Bottom-up Rebalancing Binary Search Trees by Flipping a Coin

Gerth Stølting Brodal

Aarhus University

12th International Conference on Fun with Algorithms (FUN 2024), Island of La Maddalena, Sardinia, Italy, June 4-8, 2024

Review

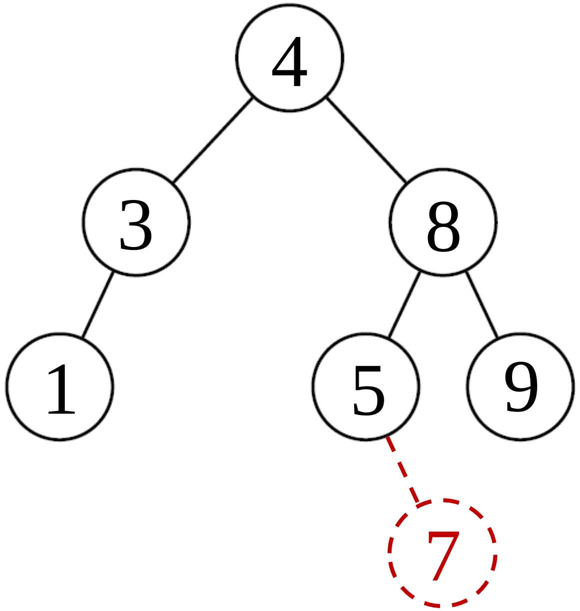
The paper considers an important problem…

The paper fails to solve the problem…

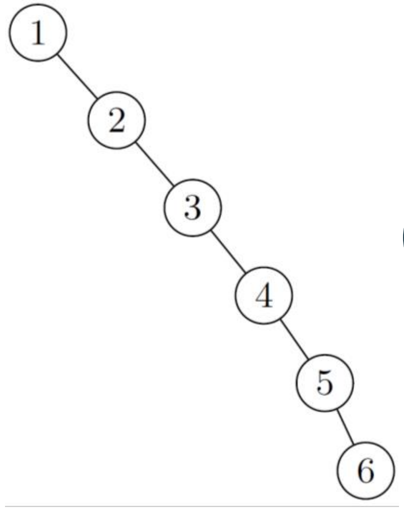
Unbalanced binary search trees – Insertions

1) Locate **insertion point** (empty leaf)

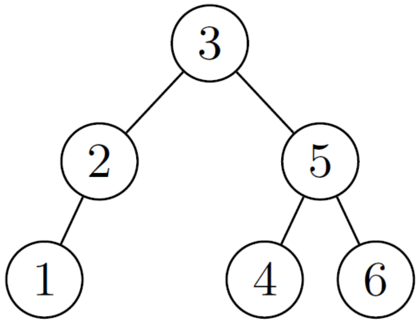
2) Create new leaf node



Unbalanced binary search trees



Inserting 1 2 3 4 5 6



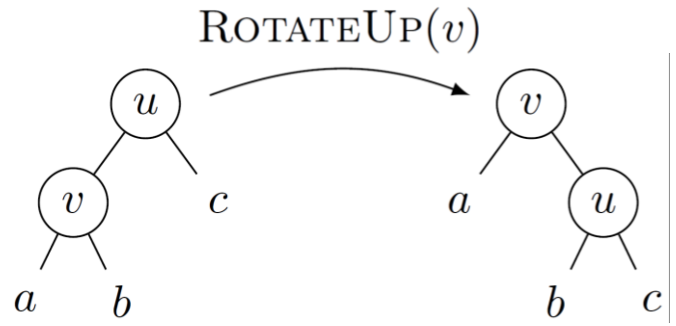
Inserting 3 5 2 4 1 6

increasing sequence gives linear depth

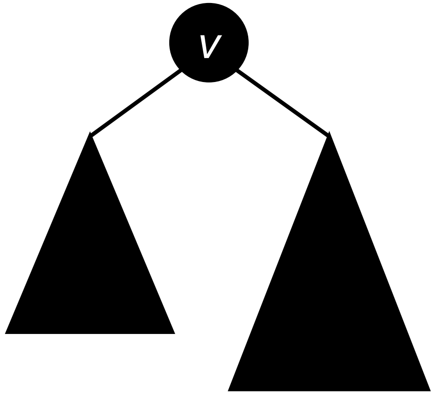
random permutation gives expected logarithmic depth [Hilbard 1962]

Balanced binary search trees

- Structural **invariants** implying logarithmic depth
- Rebalance using **rotations**

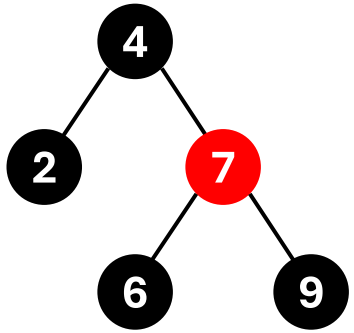


60s
AVL-tree



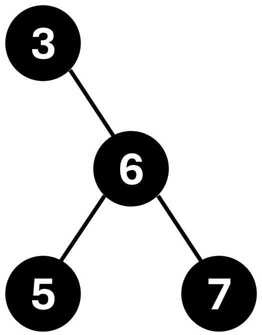
$|h(v.left) - h(v.right)| \leq 1$

70s
Red-black tree



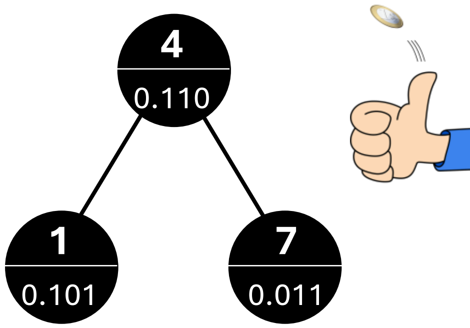
No two adjacent red &
root-to-leaf paths same #black

80s
Splay tree



Always rotate accessed
node to the root

90s
Treap



Elements random priority,
stored using heap order

Balanced binary search tree insertions : rebalancing cost

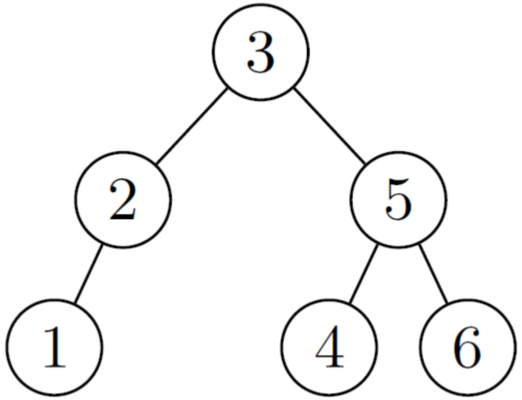
Reference	Name	Time	Rotations	Random bits	Space pr node
[AVL62]	AVL-trees	$O_A(1)$	O(1)	0	2
[GS78]	Red-black trees	$O_A(1)$	O(1)	0	1
[B79]	Encoded 2-3 trees	$O(\lg n)$	$O(\lg n)$	0	0
[ST85]	Splay trees	$O_A(\lg n)$	$O_A(\lg n)$	0	0
[A89,GR93]	Scapegoat	$O(\lg n)$	$O(\lg n)$	0	global $O(\lg n)$
[SA96]	Treaps	$O_E(1)$	$O_E(1)$	$O_E(1)$	$O_E(1)$
[MR98]	Randomized BST	$O_E(\lg n)$	$O_E(1)$	$O_E(\lg^2 n)$	$O(\lg n)$
[S09]	Seidel	$O_E(\lg^2 n)$	$O_E(1)$	$O_E(\lg^3 n)$	0
	Open problem	$O_E(1)$	$O(1)$	$O_E(1)$	0

random BST

Question asked in this paper

Does there exists a randomized rebalancing scheme satisfying...

