Red-Black Trees

Why Even the Inventor Moved On...

Your Name

February 8, 2026

We All Love to Sort Things!

- Organizing bookshelves

We All Love to Sort Things!

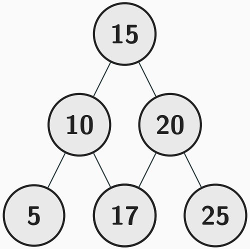
- Organizing bookshelves
- Arranging files on computers

We All Love to Sort Things!

- Organizing bookshelves
- Arranging files on computers
- Putting groceries in order

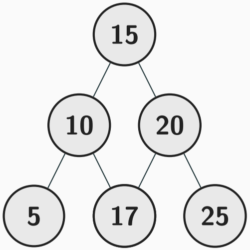
We All Love to Sort Things!

- Organizing bookshelves
- Arranging files on computers
- Putting groceries in order
- **What's a really good way?** 🌲



We All Love to Sort Things!

- Organizing bookshelves
- Arranging files on computers
- Putting groceries in order
- **What's a really good way?** 🌲



Binary Trees are Awesome!

Why Binary Trees Are Good

Beautiful Structure

- Everything has its place
- Search: $O(\log n)$
- Insert: $O(\log n)$
- Delete: $O(\log n)$

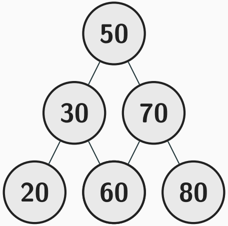
Why Binary Trees Are Good

Beautiful Structure

- Everything has its place
- Search: $O(\log n)$
- Insert: $O(\log n)$
- Delete: $O(\log n)$

The Magic

Logarithmic time = **Sports car performance!**



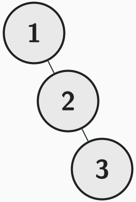
Sadly, Sometimes It Goes Wrong...

What happens when we insert: 1, 2, 3, 4, 5?

Sadly, Sometimes It Goes Wrong...

What happens when we insert: 1, 2, 3, 4, 5?

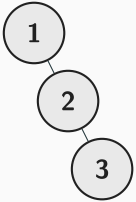
After 1, 2, 3



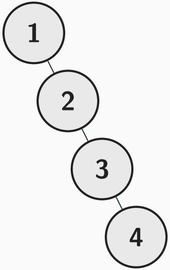
Sadly, Sometimes It Goes Wrong...

What happens when we insert: 1, 2, 3, 4, 5?

After 1, 2, 3



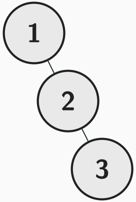
After 4



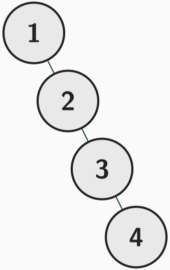
Sadly, Sometimes It Goes Wrong...

What happens when we insert: 1, 2, 3, 4, 5?

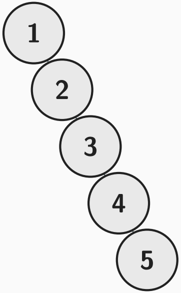
After 1, 2, 3



After 4

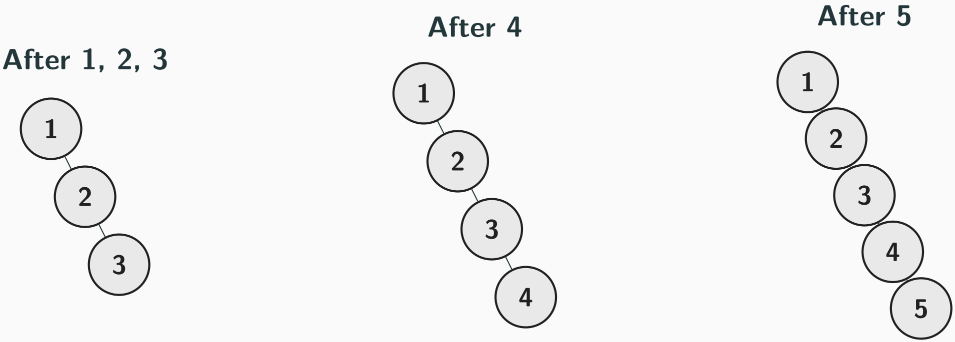


After 5



Sadly, Sometimes It Goes Wrong...

