Heaps

IntroCS

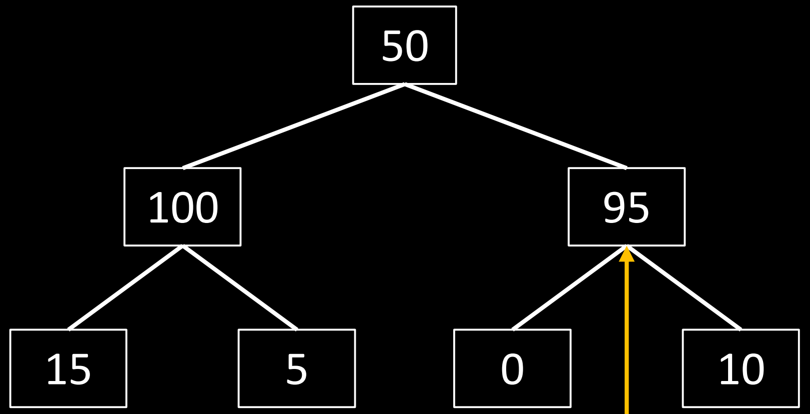
Sebastian Dunzer

Heaps are trees

$$\mathrm{node}_i = \mathrm{array}[\mathrm{i}]$$
$$\text{left child}_{\mathrm{i}} = \mathrm{array}[2\mathrm{i} + 1]$$
$$\text{right child}_{\mathrm{i}} = \mathrm{array}[2\mathrm{i} + 2]$$

One way to program trees is to flatten it into an array.
To draw a tree from an array we use the formula in the top right
Let's drill it down to the example below.
array[0] is the root. So, we draw 50 on top of the tree.

0	**1**	**2**	**3**	**4**	**5**	**6**
50	100	95	15	5	0	10



The indices of the root's children are:
$i_{left} = 2 * 0 + 1 = 1; i_{left} = 2 * 0 + 2 = 2$
So, array[1] = 100 left child and array[2] = 95 right child of root.
Let's calculate the indices for node array[1]:
$i_{left} = 2 * 1 + 1 = 3; i_{left} = 2 * 1 + 2 = 4$
So, array[3] = 15 left child and array[4] = 5 right child of left child of root

I guess you can now figure out the children of the right child of the root on your own

Let’s heapify (reheap)

$$\text{node}_i = \text{array}[i]$$
$$\text{left child}_i = \text{array}[2i+1]$$
$$\text{right child}_i = \text{array}[2i+2]$$

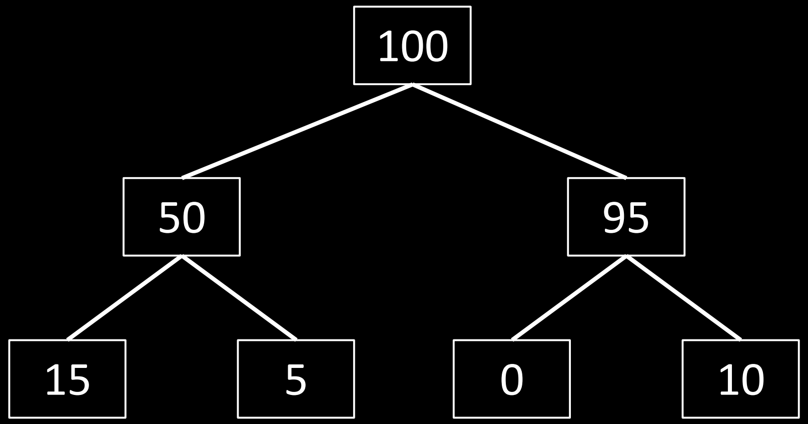
First, look at the condition that makes a max-heap on the right:
The formula says, in other words, a parent is always larger than or equal to its children.
So, to create a heap we must swap all the values in such a way that the tree is a heap.

Max-heap condition:
$\text{node}_i \geq \text{left child}_i$, and $\text{node}_i \geq \text{right child}_i$

Let’s enforce the condition from left to right.
We only need to swap once in this array to create a heap.

i	0	1	2	3	4	5	6
1.	50	100	95	15	5	0	10
2.	**100**	**50**	**95**	**15**	**5**	**0**	**10**

=



Let’s heapify (reheap)

Analog to the Max-heap there are Min-heaps.

