# k-AVL Trees

## An arbitrary balance factor tree

Advanced topics in Data Structures

Hostile Robot Productions

April 7, 2023

Focus

# AVL Background

# AVL Background

Invented in 1962, coinciding with the invention of the printing press and cave painting.

# AVL Background

Invented in 1962, coinciding with the invention of the printing press and cave painting.

$$\underbrace{\text{AV}}_{\substack{\text{Georgy} \\ \text{Adelson-Velsky}}}$$

# AVL Background

Invented in 1962, coinciding with the invention of the printing press and cave painting.

$$\underbrace{\textbf{AV}}_{\text{Georgy Adelson-Velsky}}\ \underbrace{\textbf{L}}_{\text{Evgenii Landis}}$$

# AVL Background

Invented in 1962, coinciding with the invention of the printing press and cave painting.

$$\underbrace{\text{k-}}_{\substack{\text{Zak}\\ \text{Kieda}}}\underbrace{\text{AV}}_{\substack{\text{Georgy}\\ \text{Adelson-Velsky}}}\underbrace{\text{L}}_{\substack{\text{Evgenii}\\ \text{Landis}}}$$

# AVL Background

Invented in 1962, coinciding with the invention of the printing press and cave painting.

$$\underbrace{\text{k-}}_{\substack{\text{Zak}\\ \text{Kieda}}}\underbrace{\text{AV}}_{\substack{\text{Georgy}\\ \text{Adelson-Velsky}}}\underbrace{\text{L}}_{\substack{\text{Evgenii}\\ \text{Landis}}}$$

Also $k$ is a pretty cool placeholder for the maximum balance factor like $\sqrt{n}$-AVL tree.

# Standard AVL Tree

# Standard AVL Tree

Self balancing binary search tree with balance factor BF on each node

Self balancing binary search tree with balance factor BF on each node

$$\mathrm{BF}(X) := \mathrm{Height}(\mathrm{Right}(X)) - \mathrm{Height}(\mathrm{Left}(X)) \tag{1}$$
$$\mathrm{BF}(X) \in \{-1, 0, 1\} \tag{2}$$

# Standard AVL Tree

Self balancing binary search tree with balance factor BF on each node

$$\mathrm{BF}(X) := \mathrm{Height}(\mathrm{Right}(X)) - \mathrm{Height}(\mathrm{Left}(X)) \tag{1}$$

$$\mathrm{BF}(X) \in \{-1, 0, 1\} \tag{2}$$

- Property (2) is preserved after any operation

# Standard AVL Tree

Self balancing binary search tree with balance factor BF on each node

$$\mathrm{BF}(X) := \mathrm{Height}(\mathrm{Right}(X)) - \mathrm{Height}(\mathrm{Left}(X)) \tag{1}$$

$$\mathrm{BF}(X) \in \{-1, 0, 1\} \tag{2}$$

- Property (2) is preserved after any operation
- Tree balances through a series of rotations

Self balancing binary search tree with balance factor BF on each node

$$\mathrm{BF}(X) := \mathrm{Height}(\mathrm{Right}(X)) - \mathrm{Height}(\mathrm{Left}(X)) \tag{1}$$

$$\mathrm{BF}(X) \in \{-1, 0, 1\} \tag{2}$$

- Property (2) is preserved after any operation
- Tree balances through a series of rotations
- Rotations guarantee property (2) by reducing tree height

# Standard AVL Tree

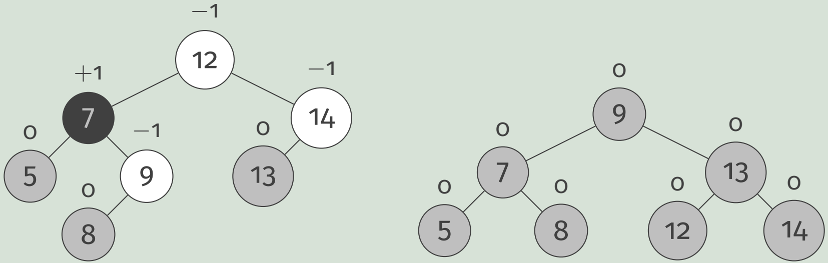
Self balancing binary search tree with balance factor BF on each node

$$\mathrm{BF}(X) := \mathrm{Height}(\mathrm{Right}(X)) - \mathrm{Height}(\mathrm{Left}(X)) \tag{1}$$

$$\mathrm{BF}(X) \in \{-1, 0, 1\} \tag{2}$$

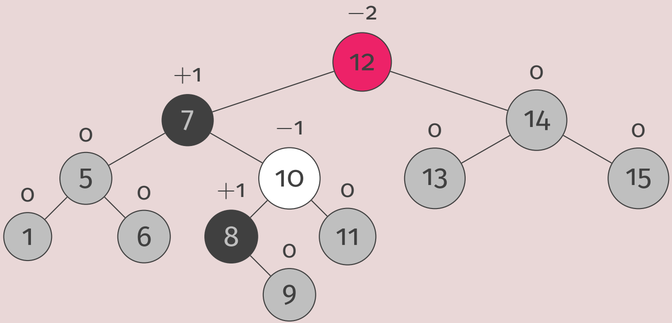
- Property (2) is preserved after any operation
- Tree balances through a series of rotations
- Rotations guarantee property (2) by reducing tree height
- Don't need to store height information, only BF