1. No balance information stored
2. Worst-case O(1) rotations
3. Most rotations near the insertion
4. Local information only
5. Expected O(1) time
6. O(1) random bits per insertion
7. Nodes expected depth O(lg *n*)

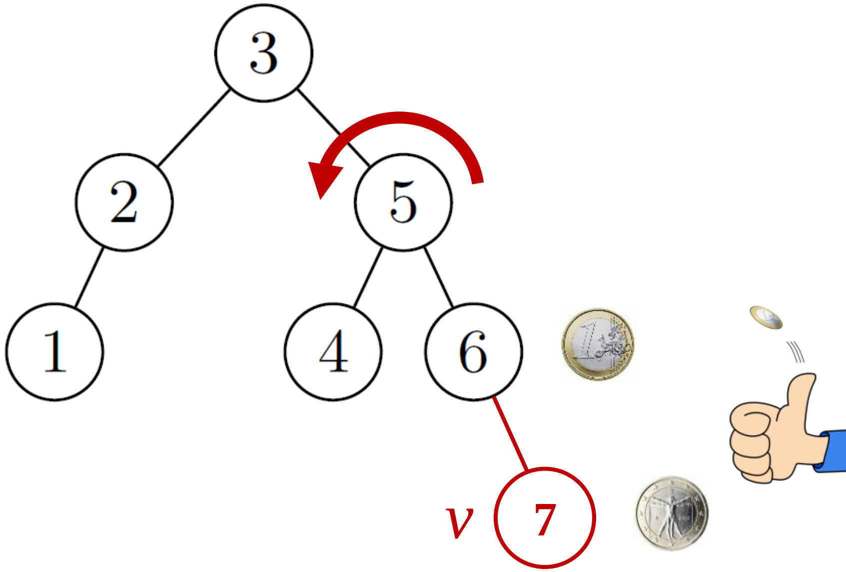


Algorithm RebalanceZig

REBALANCEZIG(v)

---

while $v.p \neq$ NIL **and** coin flip is tail **do**
  $v \leftarrow v.p$
if $v.p \neq$ NIL **then**
  ROTATEUP(v)



Fact 1 REBALANCEZIG takes expected O(1) time and performs ≤ 1 rotation

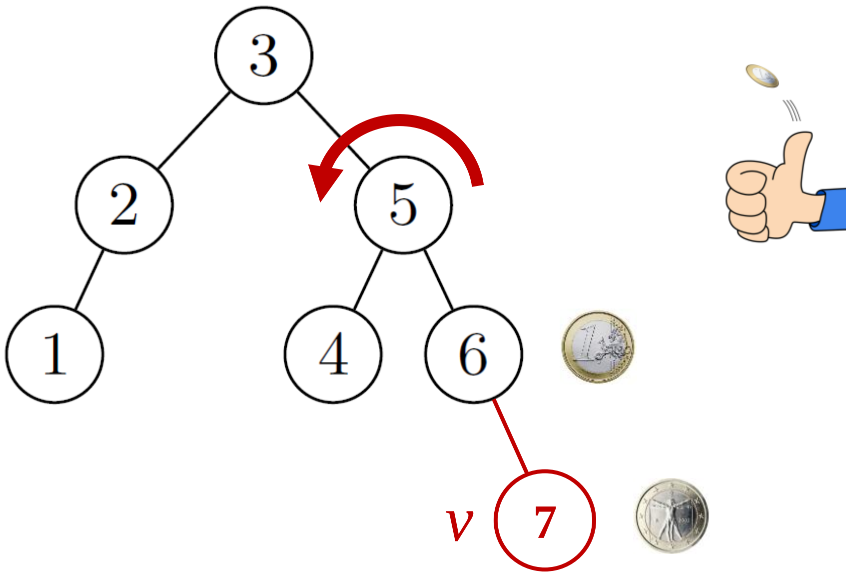
Theorem 1 REBALANCEZIG on **increasing** sequence each node expected depth O(lg n), for 0 < *p* < 1 (p = Pr[tail])

Algorithm RebalanceZig extreme probabilities

REBALANCEZIG(v)

