PRIORITY QUEUE ADT


insert
delete
minimum


priority queue, Q
delete
min
insert

LINKED LIST

IMPLEMENTATION

O(1)

O(n)

(SORTED)

LINKED LIST

IMPLEMENTATION

O(n)

O(1)

HEAP

IMPLEMENTATION

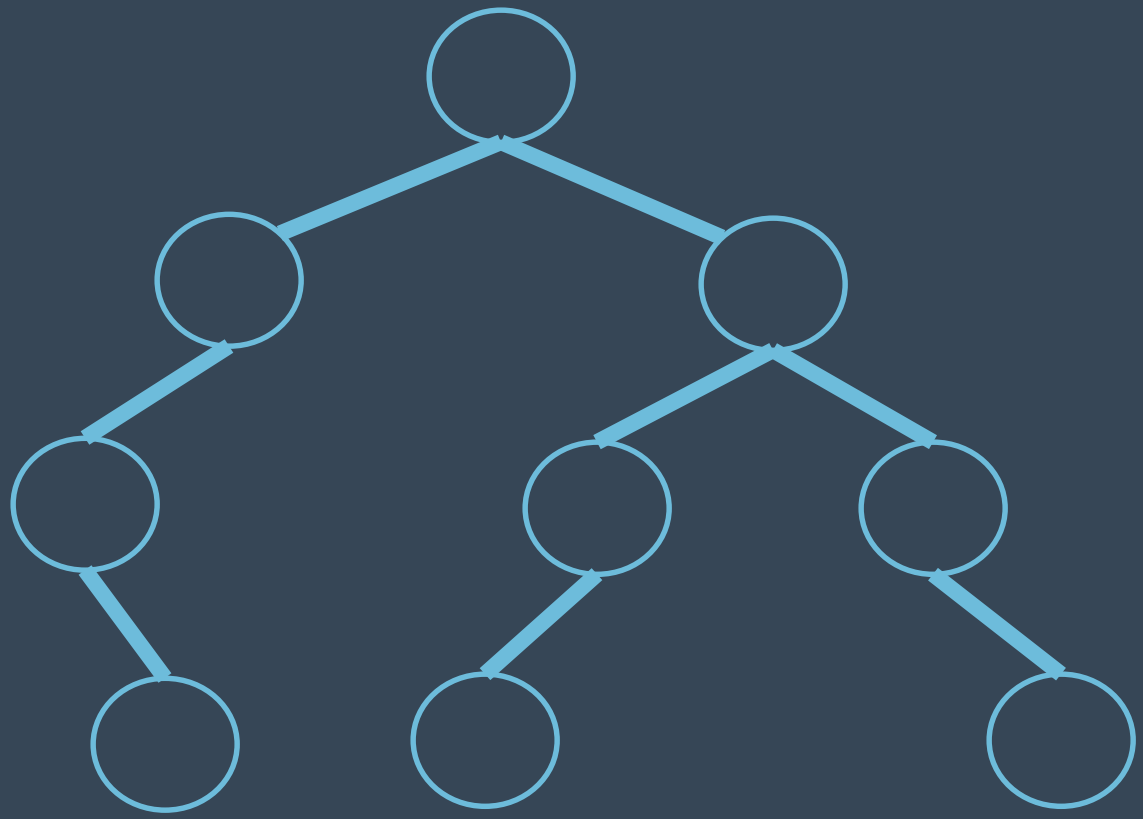
O(log n)

O(log n)

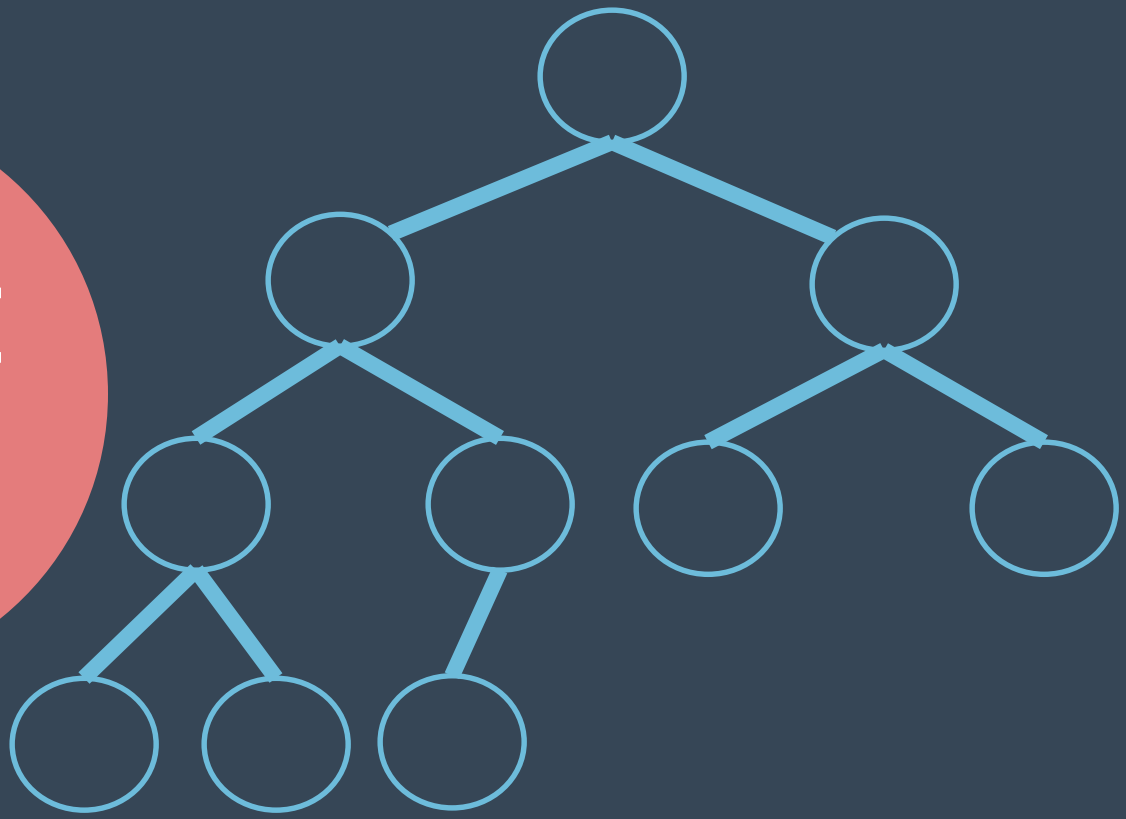
STRUCTURE PROPERTY

A binary tree that is completely filled, with the possible exception of the bottom level, which is filled from left to right.

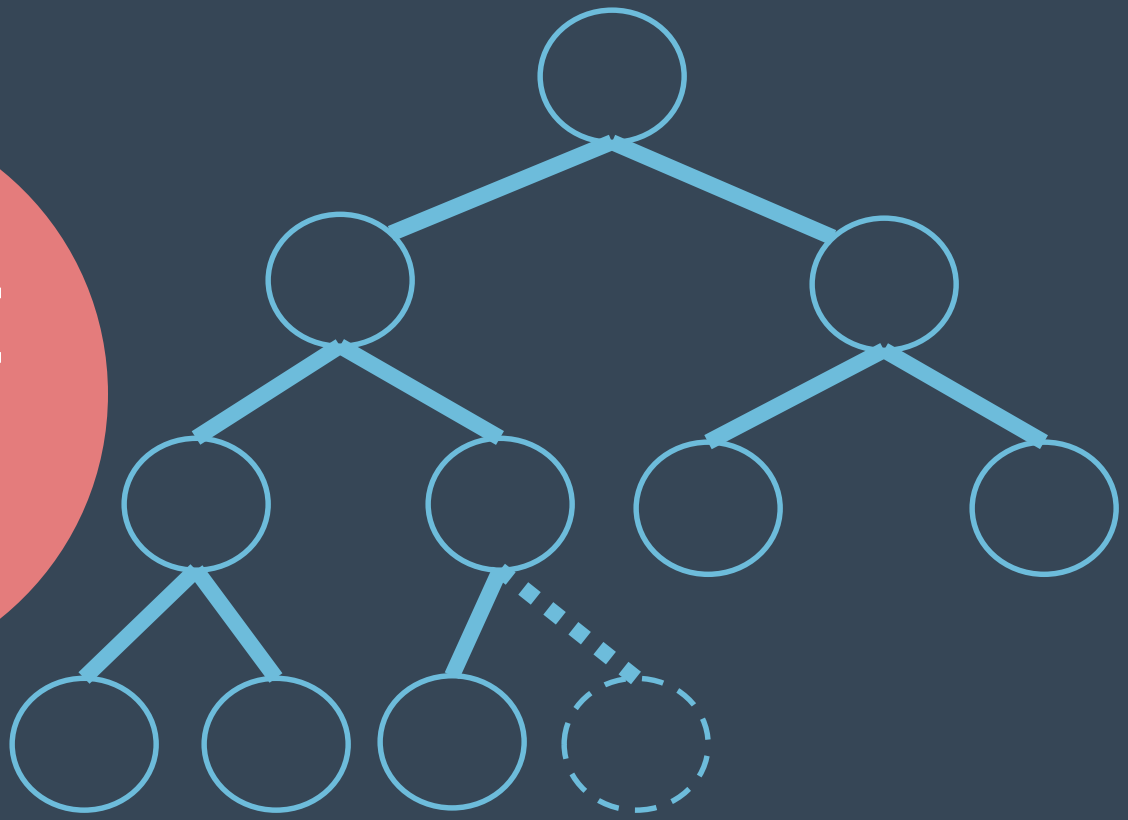
STRUCTURE PROPERTY



STRUCTURE PROPERTY



STRUCTURE PROPERTY

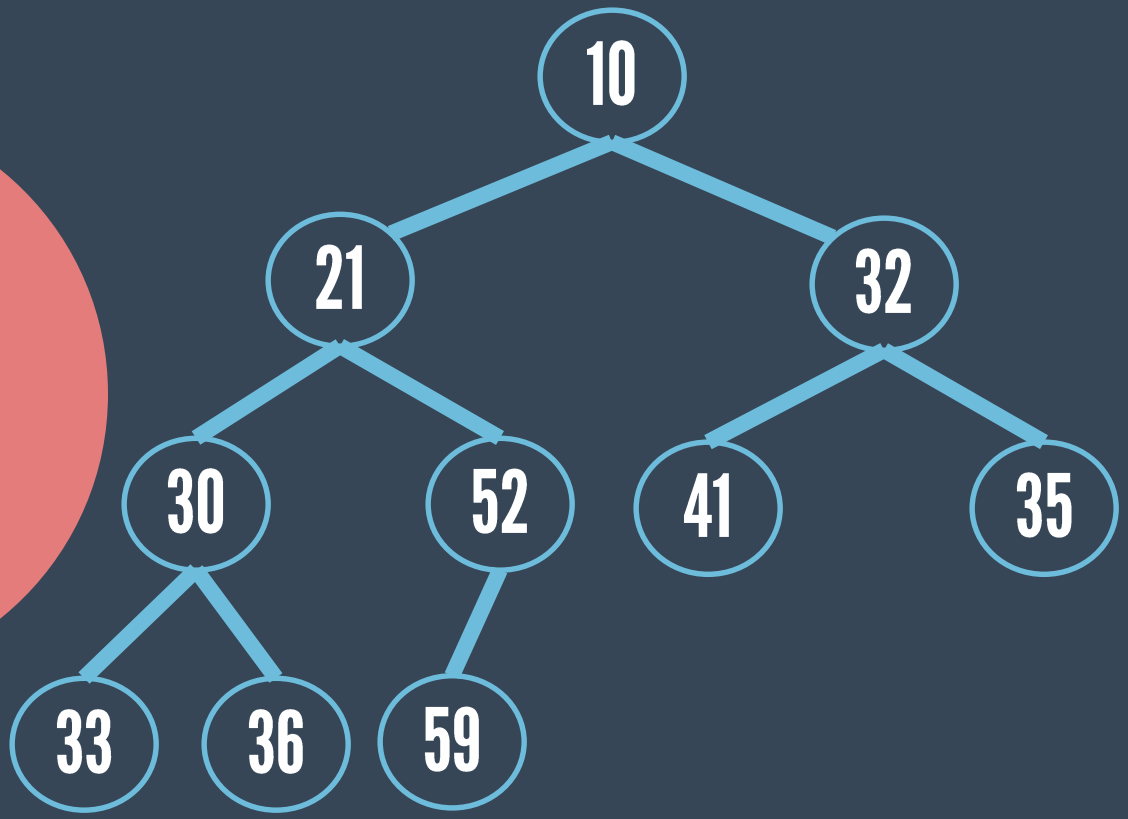


HEAP ORDER PROPERTY

Smallest element is at the root.