## Valid AVL Trees




Balance factors are listed above each respective node

## Invalid AVL Tree



## Standard AVL Tree: Cases

Balancing for a standard AVL tree can be broken down into a few cases

# Standard AVL Tree: Cases

Balancing for a standard AVL tree can be broken down into a few cases

- Left Left: Node $X$ has $\text{BF}(X) < -1$, $\text{BF}(\text{Left}(X)) \leq 0$
- Left Right: Node $X$ has $\text{BF}(X) < -1$, $\text{BF}(\text{Left}(X)) > 0$
- Right Left: Node $X$ has $\text{BF}(X) > +1$, $\text{BF}(\text{Right}(X)) < 0$
- Right Right: Node $X$ has $\text{BF}(X) > +1$, $\text{BF}(\text{Right}(X)) \geq 0$
- No-op: Node $X$ has $-1 \leq \text{BF}(X) \leq +1$

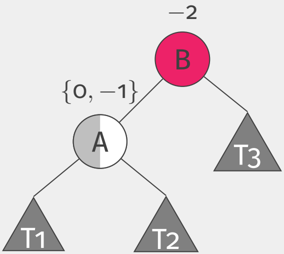
# Standard AVL Tree: Cases

Balancing for a standard AVL tree can be broken down into a few cases

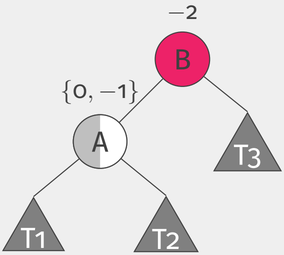
- Left Left: Node $X$ has $\mathrm{BF}(X) < -1$, $\mathrm{BF}(\mathrm{Left}(X)) \leq 0$
- Left Right: Node $X$ has $\mathrm{BF}(X) < -1$, $\mathrm{BF}(\mathrm{Left}(X)) > 0$
- Right Left: Node $X$ has $\mathrm{BF}(X) > +1$, $\mathrm{BF}(\mathrm{Right}(X)) < 0$
- Right Right: Node $X$ has $\mathrm{BF}(X) > +1$, $\mathrm{BF}(\mathrm{Right}(X)) \geq 0$
- No-op: Node $X$ has $-1 \leq \mathrm{BF}(X) \leq +1$

Note: if $\mathrm{BF}(X) > 1 \vee \mathrm{BF}(X) < -1$, then we can guarantee that $X$ has at least a child and a grandchild.

# Standard AVL Tree: Case Left Left

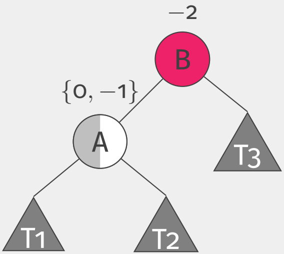


# Standard AVL Tree: Case Left Left



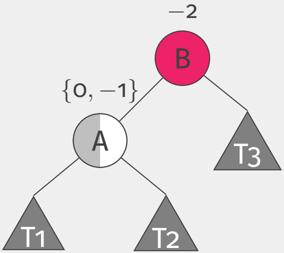

- B is the first unbalanced node we encounter after an operation

# Standard AVL Tree: Case Left Left

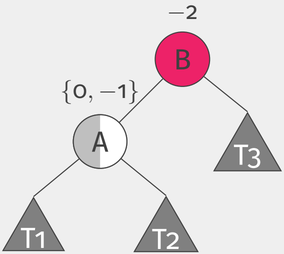



- B is the first unbalanced node we encounter after an operation
- Only operations on the decedents of B can cause B to become imbalanced

# Standard AVL Tree: Case Left Left

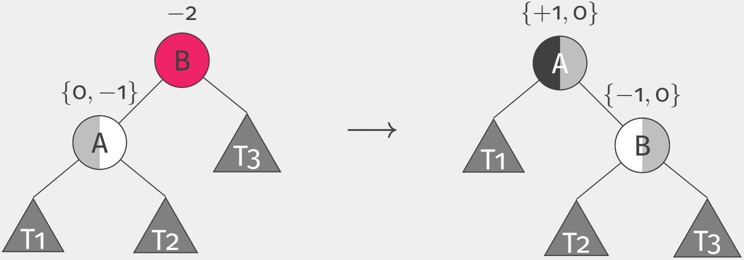



- B is the first unbalanced node we encounter after an operation
- Only operations on the decedents of B can cause B to become imbalanced
- Thus, we will always encounter highest depth imbalanced node by traversing to the root

