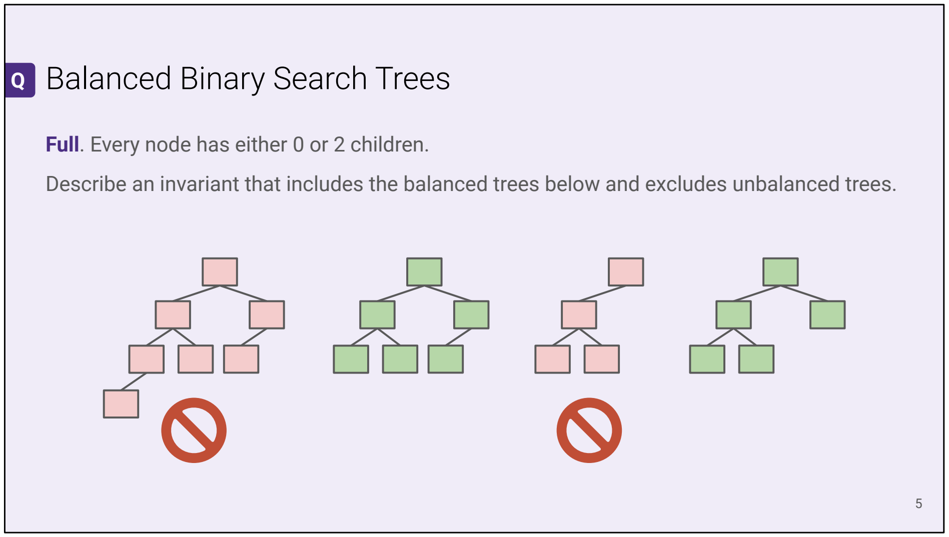


Balanced Binary Search Trees

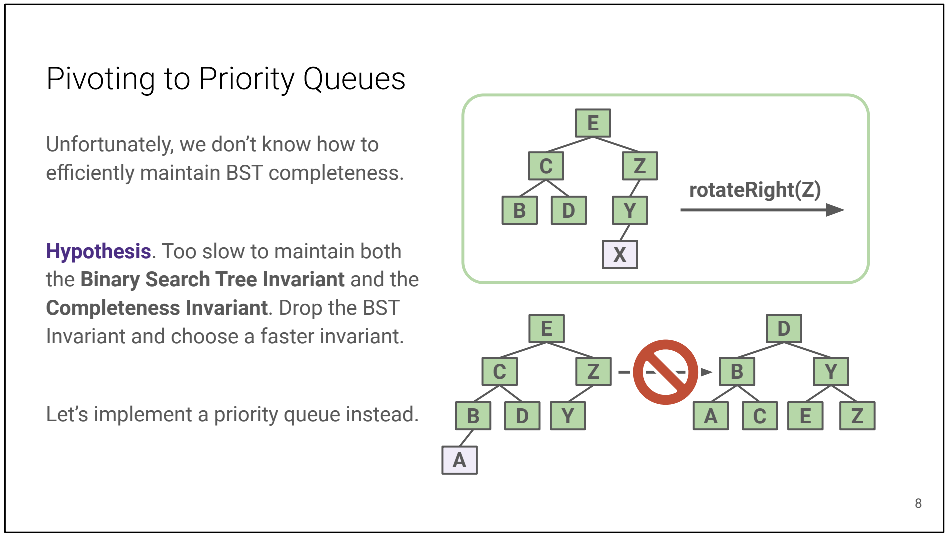
Full. Every node has either 0 or 2 children.

Describe an invariant that includes the balanced trees below and excludes unbalanced trees.

5

Let's go back in time. What if we don't accept this resolution? Sure, a full binary search tree does not guarantee balance. However, we can come up with another invariant that does guarantee balance!

Q1: Describe an invariant that includes the balanced trees below, and excludes unbalanced trees.



