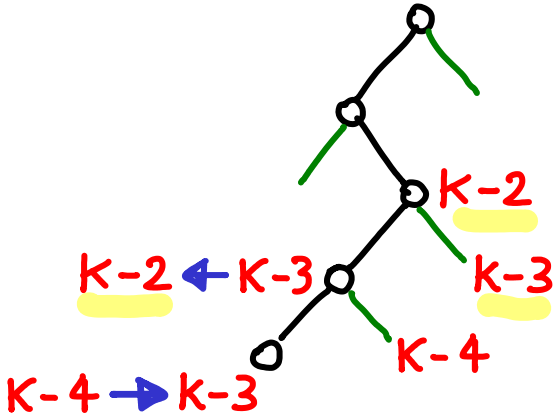
... -else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

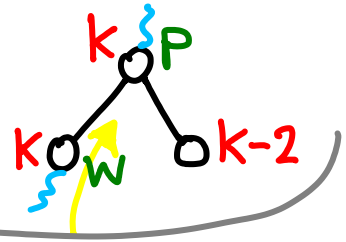


Recall:

if parent score is +1 of sibling → fix upward

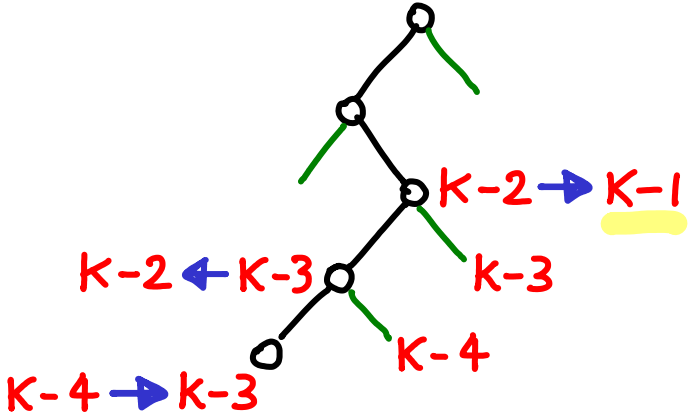
... -else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

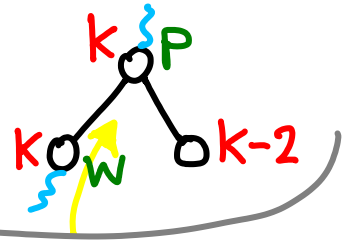


Recall:

if parent score is +1 of sibling → fix upward

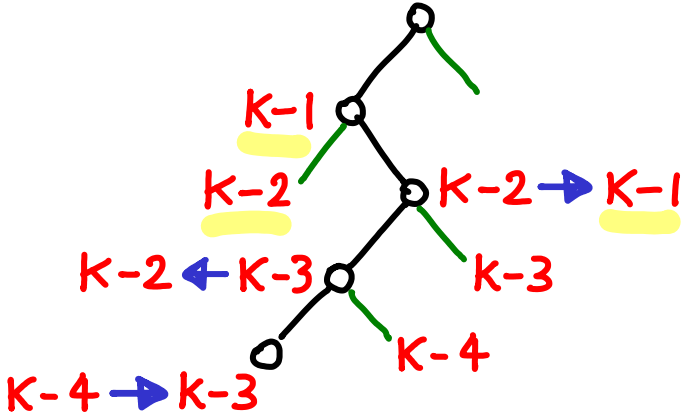
... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

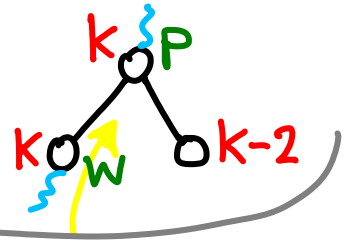


Recall:

if parent score is +1 of sibling → fix upward

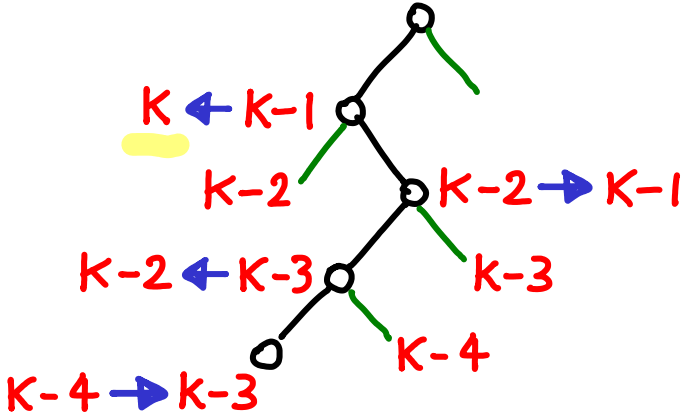
... -else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