What happens when we insert: 1, 2, 3, 4, 5?

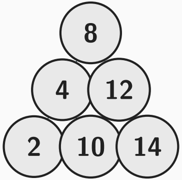


Our tree became a... LINKED LIST!

From Sports Car to Bicycle

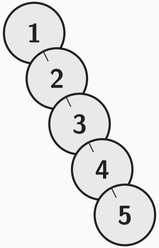
Before (Balanced)

- Search: $O(\log n)$
- Insert: $O(\log n)$
- Delete: $O(\log n)$



After (Degenerate)

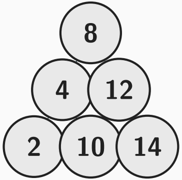
- Search: $O(n)$
- Insert: $O(n)$
- Delete: $O(n)$



From Sports Car to Bicycle

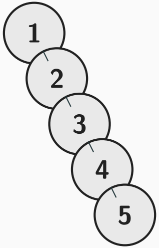
Before (Balanced)

- Search: $O(\log n)$
- Insert: $O(\log n)$
- Delete: $O(\log n)$



After (Degenerate)

- Search: $O(n)$
- Insert: $O(n)$
- Delete: $O(n)$



⚠ Not Great!

We Need Almost Balanced Trees

We want to keep $O(\log n)$ operations **even in the worst case!**

We Need Almost Balanced Trees

We want to keep $O(\log n)$ operations **even in the worst case!**

Popular Solutions

- AVL Trees

We Need Almost Balanced Trees

We want to keep $O(\log n)$ operations **even in the worst case!**

Popular Solutions

- AVL Trees
- **Red-Black Trees** ★

We Need Almost Balanced Trees

We want to keep $O(\log n)$ operations **even in the worst case!**

Popular Solutions

- AVL Trees
- **Red-Black Trees** ★
- B-Trees

We Need Almost Balanced Trees

We want to keep $O(\log n)$ operations **even in the worst case!**

Popular Solutions

- AVL Trees
- **Red-Black Trees** ★
- B-Trees
- Splay Trees

We Need Almost Balanced Trees

We want to keep $O(\log n)$ operations **even in the worst case!**

Popular Solutions

- AVL Trees
- **Red-Black Trees** ★
- B-Trees
- Splay Trees

Red-Black Trees are used EVERYWHERE!

Enter: Red-Black Trees

- Around since the 1970s

Enter: Red-Black Trees

- Around since the 1970s
- Used in Linux kernels

Enter: Red-Black Trees

- Around since the 1970s
- Used in Linux kernels
- Java's TreeMap and TreeSet

Enter: Red-Black Trees

- Around since the 1970s
- Used in Linux kernels
- Java's TreeMap and TreeSet
- Databases and file systems

Enter: Red-Black Trees

- Around since the 1970s
- Used in Linux kernels
- Java's TreeMap and TreeSet
- Databases and file systems

Fun Fact

Even the original inventor **didn't mention** Red-Black Trees in his main DSA book!

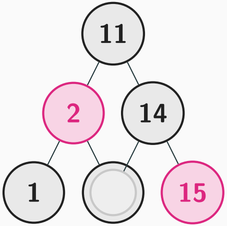
He introduced *Left-Leaning Red-Black Trees* instead.
😃

Enter: Red-Black Trees

- Around since the 1970s
- Used in Linux kernels
- Java's TreeMap and TreeSet
- Databases and file systems

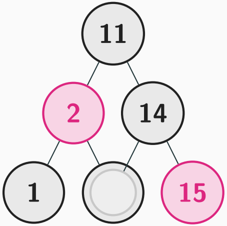
Fun Fact

Even the original inventor **didn't mention** Red-Black Trees in his main DSA book!
He introduced *Left-Leaning Red-Black Trees* instead. 😃



Enter: Red-Black Trees

- Around since the 1970s
- Used in Linux kernels
- Java's TreeMap and TreeSet
- Databases and file systems



Fun Fact

Even the original inventor **didn't mention** Red-Black Trees in his main DSA book!