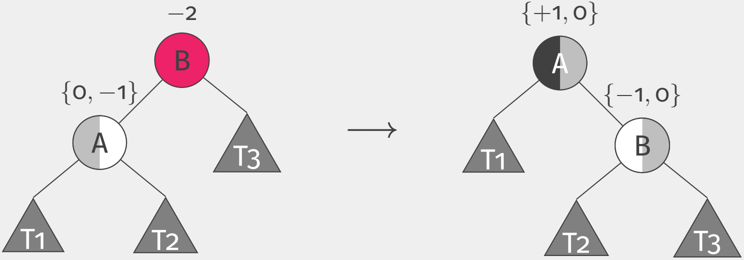



- B is the first unbalanced node we encounter after an operation
- Only operations on the decedents of B can cause B to become imbalanced
- Thus, we will always encounter highest depth imbalanced node by traversing to the root
- Thus T1, T2, T3 must be balanced

# Standard AVL Tree: Case Left Left

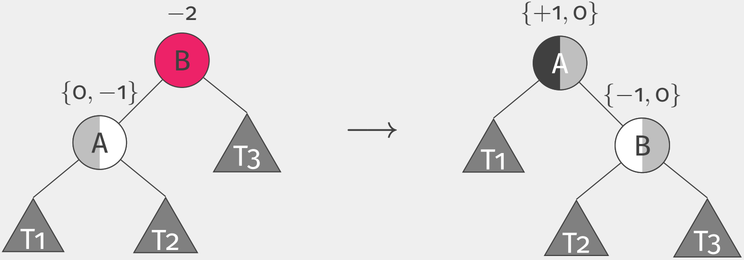


# Standard AVL Tree: Case Left Left

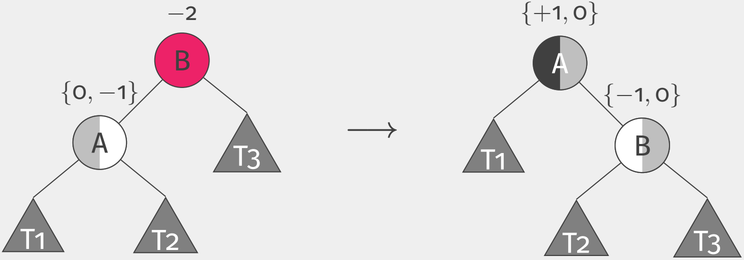



- Rotate B to the right (moving it down)

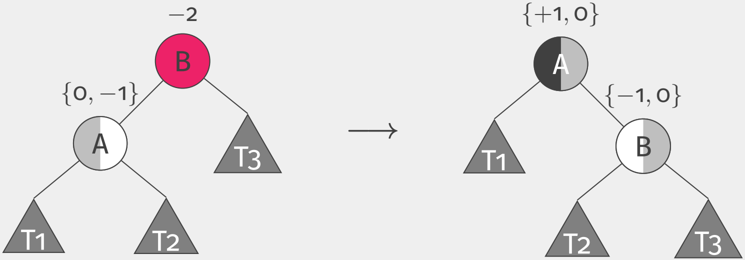
# Standard AVL Tree: Case Left Left



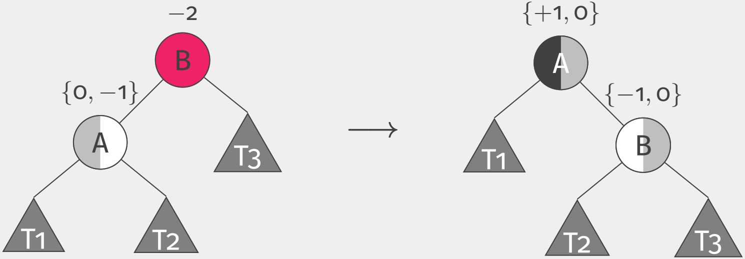

- Rotate B to the right (moving it down)
- Resulting balance factors BF'($A$) and BF'($B$) dependent on initial balance factor BF($A$) as indicated by the colors




- Rotate B to the right (moving it down)
- Resulting balance factors BF'$(A)$ and BF'$(B)$ dependent on initial balance factor BF$(A)$ as indicated by the colors
- Only two cases BF$(A) \in \{0, -1\}$