Pivoting to Priority Queues

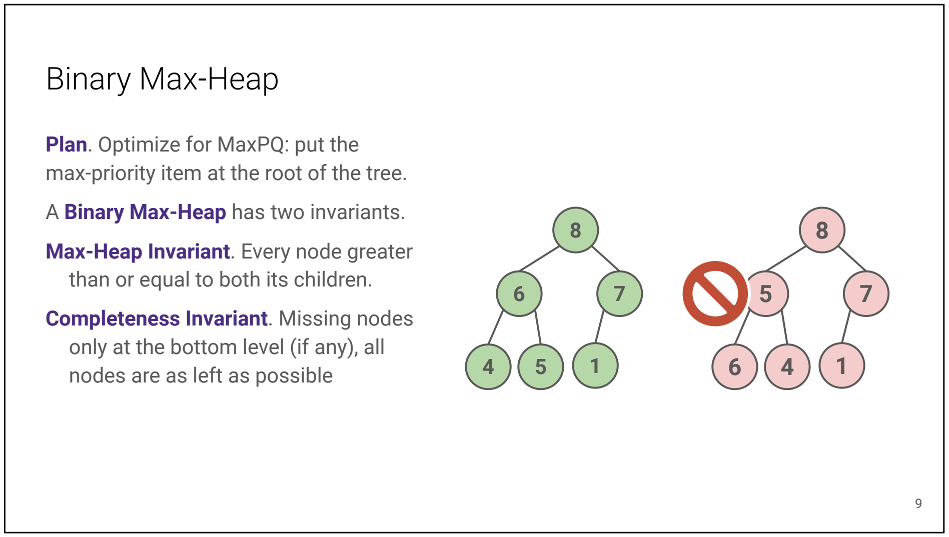
Unfortunately, we don't know how to efficiently maintain BST completeness.

Hypothesis. Too slow to maintain both the **Binary Search Tree Invariant** and the **Completeness Invariant**. Drop the BST Invariant and choose a faster invariant.

Let's implement a priority queue instead.

8

?: For the bottom example, give an asymptotic lower bound (i.e. Big-Omega) for the runtime to fix the tree where N is the number of items.



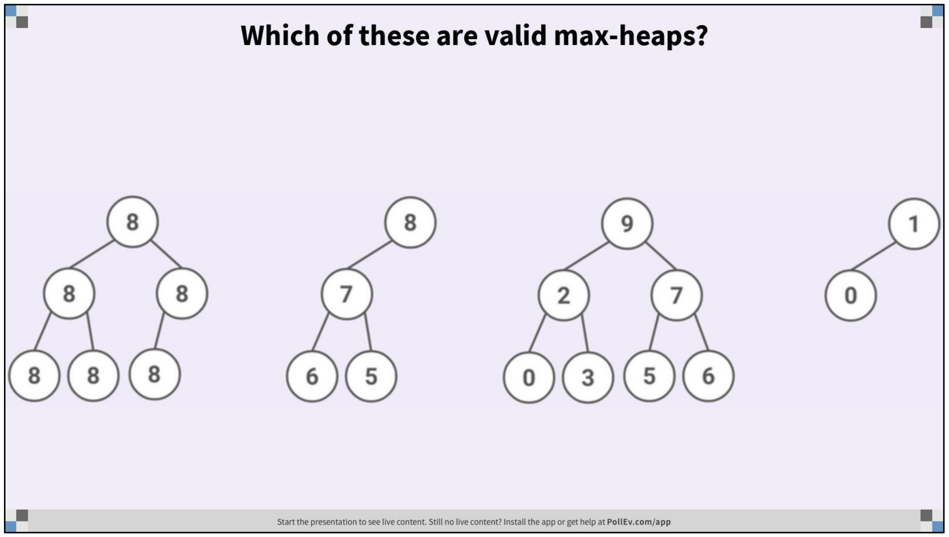
Binary Max-Heap

Plan. Optimize for MaxPQ: put the max-priority item at the root of the tree.

A **Binary Max-Heap** has two invariants.