For every node X, the key in the parent of X is smaller than the key in X.

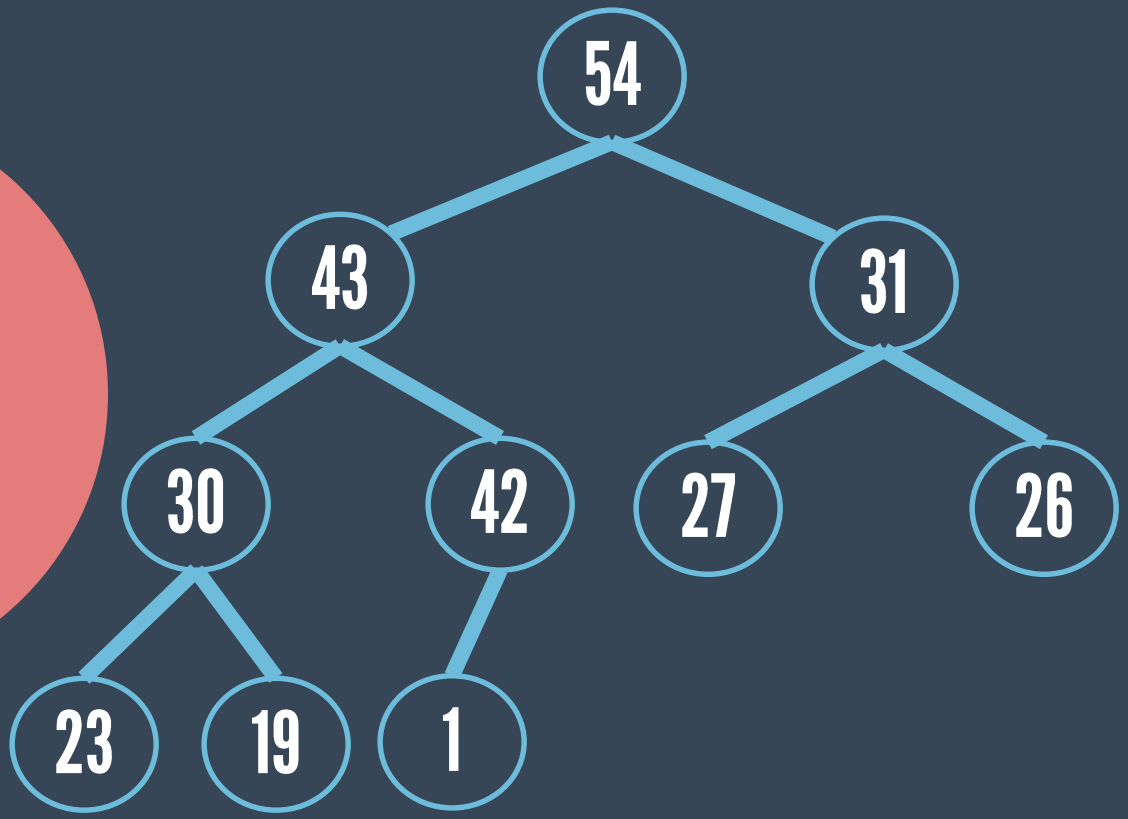
HEAP ORDER PROPERTY



MIN HEAP

MAX HEAP

HEAP ORDER PROPERTY

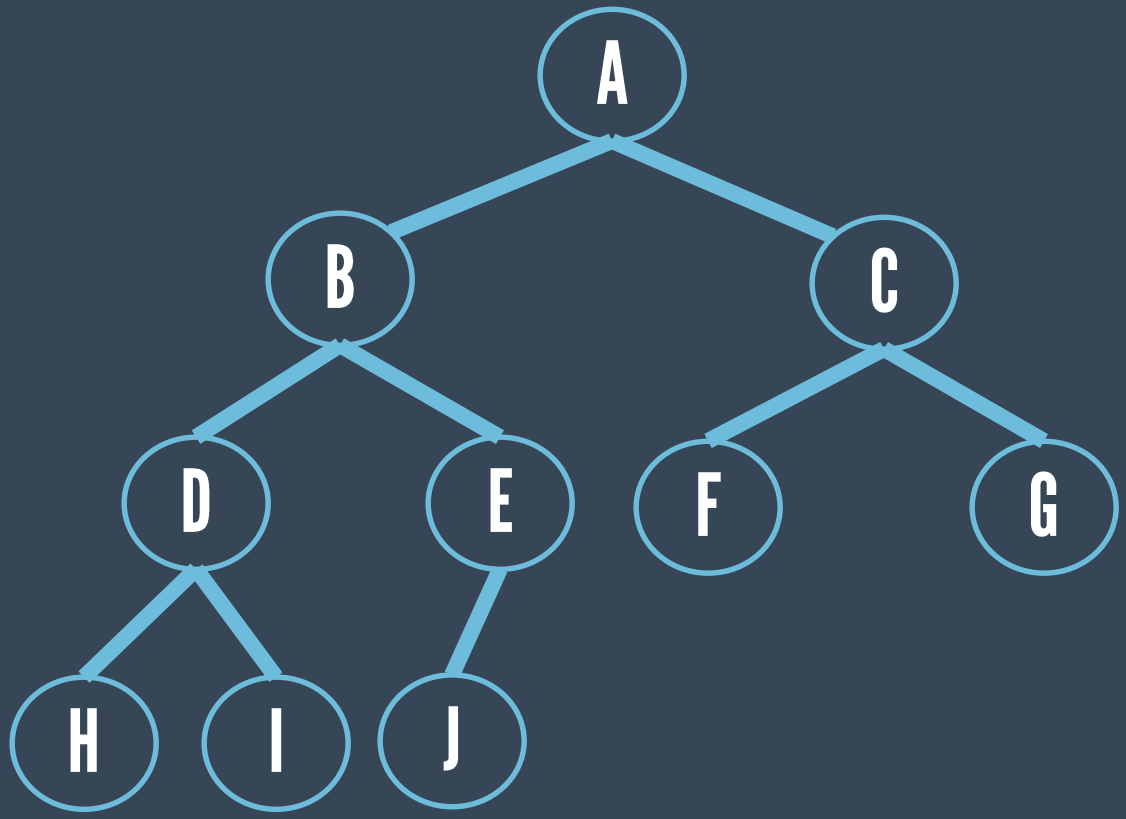


ARRAY

IMPLEMENTATION

For any element in array[i]:

- the left child is in position 2*i
- the right child is in position 2*i+1
- The parent is in $\lfloor i/2 \rfloor$

