Heaps Analysis

CS 225 - Fall 2025

Mattox Beckman

Objectives

Objectives

Demonstrate the time complexity of insert and delete

Implement the heapify function

Implement heap sort

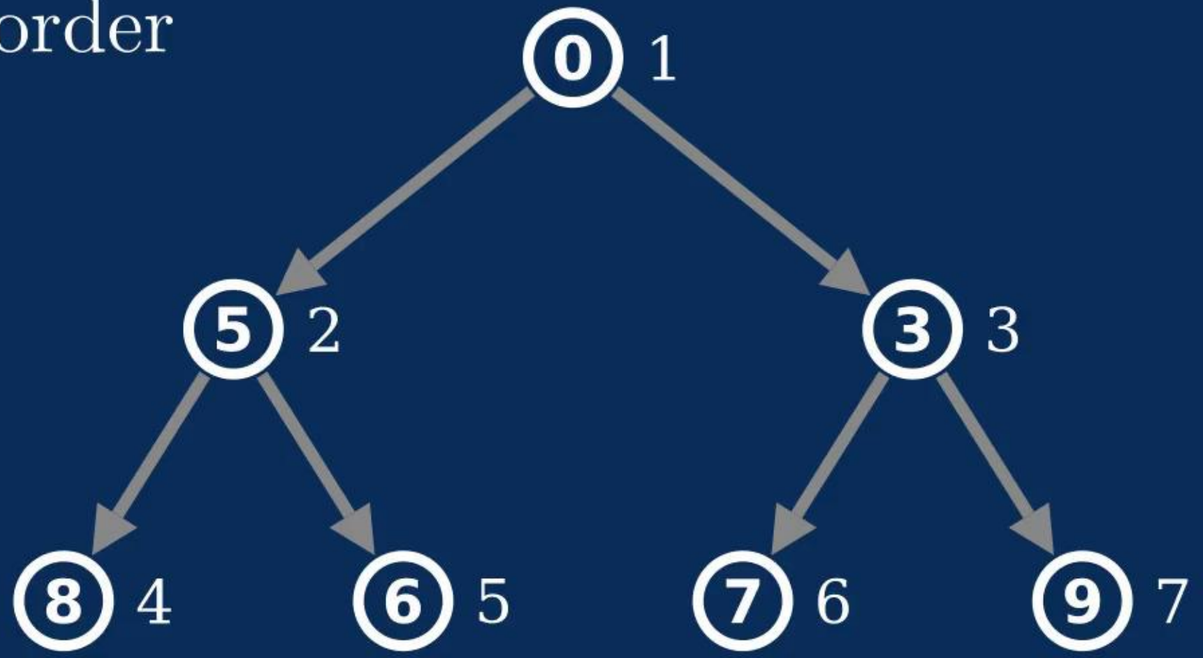
Nodes are placed in complete tree order

- Filled level by level
- Filled from left to right

Node priority higher than children

Nodes are numbered

We use an array for storage



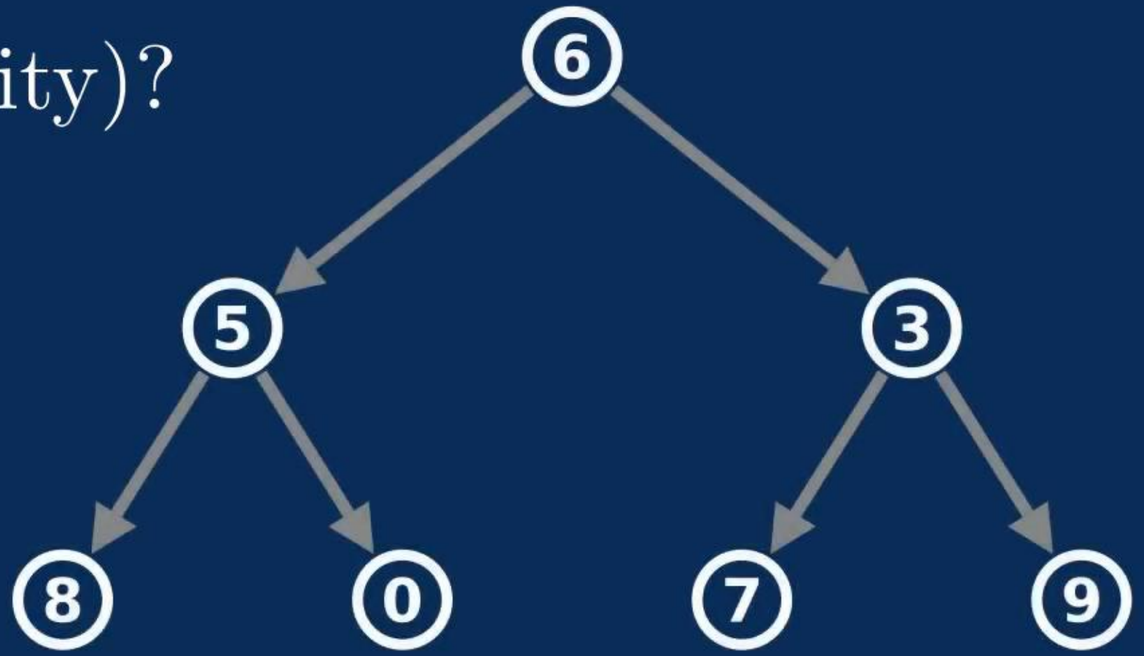
0	1	2	3	4	5	6	7
S	0	5	3	8	6	7	9

Percolate Up

Node n and parent p

Is $n < p$ (higher priority)?

- swap(n,p)
- recurse up.

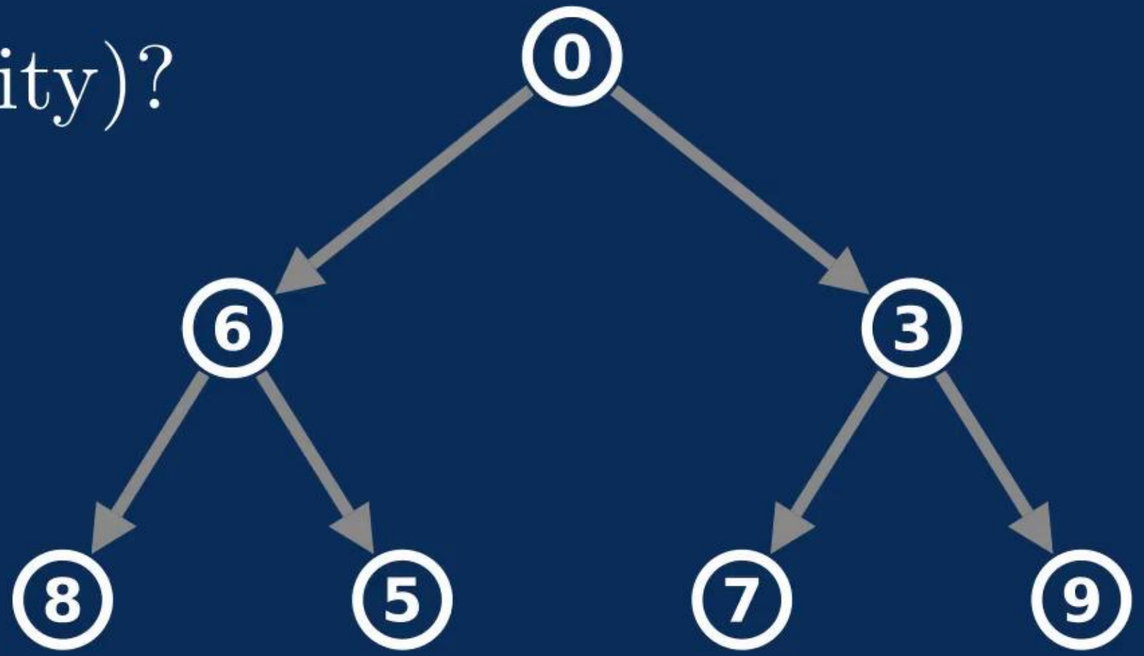


Percolate Up

Node n and parent p

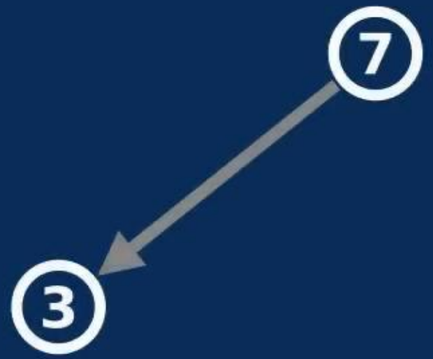
Is $n < p$ (higher priority)?

- swap(n,p)
- recurse up.



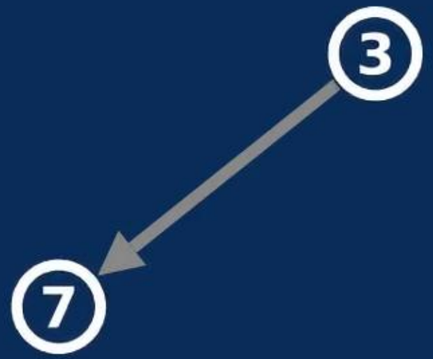
Percolate Down

If a child is smaller, swap with smallest child



Percolate Down

If a child is smaller, swap with smallest child



Percolate Down

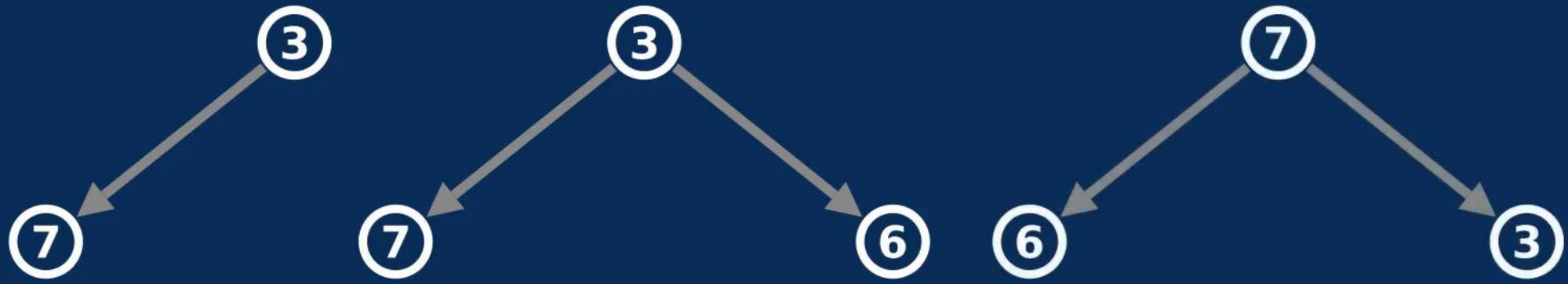
If a child is smaller, swap with smallest child

Percolate Down

If a child is smaller, swap with smallest child

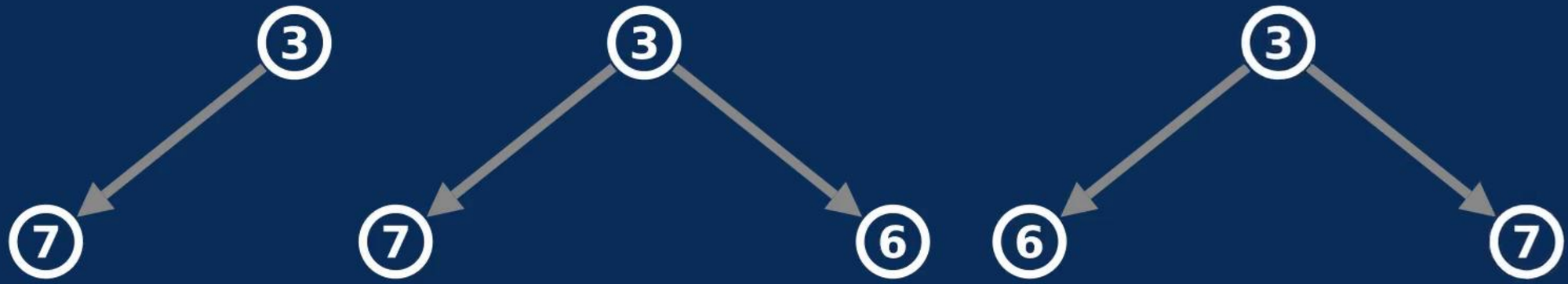
Percolate Down

If a child is smaller, swap with smallest child



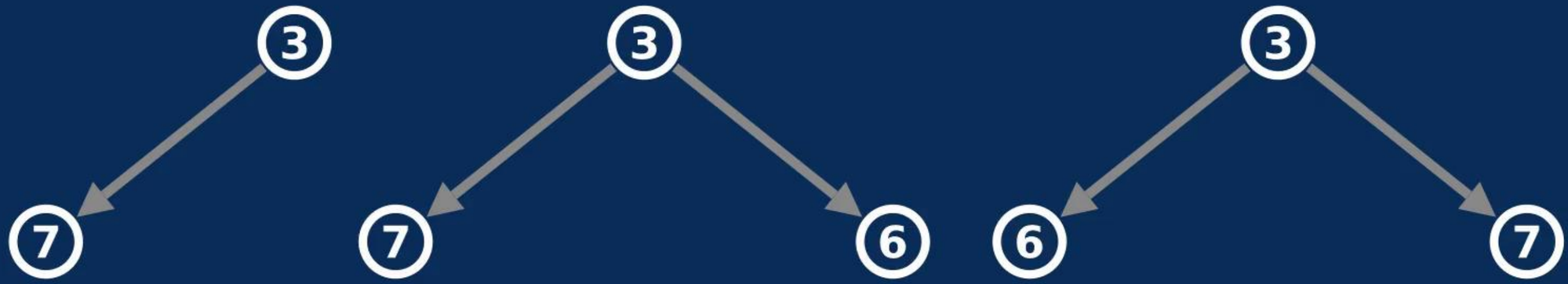
Percolate Down

If a child is smaller, swap with smallest child



Percolate Down

If a child is smaller, swap with smallest child

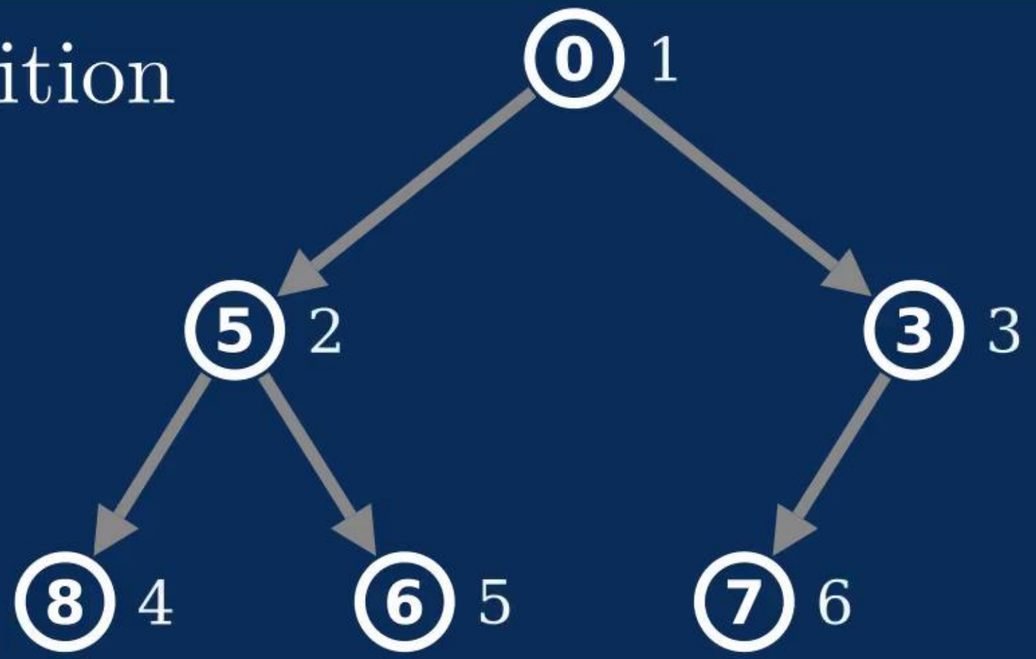



```
 1 while (true) {
 2    int left = idx * 2; int right = idx * 2 + 1; int smallest = idx;
 3    if (left < heap.size() &&
 4        heap[left] < heap[idx]) smallest = left;
 5    if (right < heap.size() &&
 6        heap[right] < heap[smallest]) smallest = right;
 7    if (smallest == idx) break;
 8    swap(heap[idx],heap[smallet]);
 9    idx = smallest;
10 }}
```