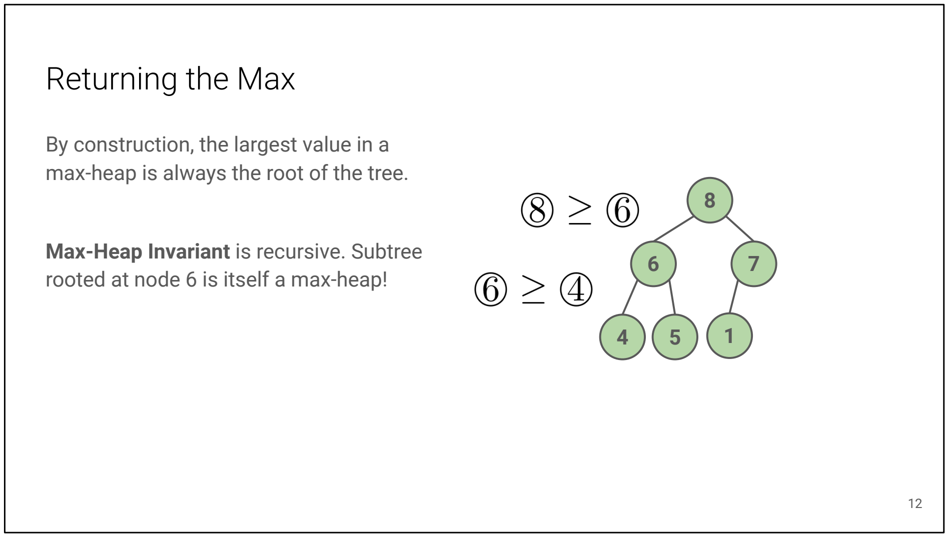
Max-Heap Invariant. Every node greater than or equal to both its children.

Completeness Invariant. Missing nodes only at the bottom level (if any), all nodes are as left as possible



Which of these are valid max-heaps?

Note that, in this visualization, the priority is the value shown in the node. Hereafter, we'll refer to the max-priority item as just "max item" for brevity.



Returning the Max

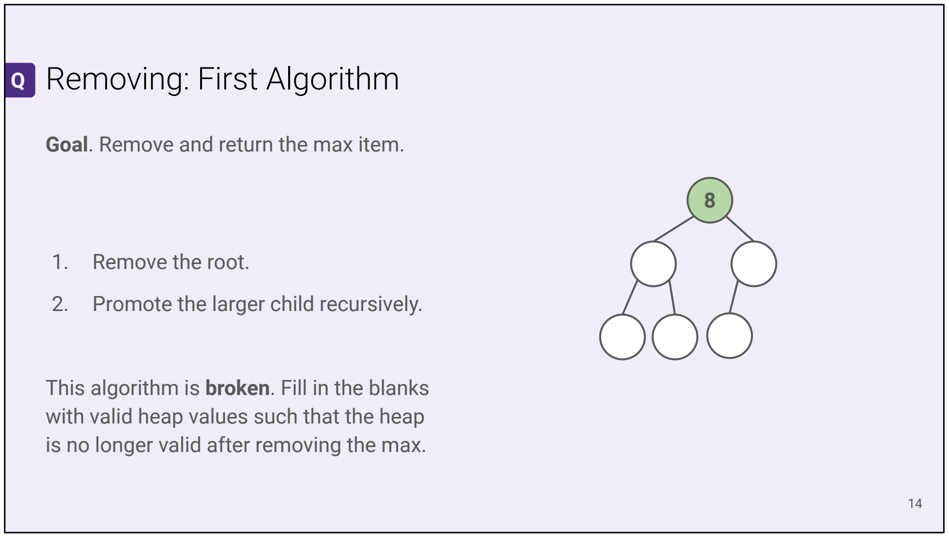
By construction, the largest value in a max-heap is always the root of the tree.

Max-Heap Invariant is recursive. Subtree rooted at node 6 is itself a max-heap!

$⑧ \geq ⑥$

$⑥ \geq ④$

?: What does the fact that the invariant is recursive guarantee about the relationship between the root 8 and its grandchildren, 4, 5, and 1? What about potential great-grandchildren?