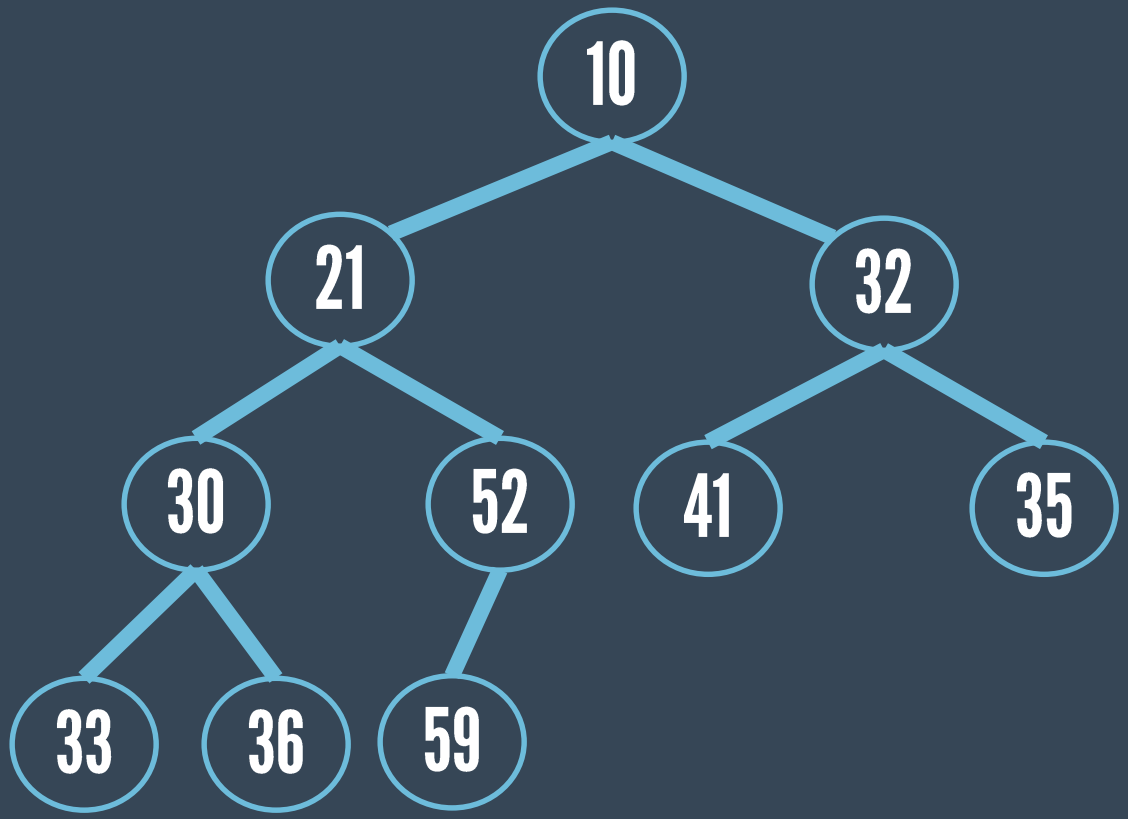


	A	B	C	D	E	F	G	H	I	J		
0	1	2	3	4	5	6	7	8	9	10	11	12

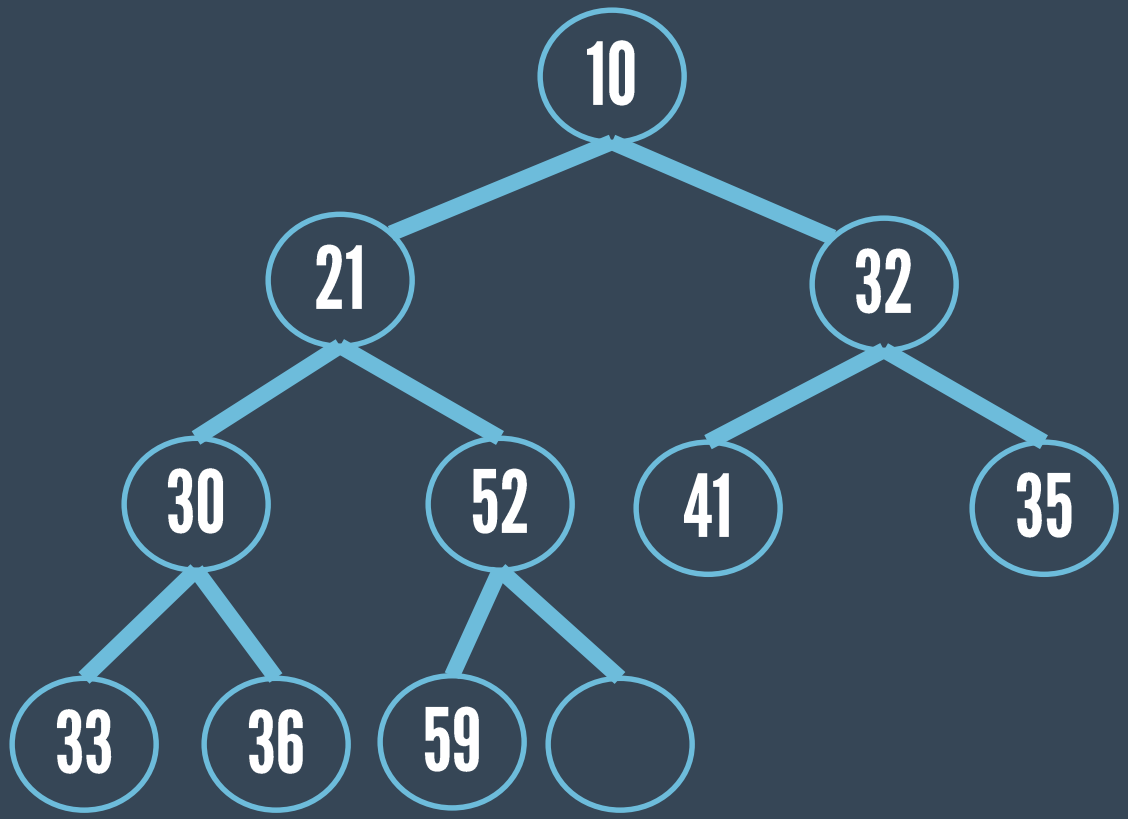
```
typedef struct node{
  int max_heap_size;
  int size;
  int *elements;
}
```