Insert

Place new element in last position

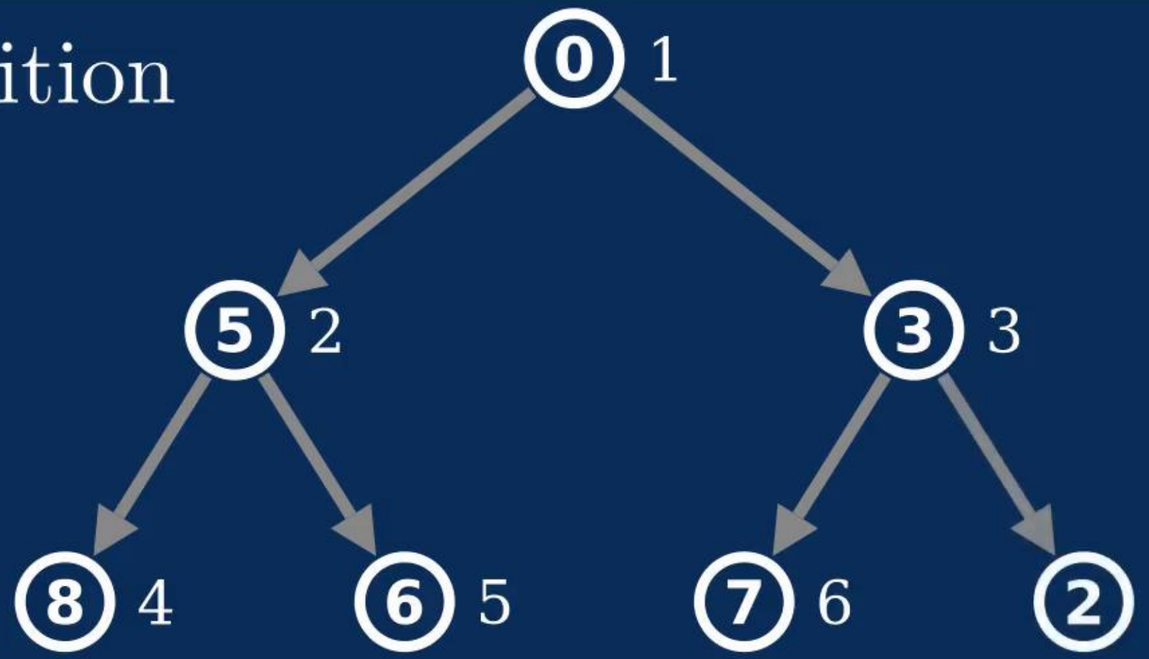
Call `percolate up` on it.



Insert

Place new element in last position

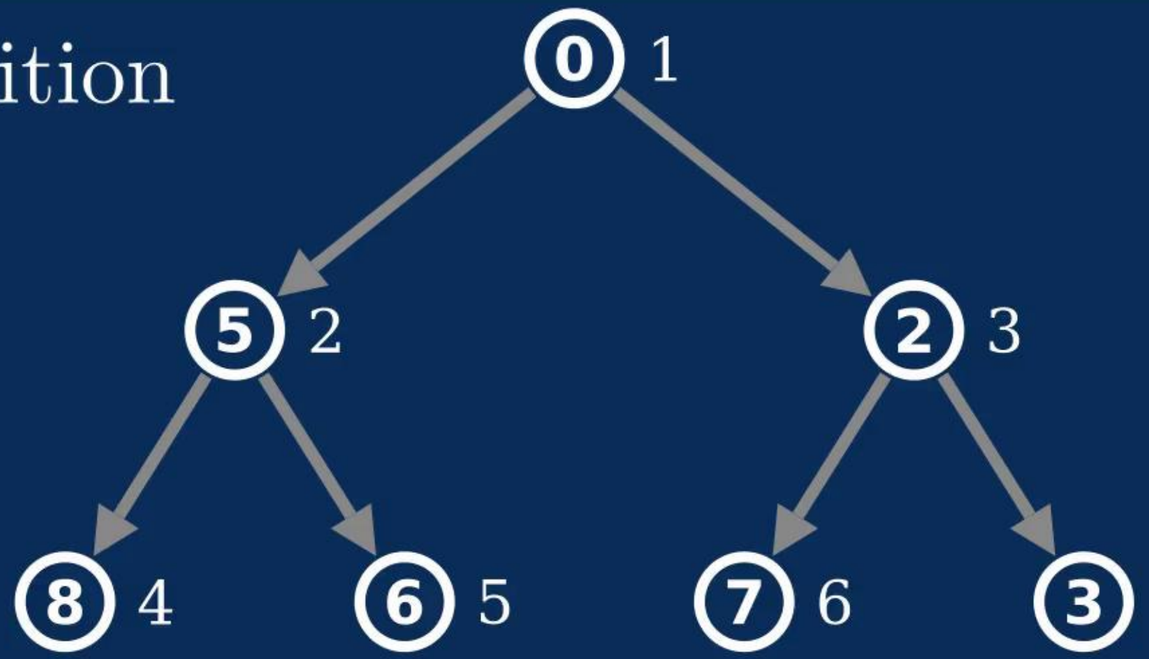
Call `percolate up` on it.



Insert

Place new element in last position

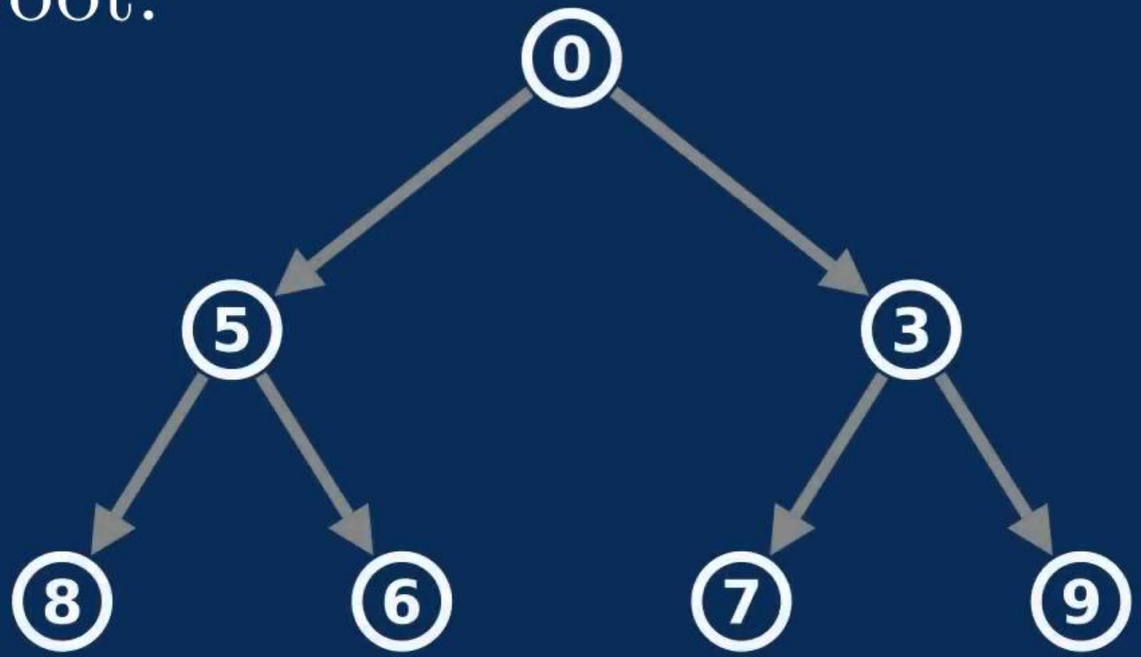
Call `percolate up` on it.



Delete Min

Overwrite last element with root.

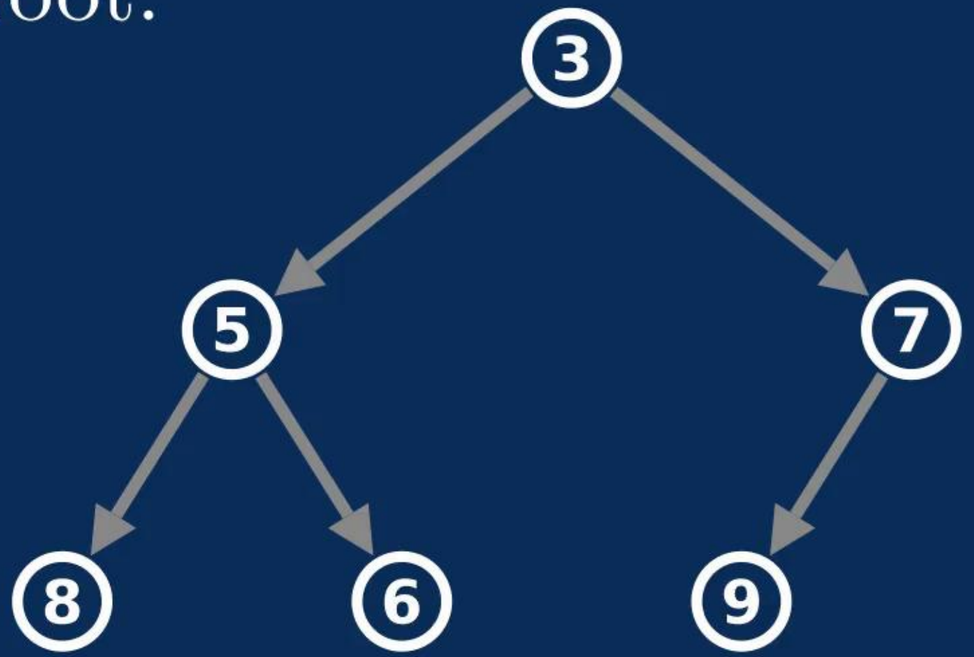
Call `percolate down` on it.



Delete Min

Overwrite last element with root.

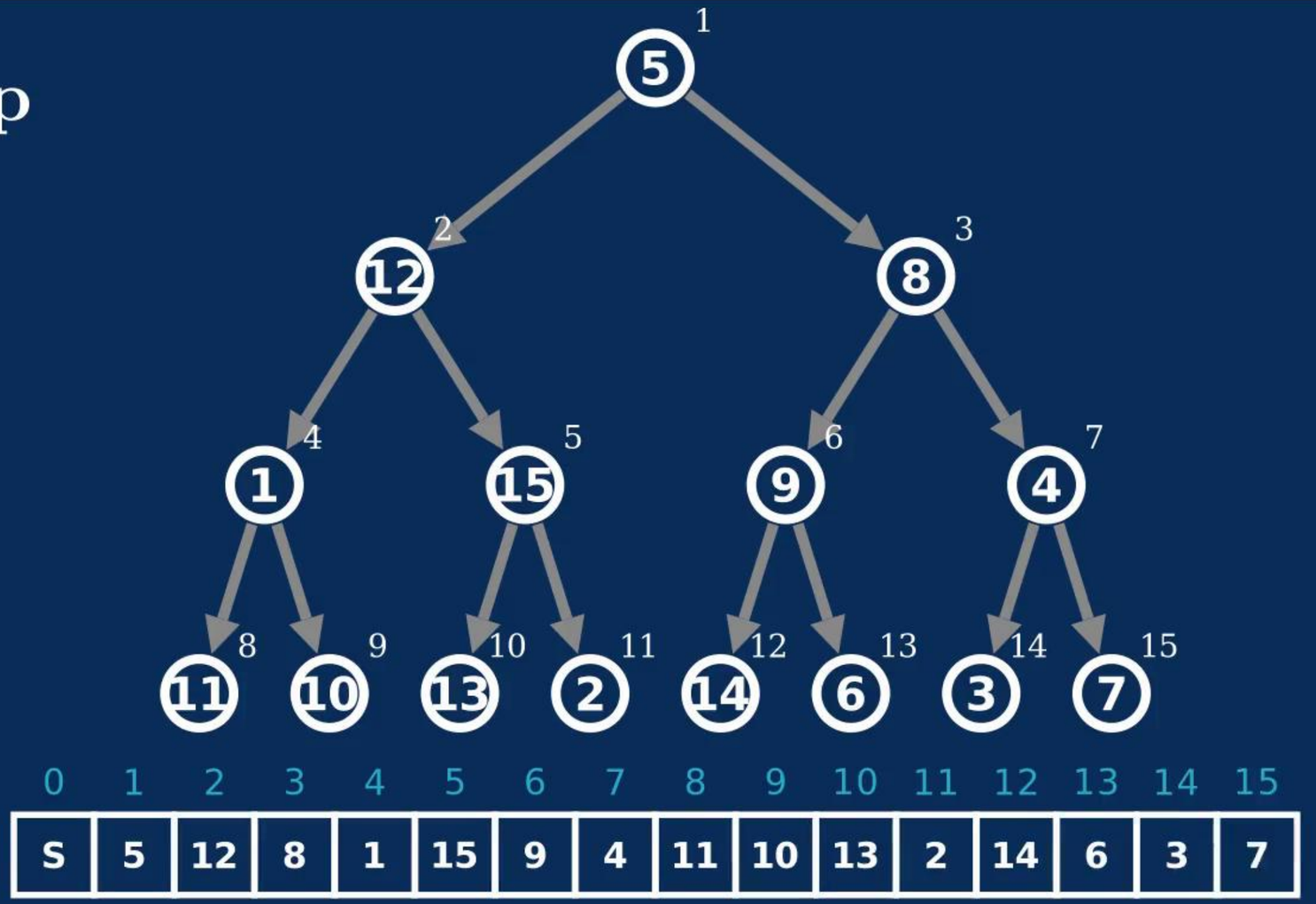
Call `percolate down` on it.