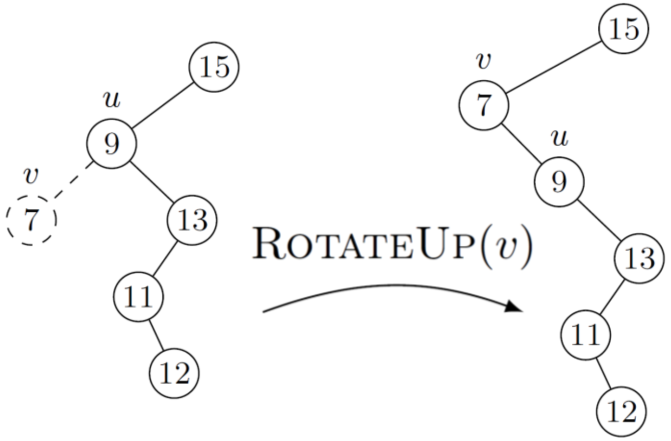
---

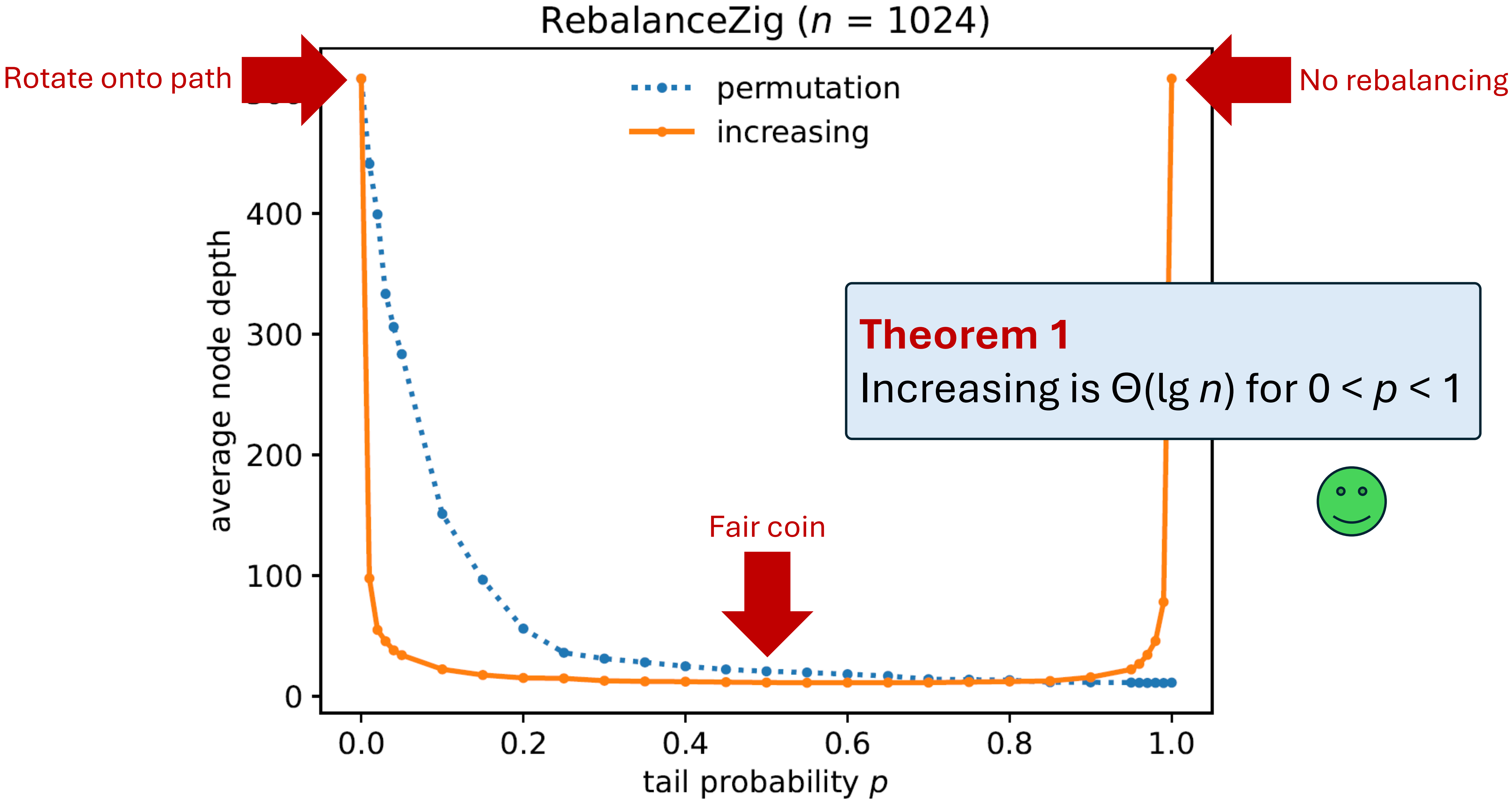
while $v.p \neq$ NIL **and** coin flip is tail **do**
  $v \leftarrow v.p$
if $v.p \neq$ NIL **then**
  ROTATEUP(v)



- **Pr[tail] = 1** never performs rotation
  i.e. unbalanced binary search tree

- **Pr[tail] = 0** always rotates up new leaf
  i.e. tree is always a path

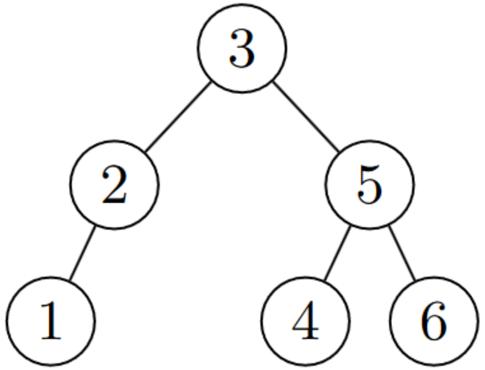




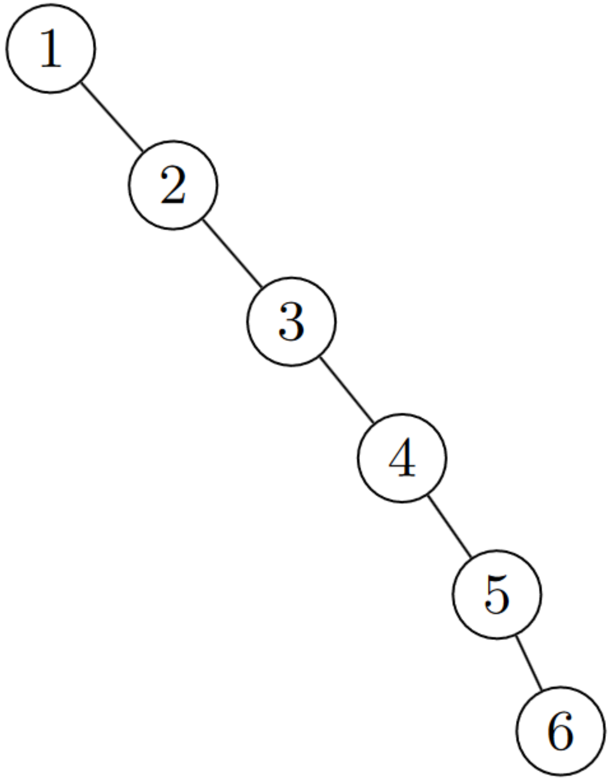
Theorem 1

Increasing is $\Theta(\lg n)$ for $0 < p < 1$

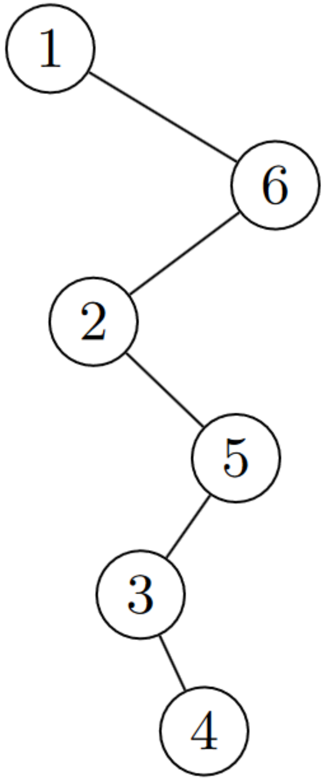
Different insertion sequences



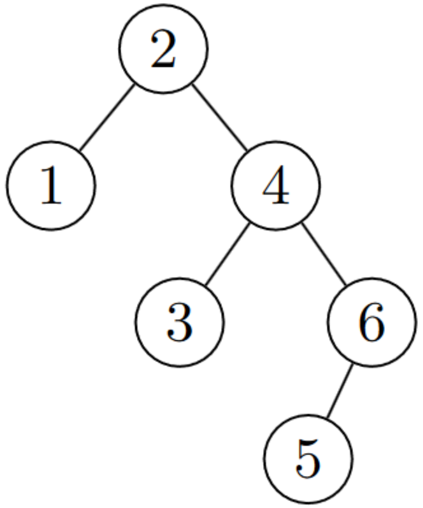
3 5 2 4 1 6
(permutation)