Strategy: percolate up

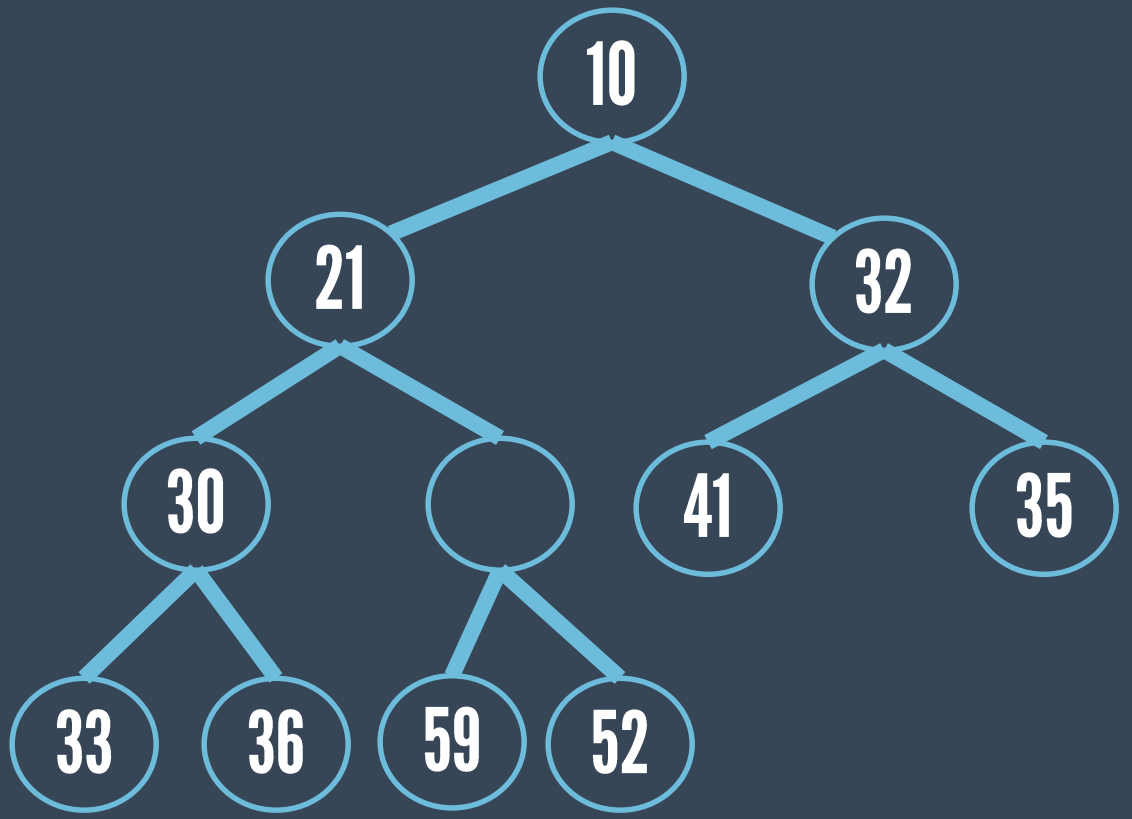
```
insert(9)
```



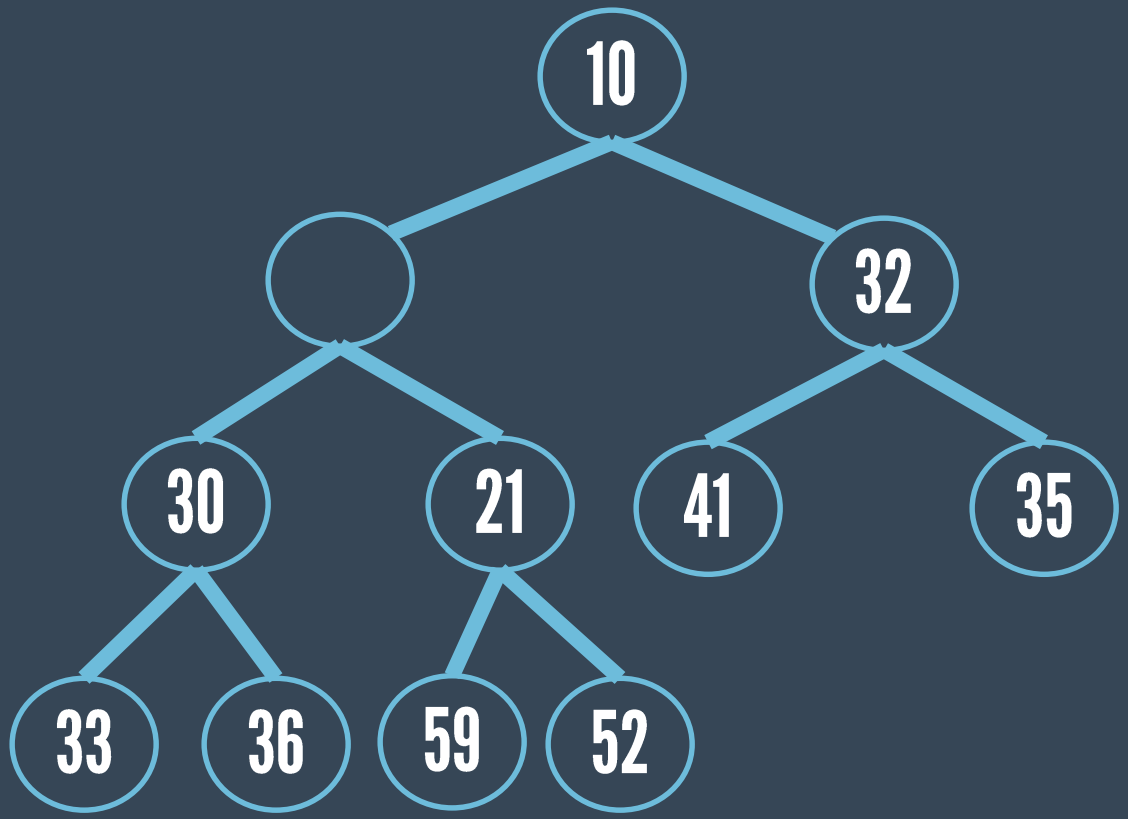
```
insert(9)
```



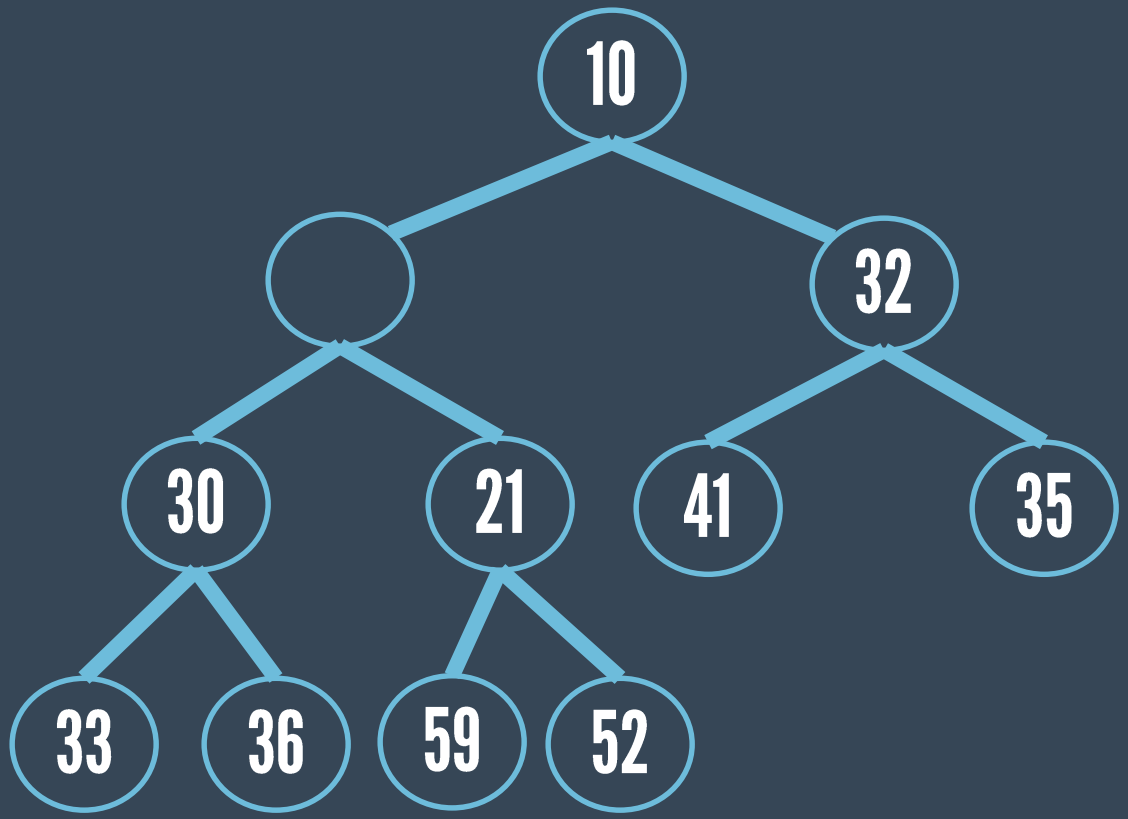
```
insert(9)
```



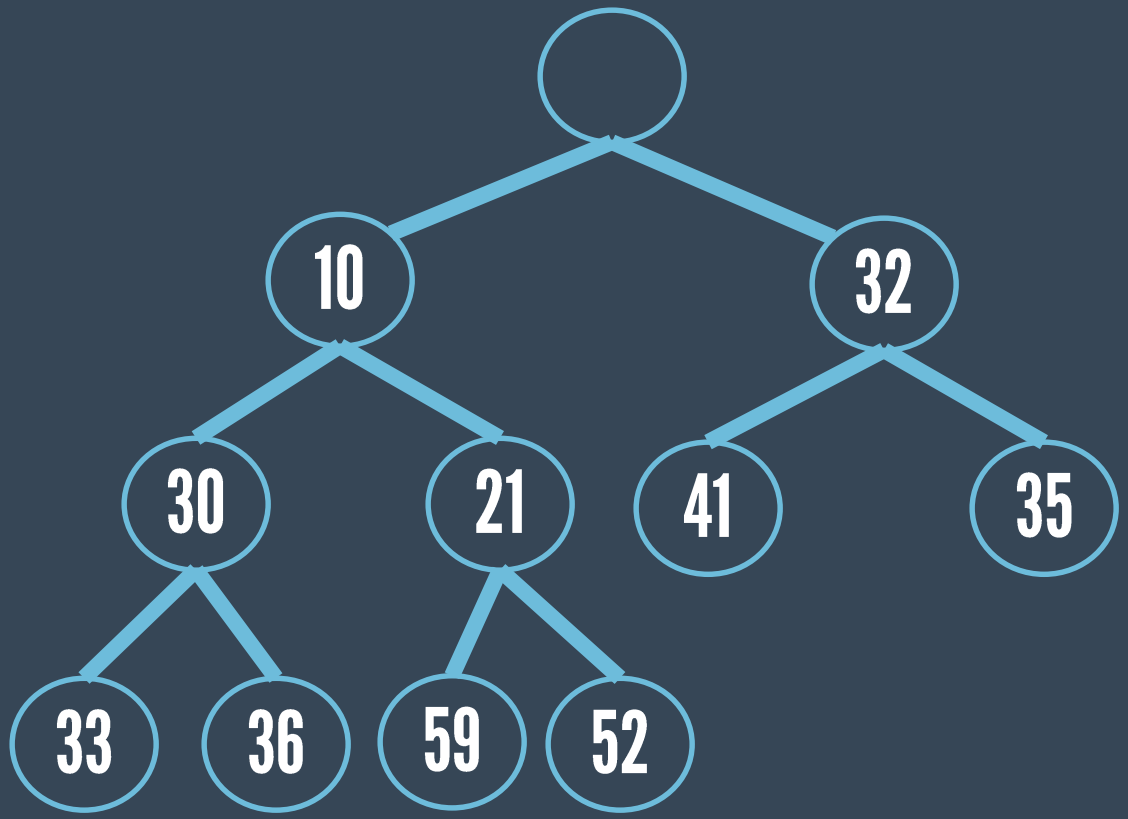
```
insert(9)
```



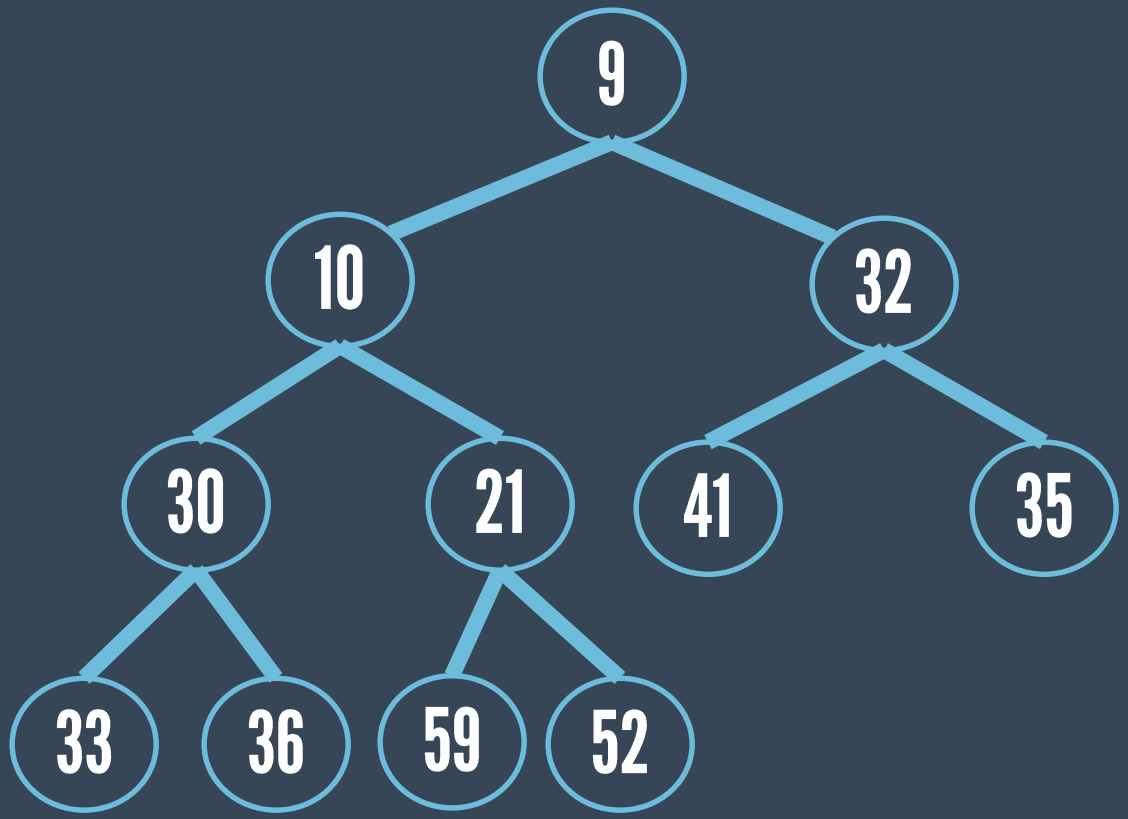
```
insert(9)
```



```
insert(9)
```

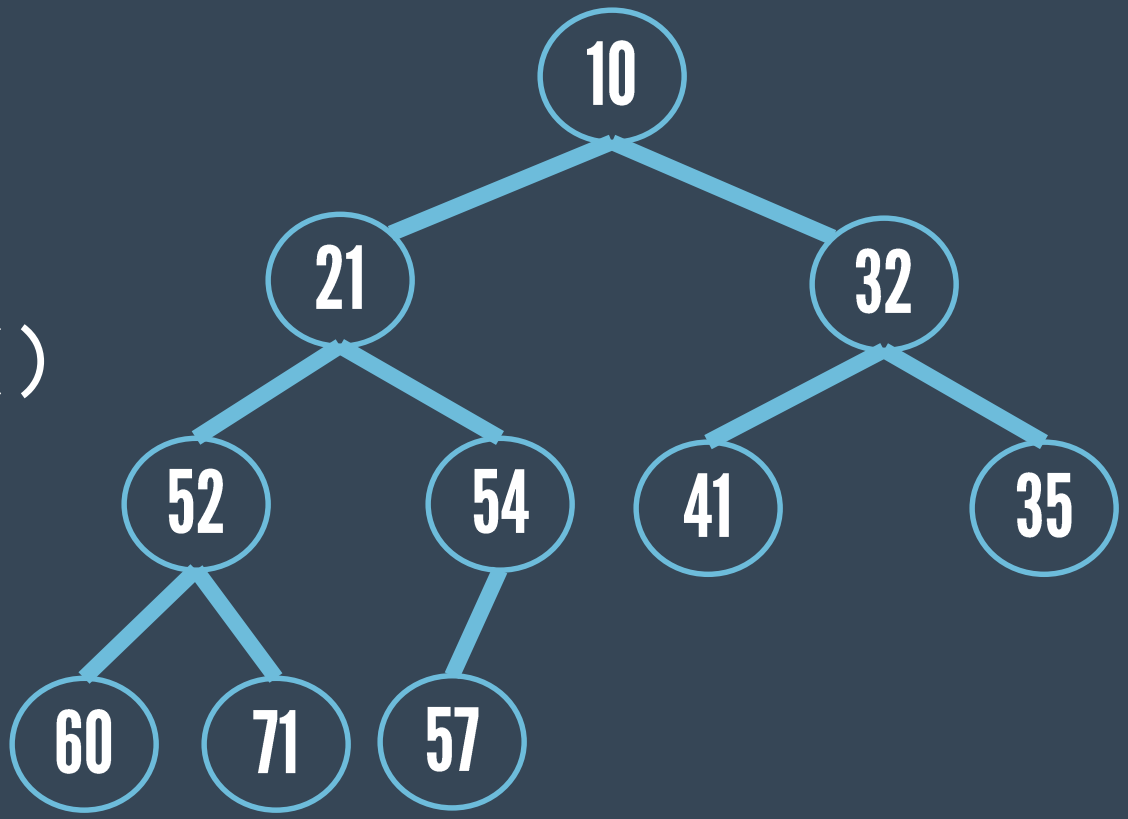