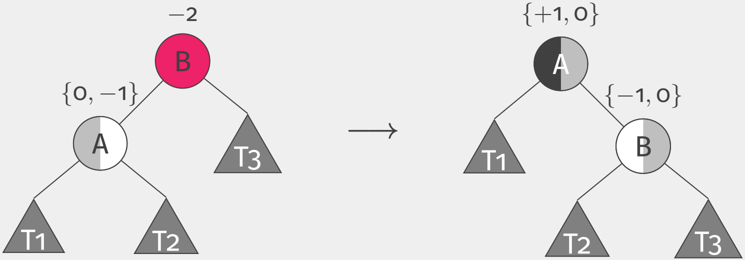



- Let Height(T1) $= t_1$, Height(T2) $= t_2$, Height(T3) $= t_3$

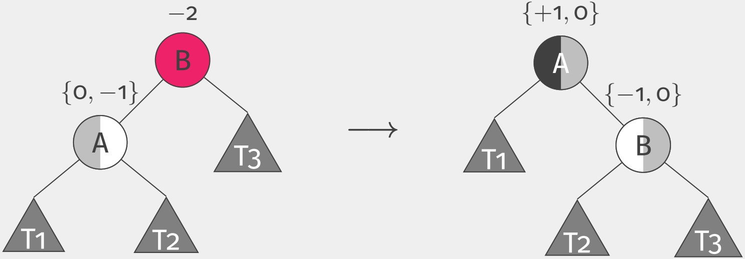



- Let Height(T1) $= t_1$, Height(T2) $= t_2$, Height(T3) $= t_3$
- We know BF(A) $= t_2 - t_1 \in \{0, -1\}$,
  BF(B) $= t_3 - (\max(t_1, t_2) + 1) = -2$

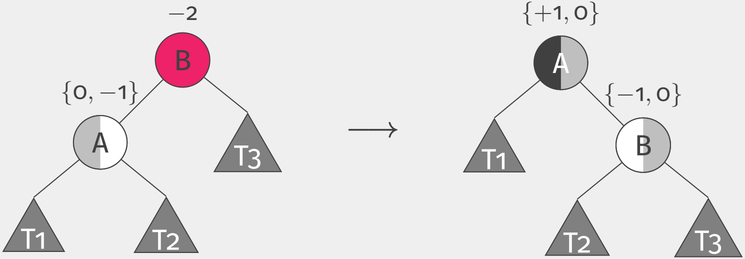
# Standard AVL Tree: Case Left Left



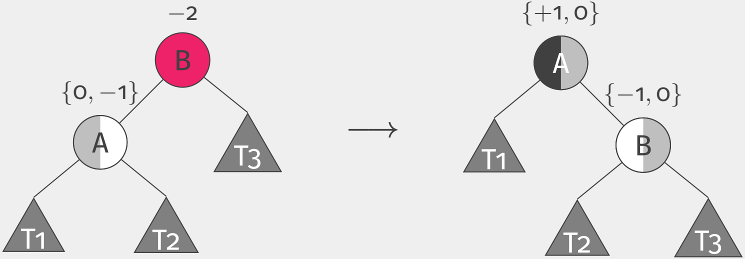

- Let Height(T1) $= t_1$, Height(T2) $= t_2$, Height(T3) $= t_3$
- We know BF(A) $= t_2 - t_1 \in \{0, -1\}$,
  BF(B) $= t_3 - (\max(t_1, t_2) + 1) = -2$
- We know $t_1 \geq t_2 \wedge t_3 < t_1$. Thus BF(B) $= t_3 - t_1 - 1 = -2$



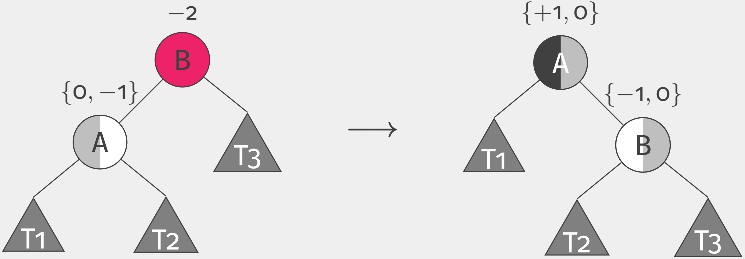

- We know BF'(B) $= t_3 - t_2$ and $t_3 - t_1 = -1$




- We know $\mathrm{BF'}(B) = t_3 - t_2$ and $t_3 - t_1 = -1$
  - $t_2 = t_1 \implies \mathrm{BF'}(B) = t_3 - t_1 = -1$
  - $t_2 = t_1 - 1 \implies \mathrm{BF'}(B) = t_3 - t_1 + 1 = 0$