Heapify

Creating a Heap

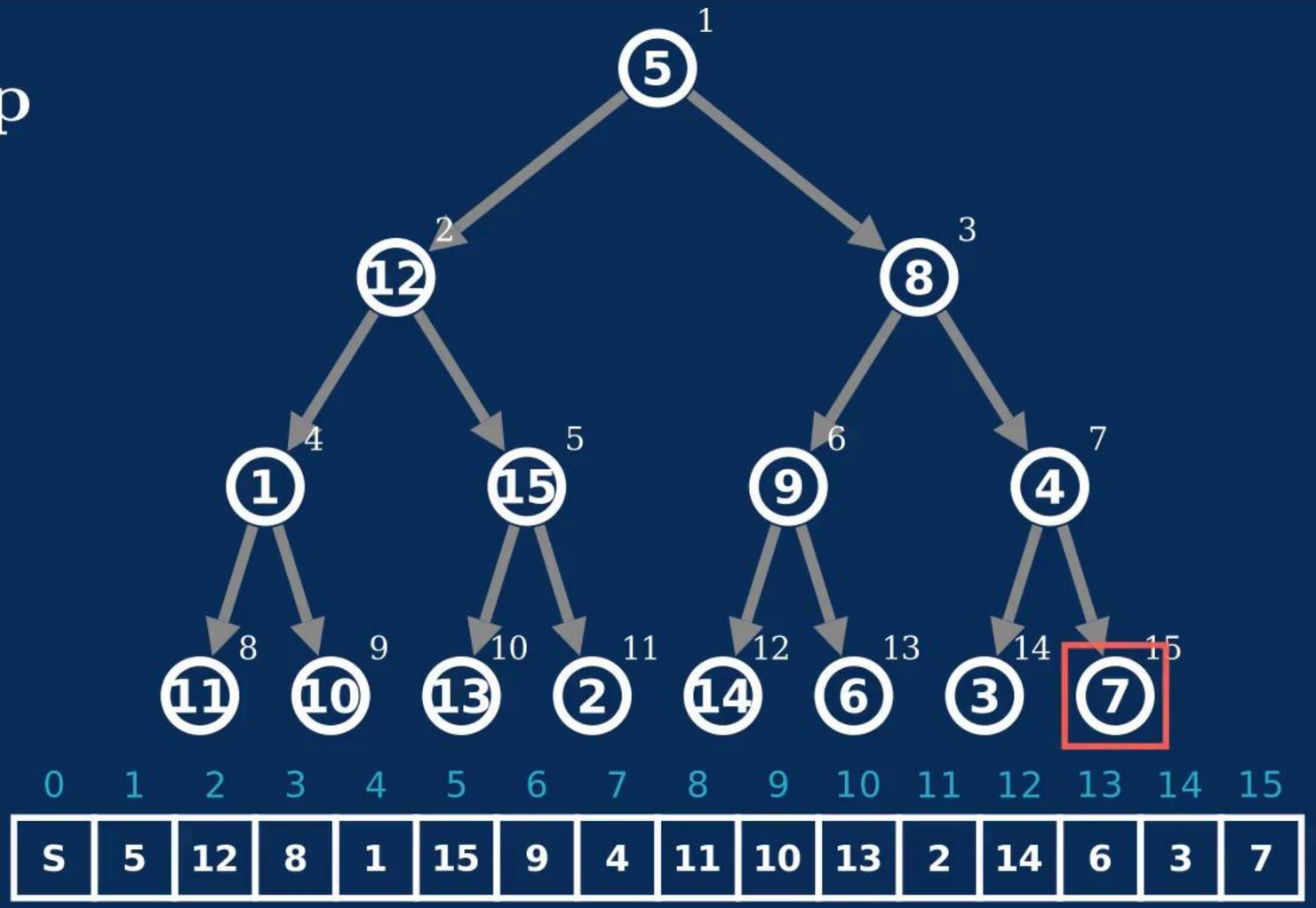
How to start?



Heapify

Creating a Heap

How to start?

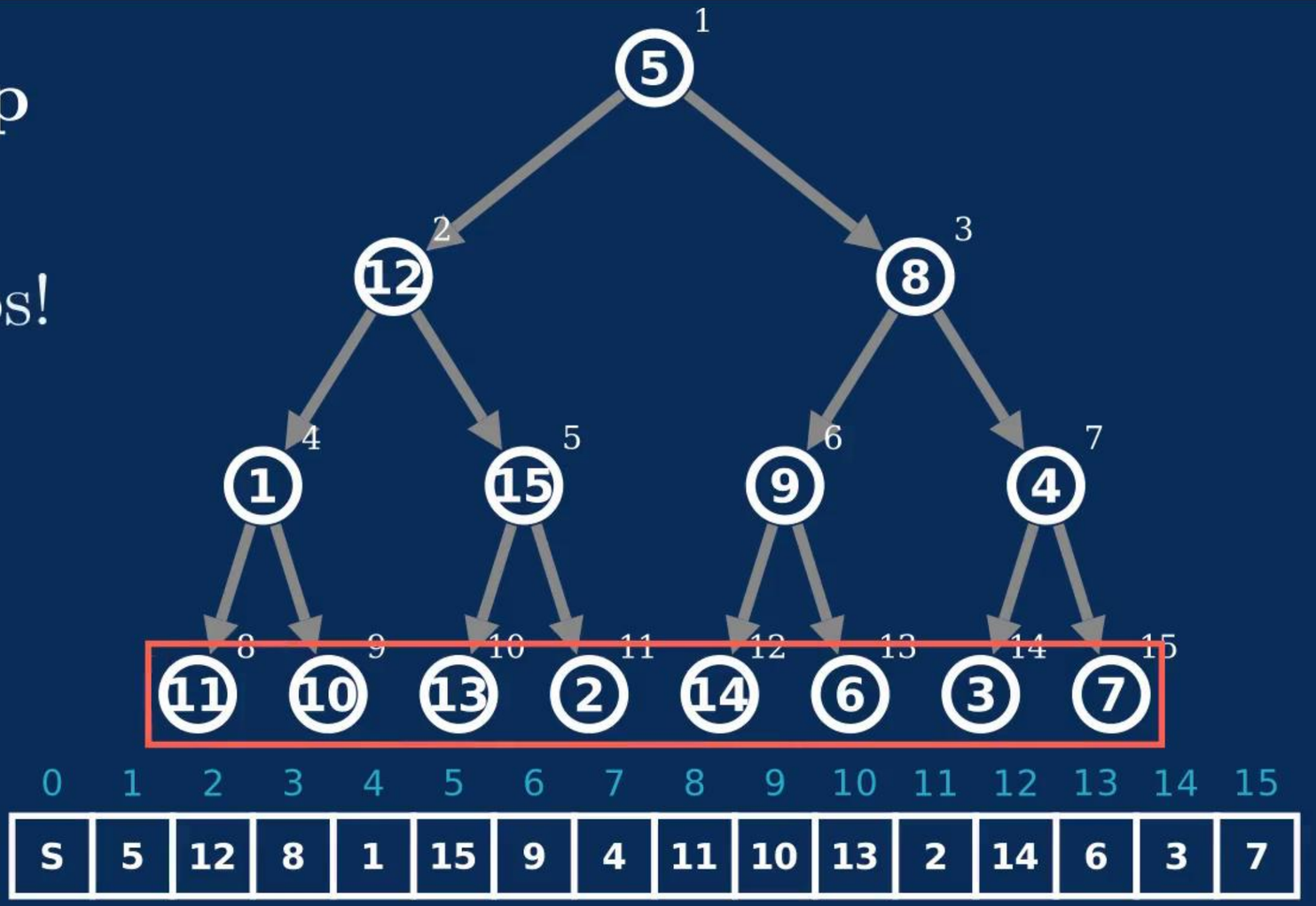


Heapify

Creating a Heap

How to start?

All these are heaps!



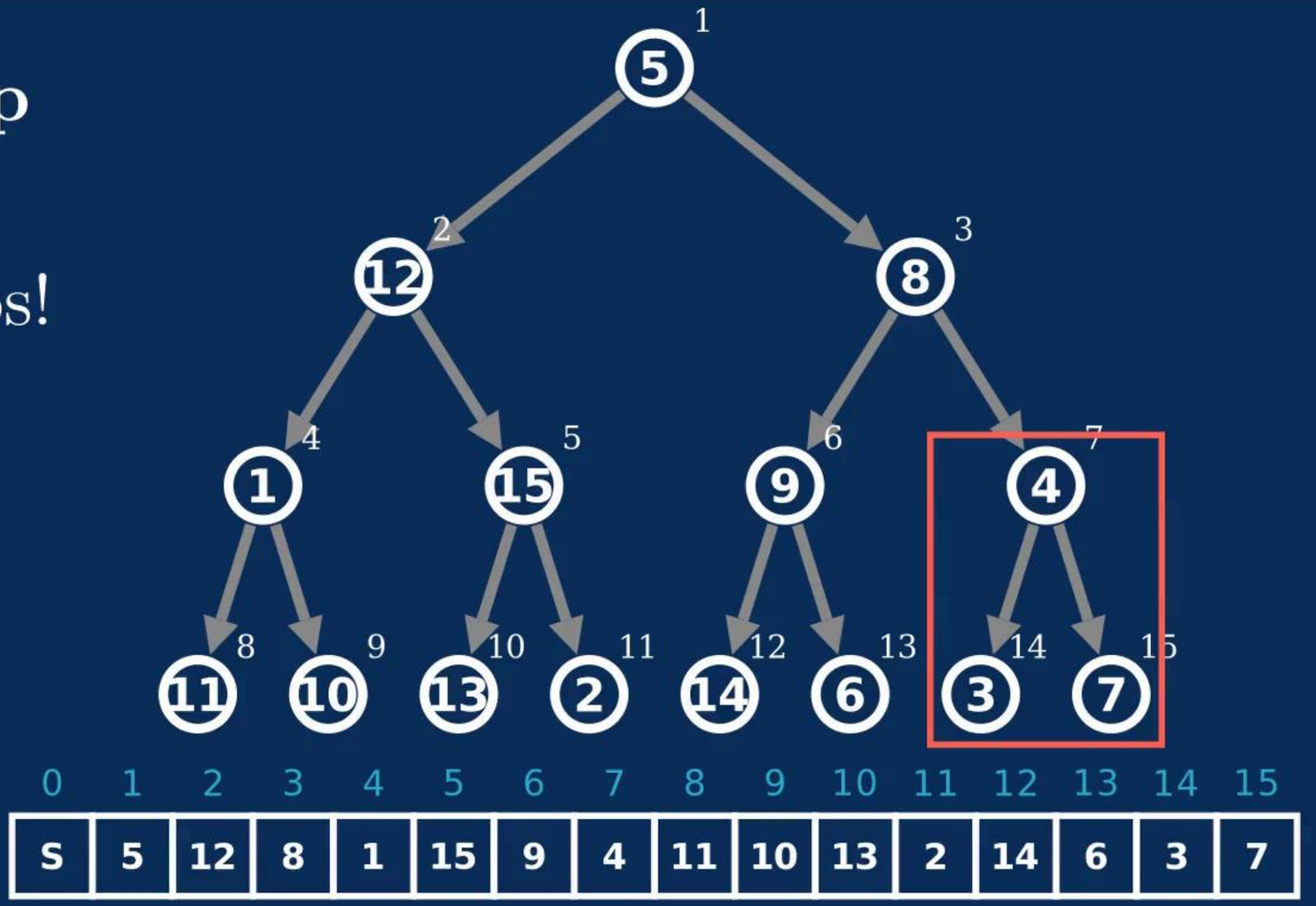
Heapify

Creating a Heap

How to start?

All these are heaps!

How about this?



Heapify

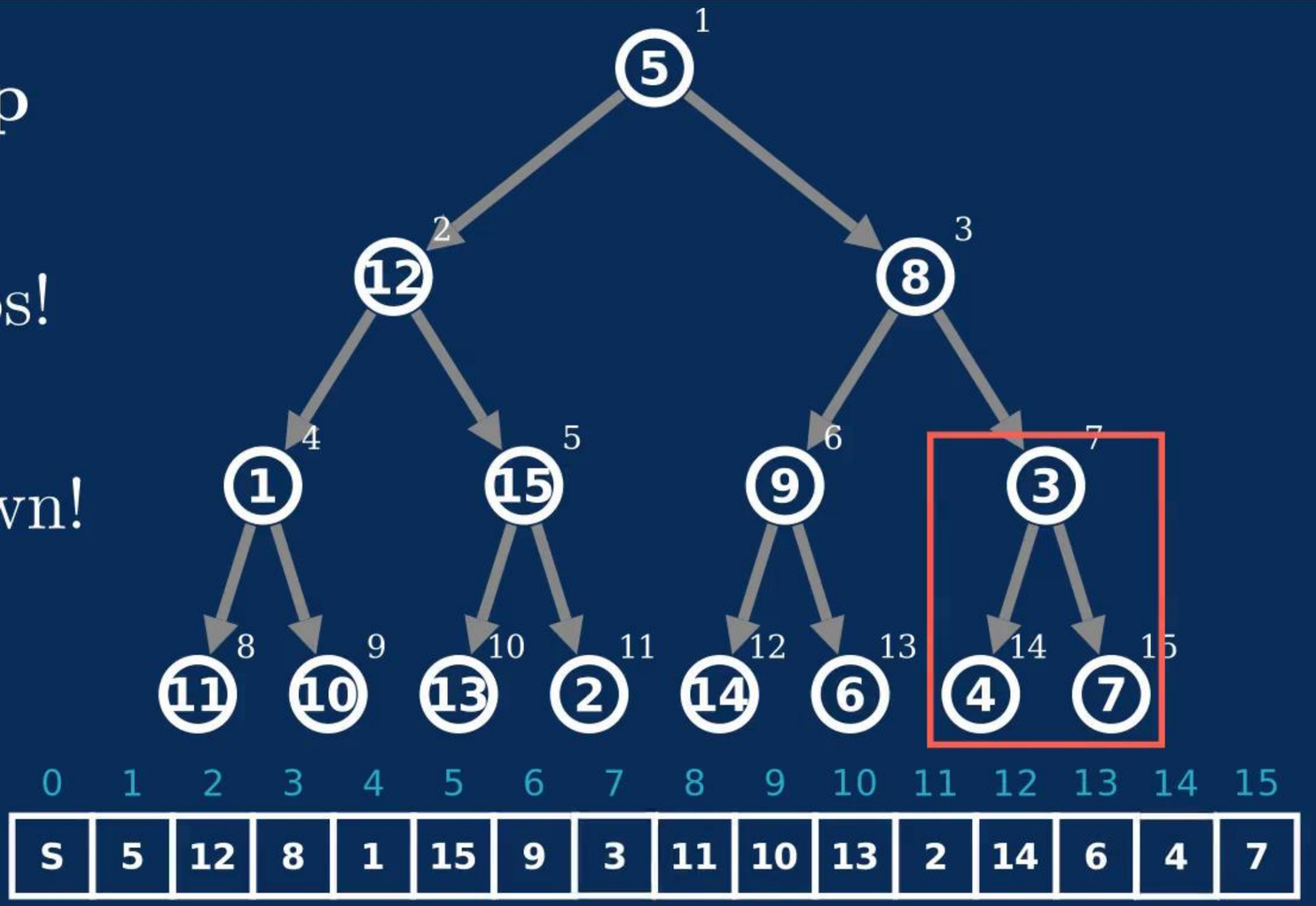
Creating a Heap

How to start?