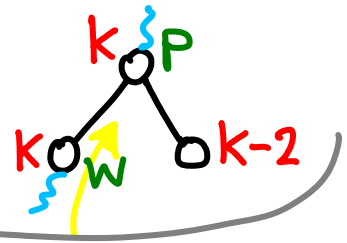


Recall:

if parent score is +1 of sibling → fix upward

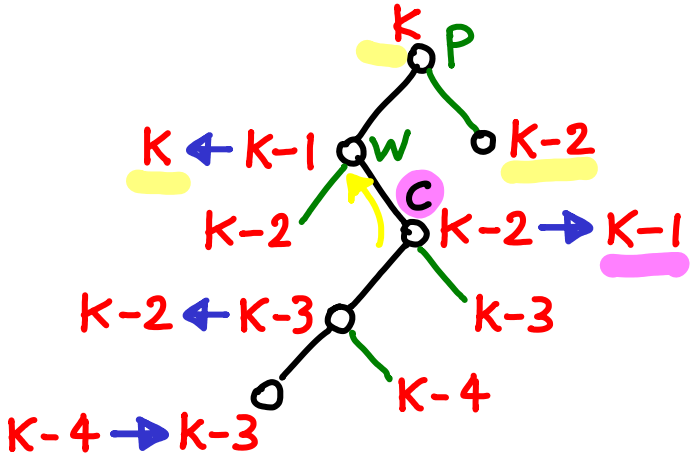
... -else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

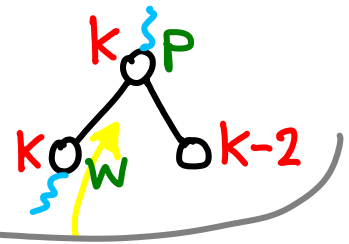


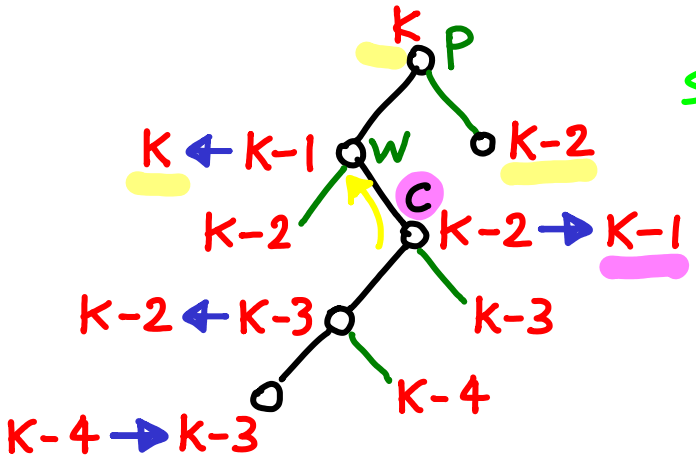
Recall:

if parent score is +1 of sibling → fix upward

...-else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE





Summary

- Observation:

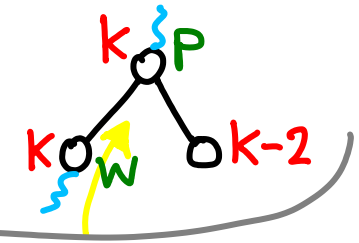
we have a child, C, with relative score -1

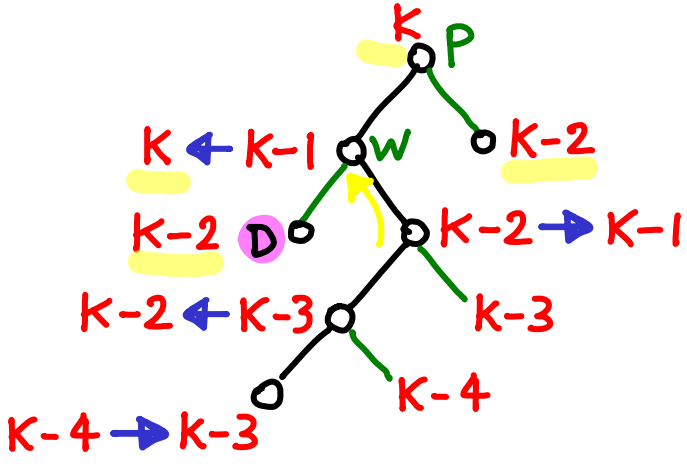
↳ by induction/construction

- true when inserted as leaf
- maintained by "fix upward" case

...-else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE





- Observation:

we have a child, C, with relative score -1

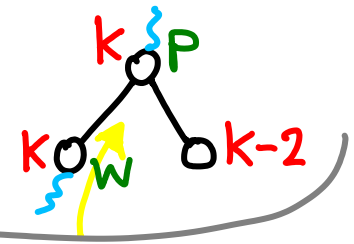
↳ by induction/construction

- true when inserted as leaf
- maintained by "fix upward" case

other child, D, has relative score -2

... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

we have a child, C, with relative score -1

↳ by induction / construction

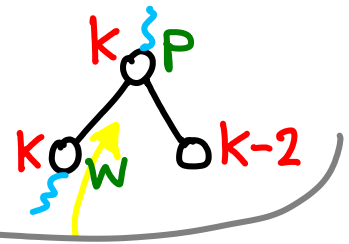
- true when inserted as leaf
- maintained by "fix upward" case

other child, D, has relative score -2

- it already had a lower score
- we incremented our score

...-else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



- Observation:

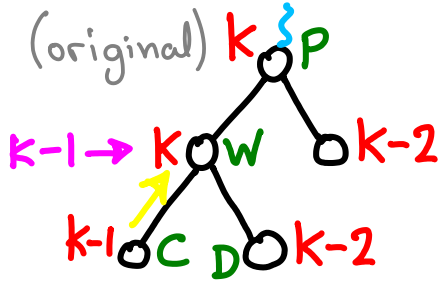
we have a child, C, with relative score -1

↳ by induction/construction

- true when inserted as leaf
- maintained by "fix upward" case

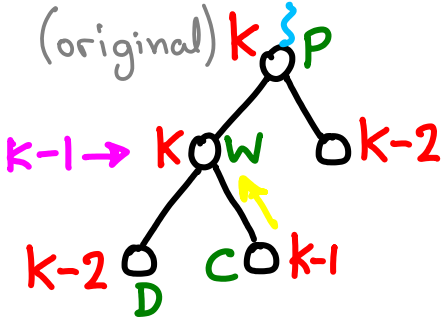
other child, D, has relative score -2

- it already had a lower score
- we incremented our score



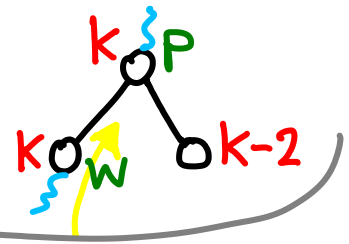
C: left child

C: right child

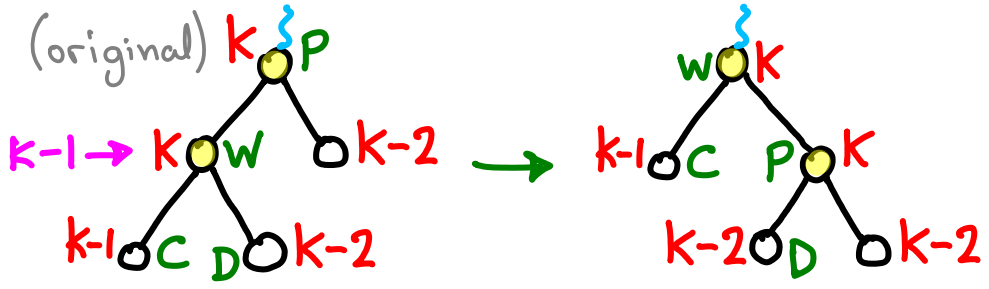


...-else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE

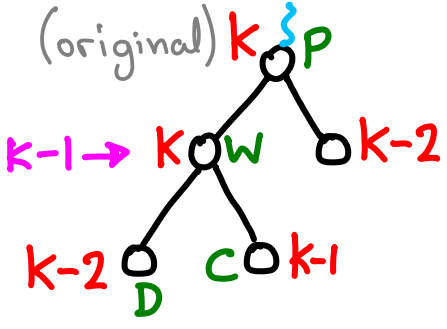


Right-rotate(P)