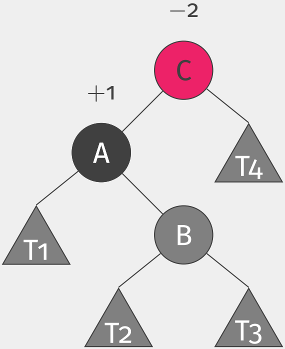



- We know $\mathrm{BF'}(\mathrm{B}) = t_3 - t_2$ and $t_3 - t_1 = -1$
  - $t_2 = t_1 \implies \mathrm{BF'}(\mathrm{B}) = t_3 - t_1 = -1$
  - $t_2 = t_1 - 1 \implies \mathrm{BF'}(\mathrm{B}) = t_3 - t_1 + 1 = 0$
- We know $\mathrm{BF'}(\mathrm{A}) = \max(t_3, t_2) + 1 - t_1$ and $t_1 > t_3$

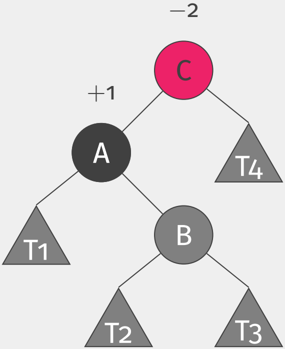



- We know BF'(B) $= t_3 - t_2$ and $t_3 - t_1 = -1$
  - $t_2 = t_1 \implies$ BF'(B) $= t_3 - t_1 = -1$
  - $t_2 = t_1 - 1 \implies$ BF'(B) $= t_3 - t_1 + 1 = 0$
- We know BF'(A) $= \max(t_3, t_2) + 1 - t_1$ and $t_1 > t_3$
  - $t_2 = t_1 \implies$ BF'(A) $= \max(t_3, t_1) + 1 - t_1 = 1$
  - $t_2 = t_1 - 1 \implies$ BF'(A) $= \max(t_3, t_1 - 1) + 1 - t_1 = 0$