All these are heaps!

How about this?

Just percolate down!



Heapify

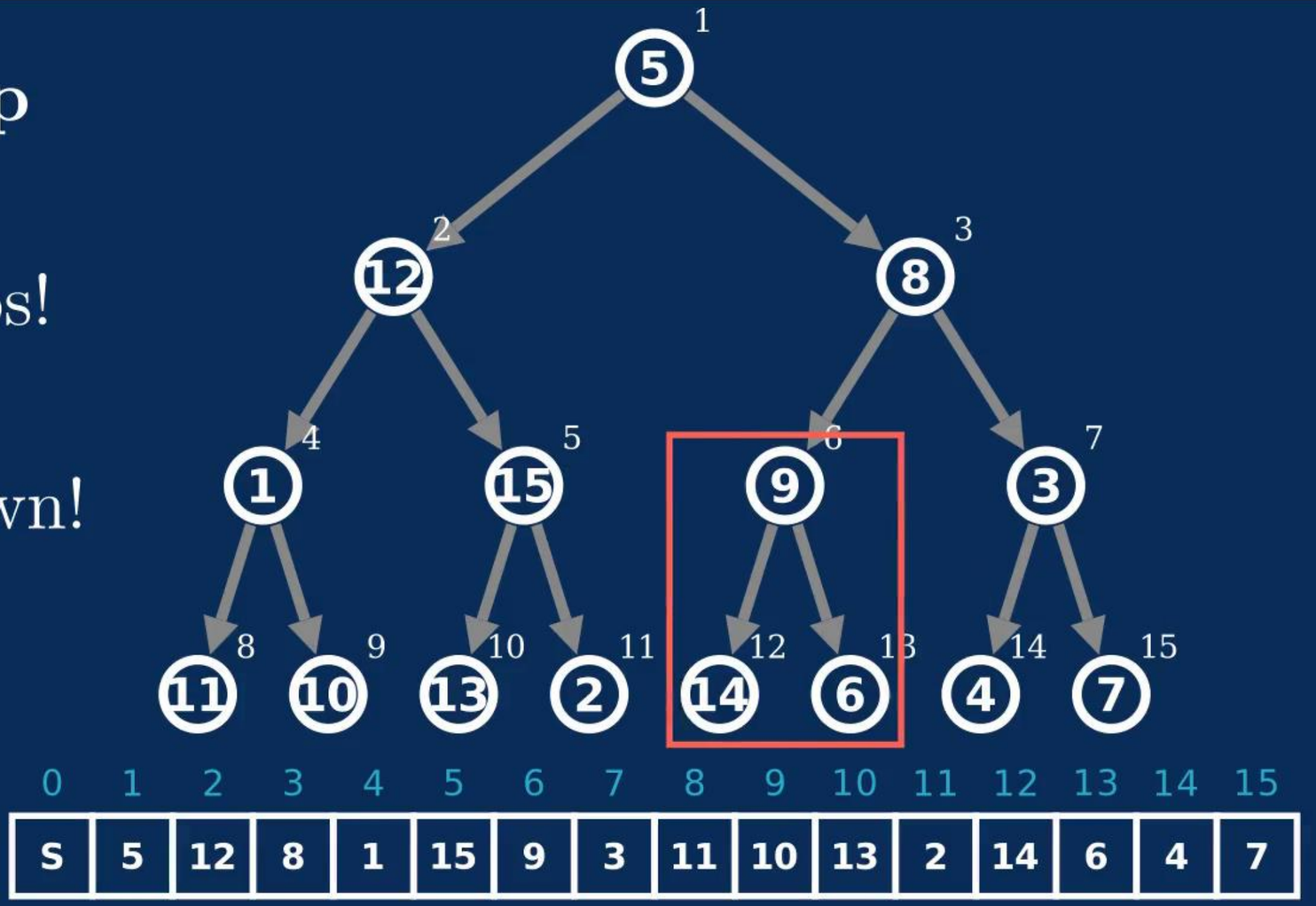
Creating a Heap

How to start?

All these are heaps!

How about this?

Just percolate down!



Heapify

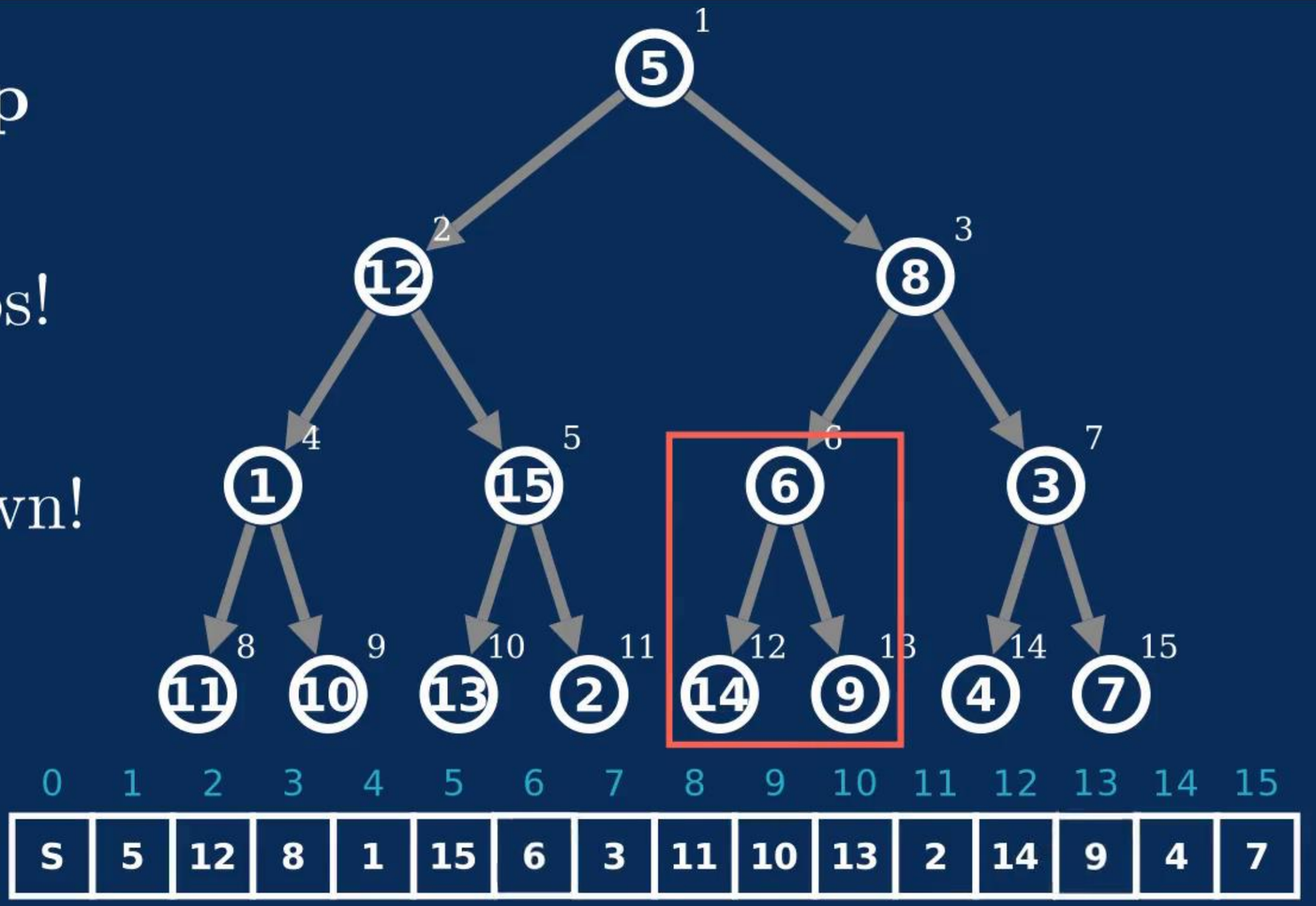
Creating a Heap

How to start?

All these are heaps!

How about this?

Just percolate down!



Heapify

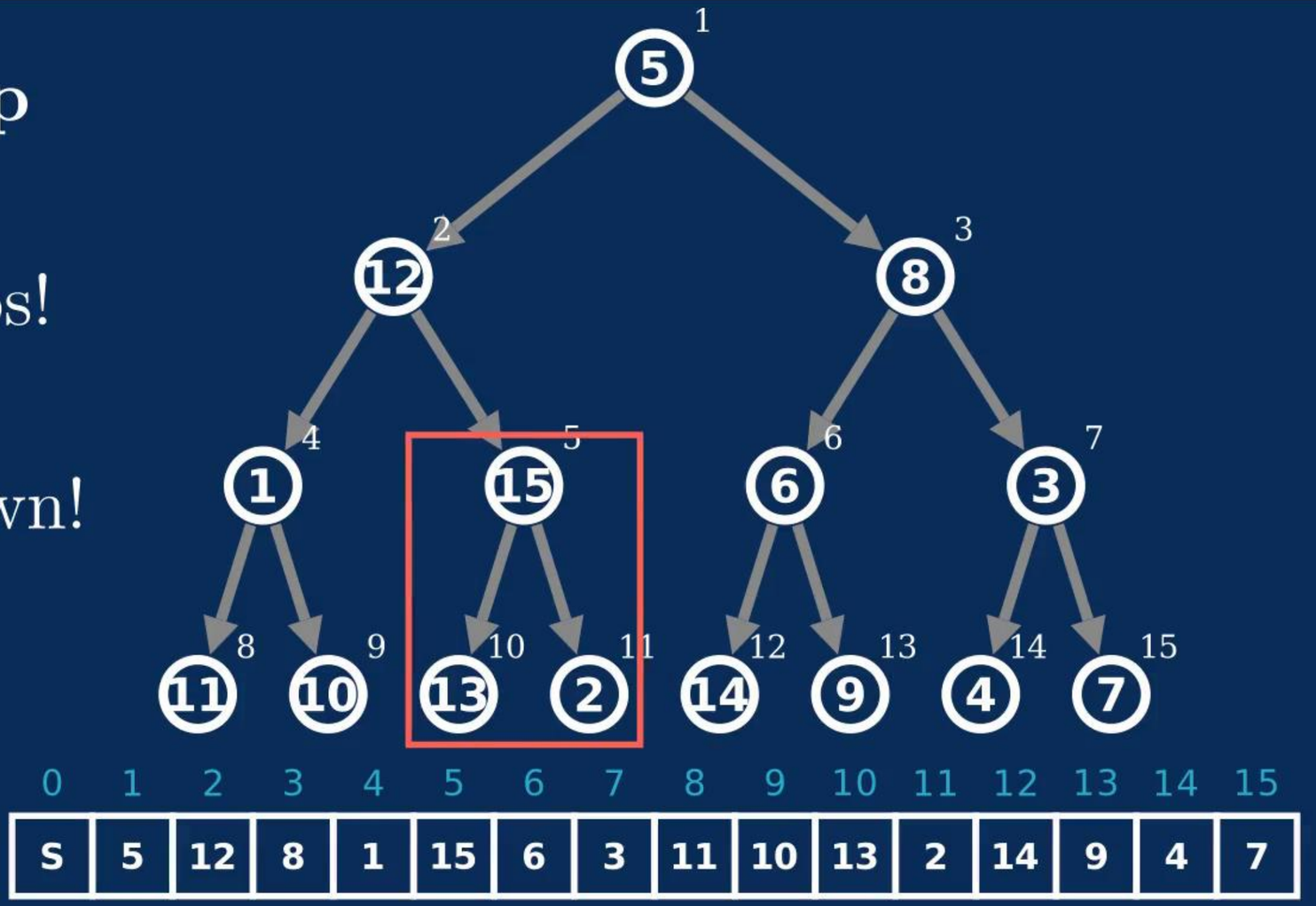
Creating a Heap

How to start?

All these are heaps!

How about this?

Just percolate down!



Heapify

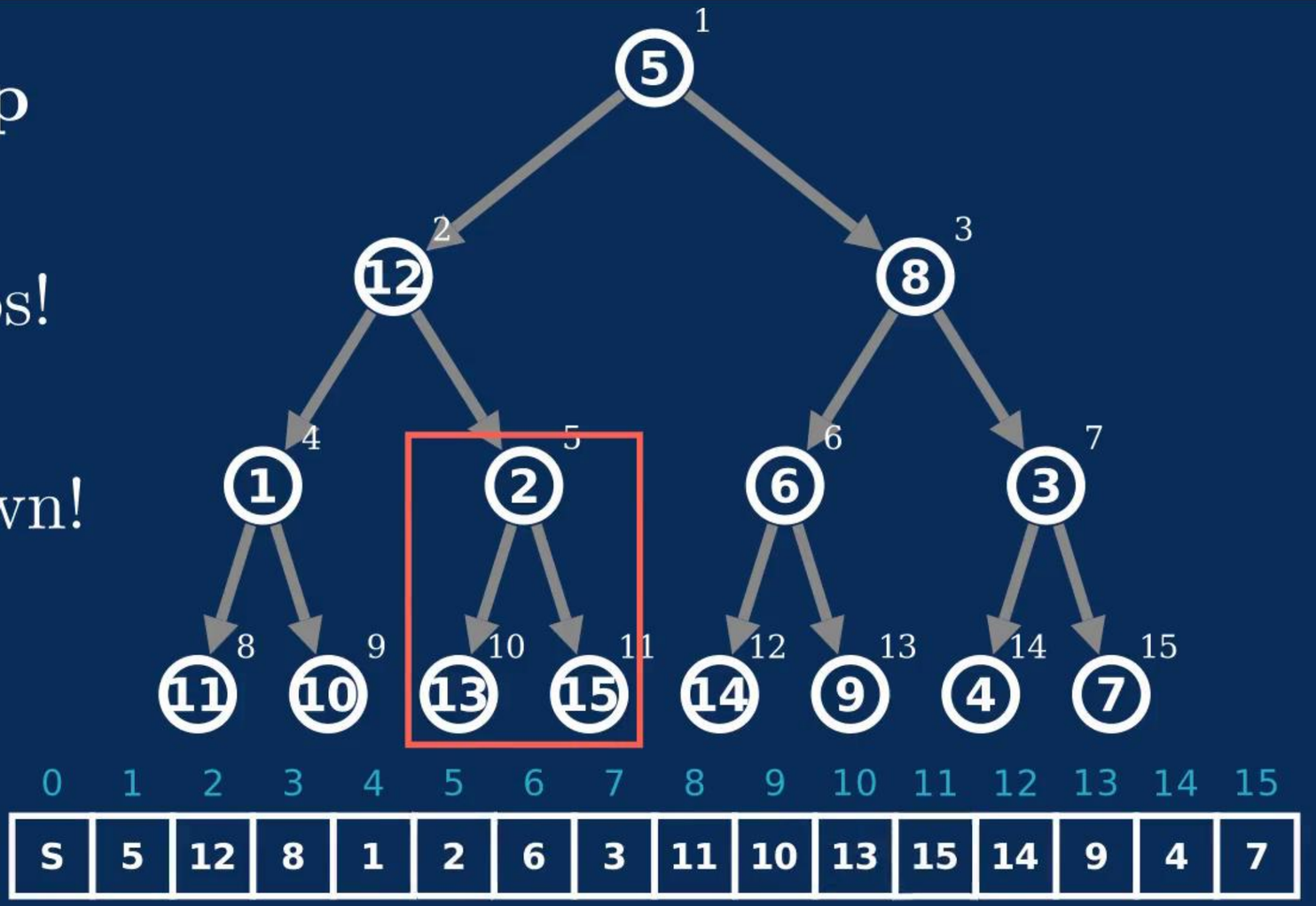
Creating a Heap

How to start?