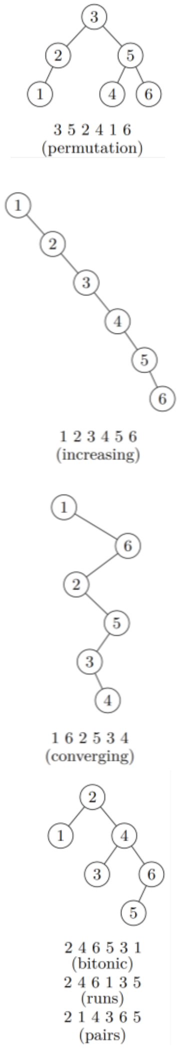
1 2 3 4 5 6
(increasing)

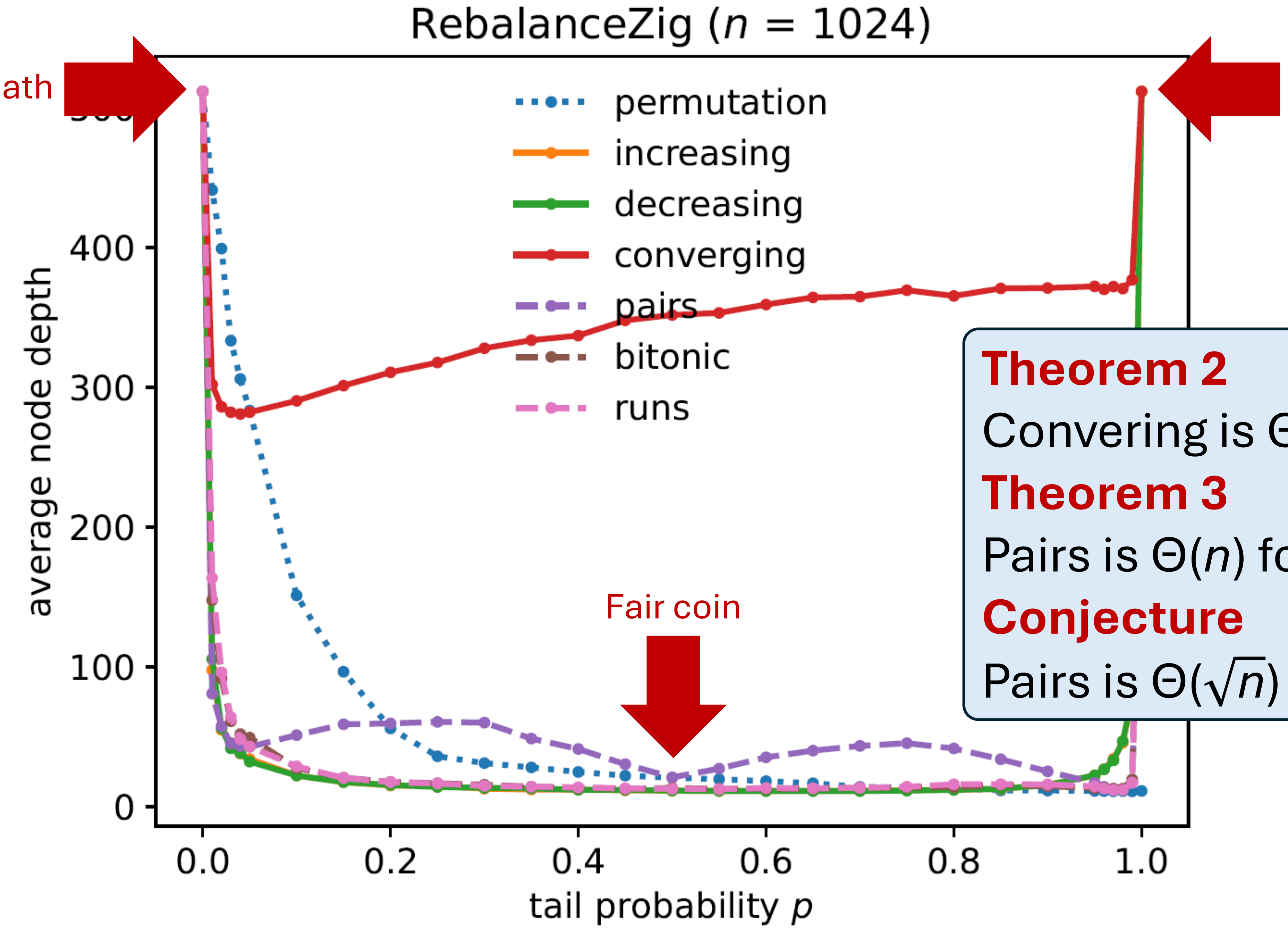


1 6 2 5 3 4
(converging)



2 4 6 5 3 1
(bitonic)
2 4 6 1 3 5
(runs)
2 1 4 3 6 5
(pairs)





Theorem 2

Convering is $\Theta(n)$

Theorem 3

Pairs is $\Theta(n)$ for $p \neq ½$

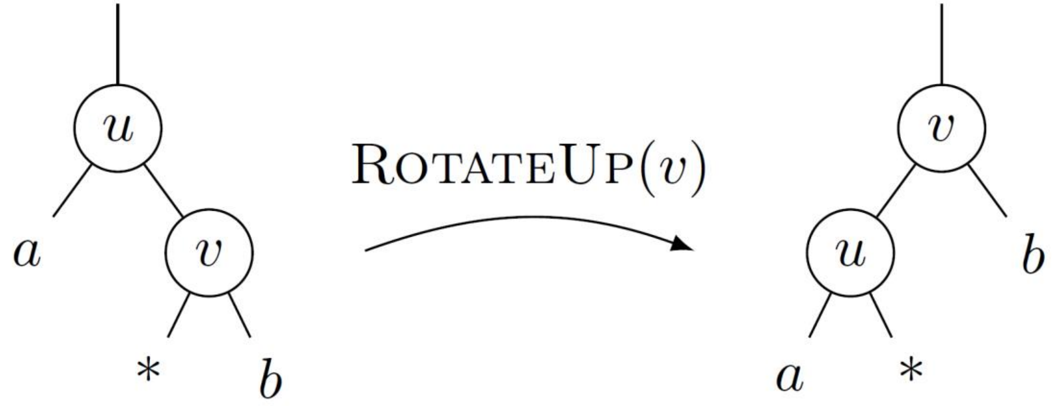
Conjecture

Pairs is $\Theta(\sqrt{n})$ for $p = ½$

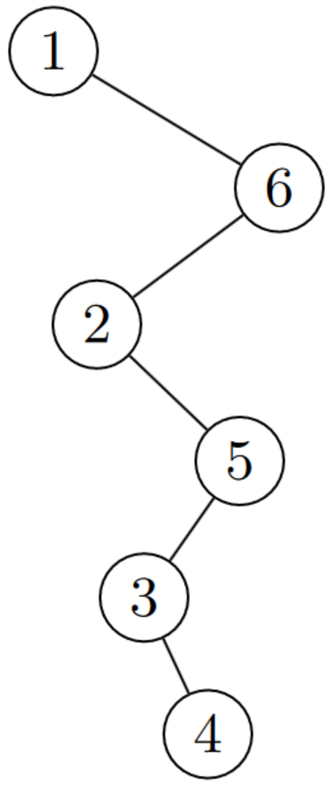
Why ReblanceZig is bad on convering

“Proof” of Theorem 2

- For each pair the depth of the insertion point increases by ≥ 1 with probability ≥ $p(1\text{-}p)$:

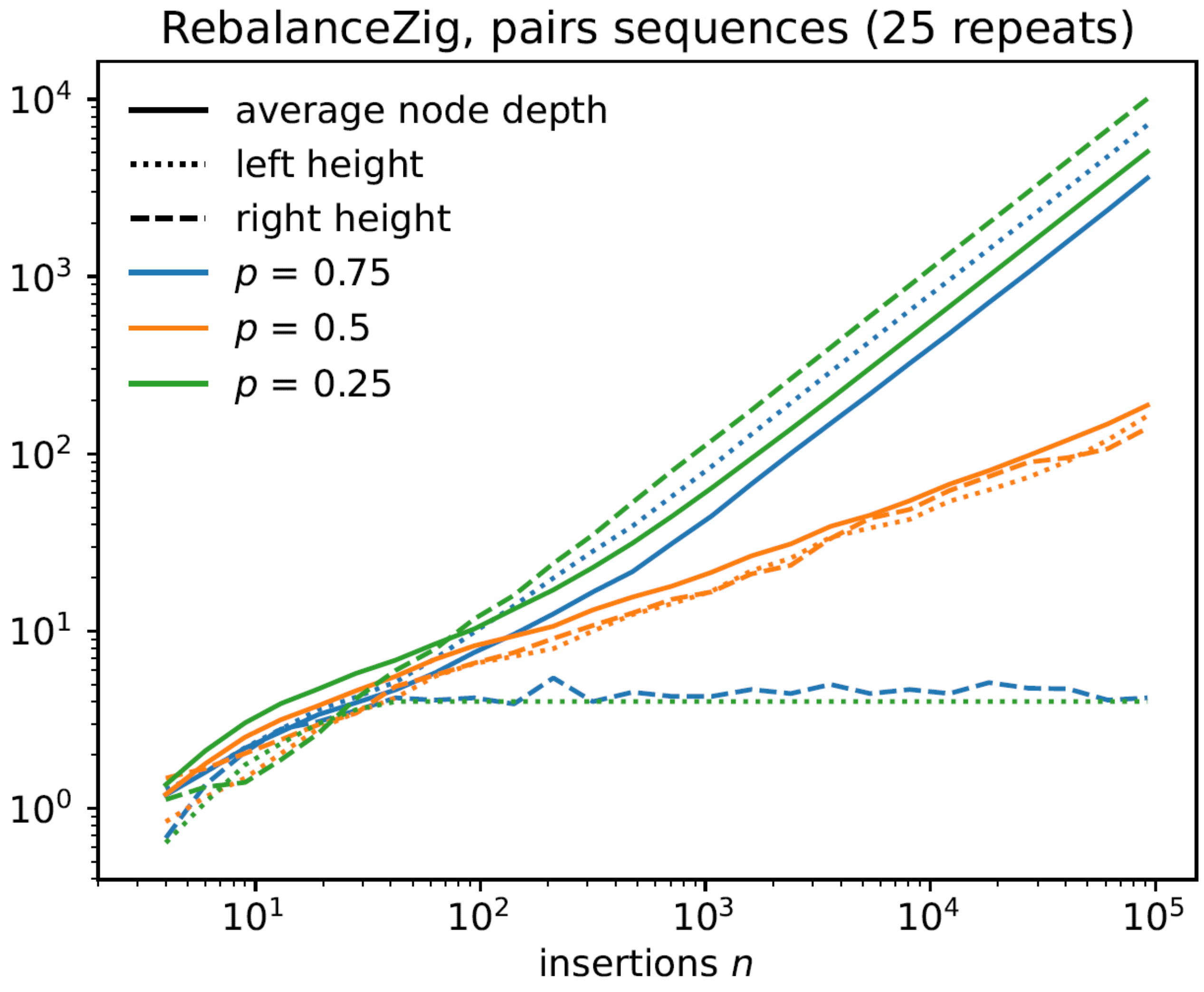


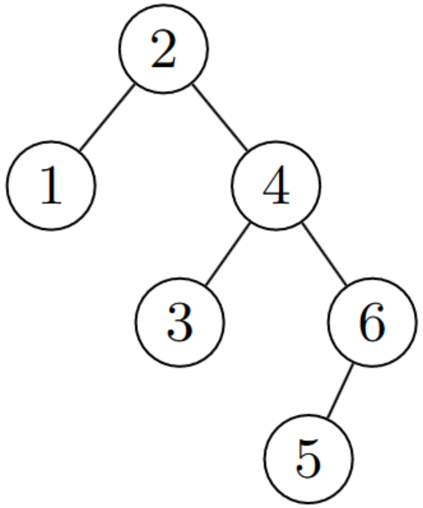
If inserting u rotates a (strict) ancestor of u up, and inserting v rotates v up, then depth of insertion point * increases by inserting the pair



1 6 2 5 3 4
(converging)