C: left child

C: right child



- Observation:

we have a child, C, with relative score -1

↳ by induction/construction

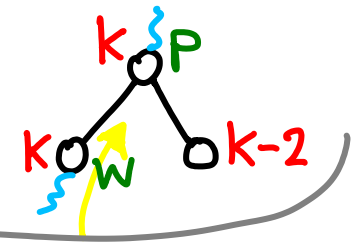
- true when inserted as leaf
- maintained by "fix upward" case

other child, D, has relative score -2

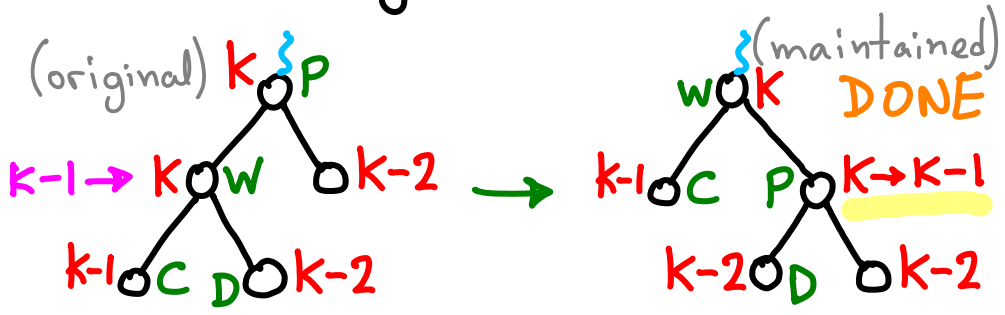
- it already had a lower score
- we incremented our score

...-else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE

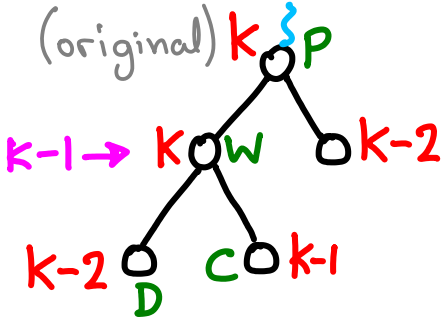


Right-rotate(P)



C: left child

C: right child



- Observation:

we have a child, C, with relative score -1

↳ by induction/construction

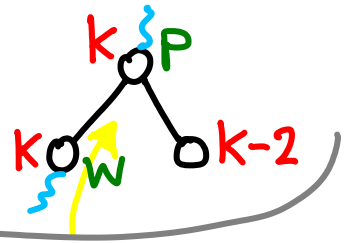
- true when inserted as leaf
- maintained by "fix upward" case

other child, D, has relative score -2

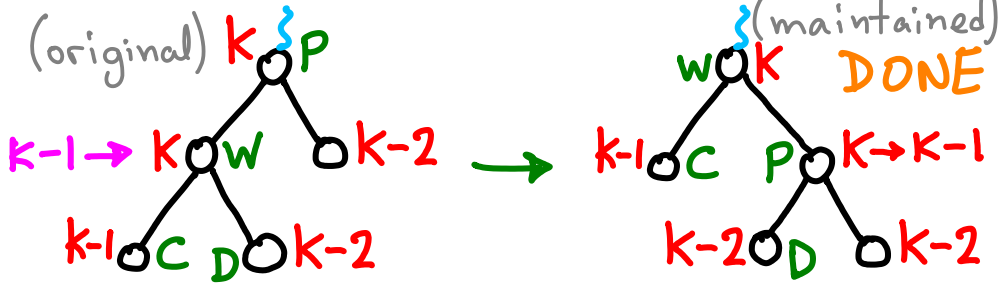
- it already had a lower score
- we incremented our score

... -else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE

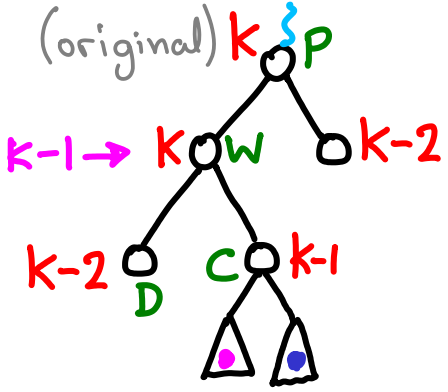


Right-rotate(P)