```
insert(9)
```

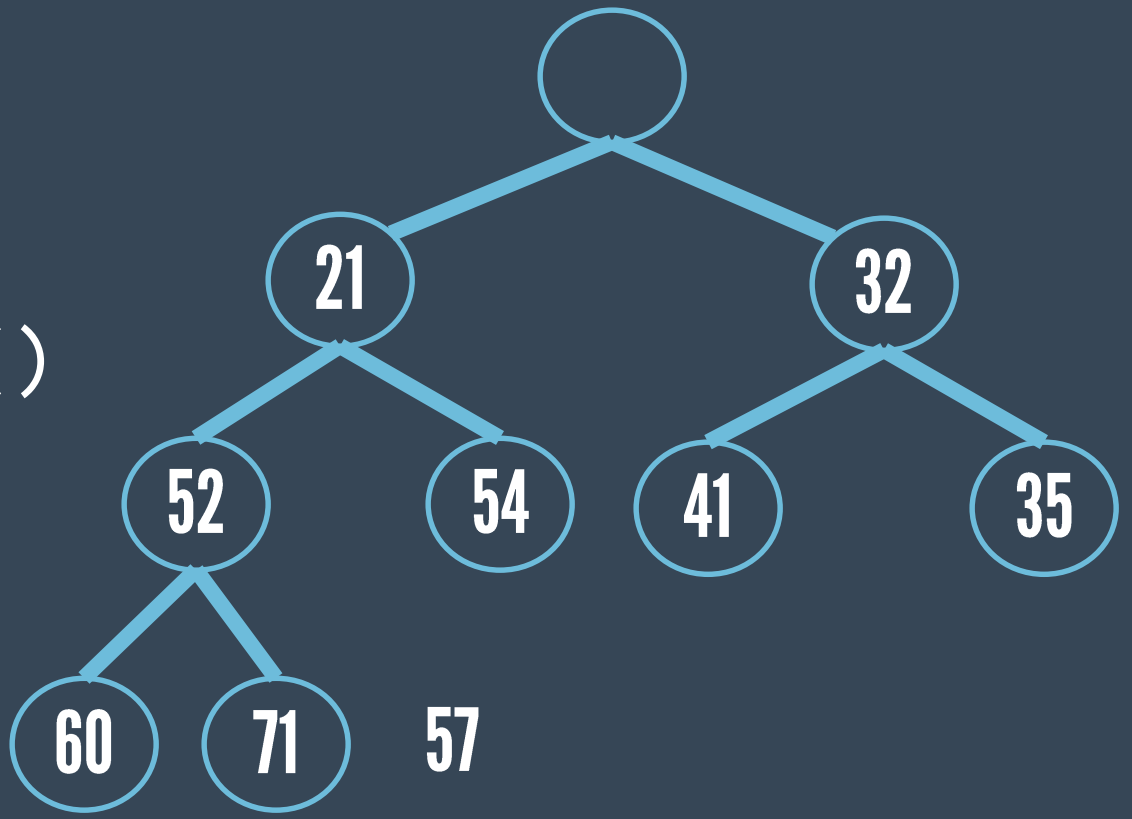


Strategy: percolate down

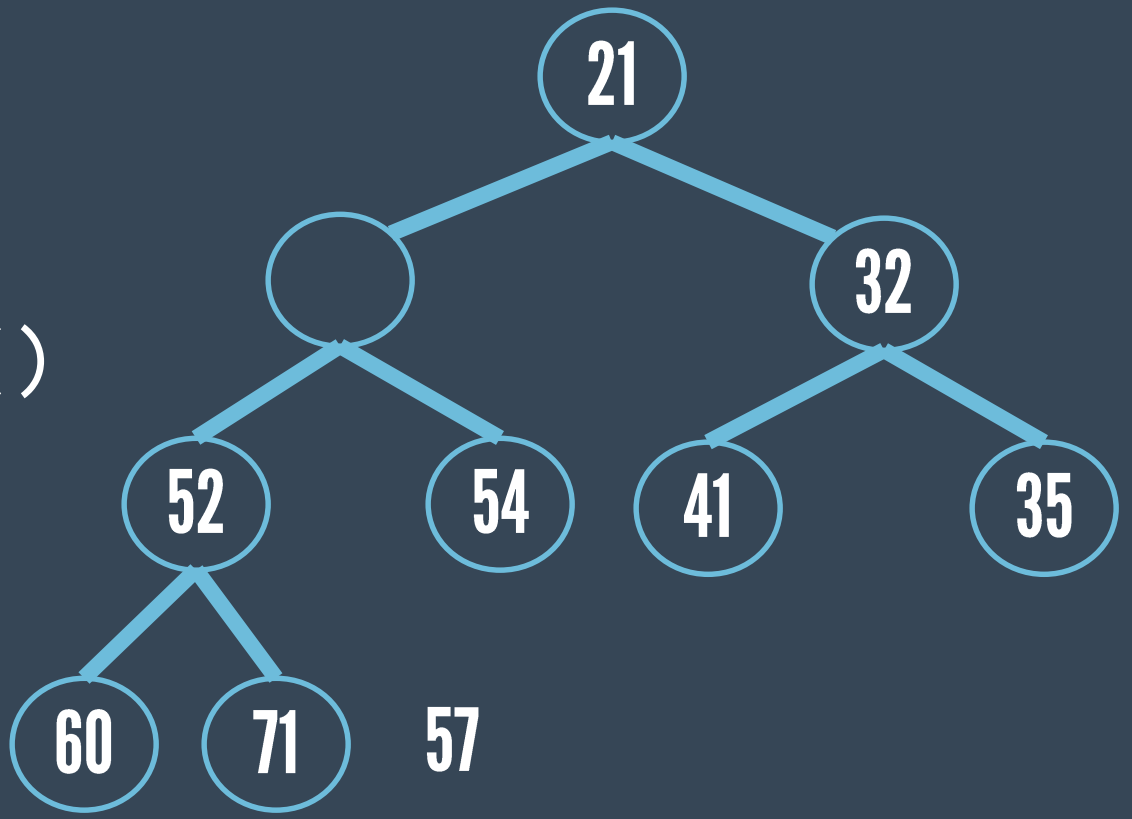
delete_min()



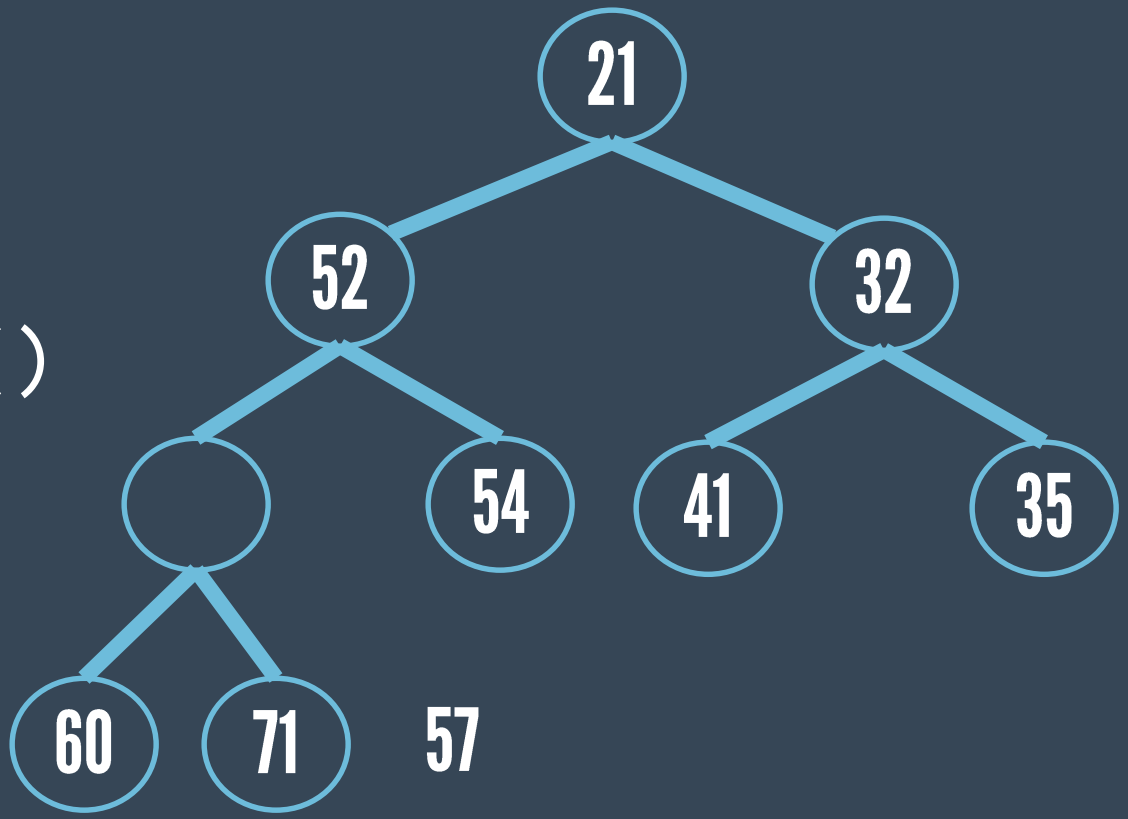
delete_min()



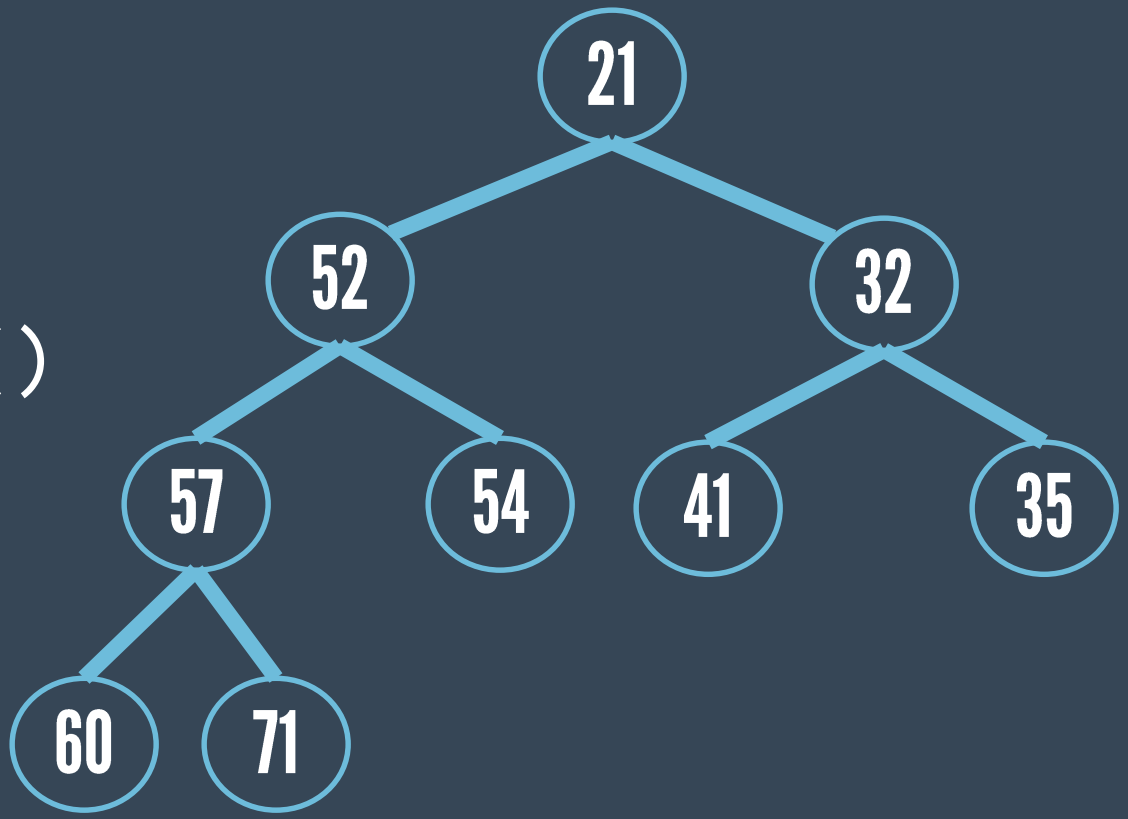
delete_min()



delete_min()



delete_min()



build heap

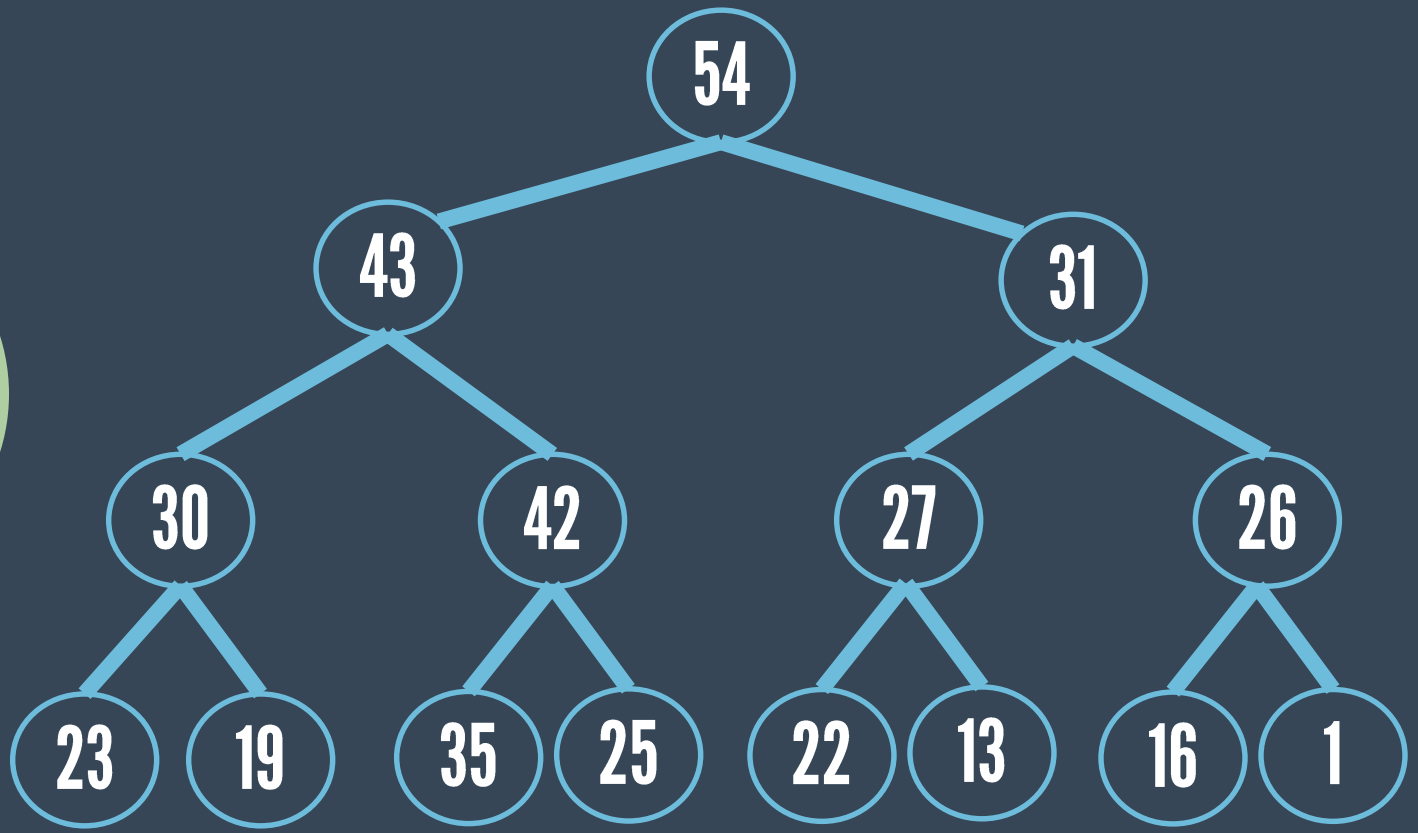
Take n inputs and place them into an empty heap.