All these are heaps!

How about this?

Just percolate down!



Heapify

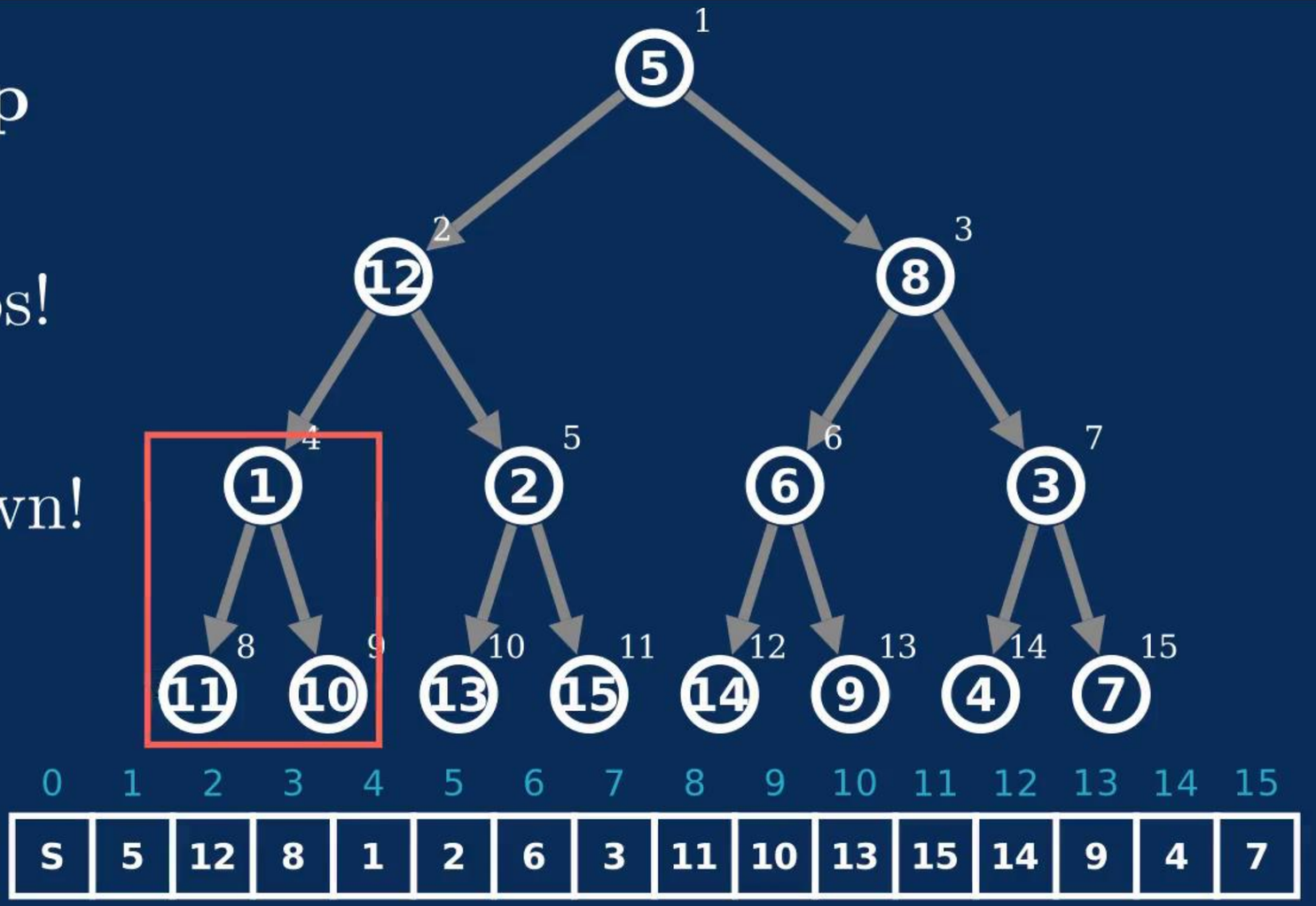
Creating a Heap

How to start?

All these are heaps!

How about this?

Just percolate down!



Heapify

Creating a Heap

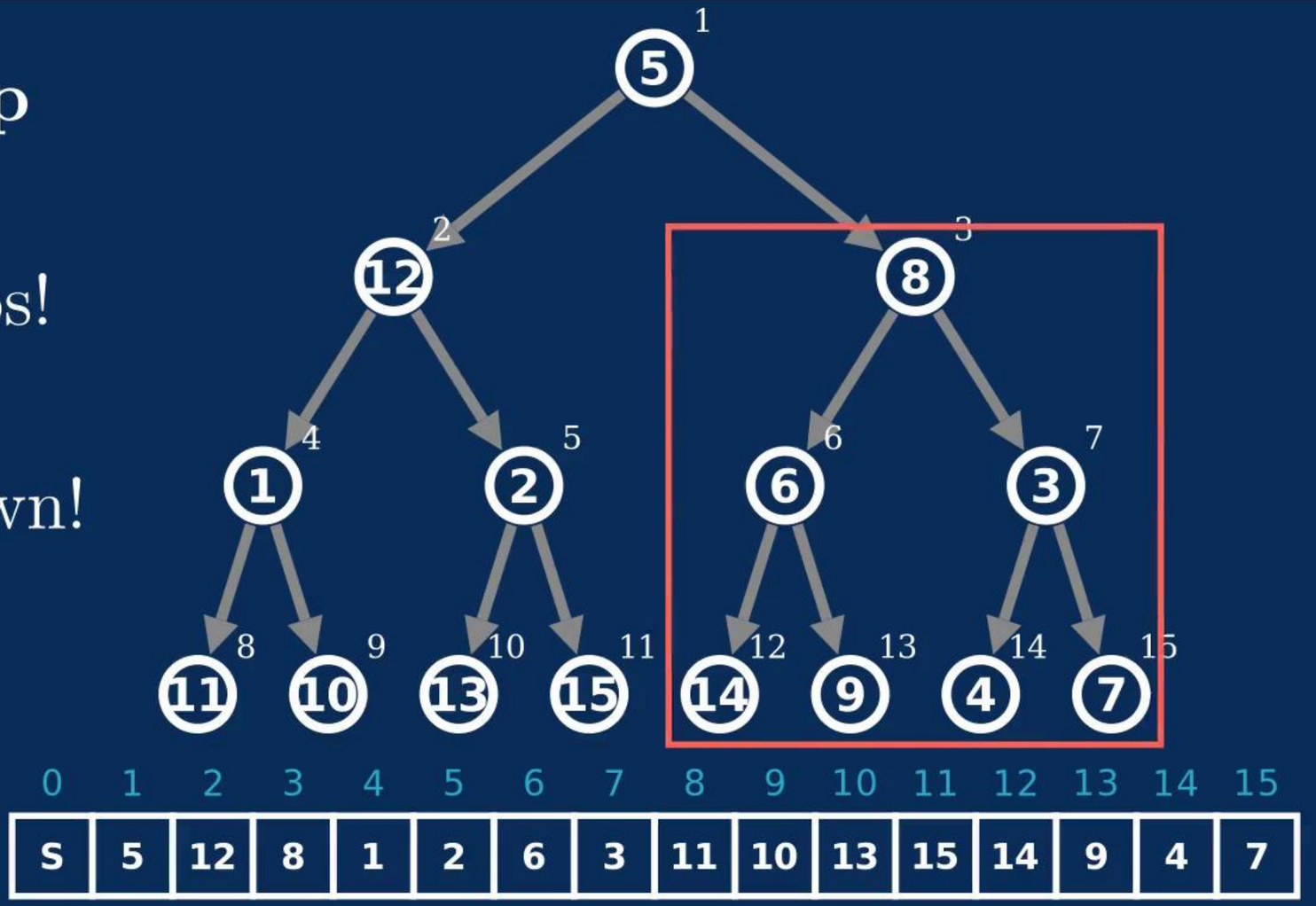
How to start?

All these are heaps!

How about this?

Just percolate down!

Next row...



Heapify

Creating a Heap

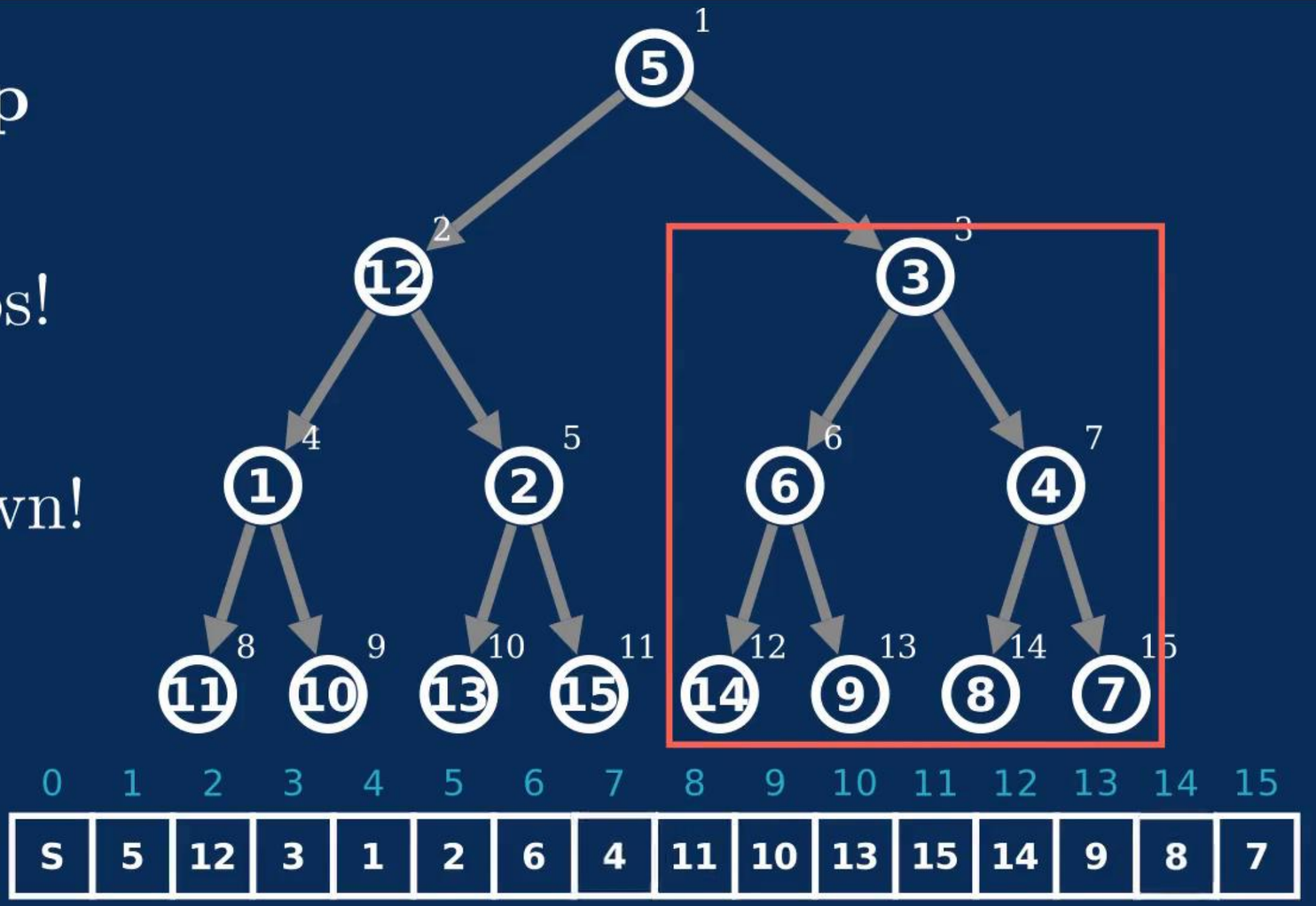
How to start?

All these are heaps!

How about this?

Just percolate down!

Next row...