C: left child

C: right child



- Observation:

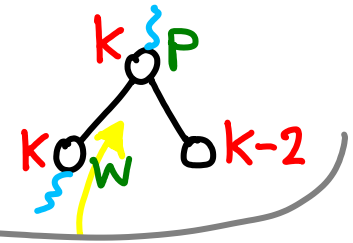
we have a child, C, with relative score -1

↳ by induction / construction
- true when inserted as leaf
- maintained by "fix upward" case

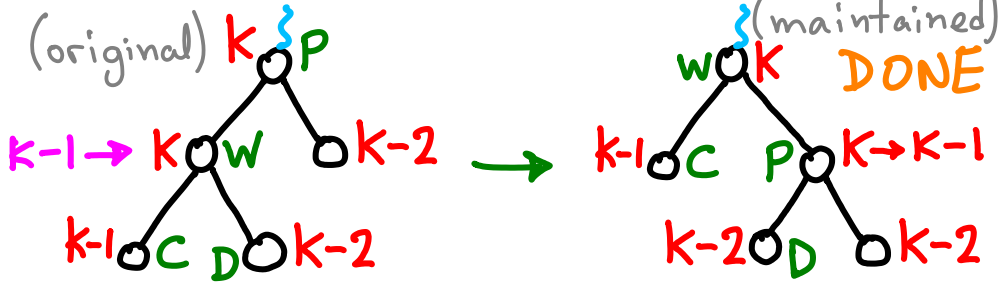
other child, D, has relative score -2
- it already had a lower score
- we incremented our score

... -else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



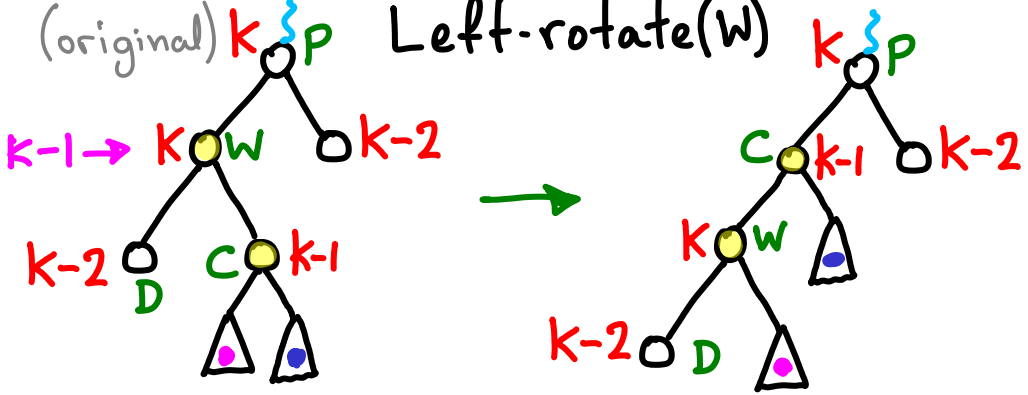
Right-rotate(P)



C: left child

C: right child

(original) k P Left-rotate(W)



- Observation:

we have a child, C, with relative score -1

↳ by induction / construction

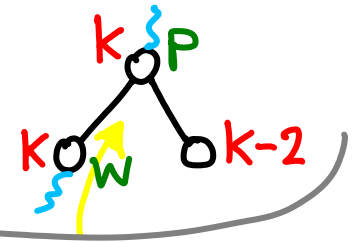
- true when inserted as leaf
- maintained by "fix upward" case

other child, D, has relative score -2

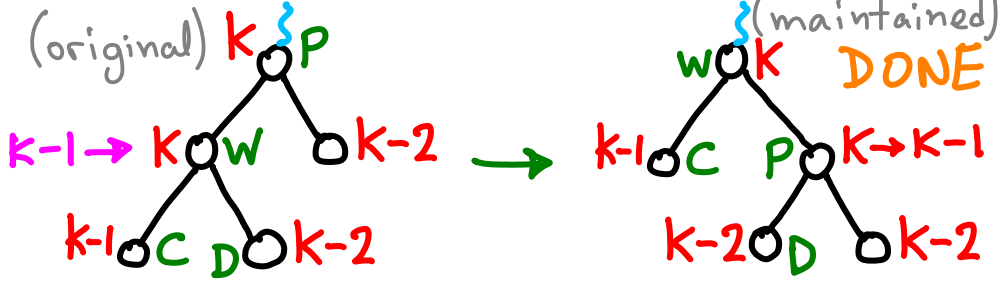
- it already had a lower score
- we incremented our score