# Standard AVL Tree: Case Left Right

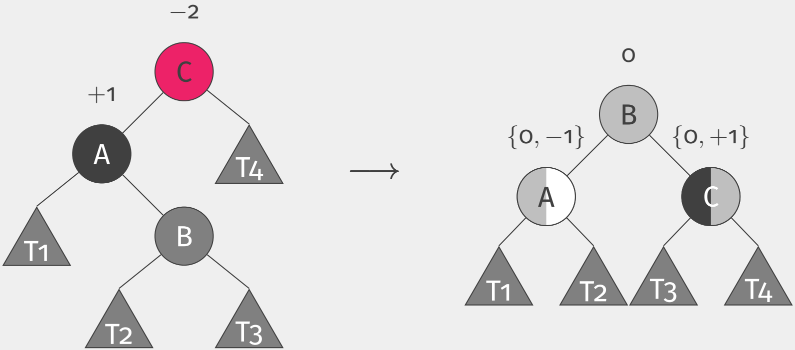


# Standard AVL Tree: Case Left Right

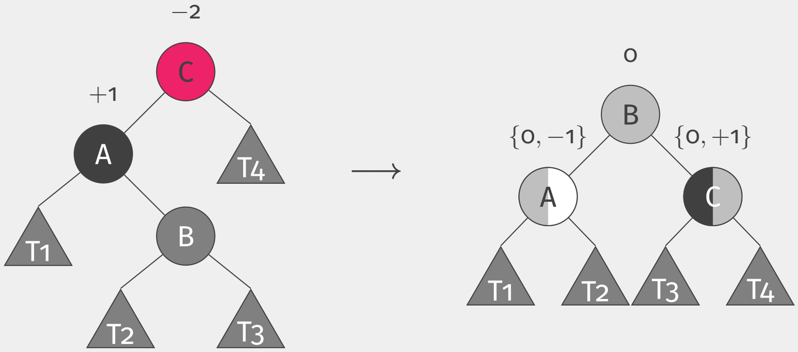



B, T1, T2, T3, and T4 are all balanced

# Standard AVL Tree: Case Left Right

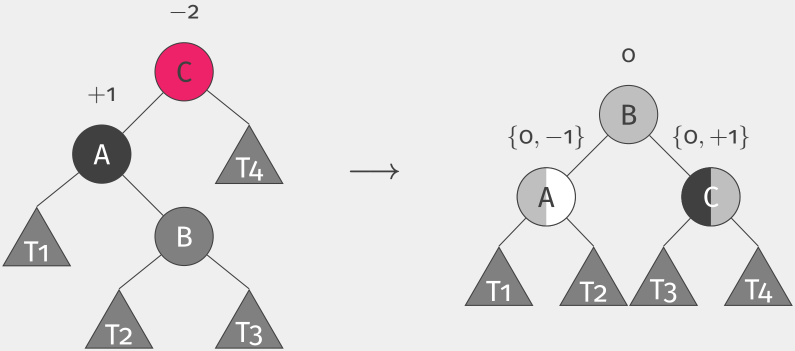


# Standard AVL Tree: Case Left Right

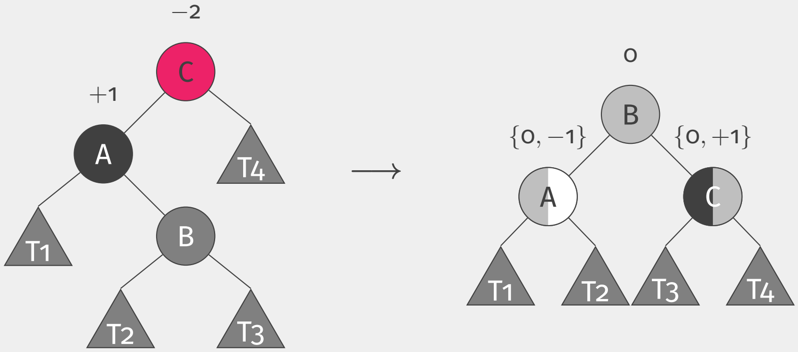



- Let Height$(T1) = t_1, \cdots,$ Height$(T4) = t_4$

# Standard AVL Tree: Case Left Right

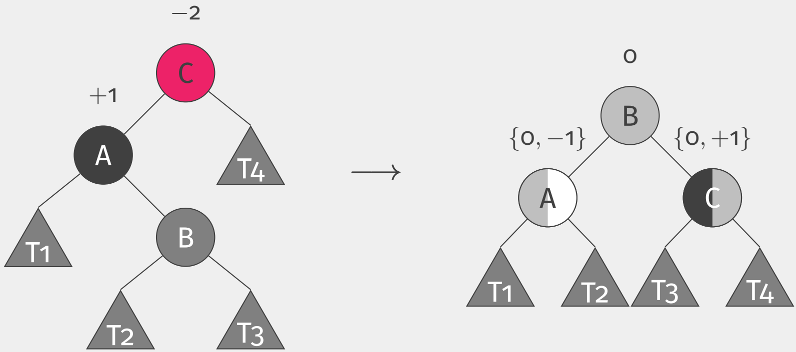



- Let Height$(T1) = t_1, \cdots,$ Height$(T4) = t_4$
- We can see/easily prove $t_1 = \max(t_2, t_3) = t_4$



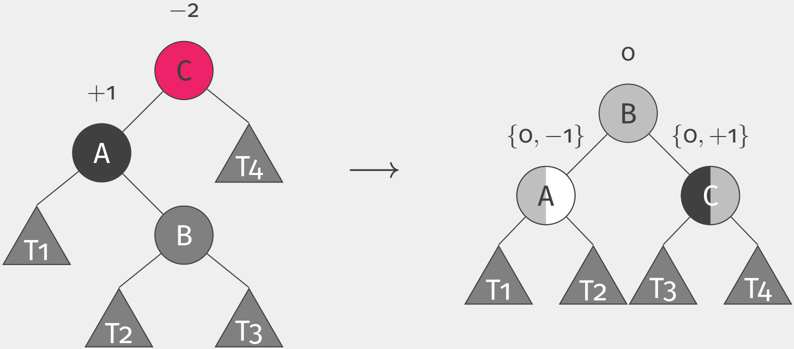

- Let Height$(T1) = t_1, \cdots,$ Height$(T4) = t_4$
- We can see/easily prove $t_1 = \max(t_2, t_3) = t_4$
- Thus BF'$(B) = \max(t_3, t_4) - \max(t_1, t_2) = 0$

