Binomial Heaps

CS2040S, AY19/20 Sem 1

Eldon Chung | eldon.chung@u.nus.edu
Wang Zhi Jian | wzhijian@u.nus.edu

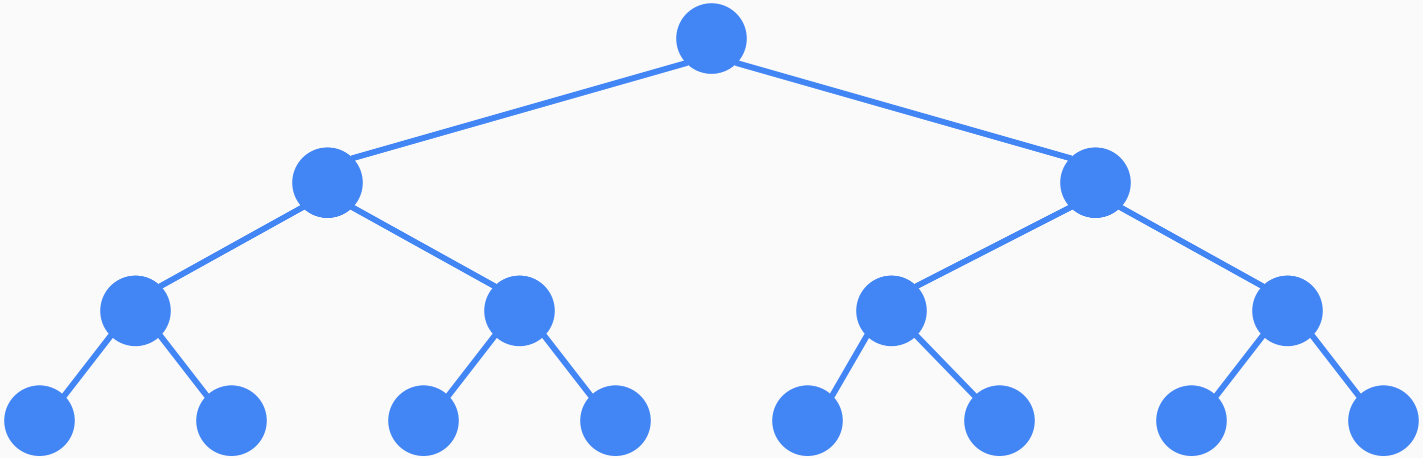
Heaps

Procedure	**Binary Heap** (worst-case)
Make-Heap	$\Theta(1)$
Insert	$\Theta(\log n)$
Minimum	$\Theta(1)$
Extract-Min	$\Theta(\log n)$
Union	$\Theta(n)$
Decrease-Key	$\Theta(\log n)$
Delete	$\Theta(\log n)$

Heaps

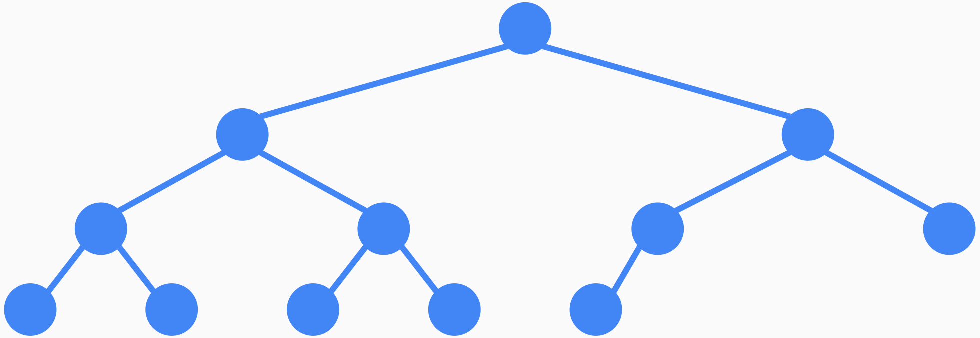
Procedure	**Binary Heap** (worst-case)	**Binomial Heap** (worst-case)	**Fibonacci Heap** (amortized)
Make-Heap	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Insert	$\Theta(\log n)$	$O(\log n)$	$\Theta(1)$
Minimum	$\Theta(1)$	$O(\log n)$	$\Theta(1)$
Extract-Min	$\Theta(\log n)$	$\Theta(\log n)$	$O(\log n)$
Union	$\Theta(n)$	$O(\log n)$	$\Theta(1)$
Decrease-Key	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(1)$
Delete	$\Theta(\log n)$	$\Theta(\log n)$	$O(\log n)$

Binomial Heap



Structure of a Binomial Heap

Binary Heaps are **complete binary trees**.



Binomial Heaps have a different structure: they are made up of **binomial trees**.

Properties of Binomial Trees

Binomial Trees are trees with the following properties:

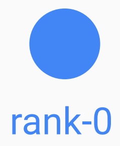
1. A rank-0 binomial tree is a tree with one node.

rank-0

2. A rank-k binomial tree is a tree constructed from two rank-(k−1) binomial trees with the root node of one tree as the root node of the other tree.

Binomial Tree

This is a **rank-0** binomial tree.



Binomial Tree

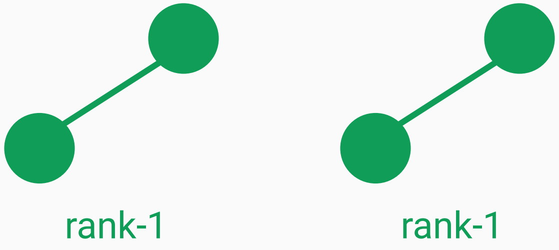
To construct a **rank-1** binomial tree, we take two **rank-0** binomial trees...

Binomial Tree

To construct a **rank-1** binomial tree, we take two **rank-0** binomial trees…
...and make one of them the child of the other’s root.