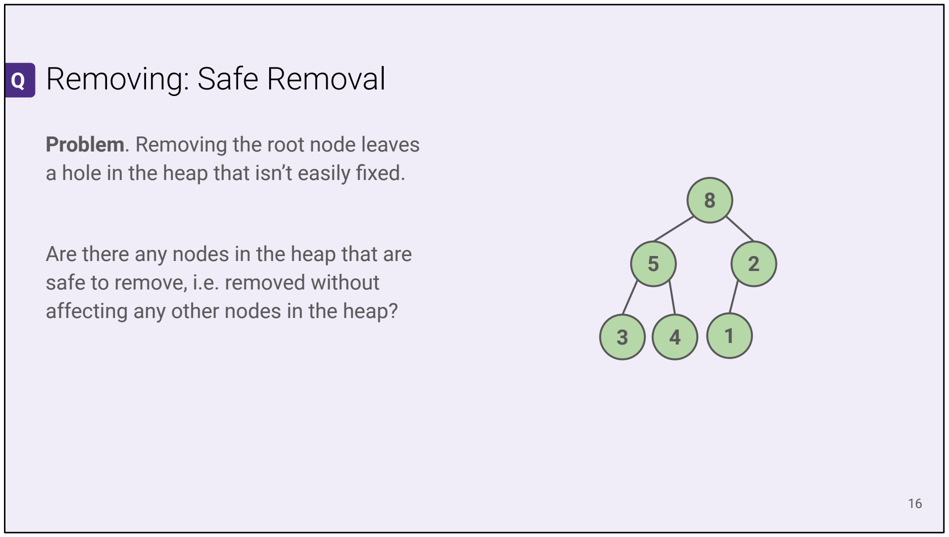
Q Removing: First Algorithm

Goal. Remove and return the max item.

1. Remove the root.
2. Promote the larger child recursively.

This algorithm is **broken**. Fill in the blanks with valid heap values such that the heap is no longer valid after removing the max.

Q1: This algorithm is **broken**. Fill in the blanks with valid heap values such that the heap is no longer valid after removing the max.



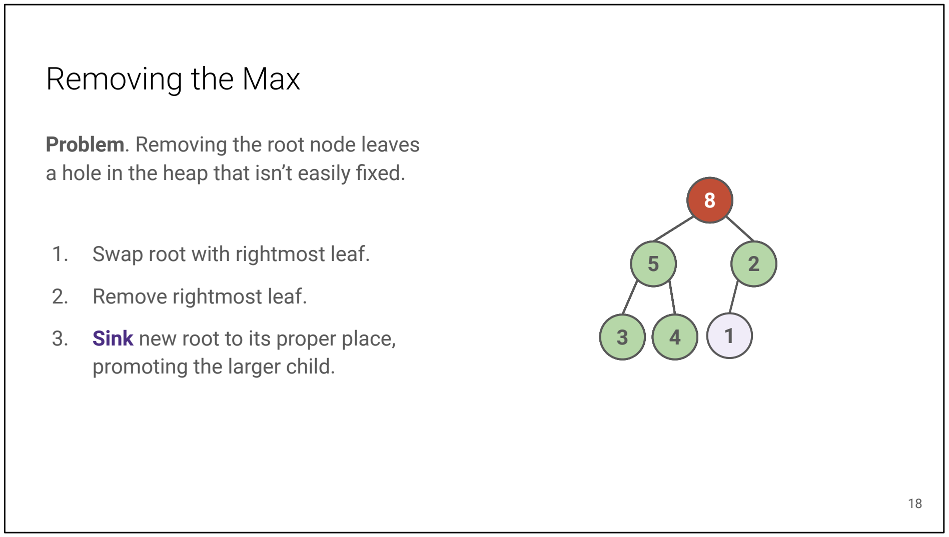
Q Removing: Safe Removal

Problem. Removing the root node leaves a hole in the heap that isn't easily fixed.

Are there any nodes in the heap that are safe to remove, i.e. removed without affecting any other nodes in the heap?

?: What invariants do we need to keep in mind when implementing remove?

Q1: Are there any nodes in the heap that are safe to remove, i.e. removed without affecting any other nodes in the heap?