Heapify

Creating a Heap

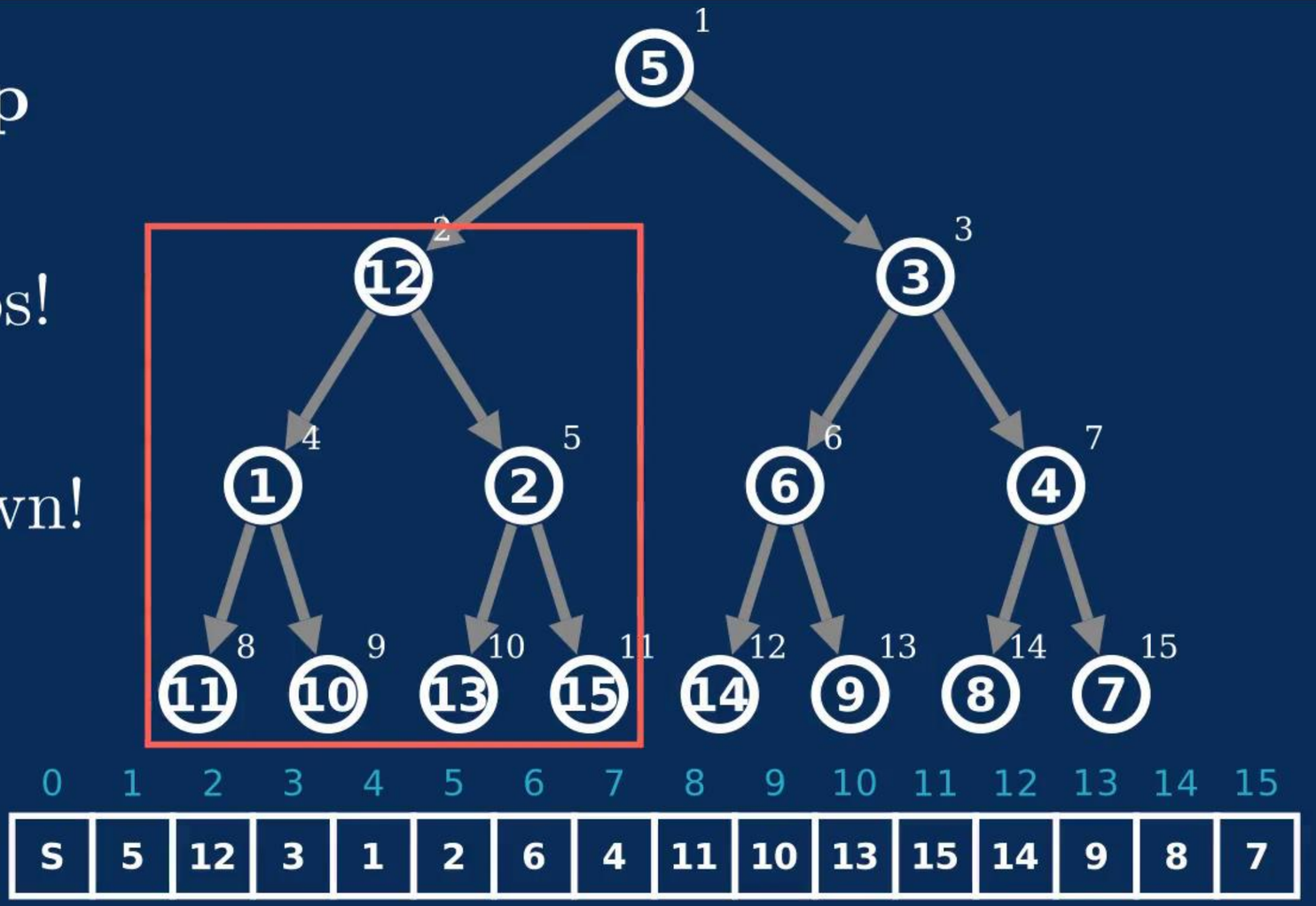
How to start?

All these are heaps!

How about this?

Just percolate down!

Next row...



Heapify

Creating a Heap

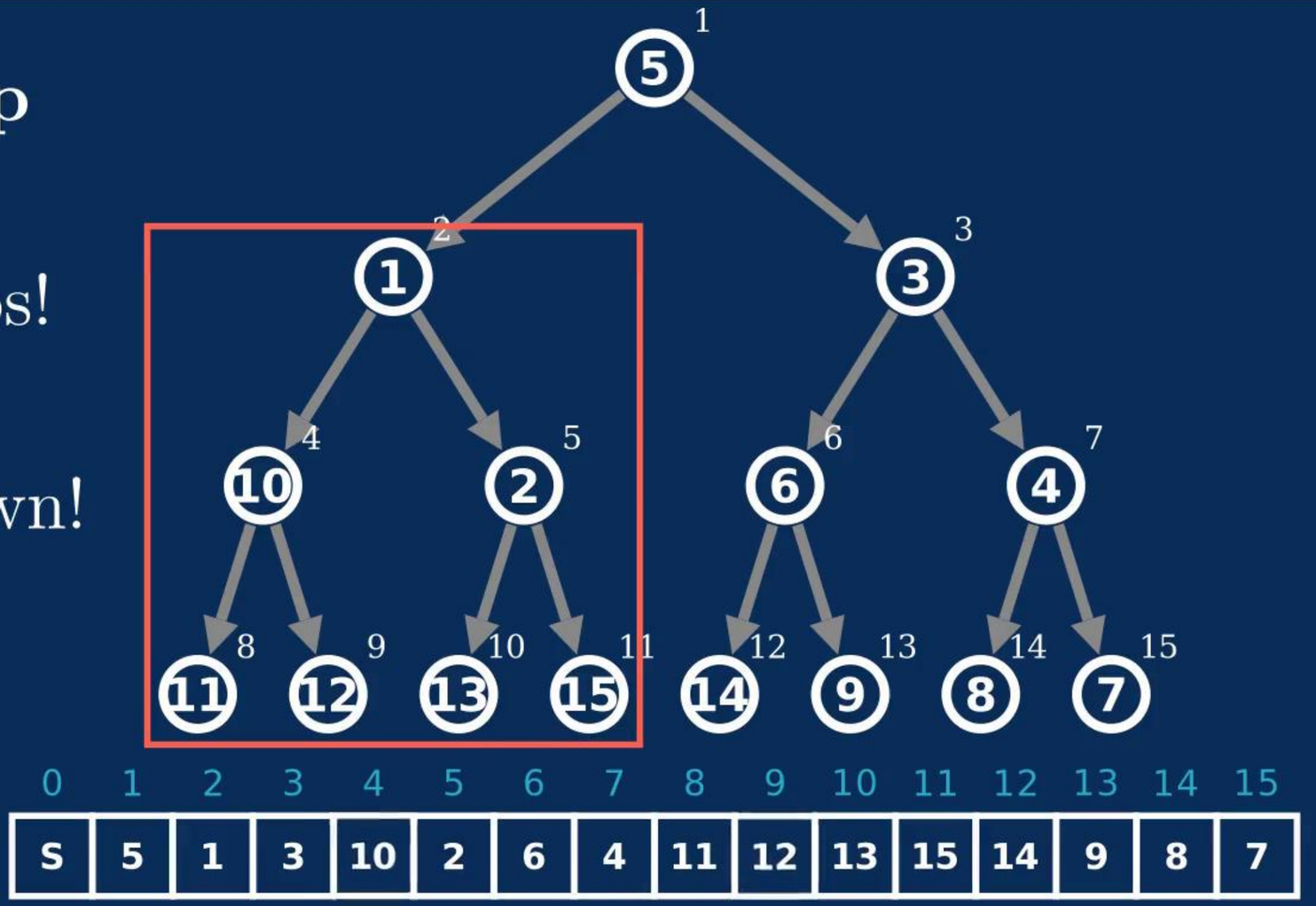
How to start?

All these are heaps!

How about this?

Just percolate down!

Next row...



Heapify

Creating a Heap

How to start?

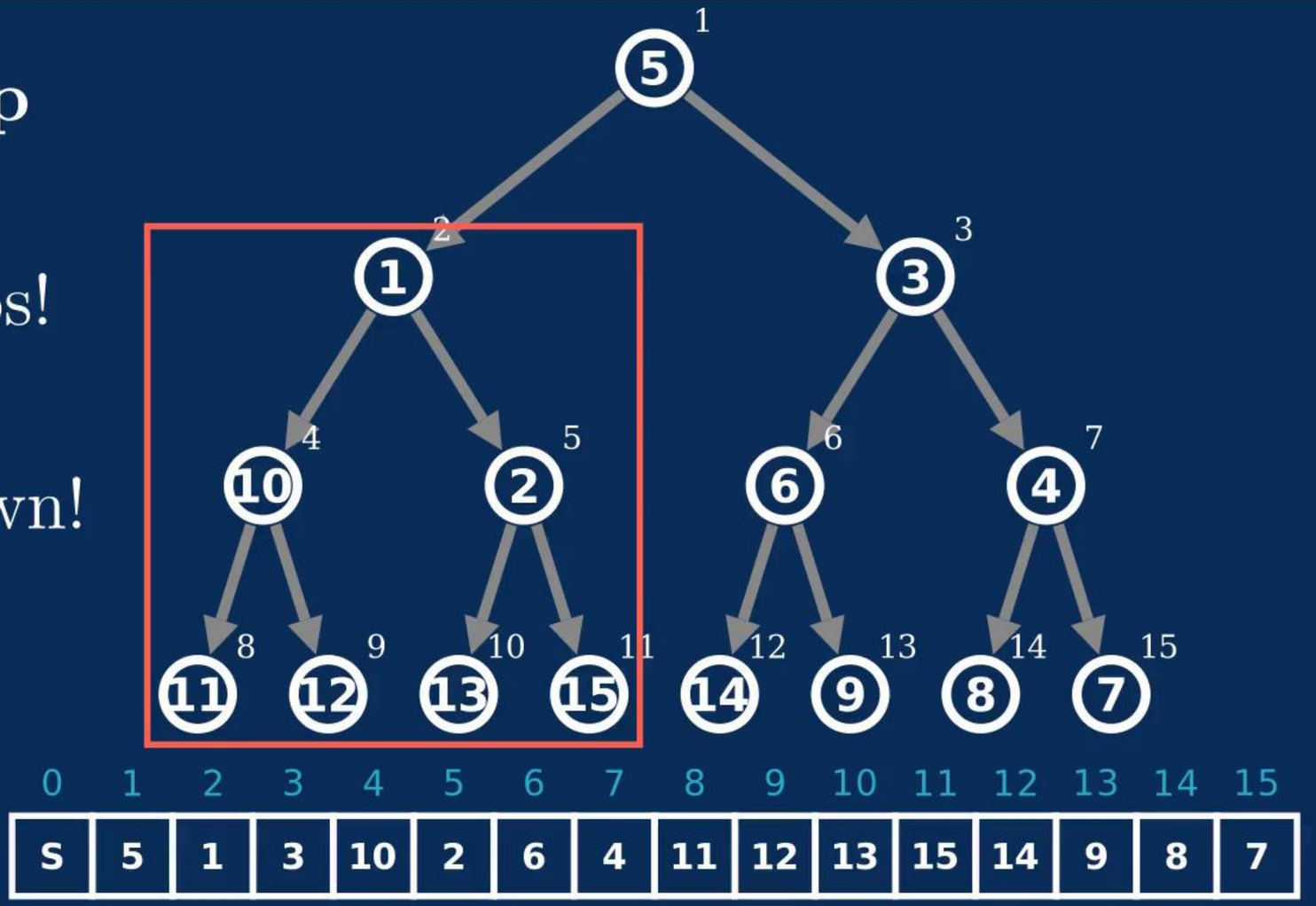
All these are heaps!

How about this?

Just percolate down!

Next row...

Root...



Heapify

Creating a Heap

How to start?

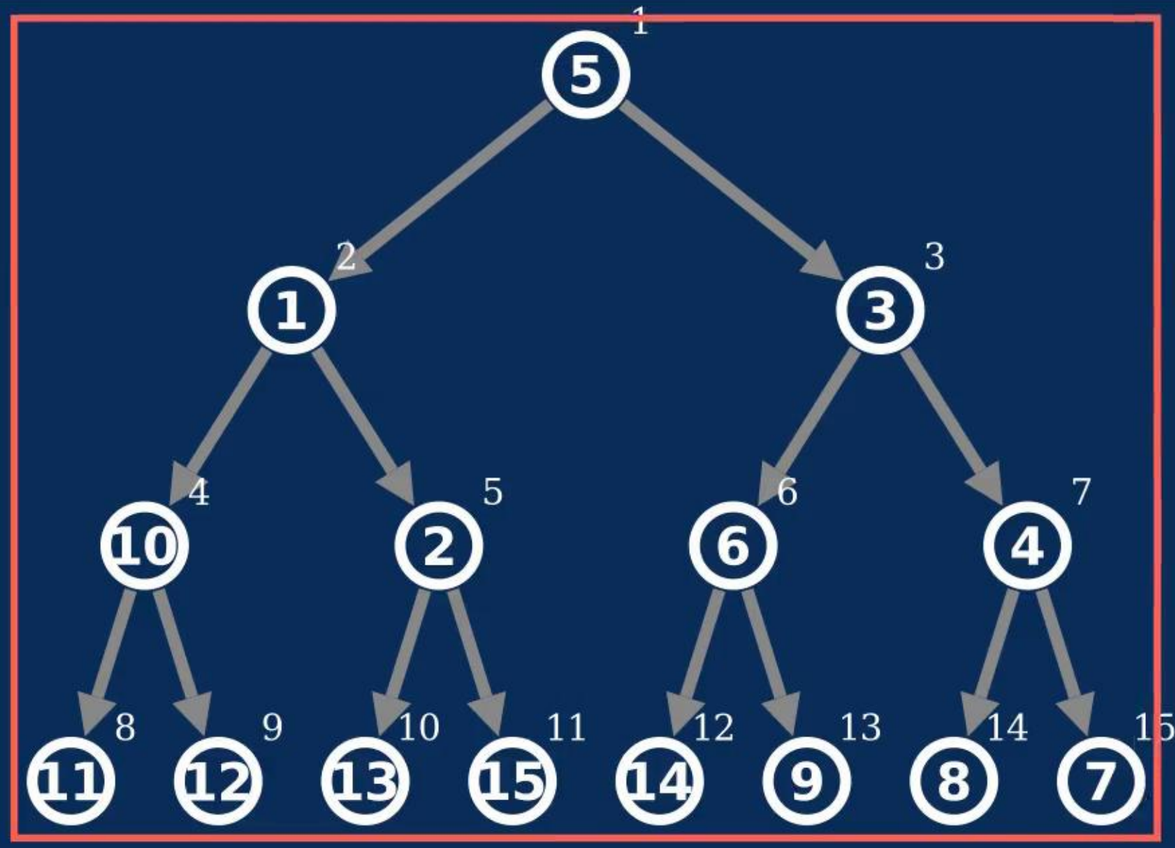
All these are heaps!

How about this?

Just percolate down!

Next row...

Root...



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	5	1	3	10	2	6	4	11	12	13	15	14	9	8	7

Heapify

Creating a Heap

How to start?