# Standard AVL Tree: Case Left Right



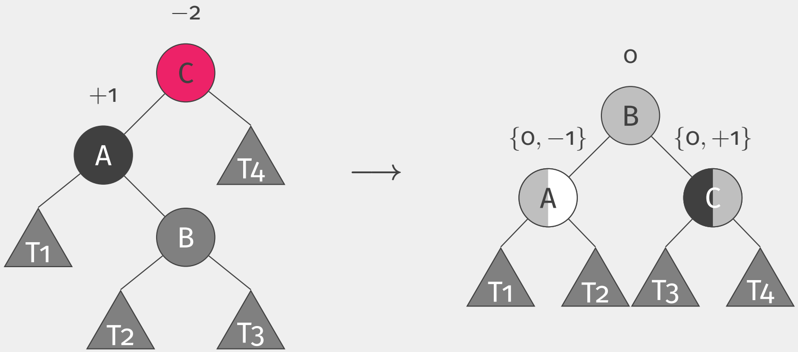

- We know BF’$(A) = t_2 - t_1$

# Standard AVL Tree: Case Left Right



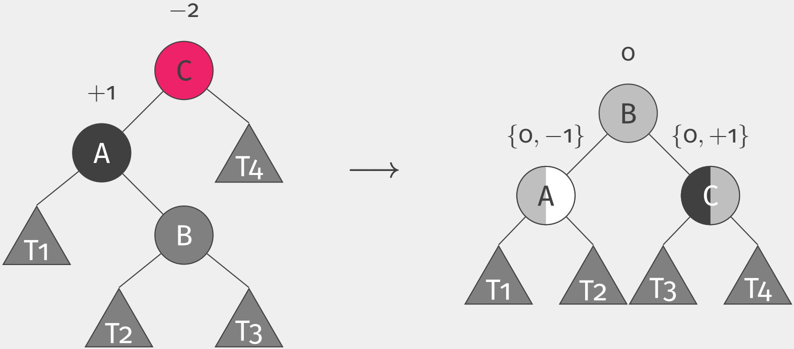

- We know $\mathrm{BF'}(A) = t_2 - t_1$
- If $\mathrm{BF}(B) \leq 0$ then $\max(t_2, t_3) = t_2$
  - So $t_1 = t_2 = t_4$
  - Thus $\mathrm{BF'}(A) = 0$

# Standard AVL Tree: Case Left Right



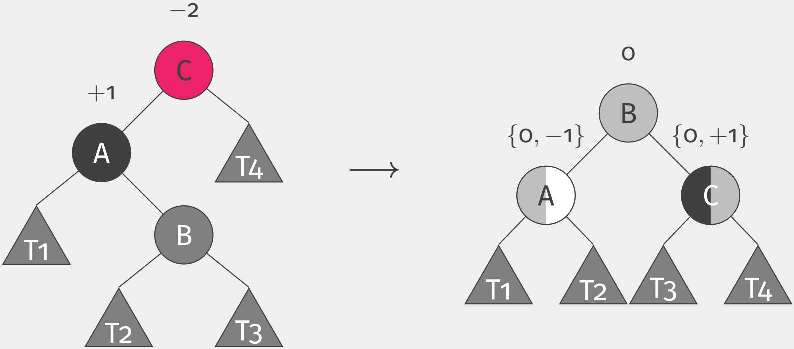

- We know BF'$(A) = t_2 - t_1$
- If BF$(B) \leq 0$ then max$(t_2, t_3) = t_2$
  - So $t_1 = t_2 = t_4$
  - Thus BF'$(A) = 0$
- If BF$(B) > 0$ then $t_3 - t_2 = 1$ since $B$ is balanced
  - So $t_1 =$ max$(t_2, t_3) =$ max$(t_2, t_2 + 1) = t_2 + 1$
  - Thus BF'$(A) = t_2 - t_1 = -1$

# Standard AVL Tree: Case Left Right



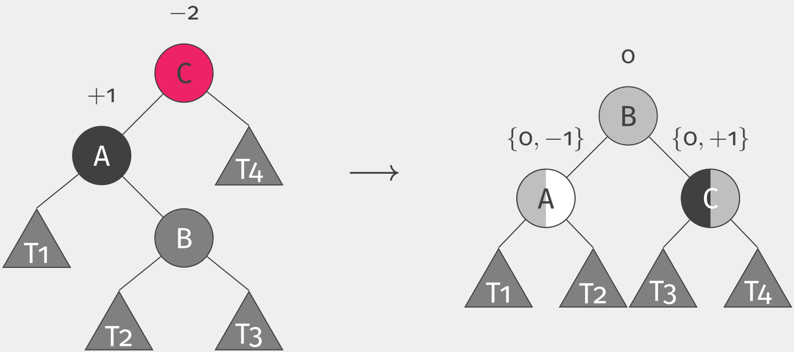

- We know BF’$(C) = t_4 - t_3$

# Standard AVL Tree: Case Left Right



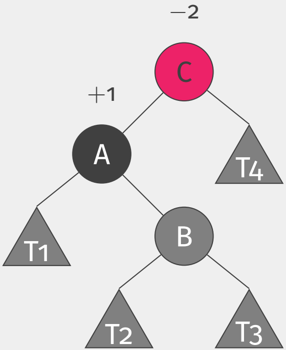

- We know $\text{BF'}(C) = t_4 - t_3$
- If $\text{BF}(B) \geq 0$ then $\max(t_2, t_3) = t_3$
  - So $t_1 = t_3 = t_4$
  - Thus $\text{BF'}(C) = 0$

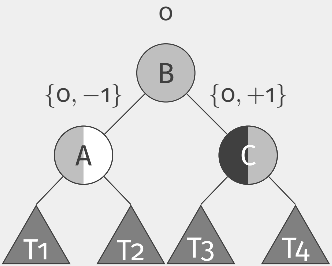
# Standard AVL Tree: Case Left Right




- We know BF’$(C) = t_4 - t_3$
- If BF$(B) \geq 0$ then max$(t_2, t_3) = t_3$
  - So $t_1 = t_3 = t_4$
  - Thus BF’$(C) = 0$
- If BF$(B) < 0$ then $t_3 - t_2 = -1$ since $B$ is balanced
  - So $t_4 =$ max$(t_2, t_3) =$ max$(t_3 + 1, t_3) = t_3 + 1$
  - Thus BF’$(C) = t_4 - t_3 = +1$