Why ReblanceZig is bad on pairs

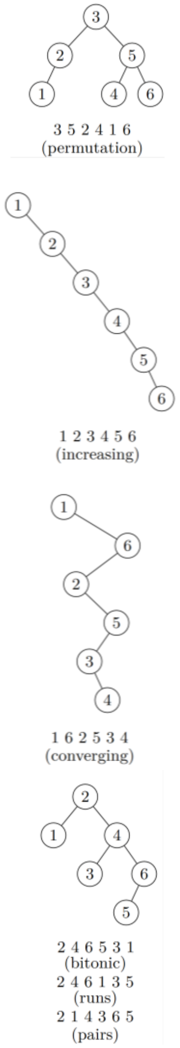


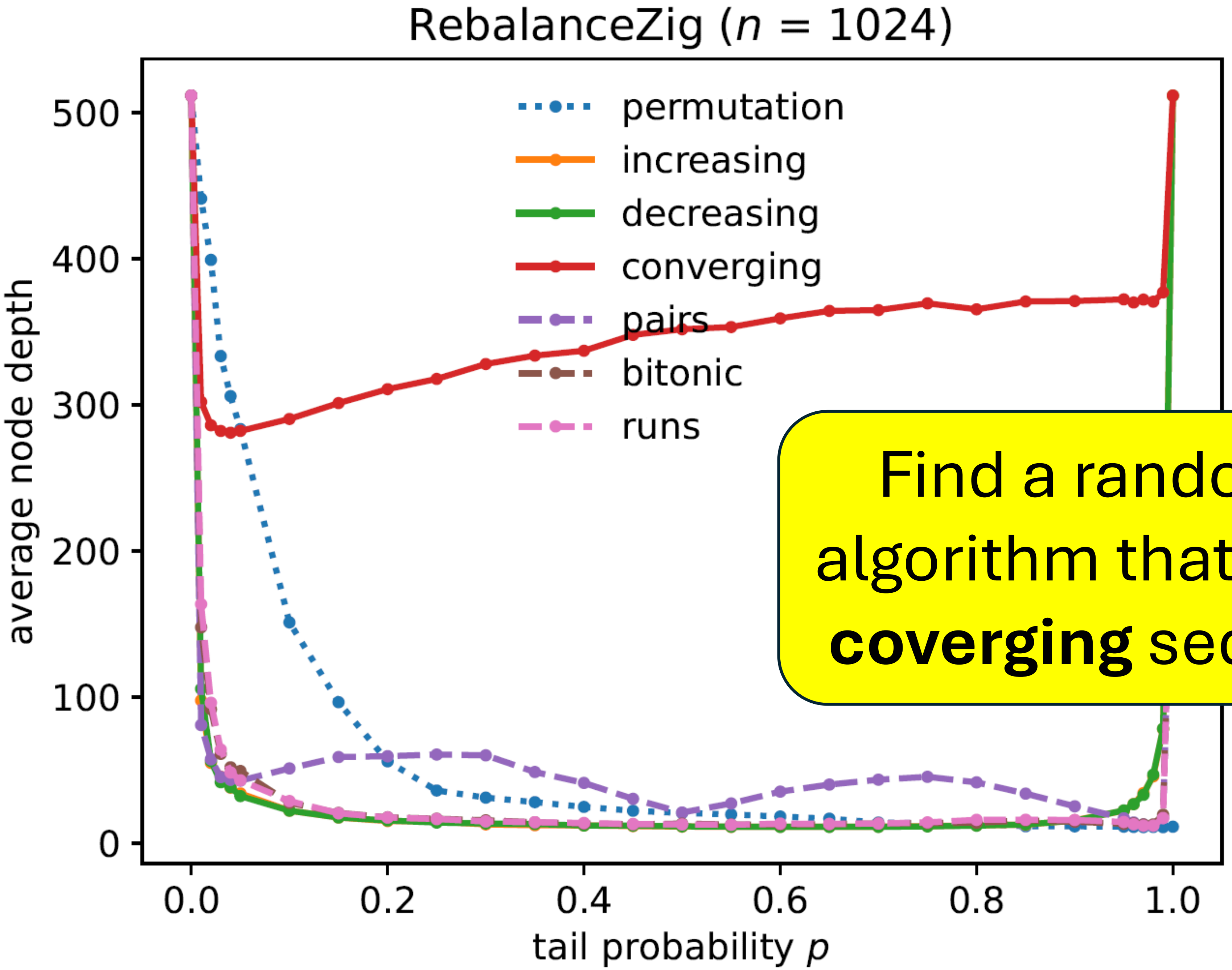


2 1 4 3 6 5
(pairs)

"Proof" of Theorem 3

- $p < ½$, odd numbers tend to be rotated on rightmost path, right path expected $\Theta(n)$ nodes
- $p > ½$, rightmost path tends to contain $O(1)$ nodes, left path expected $\Theta(n)$ nodes





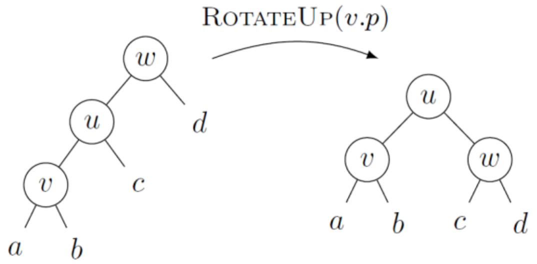
Find a randomized algorithm that handles **coverging** sequences

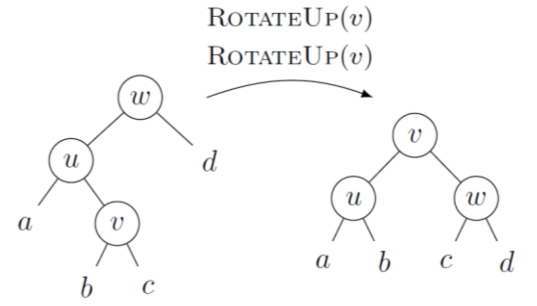
Algorithm RebalanceZigZag

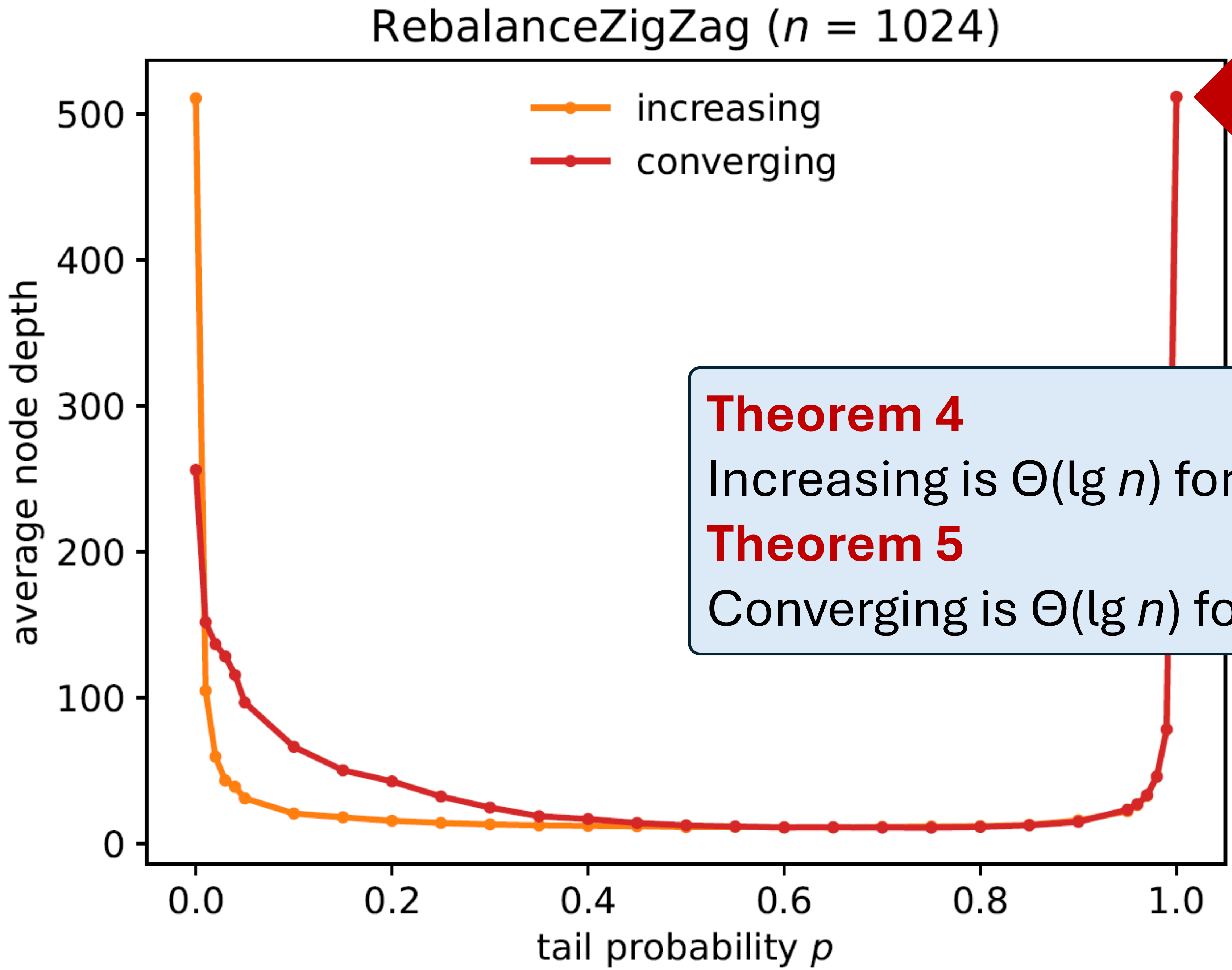
RebalanceZigZag(v)