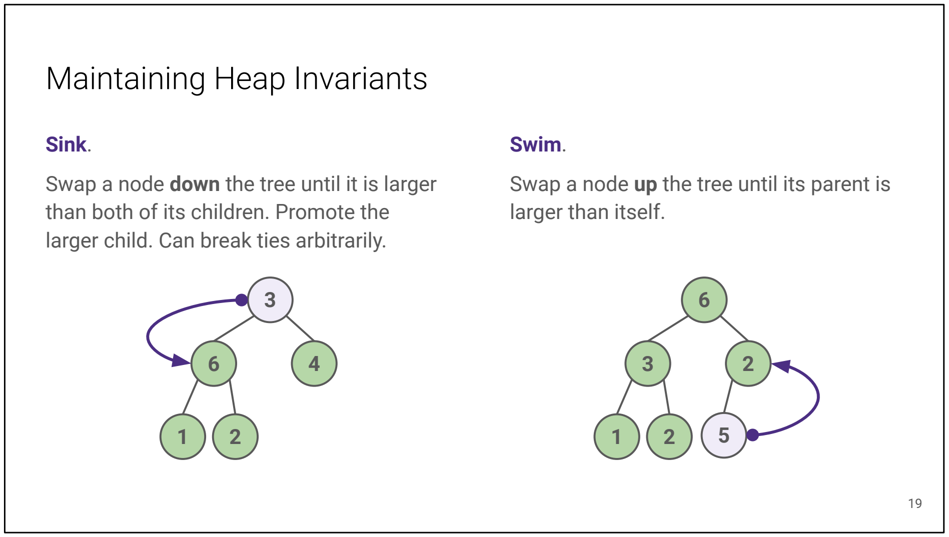


Removing the Max

Problem. Removing the root node leaves a hole in the heap that isn't easily fixed.

1. Swap root with rightmost leaf.
2. Remove rightmost leaf.
3. **Sink** new root to its proper place, promoting the larger child.

?: What about the two other leaf nodes on the bottom level? Why can't they be safely removed?



Maintaining Heap Invariants

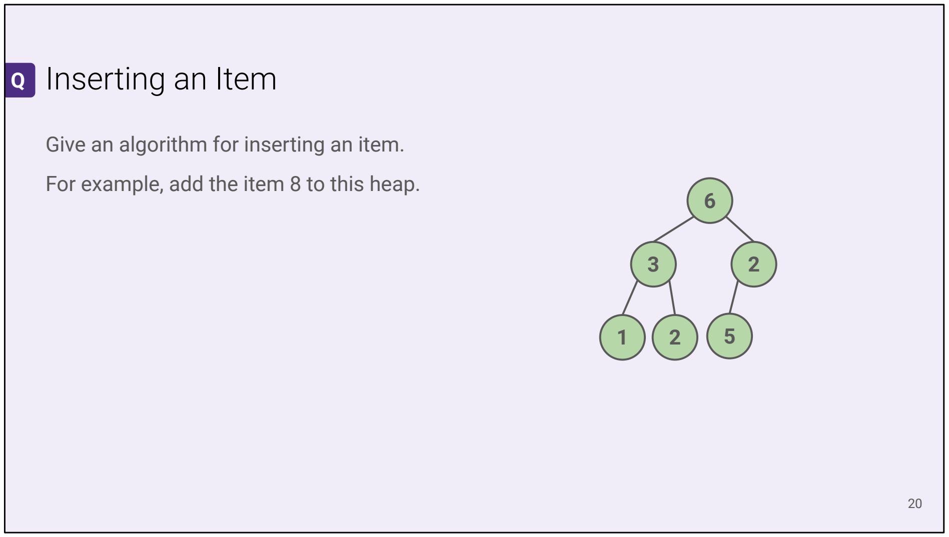
Sink.

Swap a node **down** the tree until it is larger than both of its children. Promote the larger child. Can break ties arbitrarily.

Swim.

Swap a node **up** the tree until its parent is larger than itself.

?: How can we use these operations to insert an item?