# Standard AVL Tree: Case Left Right




**Summary**

$$\text{BF'}(A) = -\max(\text{BF}(B), 0)$$
$$\text{BF'}(B) = 0$$
$$\text{BF'}(C) = -\min(\text{BF}(B), 0)$$

Same as AVL tree, except property (2) is changed to

$$\mathrm{BF}(X) \in \{x \in \mathbb{N} \mid -k \leq x \leq k\} \tag{2}$$

# $k$-AVL Tree

Same as AVL tree, except property (2) is changed to

$$\text{BF}(X) \in \{x \in \mathbb{N} \mid -k \leq x \leq k\} \tag{2}$$

- In fact, the 1-AVL tree is equivalent to our standard AVL tree

# $k$-AVL Tree

Same as AVL tree, except property (2) is changed to

$$\text{BF}(X) \in \{x \in \mathbb{N} \mid -k \leq x \leq k\} \tag{2}$$

- In fact, the 1-AVL tree is equivalent to our standard AVL tree
- $k$ does not need to be fixed, can be a dependent on number of nodes $n$

# $k$-AVL Tree

Same as AVL tree, except property (2) is changed to

$$\mathrm{BF}(X) \in \{x \in \mathbb{N} \mid -k \leq x \leq k\} \tag{2}$$

- In fact, the 1-AVL tree is equivalent to our standard AVL tree
- $k$ does not need to be fixed, can be a dependent on number of nodes $n$
- $n$-AVL tree is equivalent to a non-balancing BST

**The simple solution**

Store the height on each node, then check balance factors by subtraction

## The simple solution

Store the height on each node, then check balance factors by subtraction

## The problem

Have to traverse to the root on every operation to set height

### The simple solution

Store the height on each node, then check balance factors by subtraction

### The problem

Have to traverse to the root on every operation to set height

### Question

Is it possible to only store balance factors at each node and not have to traverse to the root?

# $k$-AVL Tree: Cases

This is a slide with the plain style and it is numbered.

This slide has an empty title and is aligned to top.

# No Slide Numbering

This slide is not numbered and is citing reference [1].

The packages `inputenc` and `FiraSans`[1,2] are used to properly set the main fonts.

This theme provides styling commands to typeset *emphasized*, alerted, **bold**, example text, …

`FiraSans` also provides support for mathematical symbols:

$$e^{i\pi} + 1 = 0.$$

[1] https://fonts.google.com/specimen/Fira+Sans
[2] http://mozilla.github.io/Fira/

# Section 2

# Blocks

These blocks are part of 1 slide, to be displayed consecutively.

### Block

Text.

# Blocks

These blocks are part of 1 slide, to be displayed consecutively.

### Block

Text.

### Alert block

Alert text.

# Blocks

These blocks are part of 1 slide, to be displayed consecutively.

### Block

Text.

### Alert block

Alert text.

### Example block

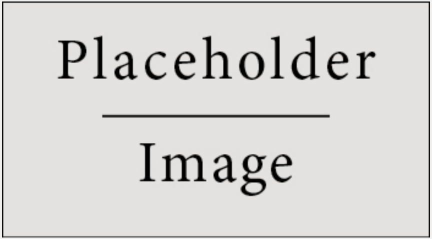
Example text.

# Columns

This text appears in the left column and wraps neatly with a margin between columns.

# Columns

This text appears in the left column and wraps neatly with a margin between columns.



# Lists

Items:

- Item 1
  - Subitem 1.1
  - Subitem 1.2
- Item 2
- Item 3

Enumerations:

1. First
2. Second
   2.1 Sub-first
   2.2 Sub-second
3. Third

Descriptions:

**First** Yes.

**Second** No.

# Table

| Discipline | Avg. Salary |
|---|---|
| **Engineering** | **$66,521** |
| Computer Sciences | $60,005 |
| Mathematics and Sciences | $61,867 |
| Business | $56,720 |
| Humanities & Social Sciences | $56,669 |
| Agriculture and Natural Resources | $53,565 |
| Communications | $51,448 |
| **Average for All Disciplines** | **$58,114** |

**Table:** Table caption

Thanks for using **Focus**!

# References


Donald E. Knuth.
**Computer programming as an art.**
*Commun. ACM*, pages 667–673, 1974.

Donald E. Knuth.
**Two notes on notation.**
*Amer. Math. Monthly*, 99:403–422, 1992.

Leslie Lamport.
***LaTeX: A Document Preparation System.***
Pearson Education India, 1994.

# Backup Slide

This is a backup slide, useful to include additional materials to answer questions from the audience.

The package `appendixnumberbeamer` is used to refrain from numbering appendix slides.