... -else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



Right-rotate(P)



C: left child

C: right child

- Observation:

we have a child, C, with relative score -1

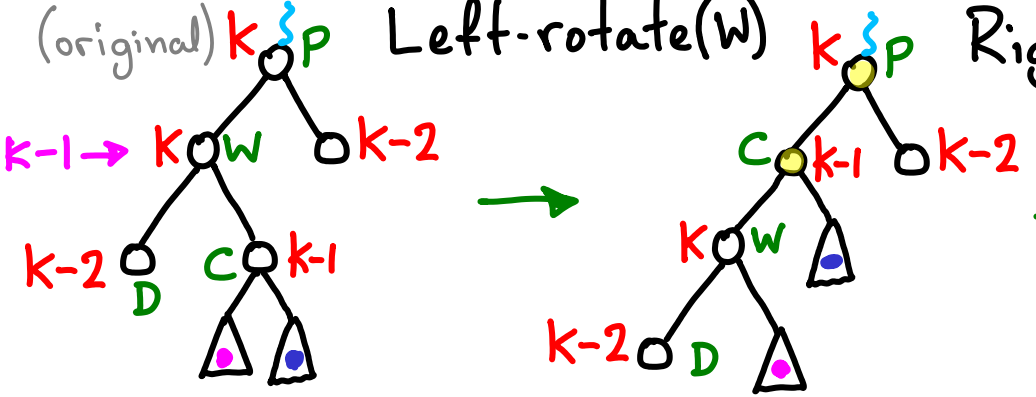
↳ by induction / construction

- true when inserted as leaf
- maintained by "fix upward" case

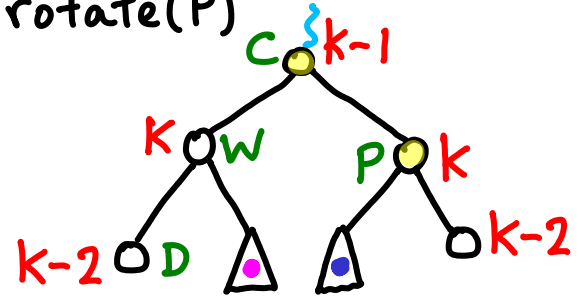
other child, D, has relative score -2

- it already had a lower score
- we incremented our score

Left-rotate(W)

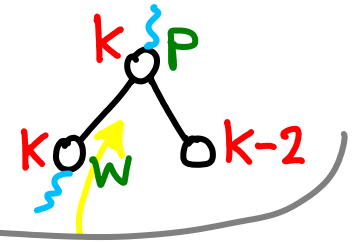


Right-rotate(P)

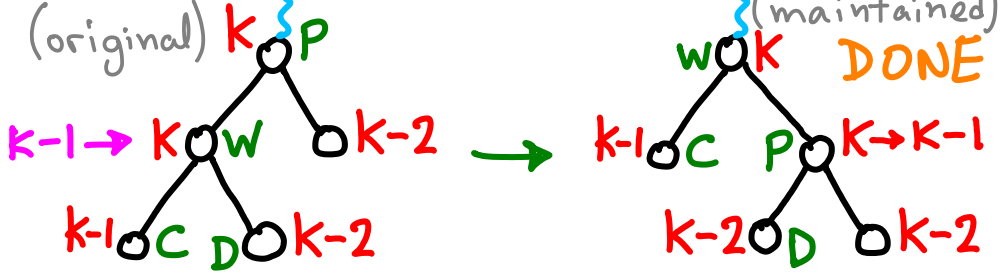


... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



Right-rotate(P)