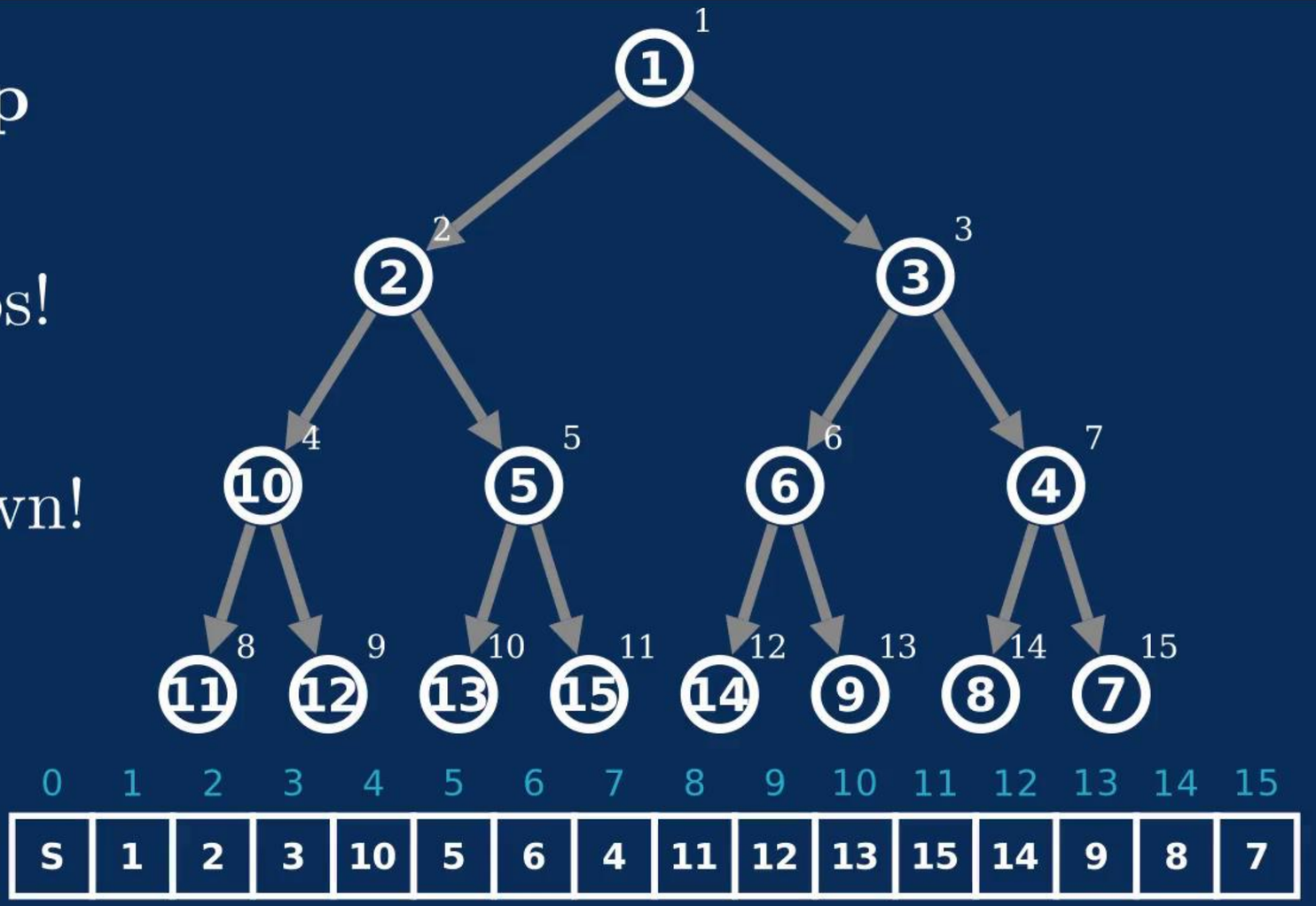
All these are heaps!

How about this?

Just percolate down!

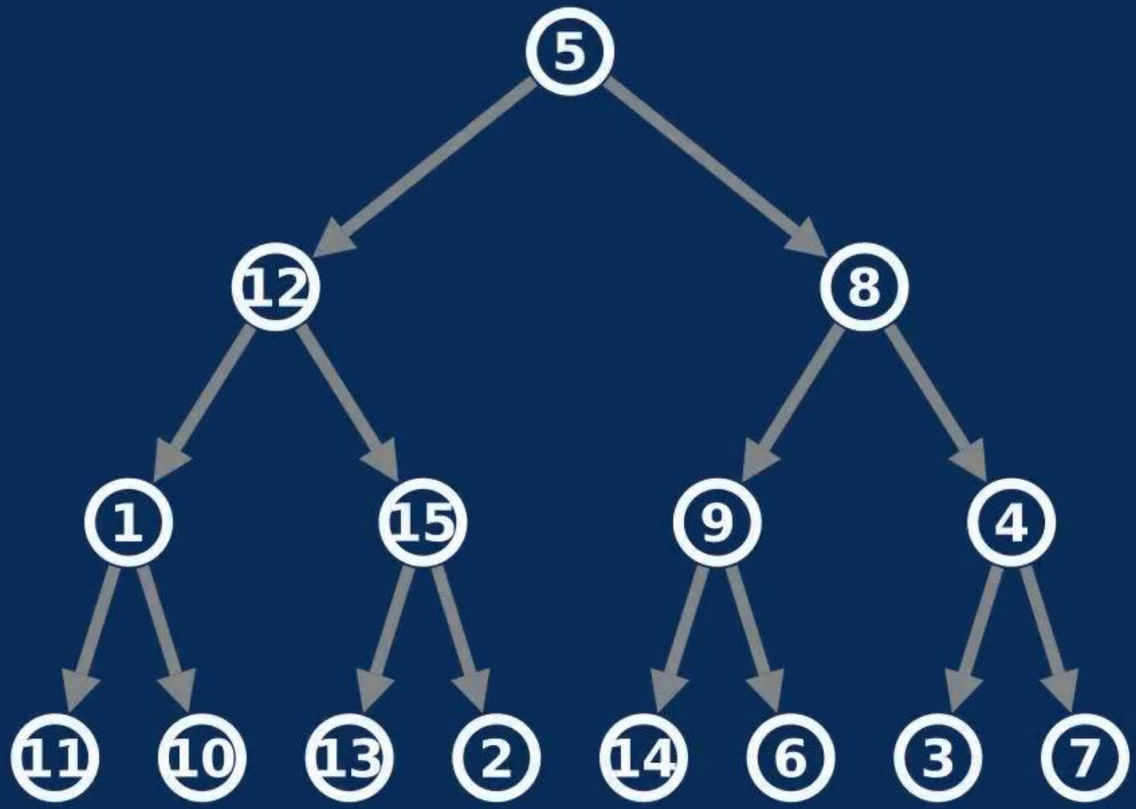
Next row...

Root...



Heapify Time

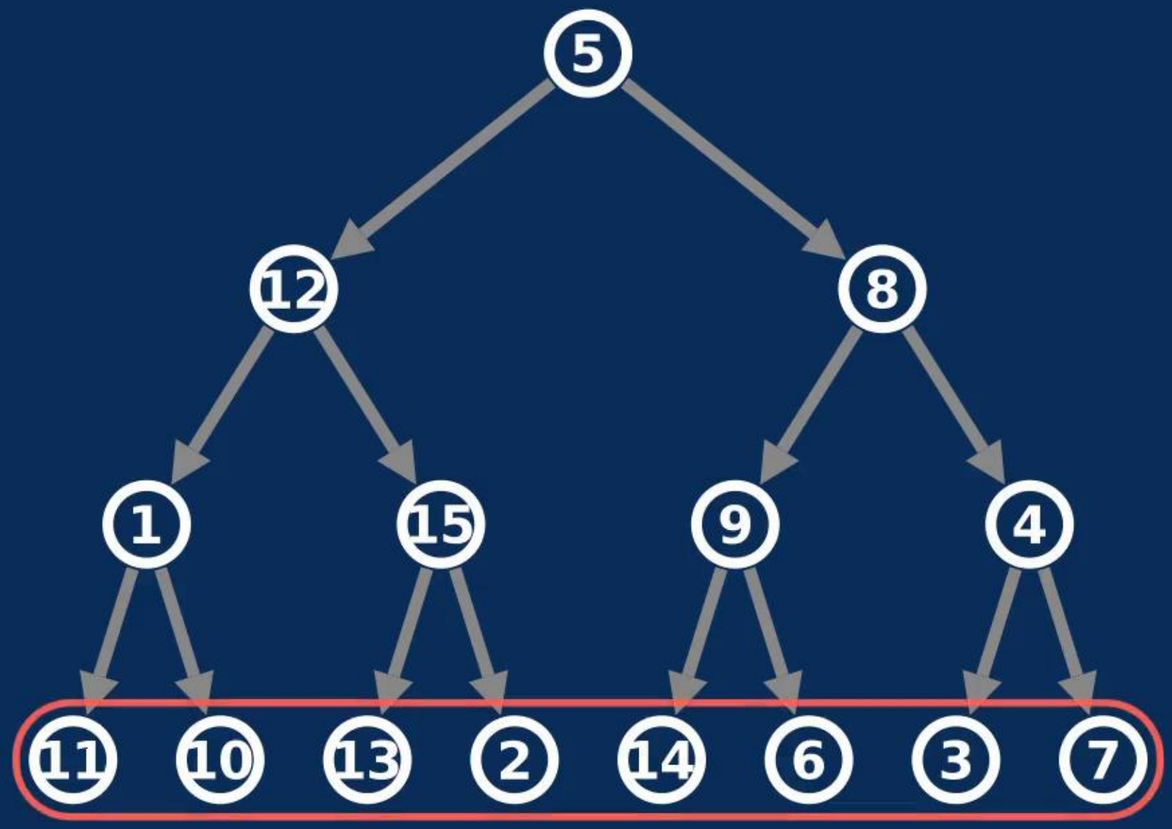
How long does this take?



Heapify Time

How long does this take?

$h = 3$ (leaves): Free!

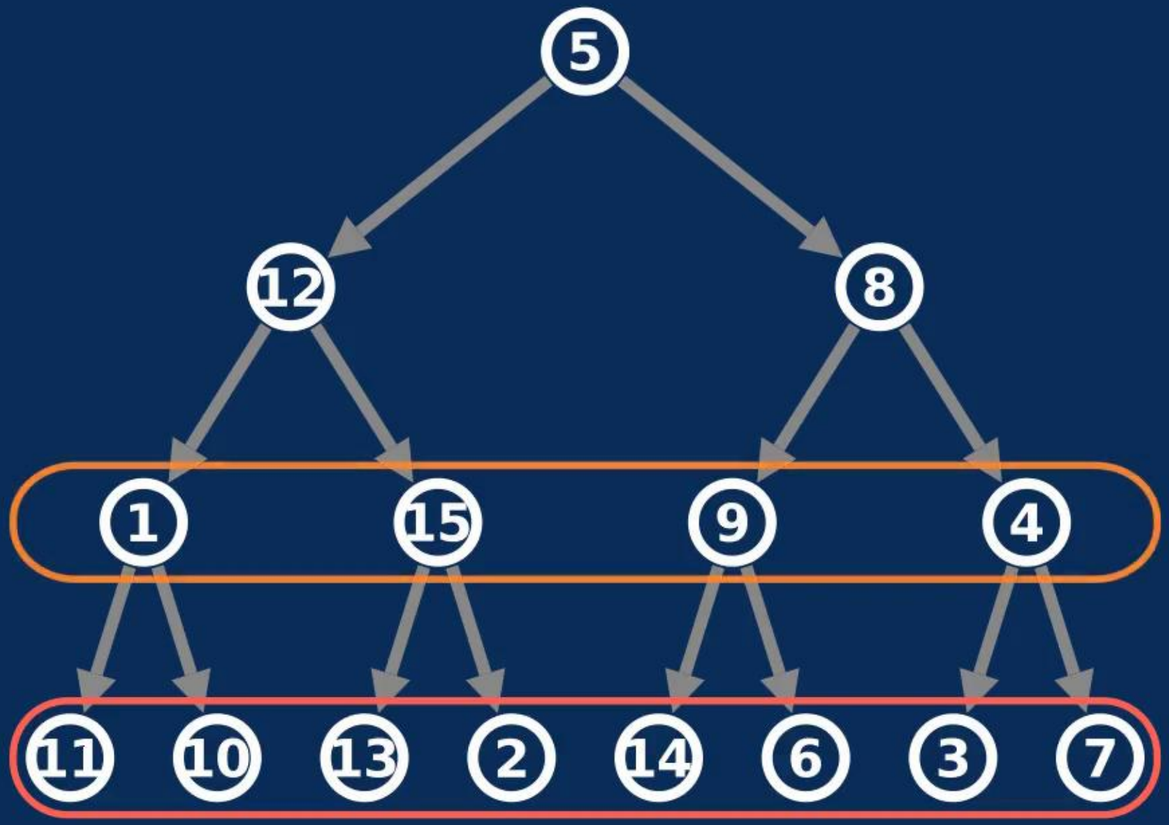


Heapify Time

How long does this take?

$h = 3$ (leaves): Free!

$h = 2$: 1 each $\times$ 4.



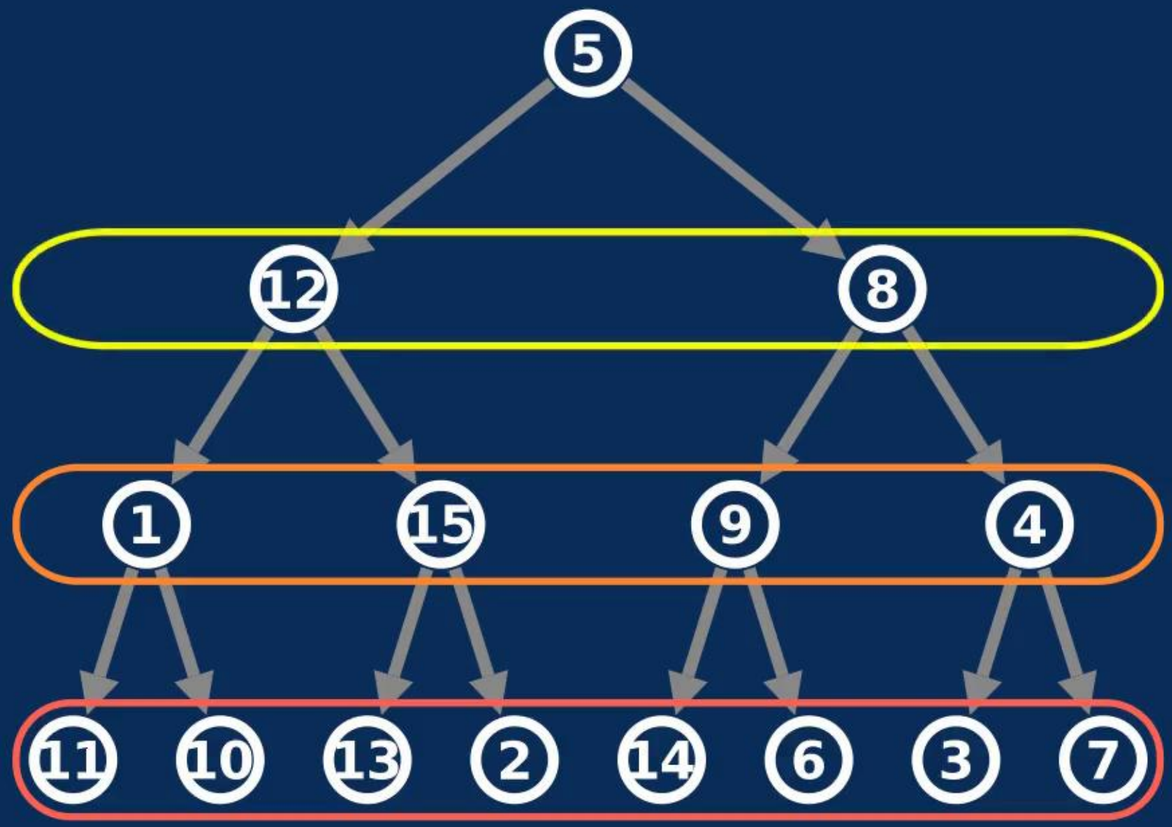
Heapify Time

How long does this take?