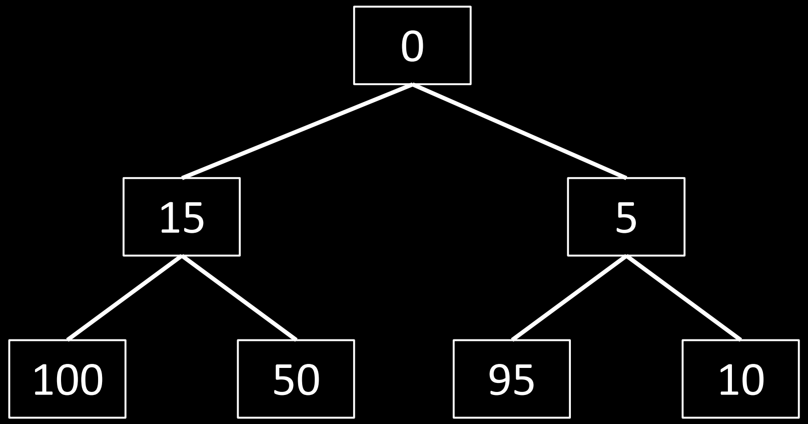
$$\text{node}_i = \text{array}[i]$$
$$\text{left child}_i = \text{array}[2i+1]$$
$$\text{right child}_i = \text{array}[2i+2]$$

Min-heap condition:

$\text{node}_i \le \text{left child}_i$, and $\text{node}_i \le \text{right child}_i$

i	0	1	2	3	4	5	6
1.	50	100	95	15	5	0	10
2.	50	15	95	100	5	0	10
3.	15	50	95	100	5	0	10
4.	15	5	95	100	50	0	10
5.	5	15	95	100	50	0	10
6.	5	15	0	100	50	95	10
7.	**0**	**15**	**5**	**100**	**50**	**95**	**10**

=



Let’s sort

Min-heap condition:

$\text{node}_i \leq \text{left child}_i$, and $\text{node}_i \leq \text{right child}_i$

We‘ll use the latter example (Min-heap) and sort the array using the heapsort-algorithm.

$\text{node}_i = \text{array}[i]$

$\text{left child}_i = \text{array}[2i+1]$

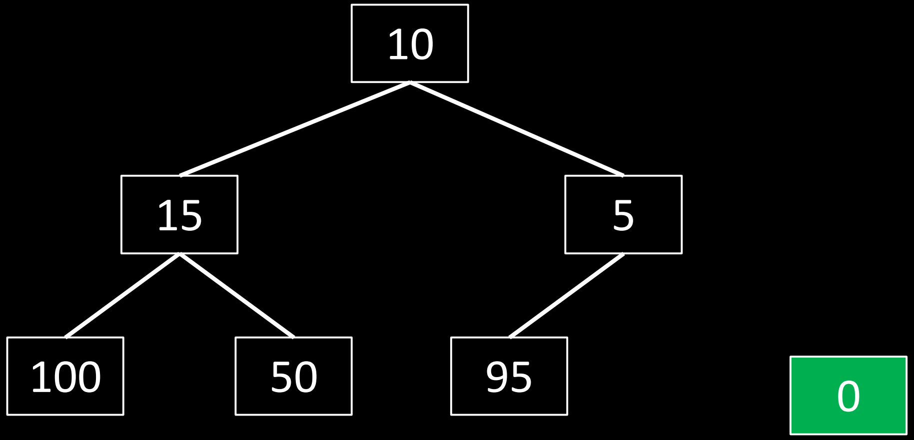
$\text{right child}_i = \text{array}[2i+2]$

Pseudocode:
1. Create Heap
2. Swap Root with last element in array
3. Consider last element as sorted
4. If whole array sorted
4.1 finished
5. Else
5.1 Go to 1

We already have created the heap before. So we just need to swap the first with the last element in the array and performed the first iteration.
Now we need to heapify (reheap) the array again.