build heap

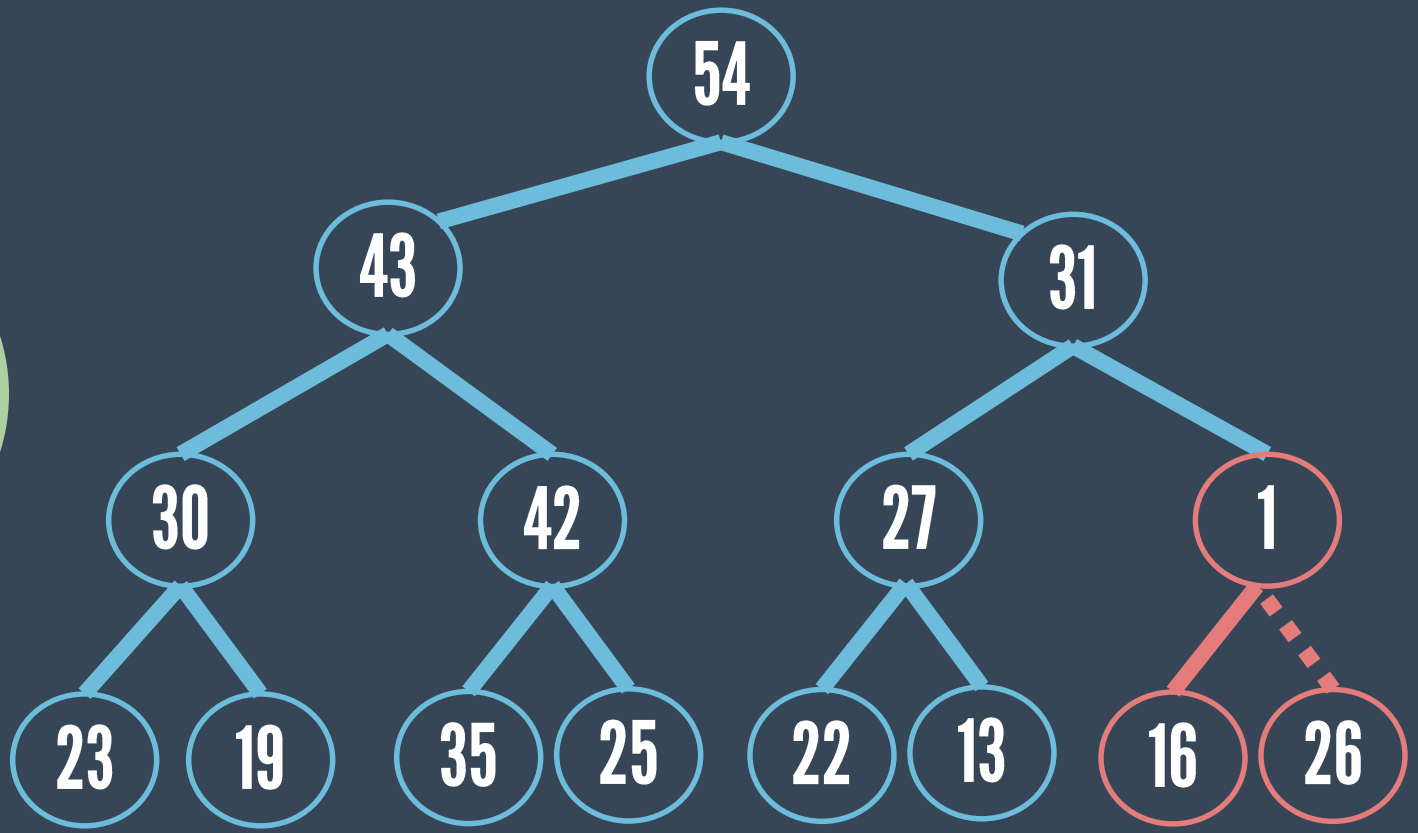
- n successive inserts
- percolate down from n/2 to 1.

```
for(i=n/2;i>0;i--)
  percolate_down(i);
```