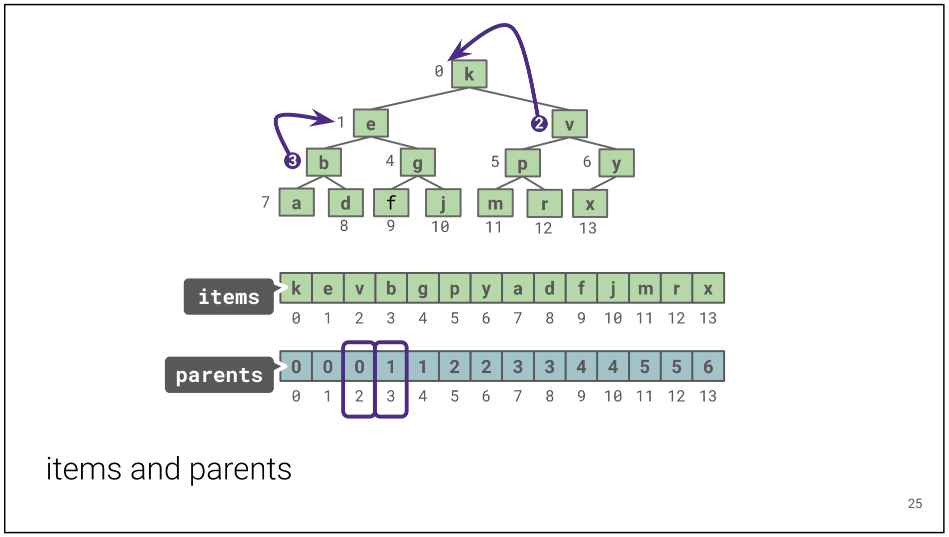


Q Inserting an Item

Give an algorithm for inserting an item.

For example, add the item 8 to this heap.

Q1: Give an algorithm for inserting an item. For example, add the item 8 to this heap.

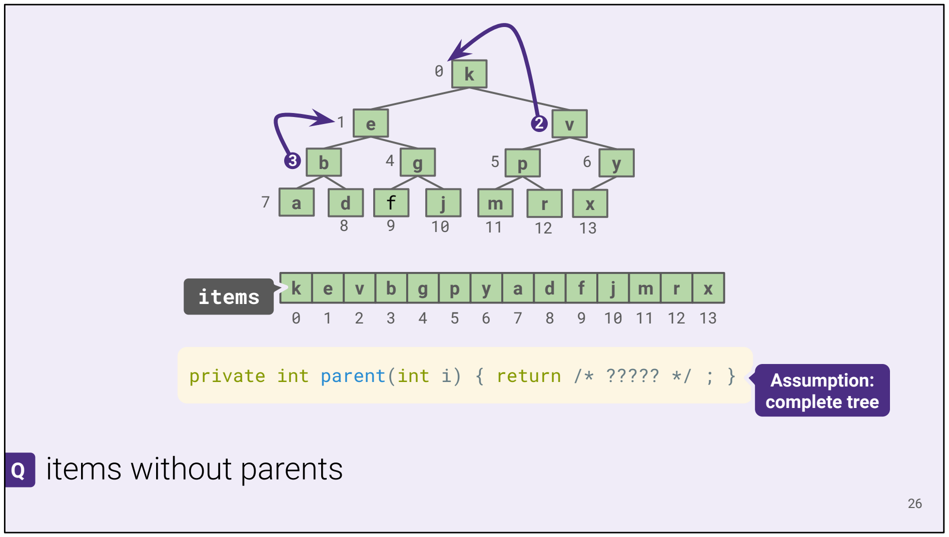


items	k	e	v	b	g	p	y	a	d	f	j	m	r	x
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

parents	0	0	0	1	1	2	2	3	3	4	4	5	5	6
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

items and parents

Note that the value of each letter (k, e, v, …) doesn't mean anything.



items	k	e	v	b	g	p	y	a	d	f	j	m	r	x
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

```
private int parent(int i) { return /* ????? */ ; }
```

Assumption: complete tree

Q items without parents

Q1: Complete the return statement in the parent method.