C: left child

C: right child

- Observation:

we have a child, C, with relative score -1

↳ by induction / construction

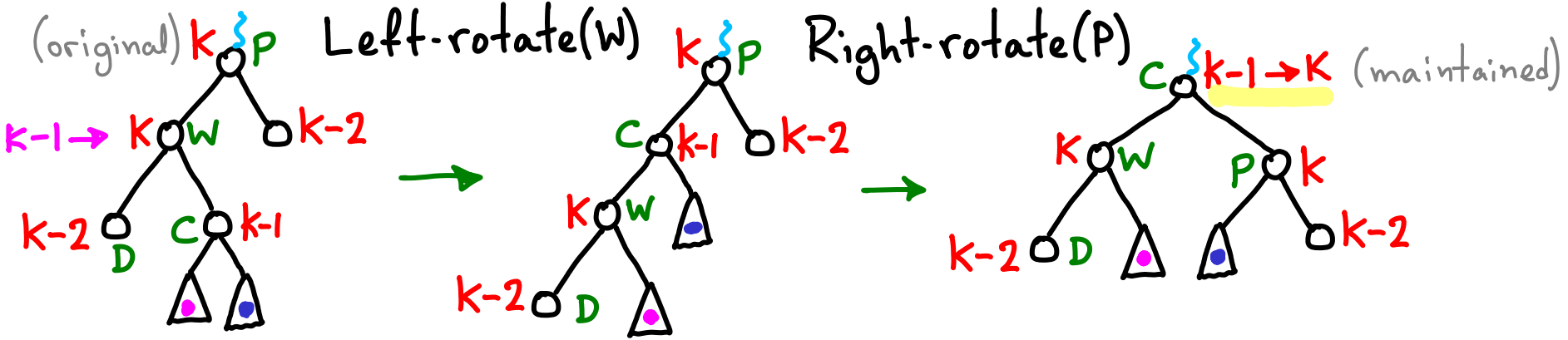
- true when inserted as leaf
- maintained by "fix upward" case

other child, D, has relative score -2

- it already had a lower score
- we incremented our score

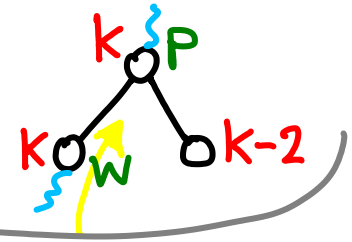
Left-rotate(W)

Right-rotate(P)

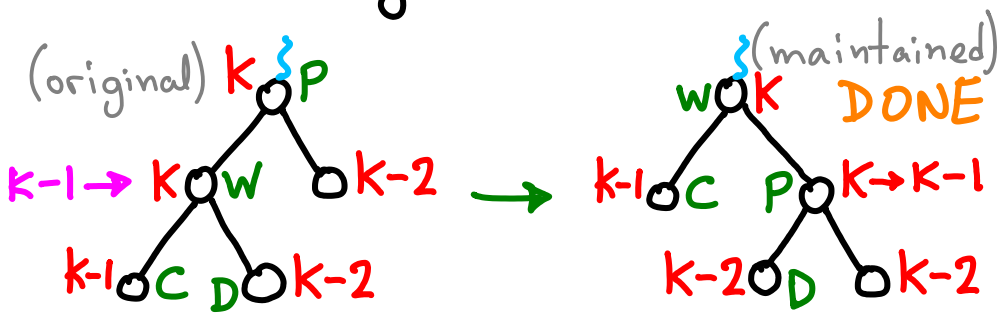


... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



Right-rotate(P)



C: left child

C: right child

• Observation:

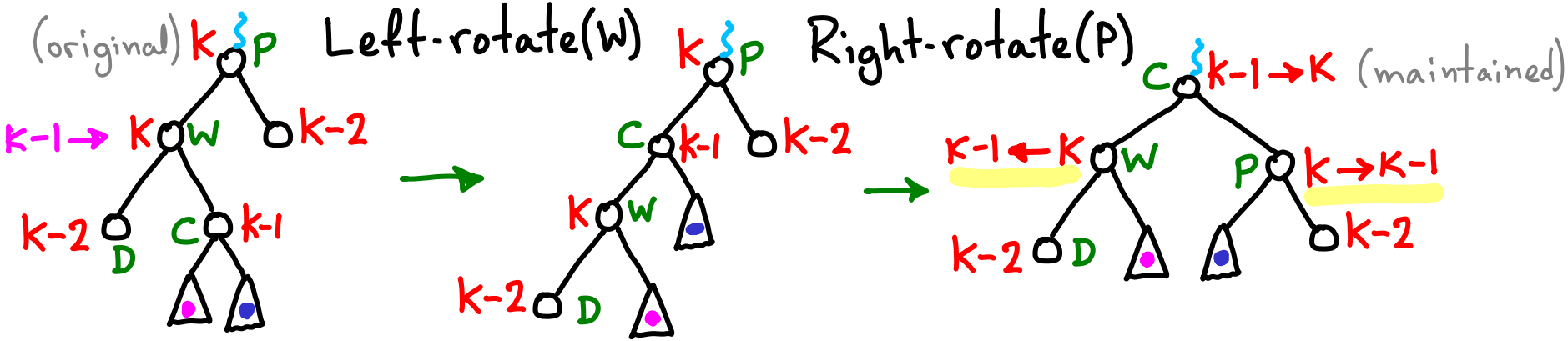
we have a child, C, with relative score -1