build heap

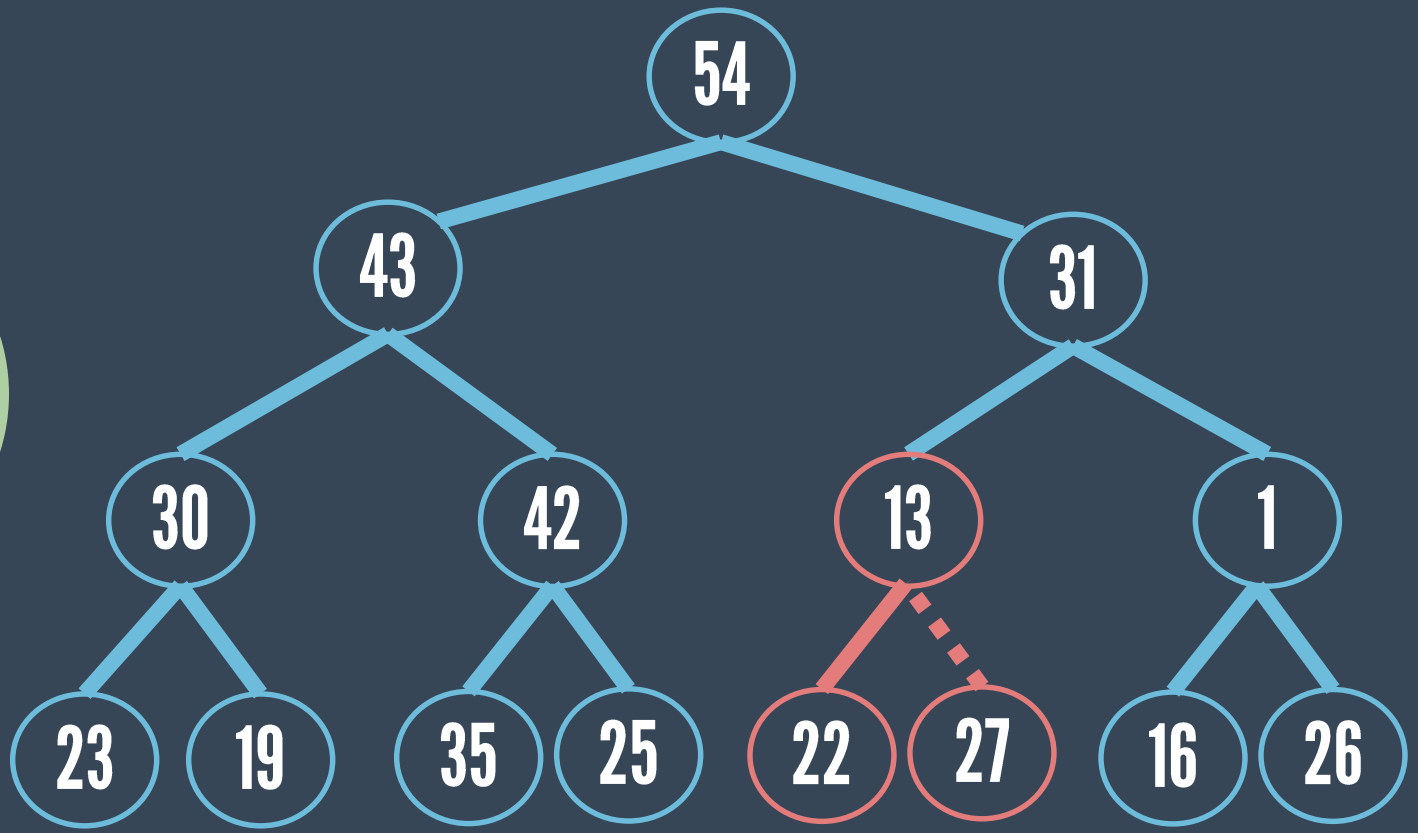


build heap



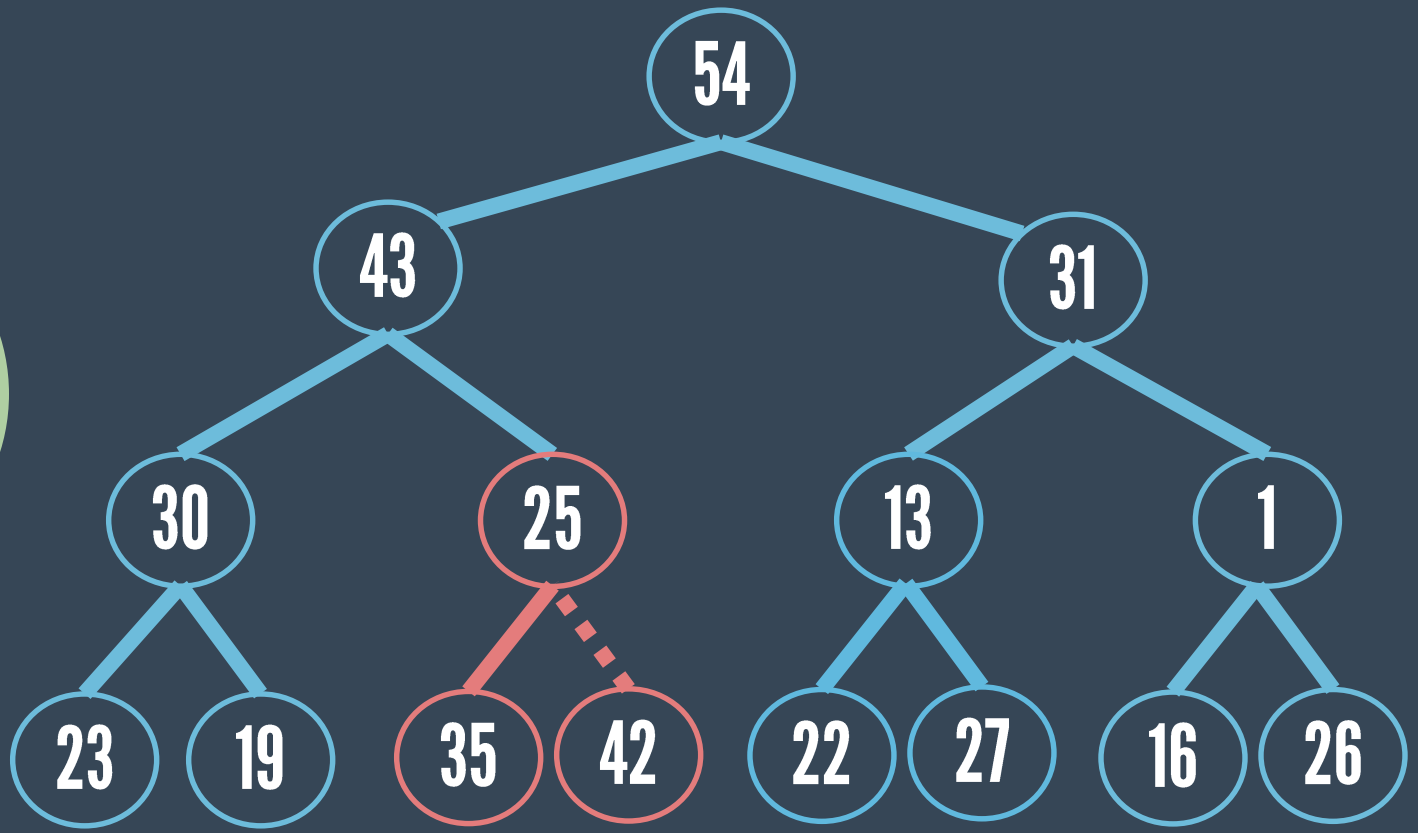
`percolate_down(7)`

build heap



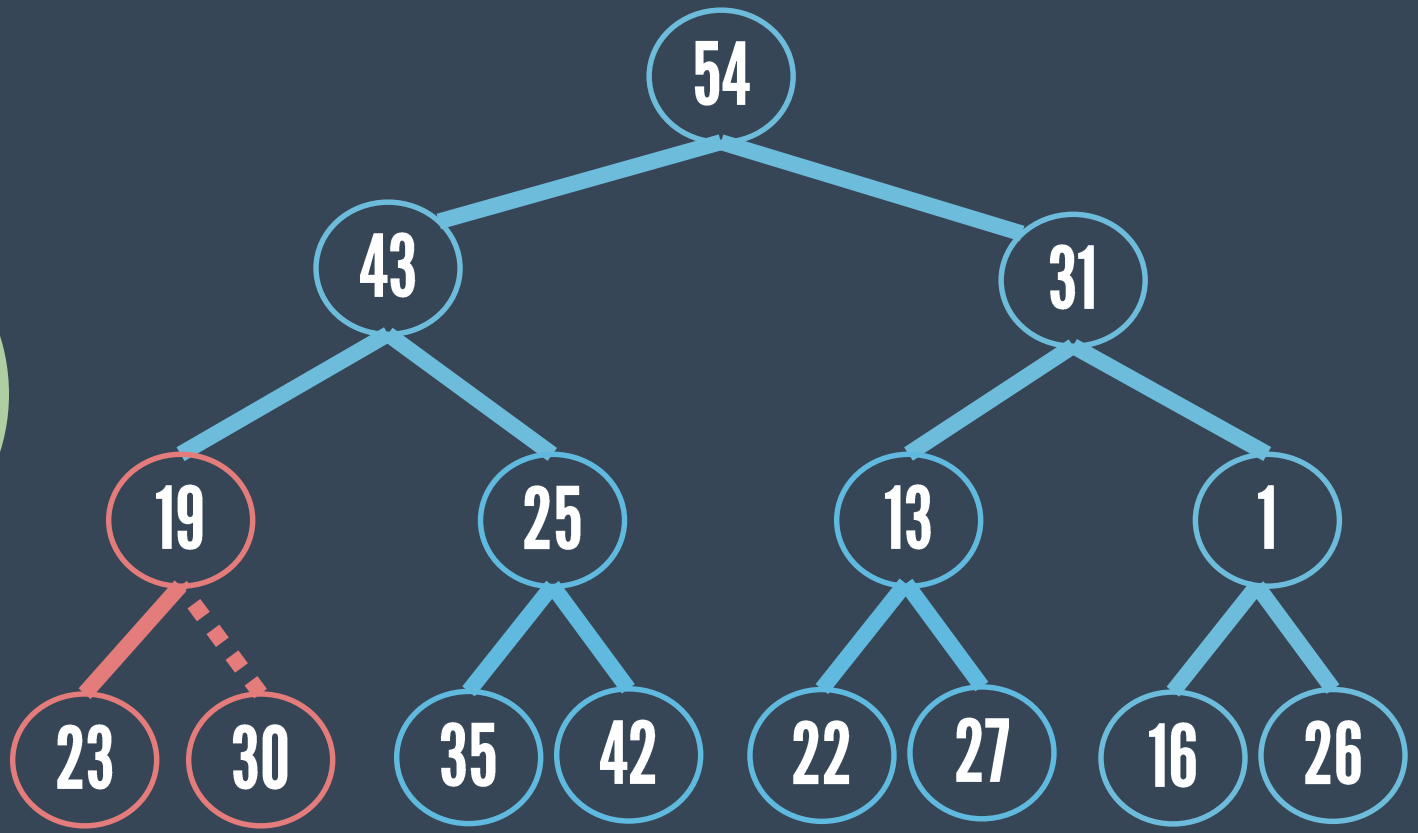
`percolate_down(6)`

build heap



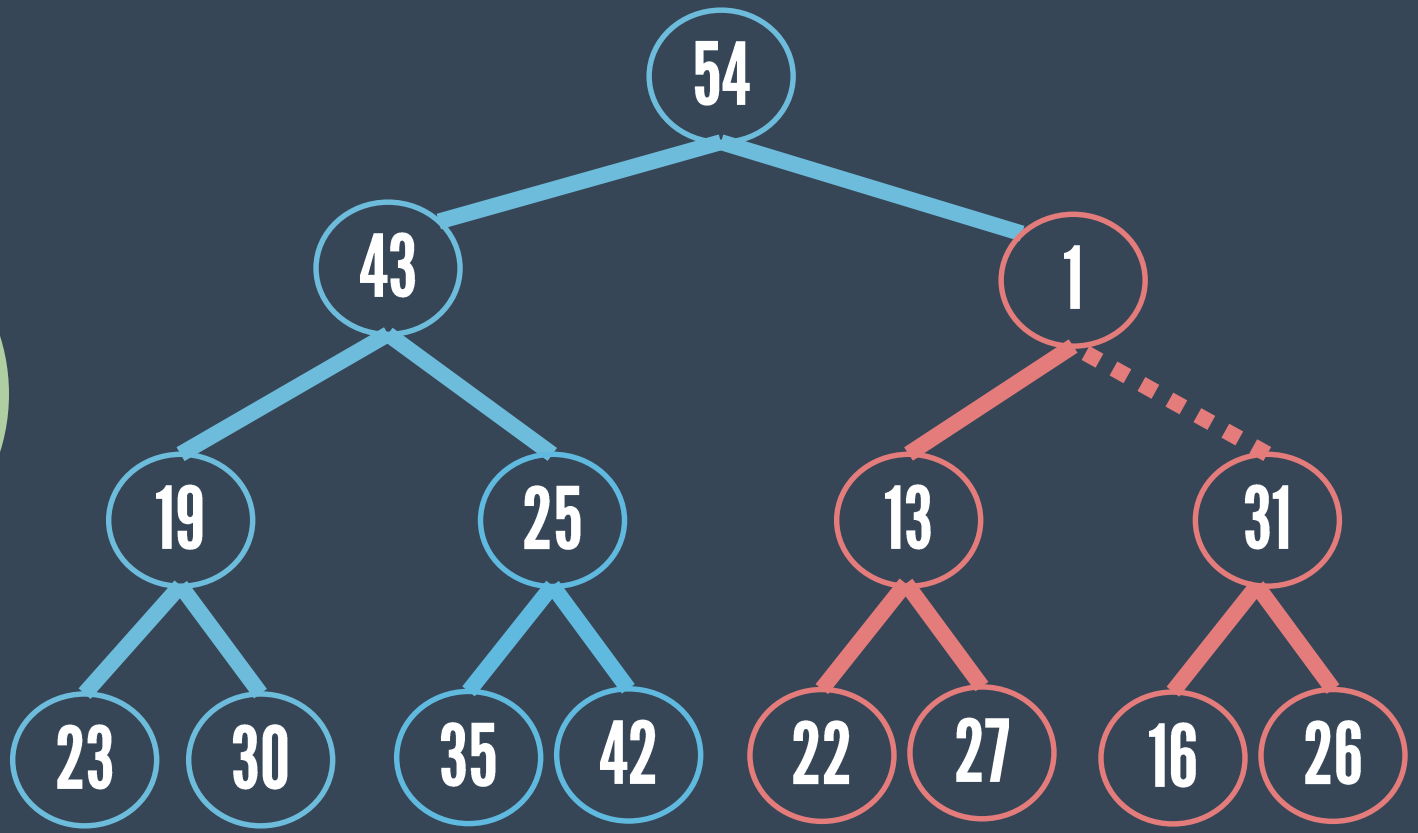

```
percolate_down(5)
```

build heap



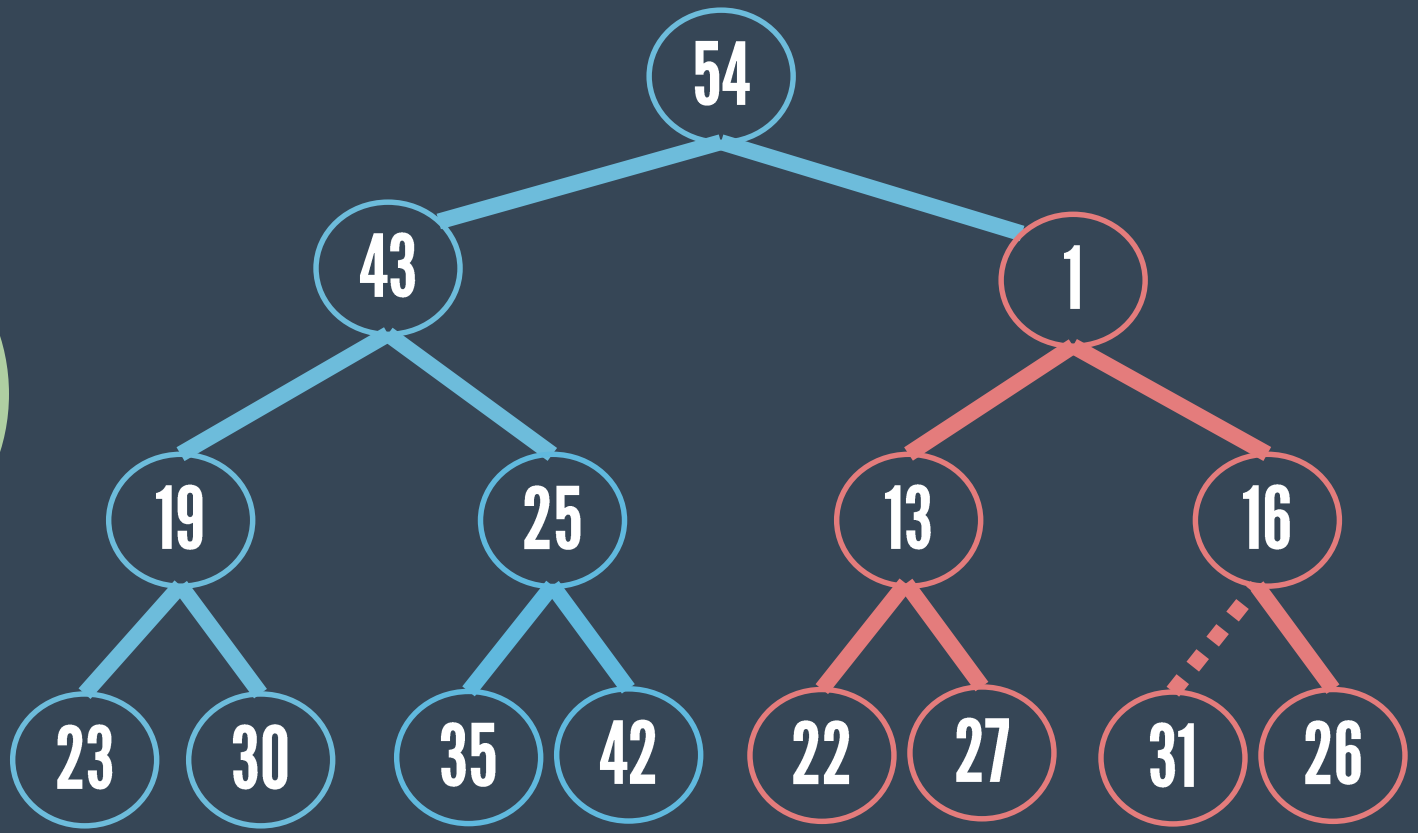
percolate_down(4)