---

while $v.p \neq$ NIL **and** coin flip is tail **do**
    $v \leftarrow v.p$
if $v.p \neq$ NIL **and** $v.p.p \neq$ NIL **then**
    if $(v = v.p.l$ **and** $v.p = v.p.p.l)$ **or** $(v = v.p.r$ **and** $v.p = v.p.p.r)$ **then**
        RotateUp($v.p$)   ▷ *zig-zig or zag-zag case*
    else
        RotateUp(v)   ▷ *zig-zag or zag-zig case*
        RotateUp(v)







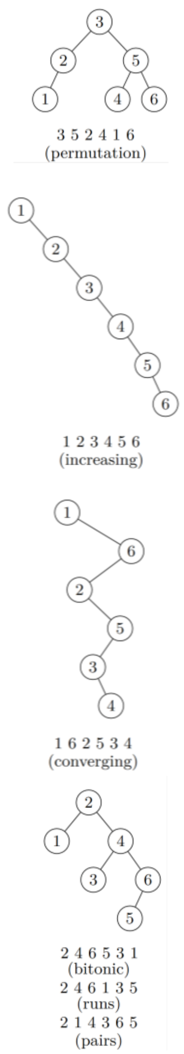
No rebalancing

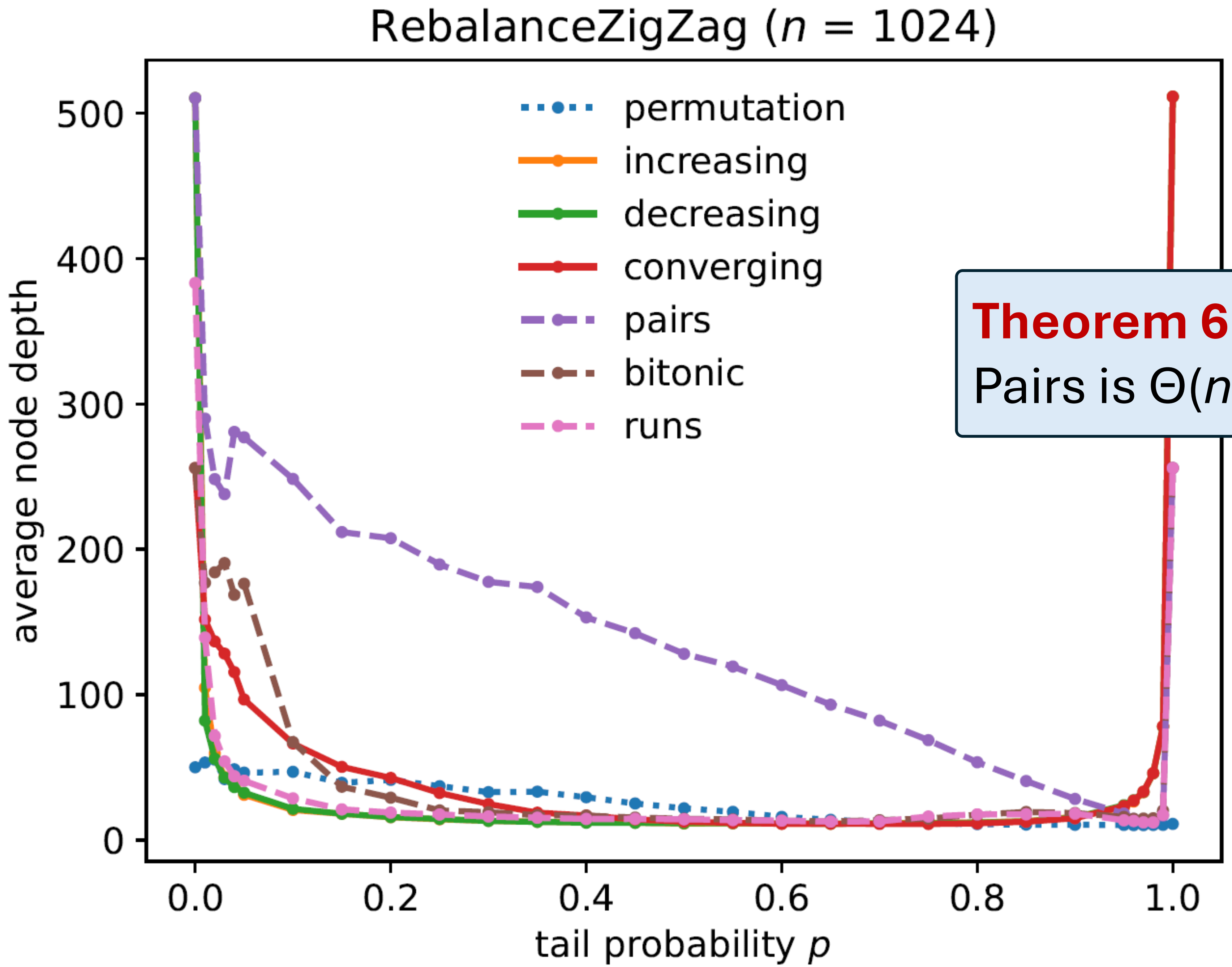
Theorem 4

Increasing is $\Theta(\lg n)$ for $0 < p < 1$

Theorem 5

Converging is $\Theta(\lg n)$ for $0.62 < p < 1$

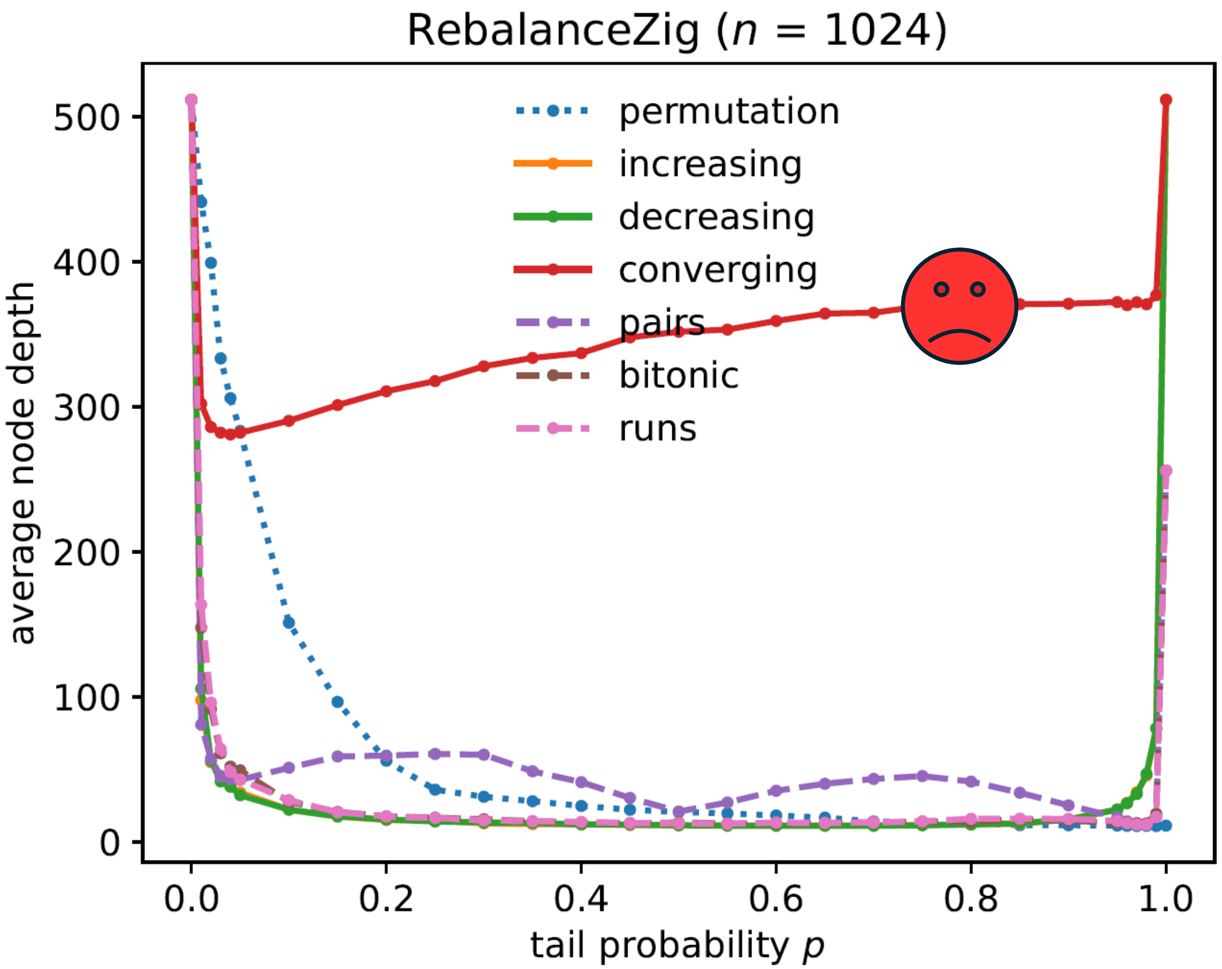