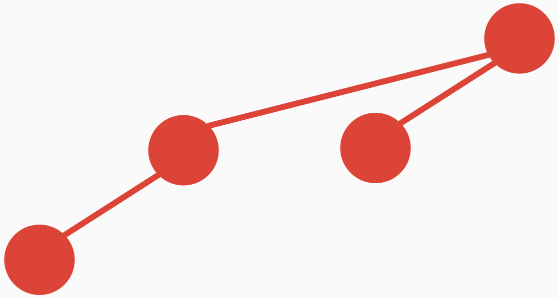
Binomial Tree

To construct a **rank-2** binomial tree, we take two **rank-1** binomial trees...



Binomial Tree

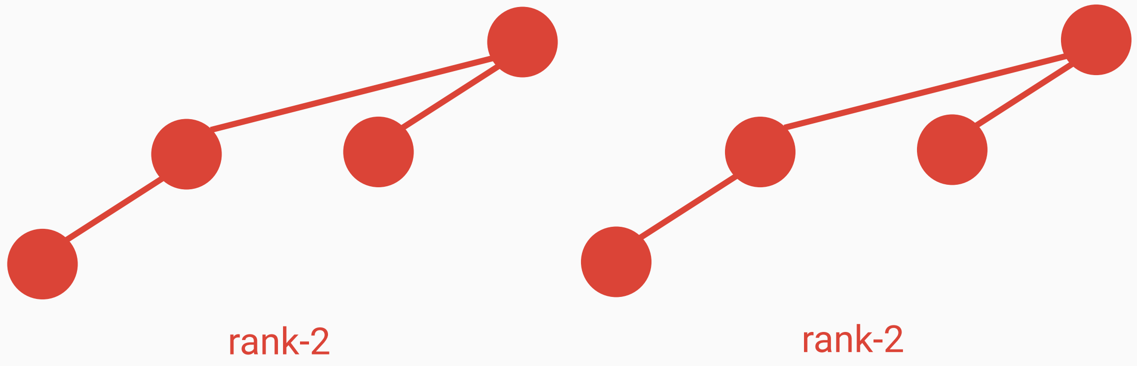
To construct a **rank-2** binomial tree, we take two **rank-1** binomial trees…
...and make one of them the child of the other's root.



rank-2

Binomial Tree

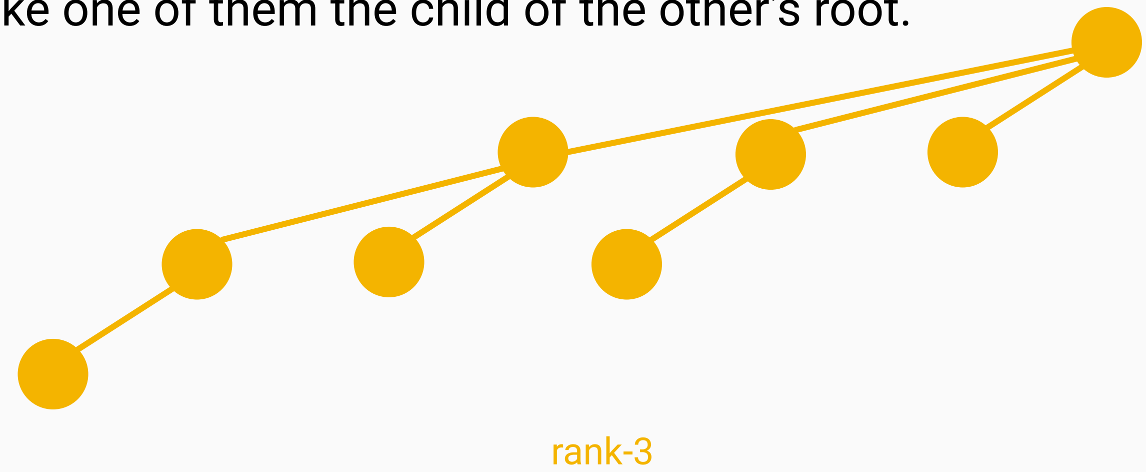
To construct a **rank-3** binomial tree, we take two **rank-2** binomial trees...



Binomial Tree

To construct a **rank-3** binomial tree, we take two **rank-2** binomial trees…

…and make one of them the child of the other's root.