i	0	1	2	3	4	5	6
heap	0	15	5	100	50	95	10
swap	10	15	5	100	50	95	0

=



Pseudocode:
1. Create Heap
2. Swap Root with last element in array
3. Consider last element as sorted
4. If whole array sorted
4.1 finished
5. Else
5.1 Go to 1

Min-heap condition:

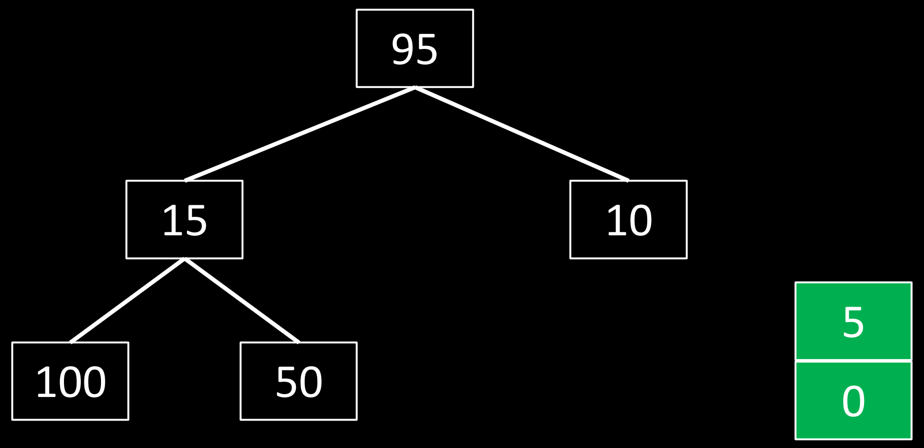
$\text{node}_i \leq \text{left child}_i$, and $\text{node}_i \leq \text{right child}_i$

$$\text{node}_i = \text{array}[i]$$
$$\text{left child}_i = \text{array}[2i+1]$$
$$\text{right child}_i = \text{array}[2i+2]$$

Iteration for Iteration the unsorted part of the array shrinks.

i	0	1	2	3	4	5	6
array	10	15	5	100	50	95	0
heap	5	15	10	100	50	95	0
swap	95	15	10	100	50	5	0

=



Pseudocode:
1. Create Heap
2. Swap Root with last element in array
3. Consider last element as sorted
4. If whole array sorted
4.1 finished
5. Else
5.1 Go to 1

Min-heap condition:

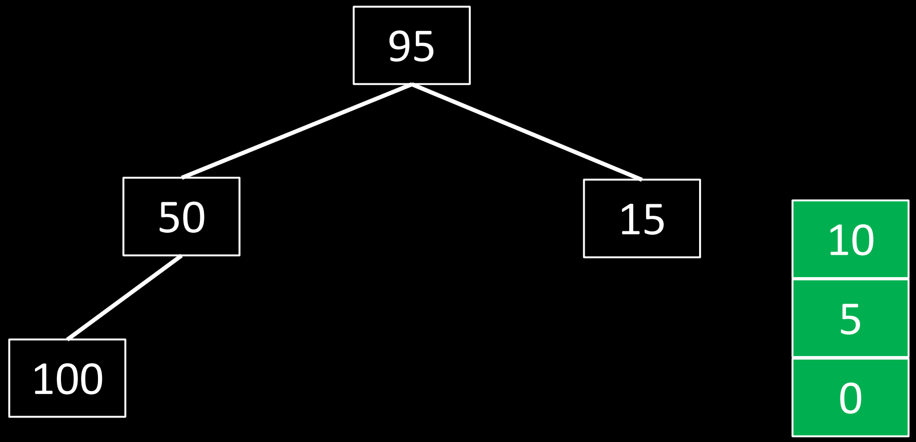
$\text{node}_i \le \text{left child}_i$, and $\text{node}_i \le \text{right child}_i$

$$\text{node}_i = \text{array}[i]$$
$$\text{left child}_i = \text{array}[2i + 1]$$
$$\text{right child}_i = \text{array}[2i + 2]$$

Iteration for Iteration the unsorted part of the array shrinks.

i	0	1	2	3	4	5	6
array	95	15	10	100	50	5	0
heap	10	50	15	100	95	5	0
swap	95	50	15	100	10	5	0