$h = 3$ (leaves): Free!

$h = 2$: 1 each $\times$ 4.

$h = 1$: 2 each $\times$ 2.



Heapify Time

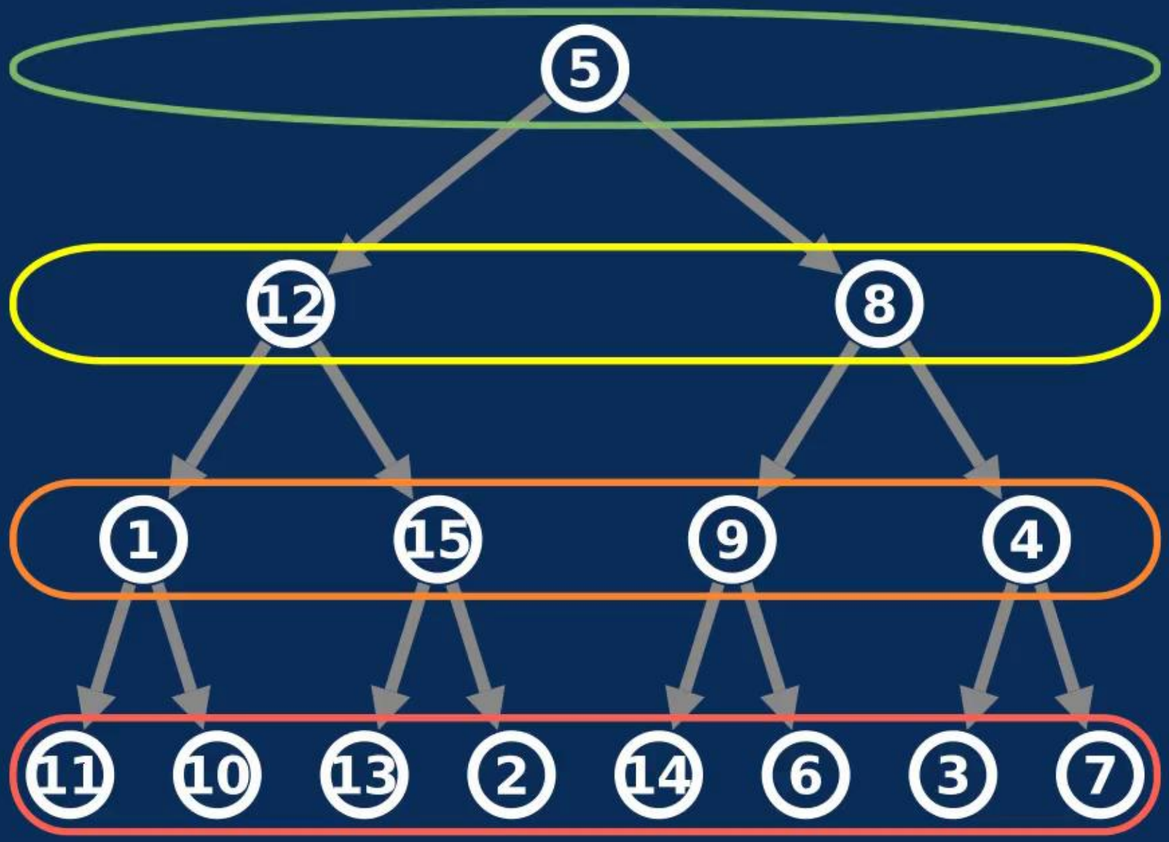
How long does this take?

$h = 3$ (leaves): Free!

$h = 2$: 1 each $\times$ 4.

$h = 1$: 2 each $\times$ 2.

$h = 0$: (root): 3.



Heapify Time

How long does this take?

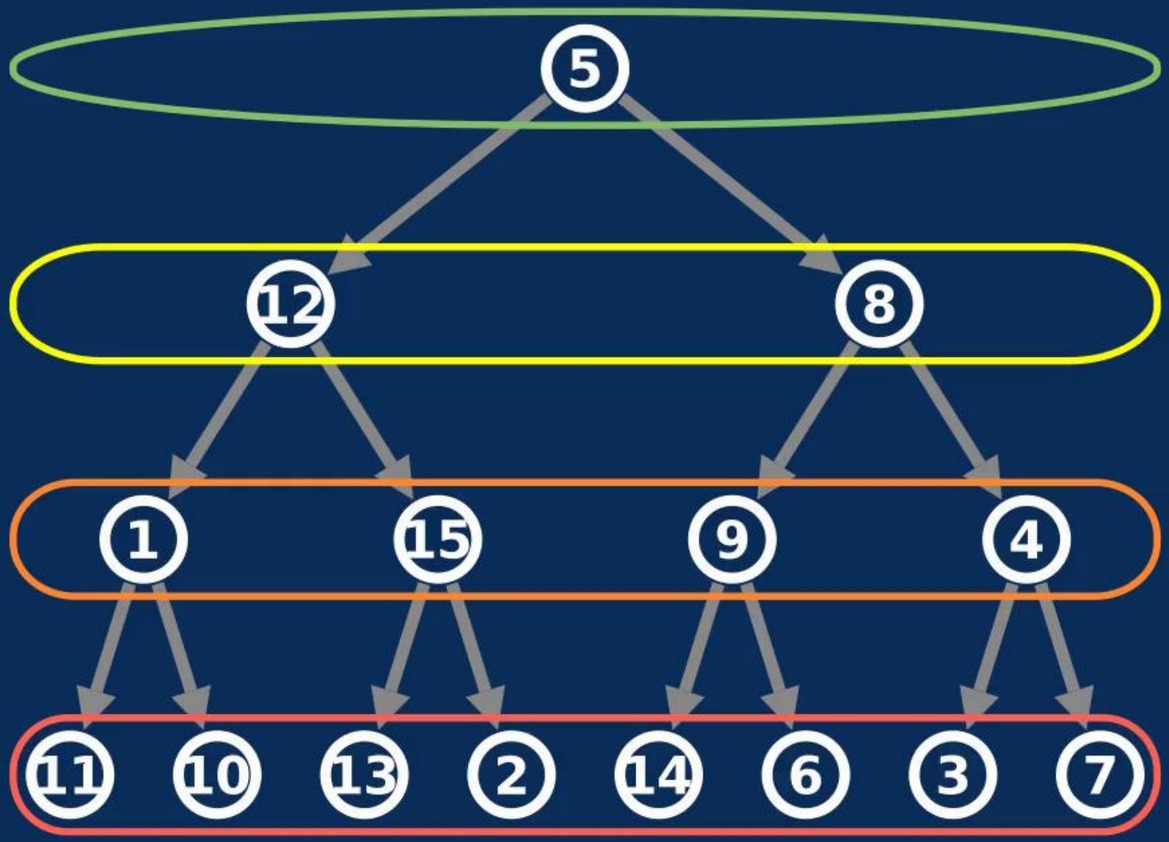
$h = 3$ (leaves): Free!

$h = 2$: 1 each $\times$ 4.

$h = 1$: 2 each $\times$ 2.

$h = 0$: (root): 3.

$$\text{Total} = \sum_{i=0}^{h} (h - i) \times n/2^{i+1}$$



Heapify Time

How long does this take?

$h = 3$ (leaves): Free!

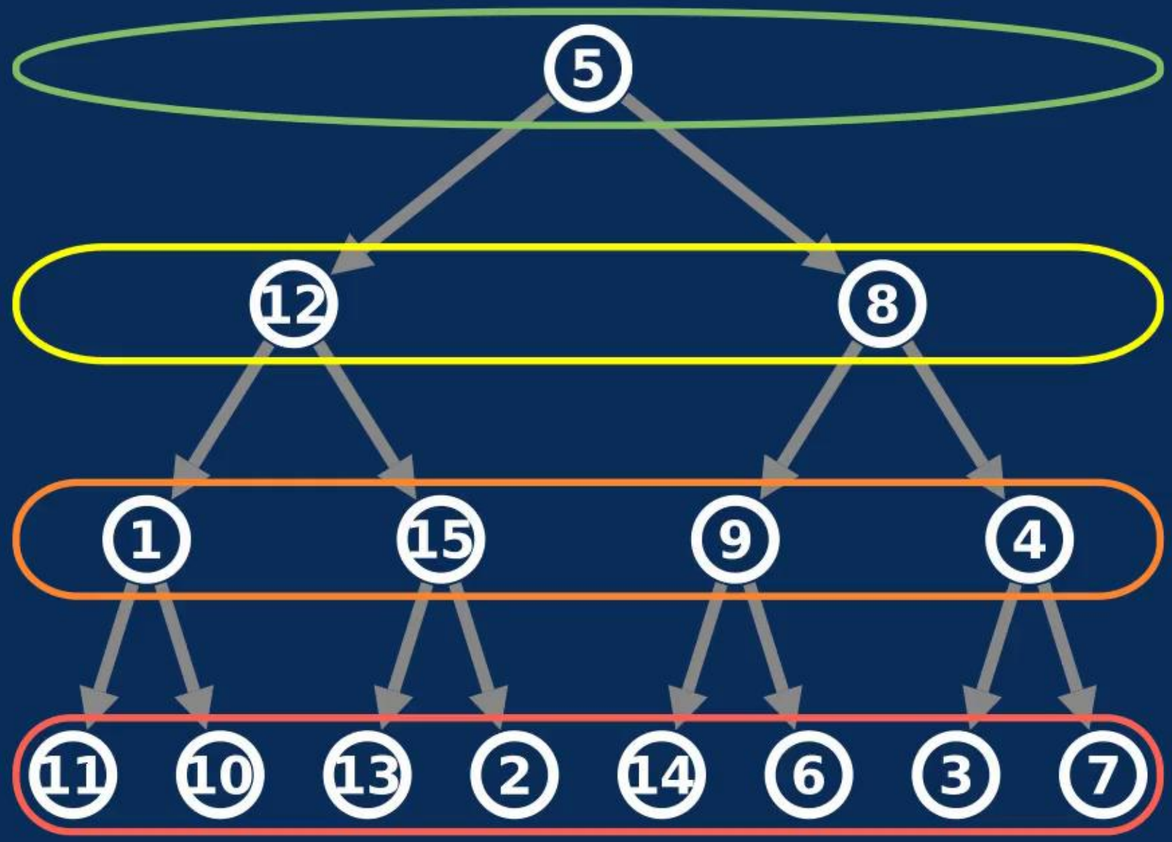
$h = 2$: 1 each $\times$ 4.

$h = 1$: 2 each $\times$ 2.

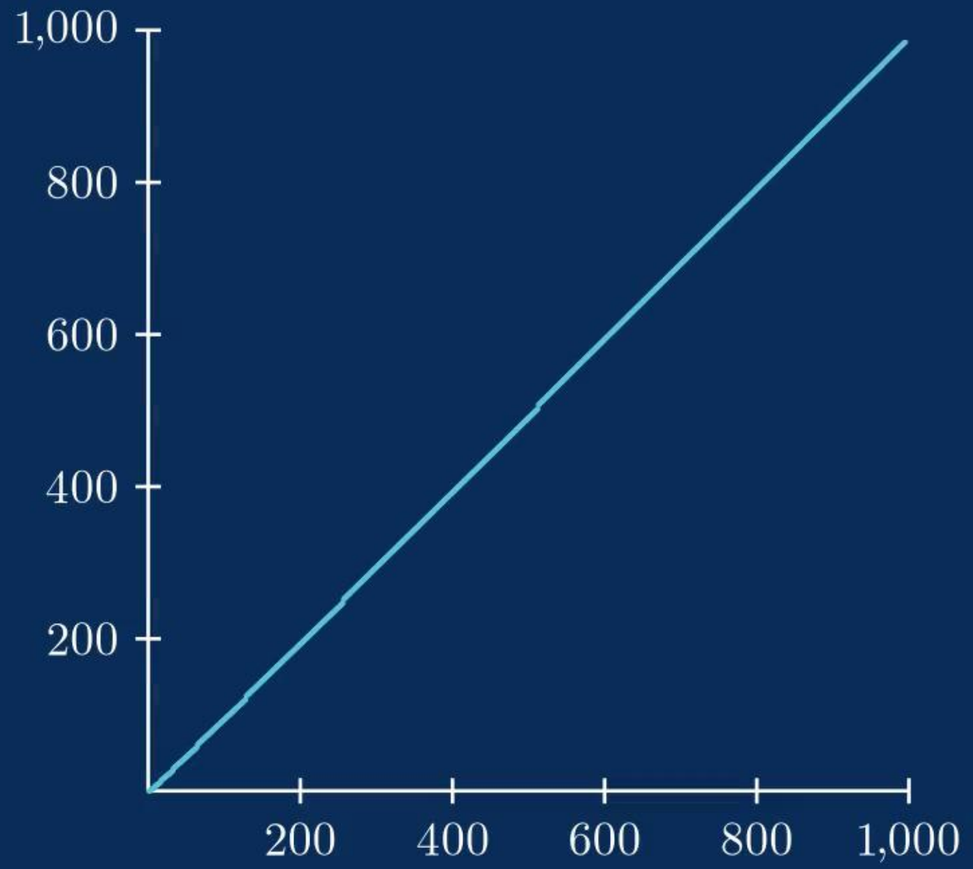
$h = 0$: (root): 3.

Total $= \sum_{i=0}^{h}(h-i) \times n/2^{i+1}$

For this: 11 swaps.



Heapify Time



Heap Sort

One final trick: We can use a heap to sort things!

Heap Sort

One final trick: We can use a heap to sort things!

How do we do it? And how long does it take?

Heap Sort

One final trick: We can use a heap to sort things!

How do we do it? And how long does it take?

- Call Heapify on an array: $\mathcal{O}(n)$

Heap Sort

One final trick: We can use a heap to sort things!

How do we do it? And how long does it take?

- Call Heapify on an array: $\mathcal{O}(n)$
- Remove n nodes: $n \times \log n$

Heap Sort

One final trick: We can use a heap to sort things!

How do we do it? And how long does it take?

- Call Heapify on an array: $\mathcal{O}(n)$
- Remove n nodes: $n \times \log n$
- Total time: $\mathcal{O}(n \log n)$

Heap Sort

One final trick: We can use a heap to sort things!

How do we do it? And how long does it take?

- Call Heapify on an array: $\mathcal{O}(n)$
- Remove n nodes: $n \times \log n$
- Total time: $\mathcal{O}(n \log n)$

Next time: Disjoint sets!

Heap Sort

One final trick: We can use a heap to sort things!

How do we do it? And how long does it take?

- Call Heapify on an array: $\mathcal{O}(n)$
- Remove n nodes: $n \times \log n$
- Total time: $\mathcal{O}(n \log n)$

Next time: Disjoint sets!