Structure of a Binomial Heap

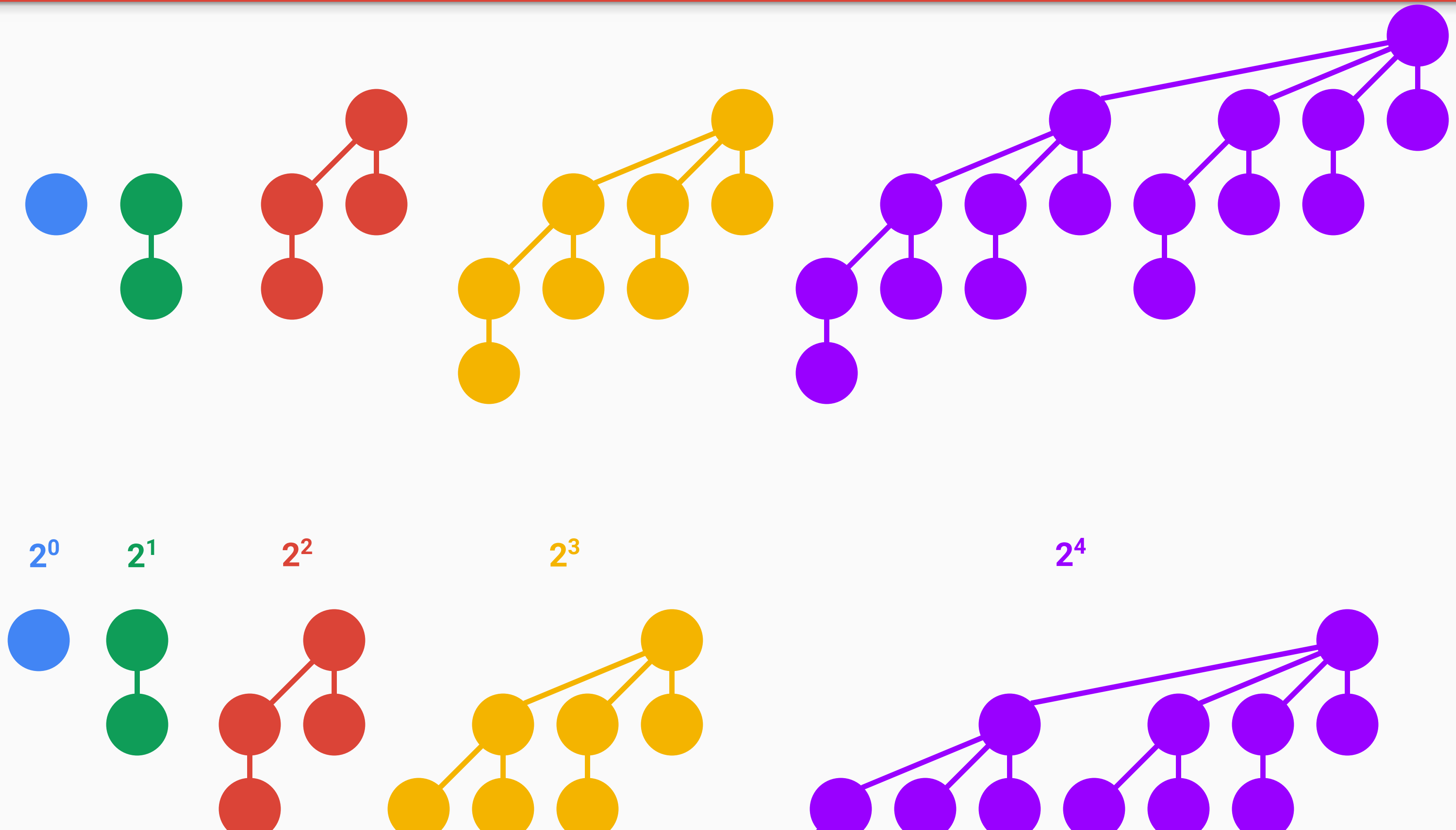
A binomial heap is a collection of binomial trees, where each binomial tree has a **unique rank** (and hence the number of binomial trees is minimised).
Each binomial tree must maintain the **heap property**.

For a binomial heap with *n* nodes, express *n* as a **sum of powers of 2**.

e.g. $n = 14$, $14 = 8 + 4 + 2 = 2^3 + 2^2 + 2^1$

We represent a binomial heap with 14 nodes as a collection of 3 binomial trees: one **rank-3** binomial tree, one **rank-2** binomial tree and one **rank-1** binomial tree.

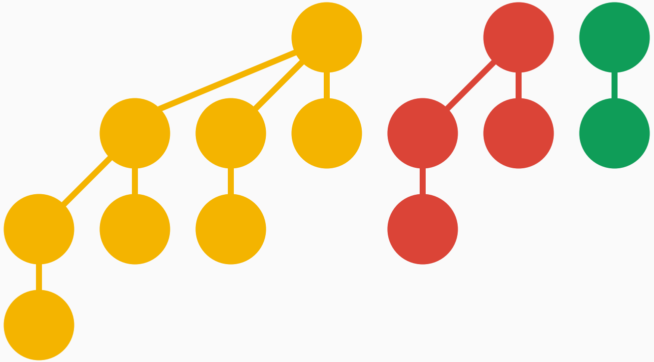
Binomial Tree Storage



Structure of a Binomial Heap

Binomial heap with 14 nodes:

$$14 = 2^3 + 2^2 + 2^1$$

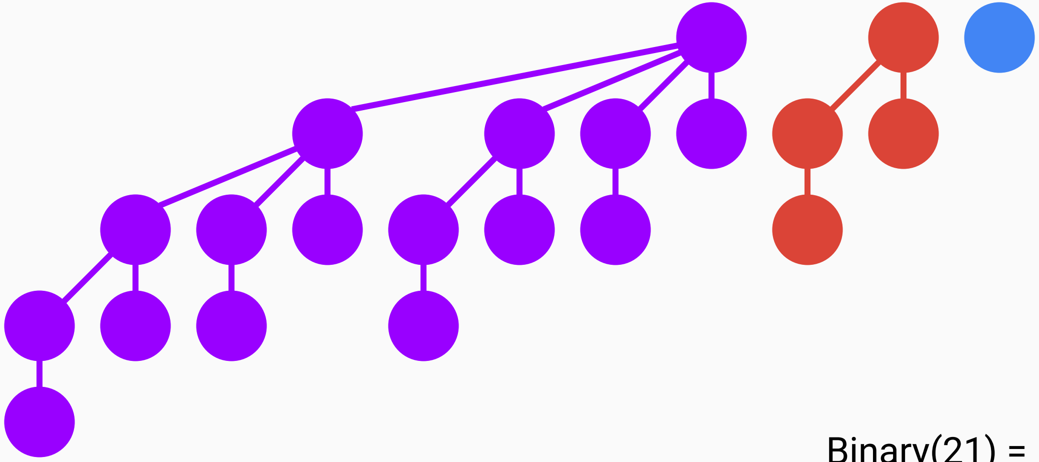


Binary(14) = 1110

Structure of a Binomial Heap

Binomial heap with 21 nodes:

$$21 = 2^4 + 2^2 + 2^0$$



Binary(21) = 10101

Structure of a Binomial Heap

How many binomial trees (k) are there for a binomial heap with n nodes?

Hint: How many bits does Binary(n) have?

Structure of a Binomial Heap

How many binomial trees (*k*) are there for a binomial heap with *n* nodes?