↳ by induction / construction

- true when inserted as leaf
- maintained by "fix upward" case

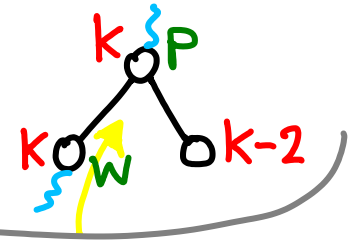
other child, D, has relative score -2

- it already had a lower score
- we incremented our score

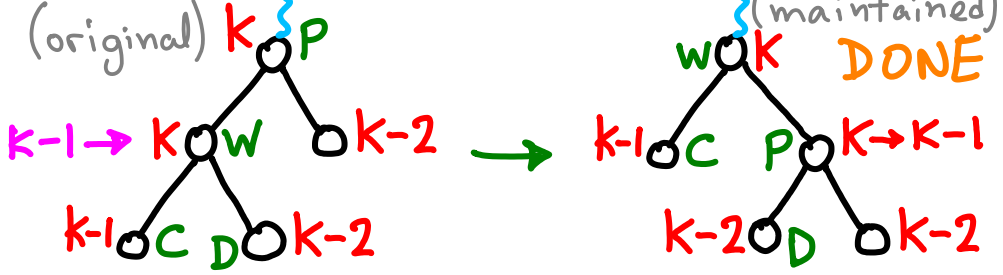


... - else (we have same score as parent, which is +2 from sibling)

do some rotating and we're DONE



Right-rotate(P)



C: left child

C: right child

- Observation:

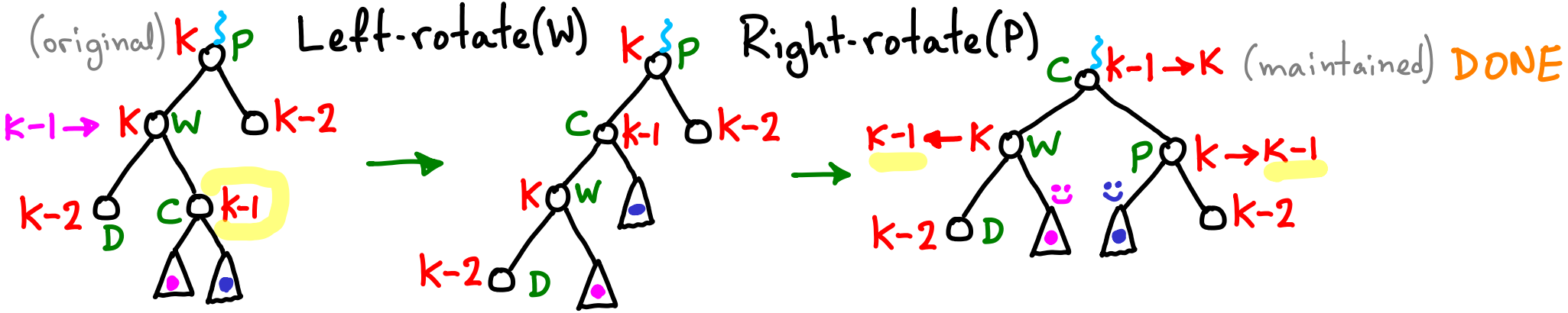
we have a child, C, with relative score -1

↳ by induction / construction

- true when inserted as leaf
- maintained by "fix upward" case

other child, D, has relative score -2

- it already had a lower score
- we incremented our score



WAVL deletion is also straightforward

worst case	AVL	RB	WAVL
height	1.4 log n	2 log n	2 log n 1.4 log n insert only
insert # rotations	O(1)	O(1)	O(1) better than RB
delete # rotations	O(log n)	O(1)	O(1)