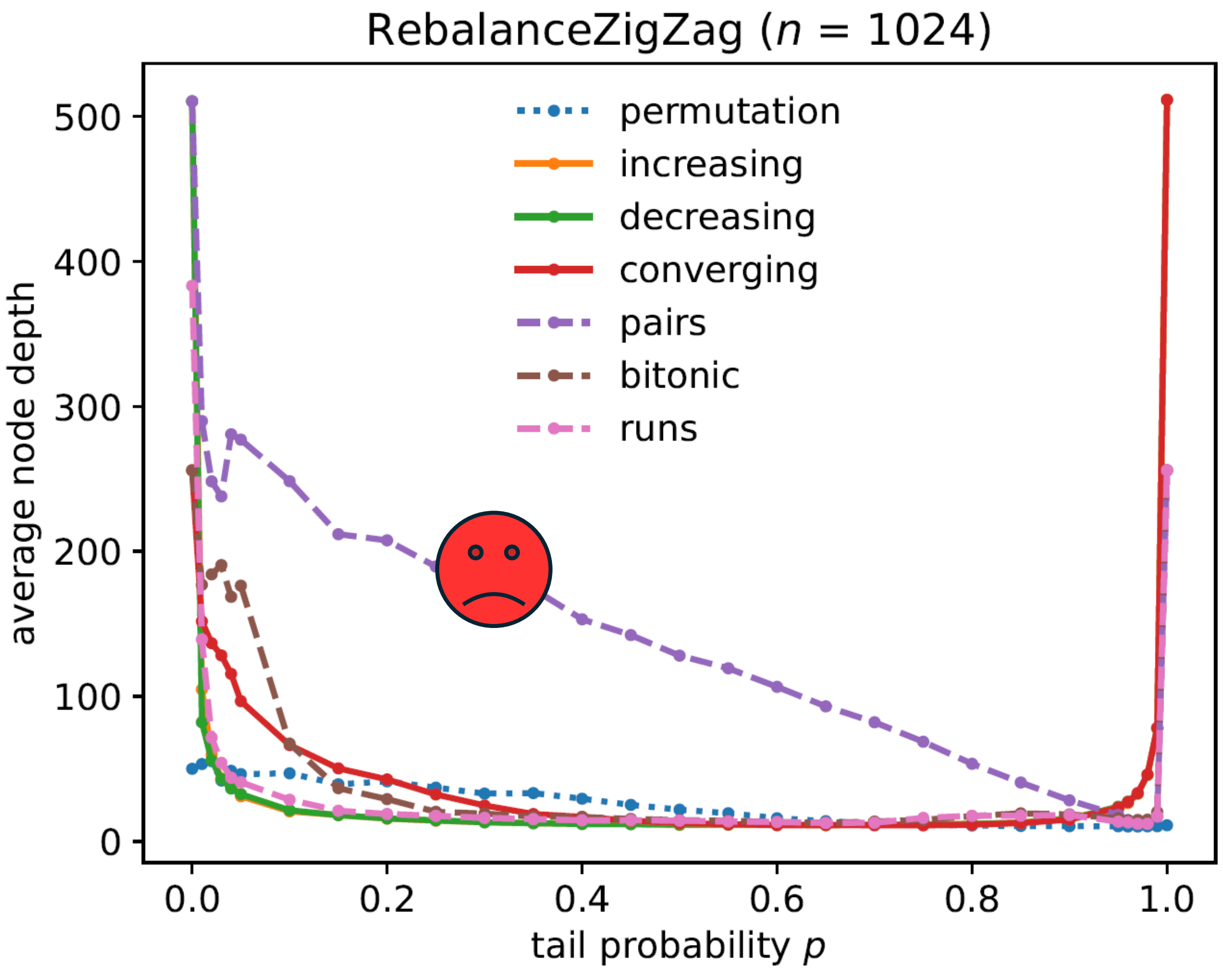


Theorem 6

Pairs is $\Theta(n)$ for $0 \leq p \leq 1$

Summary


RebalanceZig (n = 1024)
permutation
increasing
decreasing
converging
pairs
bitonic
runs
average node depth
tail probability p
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0


RebalanceZigZag (n = 1024)
permutation
increasing
decreasing
converging
pairs
bitonic
runs
average node depth
tail probability p
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0

Bottom-up Rebalancing Binary Search Trees by Flipping a Coin

Bottom-up Rebalancing Balanced Binary Search Trees by Flipping a Coin

?