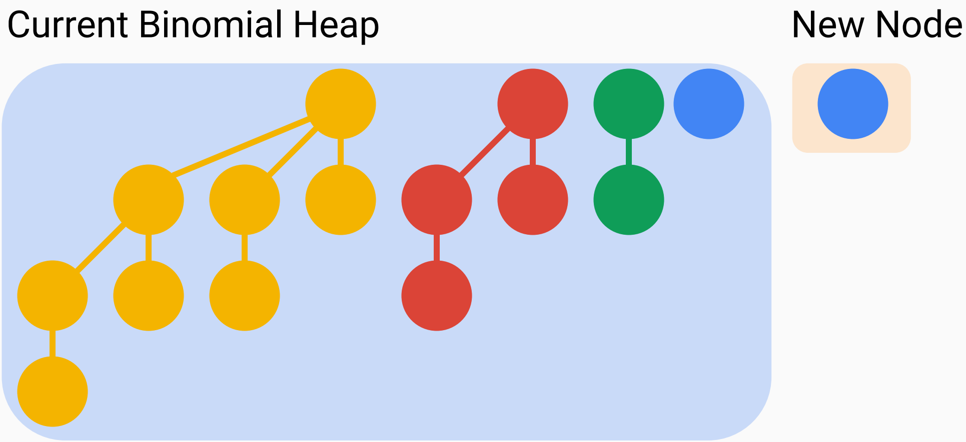
Hint: How many bits does Binary(*n*) have?

No. of bits = $\lfloor \log(n) \rfloor + 1$

Thus, for a binomial heap with *n* nodes, there are **O(log *n*)** binomial trees.

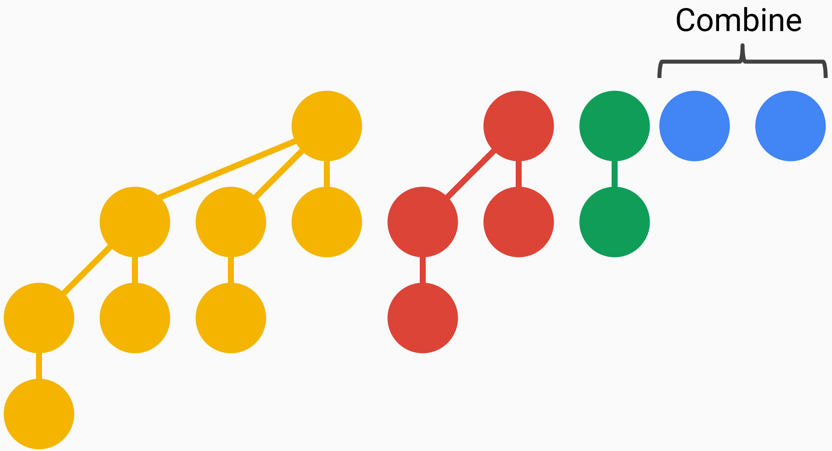
Insert

When we insert a new node into a Binomial Heap, first we insert it as a **rank-0** binomial heap.



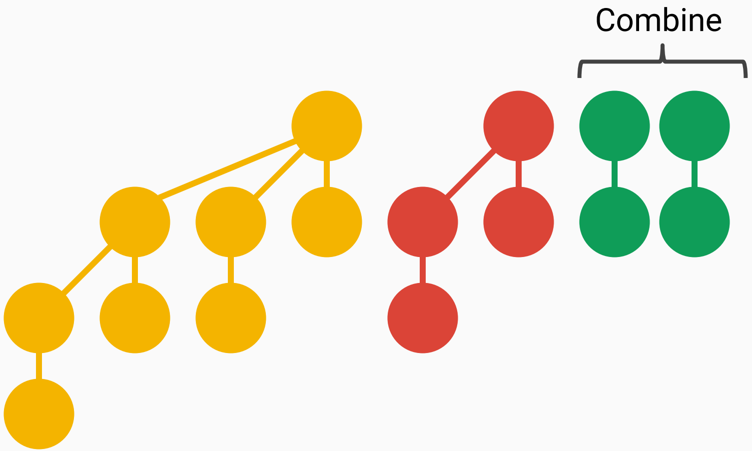
Insert

Then, we repeatedly combined binomial trees of the same rank together.
Two binomial trees of rank-*k* can be combined into one binomial tree of rank-(*k*+1).



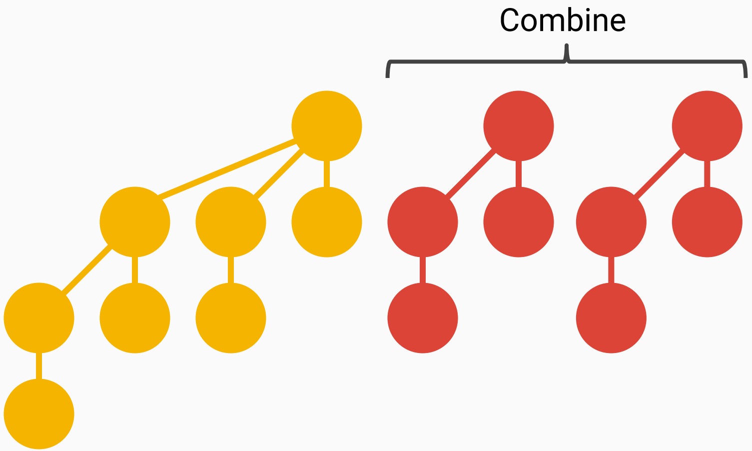
Insert

Then, we repeatedly combined binomial trees of the same rank together.
Two binomial trees of rank-k can be combined into one binomial tree of rank-$(k+1)$.