He introduced *Left-Leaning Red-Black Trees* instead. 😀

Let's get started, shall we?

Everything Has Unique Properties

Fundamental properties are better than mathematical definitions!

Everything Has Unique Properties

Fundamental properties are better than mathematical definitions!

Why Properties?

- Easier to understand

Everything Has Unique Properties

Fundamental properties are better than mathematical definitions!

Why Properties?

- Easier to understand
- Audience feels good

Everything Has Unique Properties

Fundamental properties are better than mathematical definitions!

Why Properties?

- Easier to understand
- Audience feels good
- Been around for ages!

Everything Has Unique Properties

Fundamental properties are better than mathematical definitions!

Why Properties?

- Easier to understand
- Audience feels good
- Been around for ages!

Red-Black Trees have 5 properties

Let's see them one by one...

The Five Properties

1. Every node is either Red or Black

The Five Properties

1. Every node is either Red or Black
2. The **root** is always Black

The Five Properties

1. Every node is either Red or Black
2. The **root** is always Black
3. All **leaves** (NIL) are Black

The Five Properties

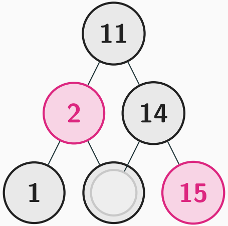
1. Every node is either Red or Black
2. The **root** is always Black
3. All **leaves** (NIL) are Black
4. No two Red nodes can be adjacent

The Five Properties

1. Every node is either Red or Black
2. The **root** is always Black
3. All **leaves** (NIL) are Black
4. No two Red nodes can be adjacent
5. Equal black height on all paths

The Five Properties

1. Every node is either Red or Black
2. The **root** is always Black
3. All **leaves** (NIL) are Black
4. No two Red nodes can be adjacent
5. Equal black height on all paths



Black Height: The Core Property

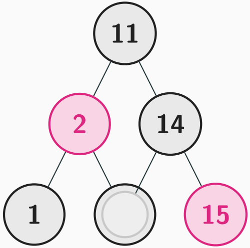
Definition

The **black-height** of a node is the number of black nodes on any path from that node to a leaf (not counting the node itself).

Black Height: The Core Property

Definition

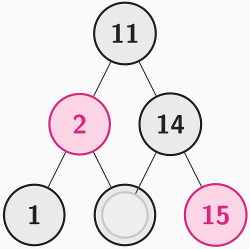
The **black-height** of a node is the number of black nodes on any path from that node to a leaf (not counting the node itself).



Black Height: The Core Property

Definition

The **black-height** of a node is the number of black nodes on any path from that node to a leaf (not counting the node itself).



Why This Matters

This property helps us understand the **balance** of the tree!

The Amazing Height Bound

Because of black height, we can prove:

The Amazing Height Bound

Because of black height, we can prove:

Theorem

A Red-Black Tree with n nodes has height $h \leq 2\log_2(n+1)$

The Amazing Height Bound

Because of black height, we can prove:

Theorem

A Red-Black Tree with n nodes has height $h \leq 2\log_2(n+1)$

What This Means

- Even worst case is logarithmic!
- Never more than twice as tall as perfect tree
- $O(\log n)$ performance guaranteed

The Amazing Height Bound

Because of black height, we can prove:

Theorem

A Red-Black Tree with n nodes has height $h \leq 2\log_2(n+1)$

What This Means

- Even worst case is logarithmic!
- Never more than twice as tall as perfect tree
- $O(\log n)$ performance guaranteed

Perfect Tree: height $= \log_2(n+1)$

RBT Tree: height $\leq 2 \times \log_2(n+1)$

✔ Still Awesome!