=



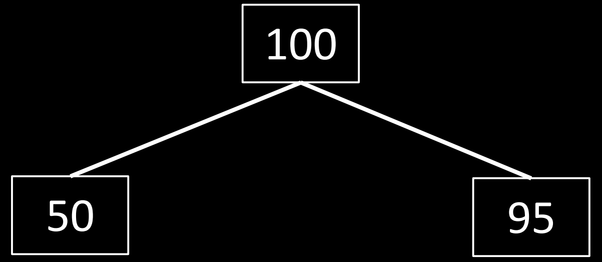
Pseudocode:
1. Create Heap
2. Swap Root with last element in array
3. Consider last element as sorted
4. If whole array sorted
4.1 finished
5. Else
5.1 Go to 1

Min-heap condition:
$\text{node}_i \leq \text{left child}_i$, and $\text{node}_i \leq \text{right child}_i$

$$\text{node}_i = \text{array}[i]$$
$$\text{left child}_i = \text{array}[2i+1]$$
$$\text{right child}_i = \text{array}[2i+2]$$

Iteration for Iteration the unsorted part of the array shrinks.

i	0	1	2	3	4	5	6
array	95	50	15	100	10	5	0
heap	15	95	50	100	10	5	0
swap	100	95	50	15	10	5	0



15
10
5
0

Pseudocode:
1. Create Heap
2. Swap Root with last element in array
3. Consider last element as sorted
4. If whole array sorted
4.1 finished
5. Else
5.1 Go to 1

Min-heap condition:

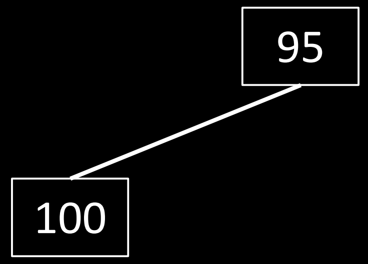
$\text{node}_i \leq \text{left child}_i$, and $\text{node}_i \leq \text{right child}_i$

$$\text{node}_i = \text{array}[i]$$
$$\text{left child}_i = \text{array}[2i + 1]$$
$$\text{right child}_i = \text{array}[2i + 2]$$

Iteration for Iteration the unsorted part of the array shrinks.

i	0	1	2	3	4	5	6
array	100	95	50	15	10	5	0
heap	50	100	95	15	10	5	0
swap	95	100	50	15	10	5	0

=



50
15
10
5
0

Pseudocode:
1. Create Heap
2. Swap Root with last element in array
3. Consider last element as sorted
4. If whole array sorted
4.1 finished
5. Else
5.1 Go to 1

Min-heap condition:

$\text{node}_i \leq \text{left child}_i$, and $\text{node}_i \leq \text{right child}_i$

$$\text{node}_i = \text{array}[i]$$
$$\text{left child}_i = \text{array}[2i+1]$$
$$\text{right child}_i = \text{array}[2i+2]$$

Iteration for Iteration the unsorted part of the array shrinks.

i	0	1	2	3	4	5	6
array	**95**	**100**	**50**	**15**	**10**	**5**	**0**
heap	**95**	**100**	**50**	**15**	**10**	**5**	**0**
swap	**100**	**95**	**50**	**15**	**10**	**5**	**0**

=

100

95
50
15
10
5
0

Pseudocode:
1. Create Heap
2. Swap Root with last element in array
3. Consider last element as sorted
4. If whole array sorted
4.1 finished
5. Else
5.1 Go to 1

Min-heap condition:
$node_i \leq$ left $child_i$, and $node_i \leq$ right $child_i$

$$node_i = array[i]$$
$$left\ child_i = array[2i + 1]$$
$$right\ child_i = array[2i + 2]$$

Yay! Finished, we sorted an array descending using min-heaps.

i	0	1	2	3	4	5	6
array	100	95	50	15	10	5	0
heap	100	95	50	15	10	5	0
swap	100	95	50	15	10	5	0

=

So much emptiness.
Nothing left to sort.

100
95
50
15
10
5
0

I hope this helps you with heap sort!

If you need further help, ask in the Tutorium or in the Forum.

Heaps are Trees

Sebastian Dunzer

IntroCS