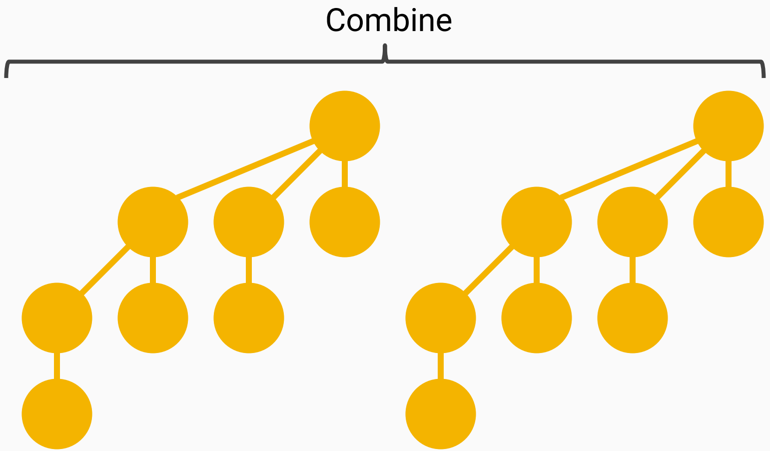
Insert

Then, we repeatedly combined binomial trees of the same rank together.
Two binomial trees of rank-k can be combined into one binomial tree of rank-(k+1).



Insert

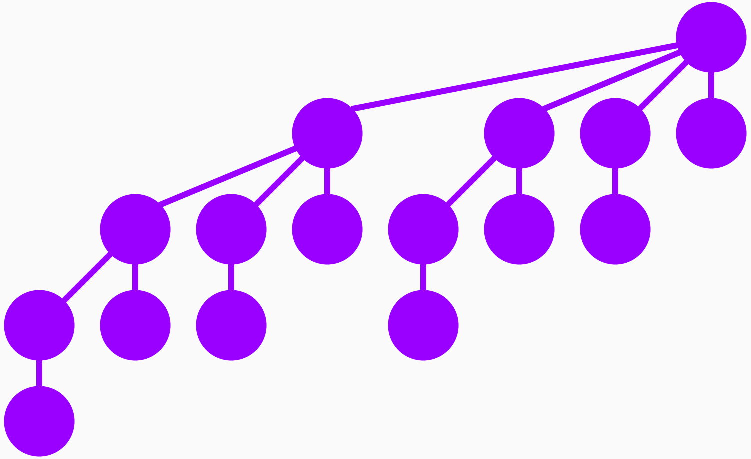
Then, we repeatedly combined binomial trees of the same rank together.
Two binomial trees of rank-k can be combined into one binomial tree of rank-(k+1).



Insert

Then, we repeatedly combined binomial trees of the same rank together.
Two binomial trees of rank-*k* can be combined into one binomial tree of rank-(*k*+1).

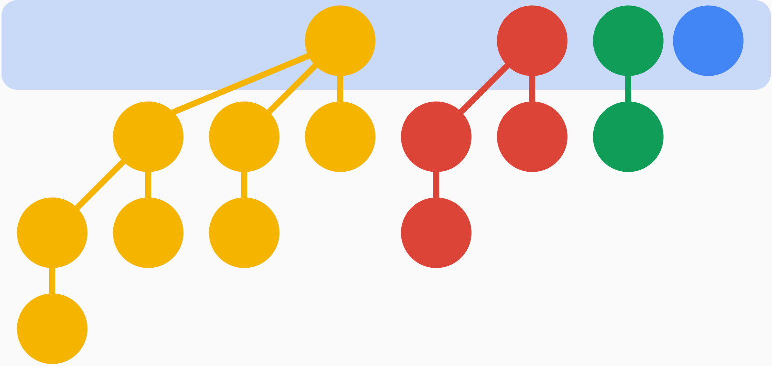
No more trees to combine! We're done!



Minimum

Since every binomial tree obeys the heap property, the minimum element must be one of the root nodes of the binomial trees.

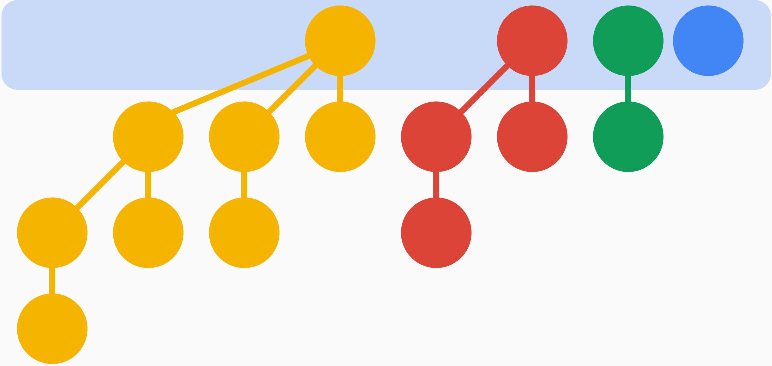
One of these must have the minimum value!



Minimum

Hence, we can iterate through the root nodes and return the one with the smallest value.

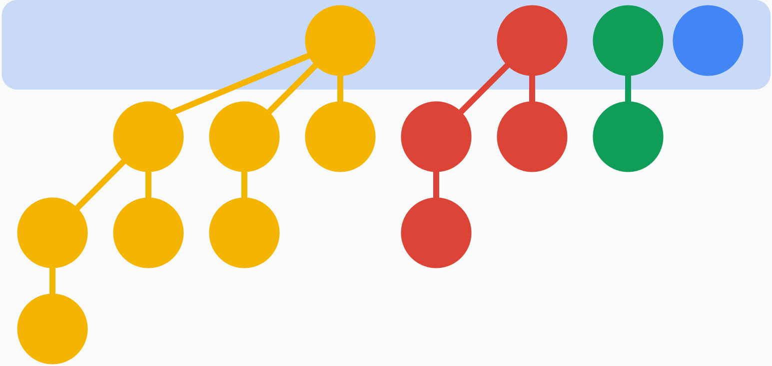
One of these must have the minimum value!