The Amazing Height Bound

Because of black height, we can prove:

Theorem

A Red-Black Tree with n nodes has height $h \leq 2\log_2(n+1)$

What This Means

- Even worst case is logarithmic!
- Never more than twice as tall as perfect tree
- $O(\log n)$ performance guaranteed

Perfect Tree: height $= \log_2(n+1)$

RBT Tree: height $\leq 2 \times \log_2(n+1)$

✔ Still Awesome!

(I skipped the detailed proof - nobody wants to sit through that!)

Wait, We're Not Done Yet!

We haven't talked about **insertion** or **deletion** yet!

Wait, We're Not Done Yet!

We haven't talked about **insertion** or **deletion** yet!

Warning: Shit's about to get real

(Also tough)

Wait, We're Not Done Yet!

We haven't talked about **insertion** or **deletion** yet!

Warning: Shit's about to get real

(Also tough)

How do we insert something?

Wait, We're Not Done Yet!

We haven't talked about **insertion** or **deletion** yet!

Warning: Shit's about to get real

(Also tough)

How do we insert something?

Insertion: The Process

1. Insert like normal BST

Insertion: The Process

1. Insert like normal BST
2. Color new node RED

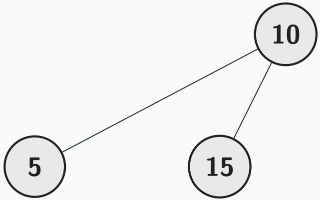
Insertion: The Process

1. Insert like normal BST
2. Color new node RED
3. Fix any violations

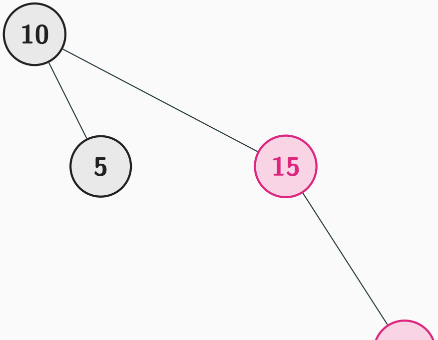
Insertion: The Process

1. Insert like normal BST
2. Color new node RED
3. Fix any violations

Before:

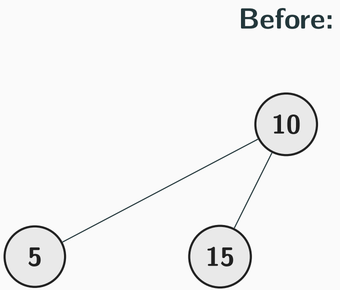


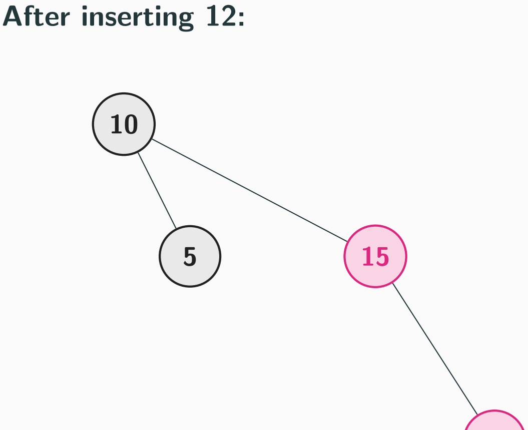
After inserting 12:



Insertion: The Process

1. Insert like normal BST
2. Color new node RED
3. Fix any violations





Fixing Violations

Uncle is RED

- Easy case!
- Just recolor
- Flip parent, uncle, grandparent

Fixing Violations

Uncle is RED

- Easy case!
- Just recolor
- Flip parent, uncle, grandparent

Uncle is BLACK

- Hard case!
- Need rotations
- Left rotate, right rotate
- Sometimes both!