Minimum

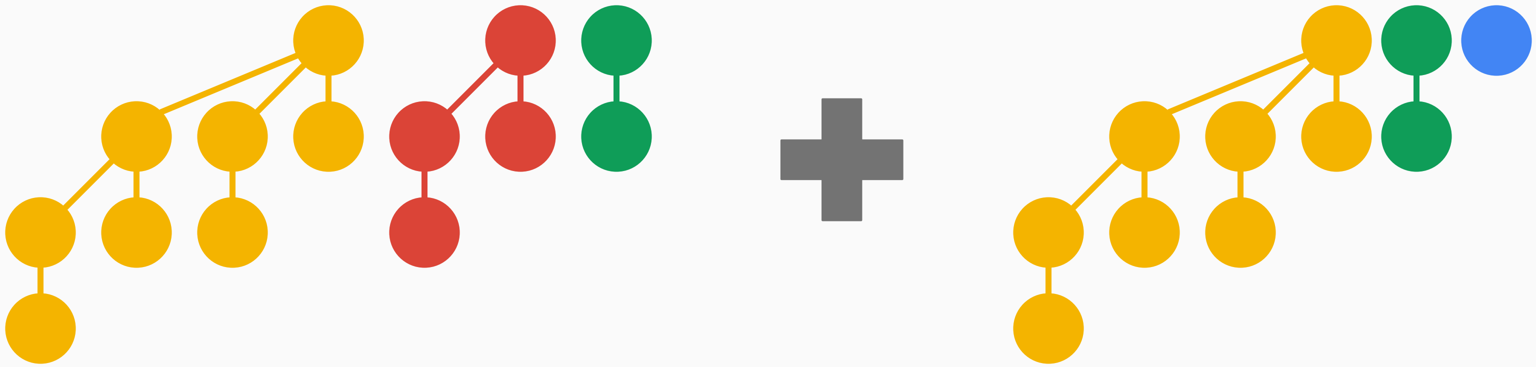
Since there are at most O(log n) binomial trees for a binomial heap with n nodes, Minimum runs in O(log n).

One of these must have the minimum value!



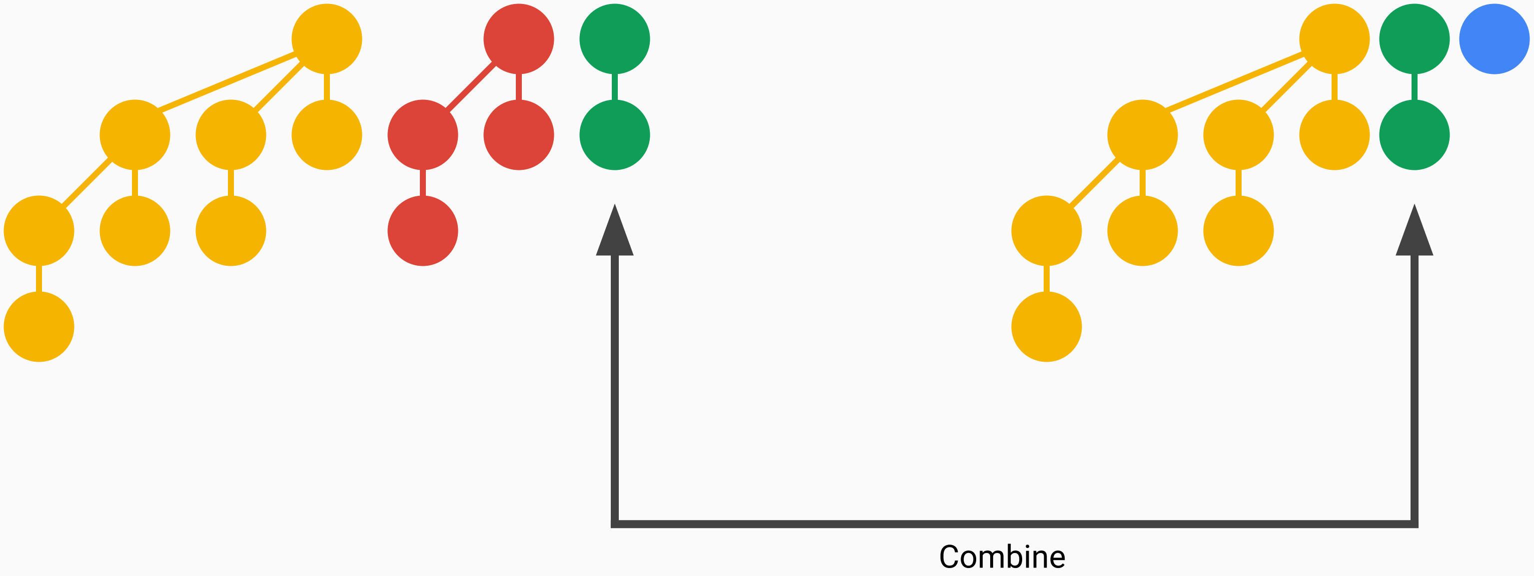
Union

Similar to the insert case, repeatedly combine binomial trees of the same rank together until all binomial trees have distinct ranks.



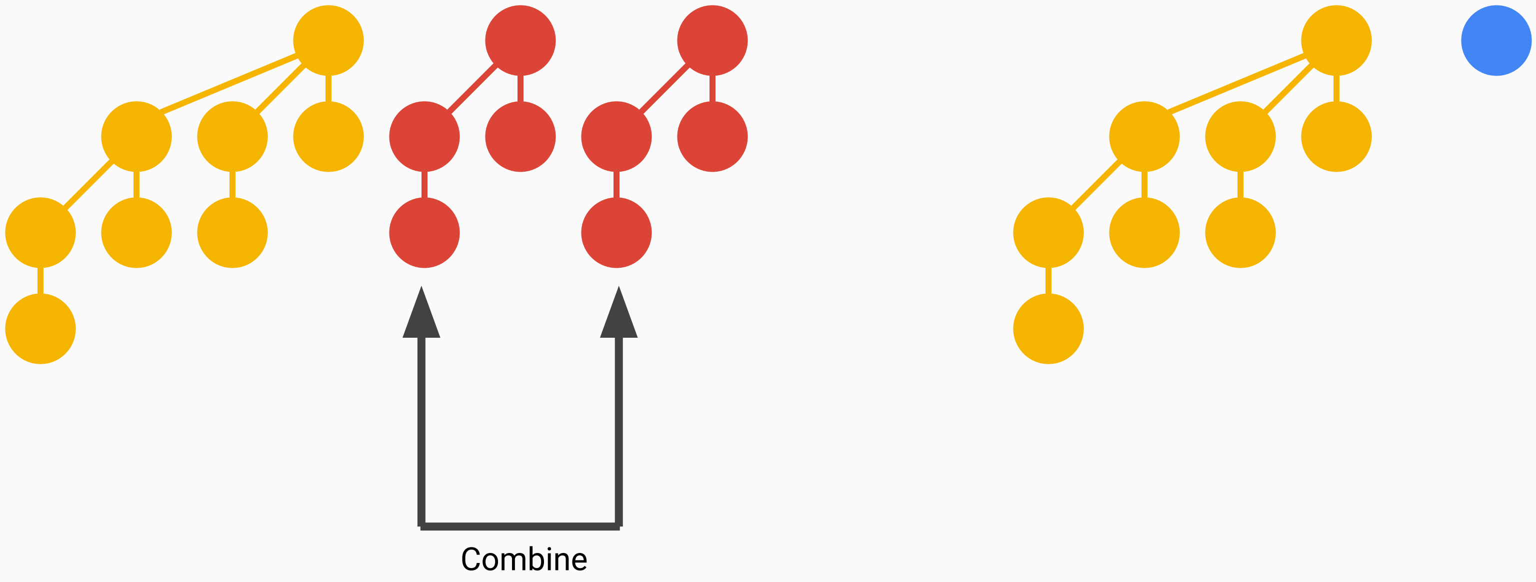
Union

Similar to the insert case, repeatedly combine binomial trees of the same rank together until all binomial trees have distinct ranks.



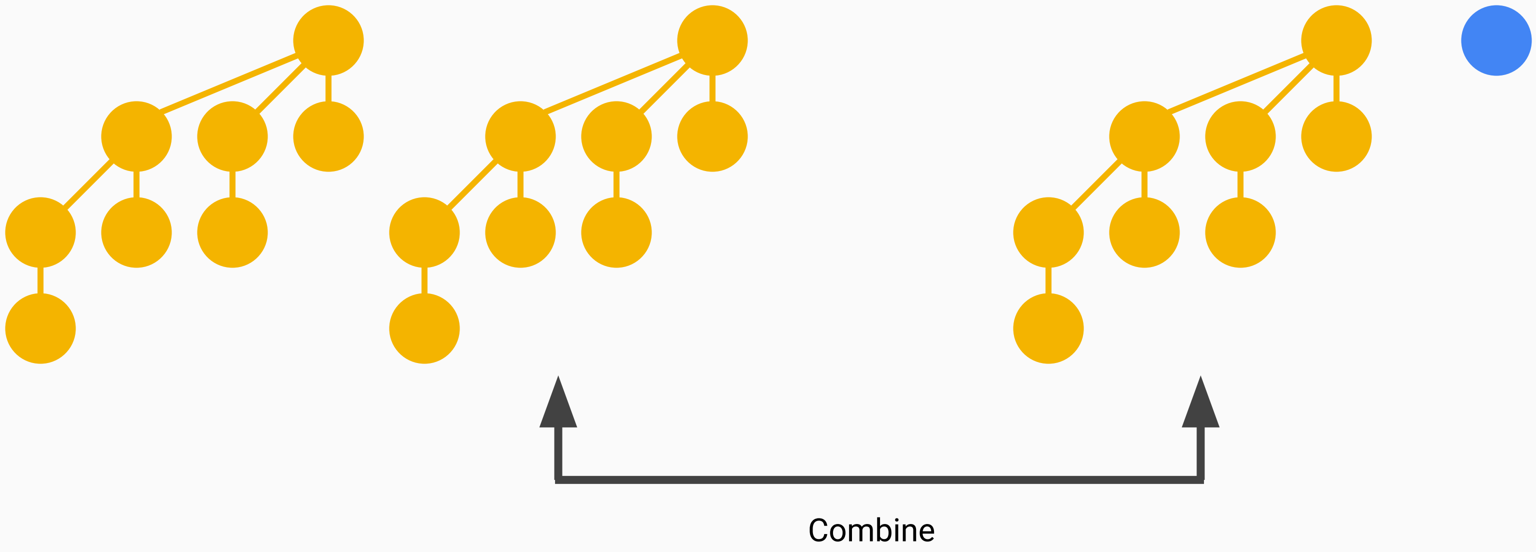
Union

Similar to the insert case, repeatedly combine binomial trees of the same rank together until all binomial trees have distinct ranks.