Fixing Violations

Uncle is RED

- Easy case!
- Just recolor
- Flip parent, uncle, grandparent

Uncle is BLACK

- Hard case!
- Need rotations
- Left rotate, right rotate
- Sometimes both!

This is where it gets tricky!

But it keeps the tree balanced!

Deletion: Even More Fun!

Deletion is even more... interesting! ☻

Deletion: Even More Fun!

Deletion is even more... interesting! ☻

If deleting RED node

- No problem!
- Just remove it
- Properties still hold

Deletion: Even More Fun!

Deletion is even more... interesting! ☻

If deleting RED node

- No problem!
- Just remove it
- Properties still hold

If deleting BLACK node

- Oh boy...
- Black height changes!
- Need "double black" fix
- Complex cases

Deletion: Even More Fun!

Deletion is even more... interesting! ☻

If deleting RED node

- No problem!
- Just remove it
- Properties still hold

If deleting BLACK node

- Oh boy...
- Black height changes!
- Need "double black" fix
- Complex cases

The key: Maintain black height at all costs!

(We'll skip the gory details - you get the idea!)

Where Are Red-Black Trees Used?

Everywhere!

Where Are Red-Black Trees Used?

Everywhere!

- 🐧 **Linux Kernel**
  Process scheduling

Where Are Red-Black Trees Used?

Everywhere!

- 🐧 **Linux Kernel**
  Process scheduling
- ☕ **Java**
  TreeMap, TreeSet

Where Are Red-Black Trees Used?

Everywhere!

- **Linux Kernel**
  Process scheduling
- **Java**
  TreeMap, TreeSet
- **Databases**
  Indexing structures

Where Are Red-Black Trees Used?

Everywhere!

- **Linux Kernel**
  Process scheduling
- **Java**
  TreeMap, TreeSet
- **Databases**
  Indexing structures
- **File Systems**
  Directory organization

Where Are Red-Black Trees Used?

Everywhere!

- **Linux Kernel**
  Process scheduling
- **Java**
  TreeMap, TreeSet
- **Databases**
  Indexing structures
- **File Systems**
  Directory organization

Every time you use these...

You're benefiting from **Red-Black Trees!**

Red-Black Trees in a Nutshell

Red-Black Trees in a Nutshell

- Complex but incredibly powerful

Red-Black Trees in a Nutshell

- Complex but incredibly powerful
- Tricky to implement

Red-Black Trees in a Nutshell

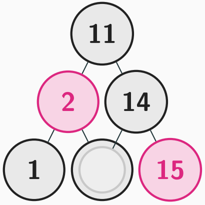
- Complex but incredibly powerful
- Tricky to implement
- Guaranteed $O(\log n)$ performance

Red-Black Trees in a Nutshell

- Complex but incredibly powerful
- Tricky to implement
- Guaranteed $O(\log n)$ performance
- Used everywhere in computing

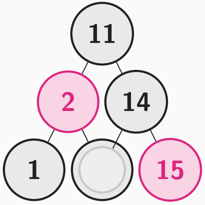
Red-Black Trees in a Nutshell

- Complex but incredibly powerful
- Tricky to implement
- Guaranteed $O(\log n)$ performance
- Used everywhere in computing



Red-Black Trees in a Nutshell

- Complex but incredibly powerful
- Tricky to implement
- Guaranteed $O(\log n)$ performance
- Used everywhere in computing



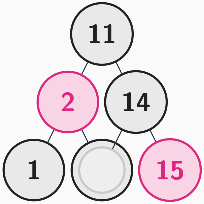
Remember

The next time you're struggling with RBT implementation...

Even the **inventor** moved on to Left-Leaning Red-Black Trees! 😃

Red-Black Trees in a Nutshell

- Complex but incredibly powerful
- Tricky to implement
- Guaranteed $O(\log n)$ performance
- Used everywhere in computing



Remember

The next time you're struggling with RBT implementation...

Even the **inventor** moved on to Left-Leaning Red-Black Trees! ☻

Any Questions?