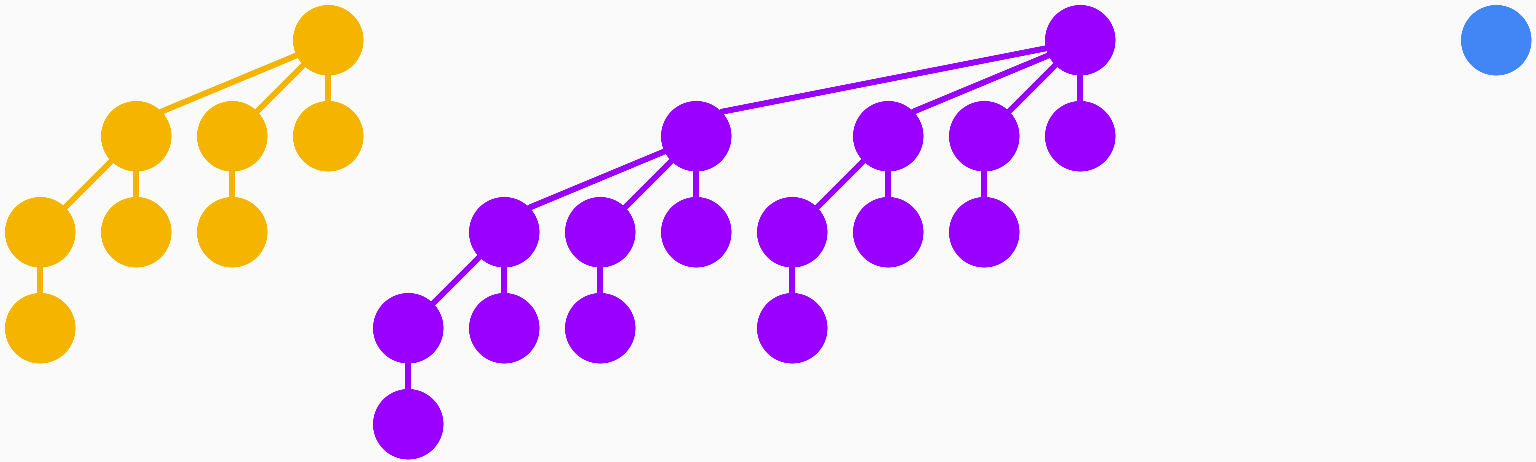
Union

Similar to the insert case, repeatedly combine binomial trees of the same rank together until all binomial trees have distinct ranks.



Union

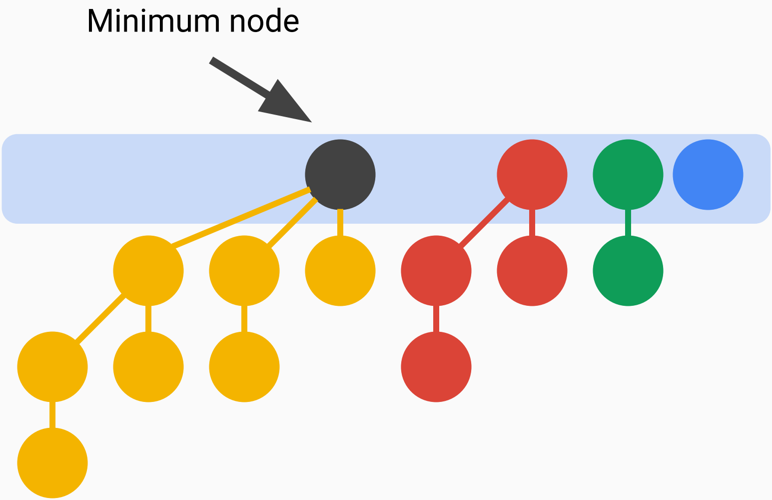
Similar to the insert case, repeatedly combine binomial trees of the same rank together until all binomial trees have distinct ranks.



No more trees to combine! We're done!

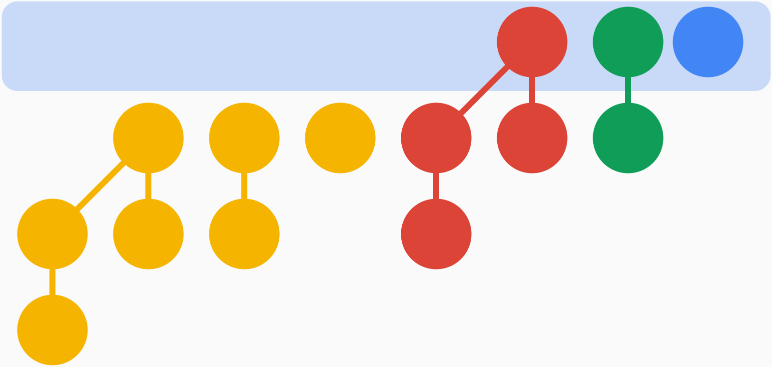
Extract-Min

First, find the minimum node (similar to Minimum).



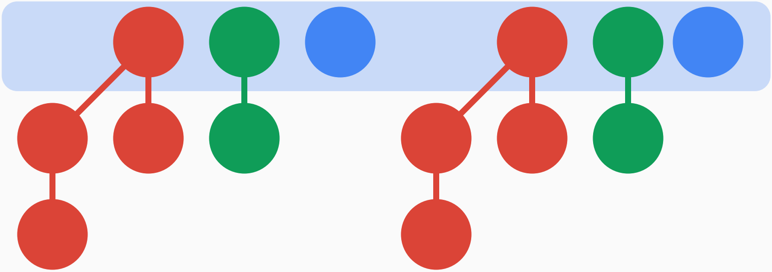
Extract-Min

Then, delete it, and place its children in the root list.



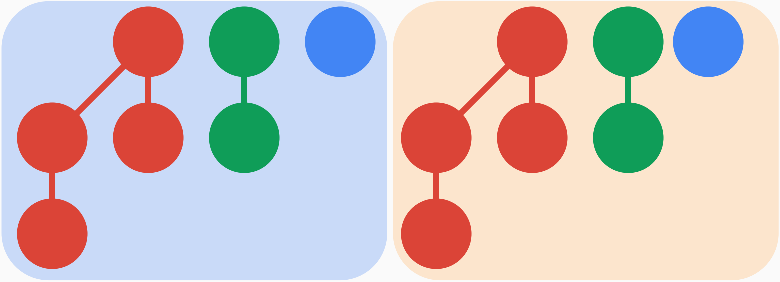
Extract-Min

Then, delete it, and place its children in the root list.



Extract-Min

Now, you have two binomial heaps. Run Union on the two heaps.



Decrease-Key

Change the value of the required node, then run heapify on the binomial tree that the node is in.

Delete

First decrease the key on the node to be deleted to $-\infty$.

Then, run Extract-Min.