build heap



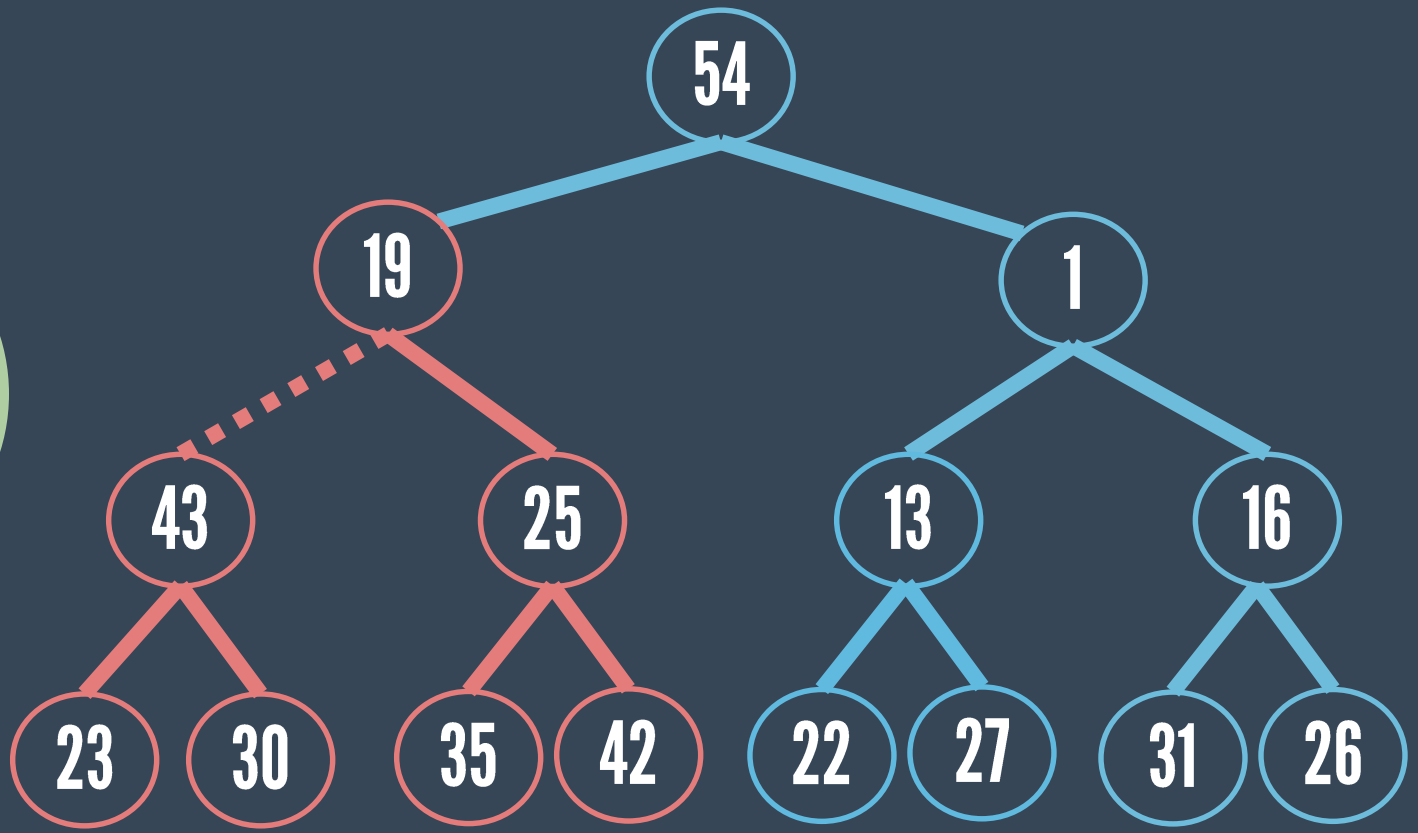
percolate_down(3)

build heap



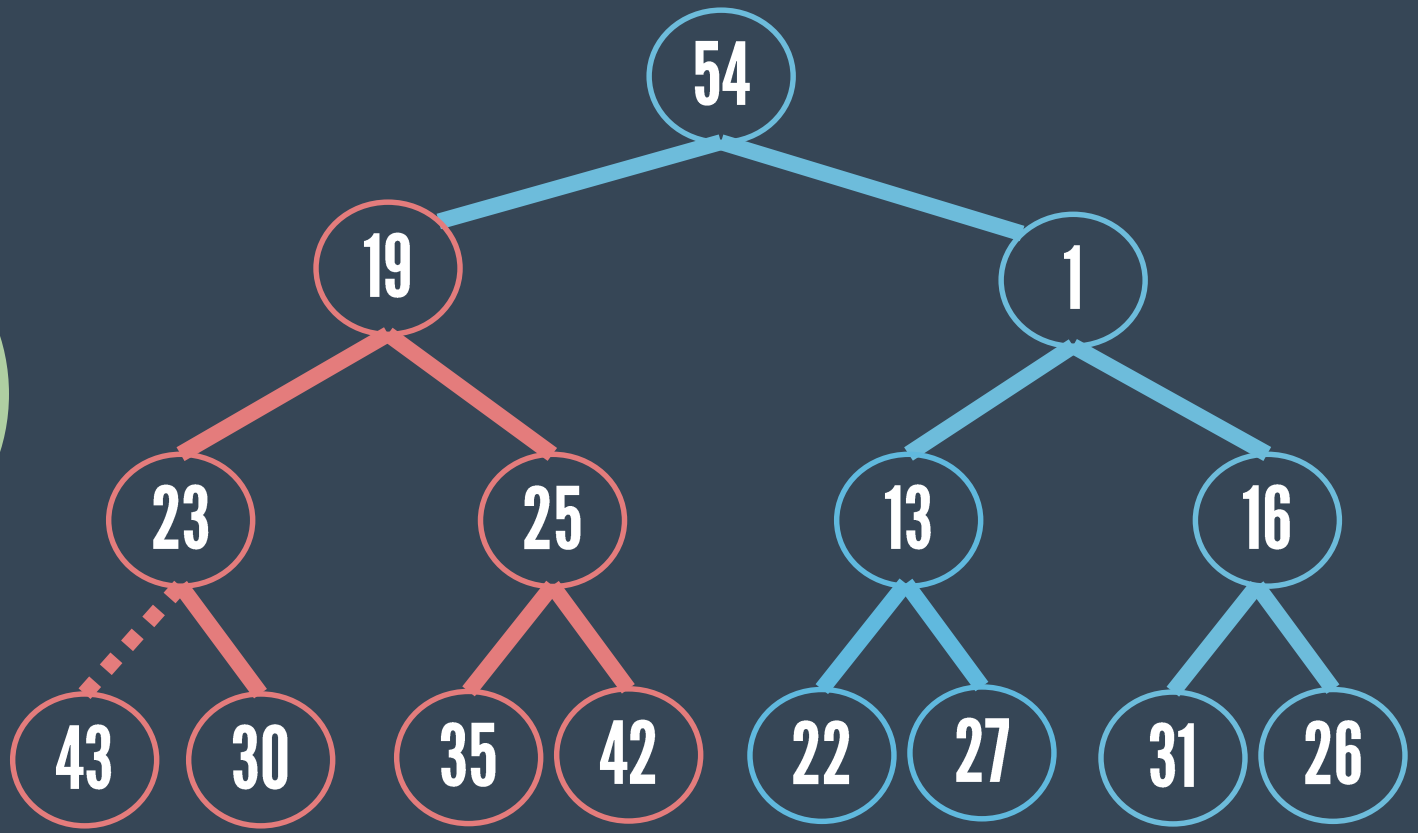
percolate_down(3)

build heap



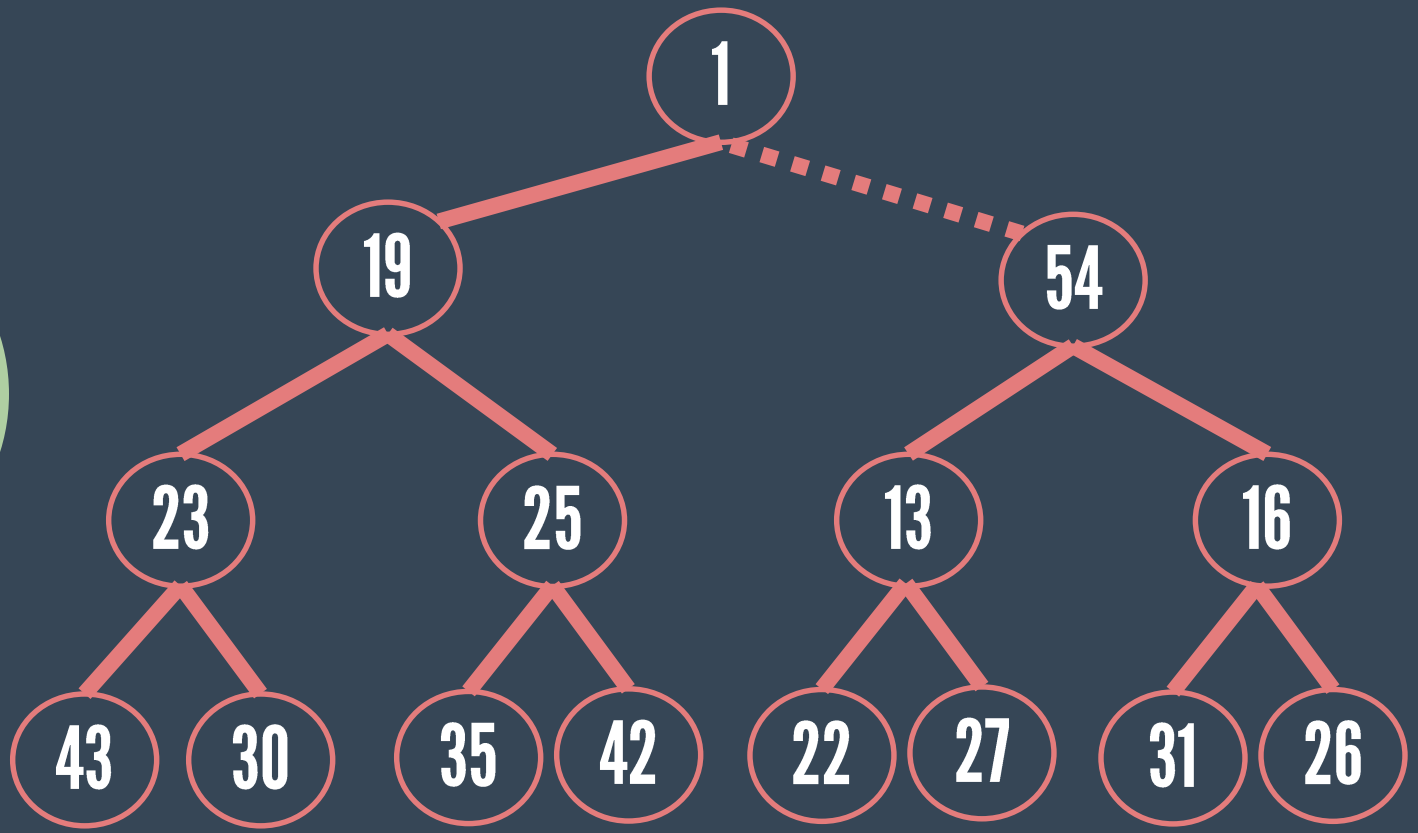

```
percolate_down(2)
```

build heap



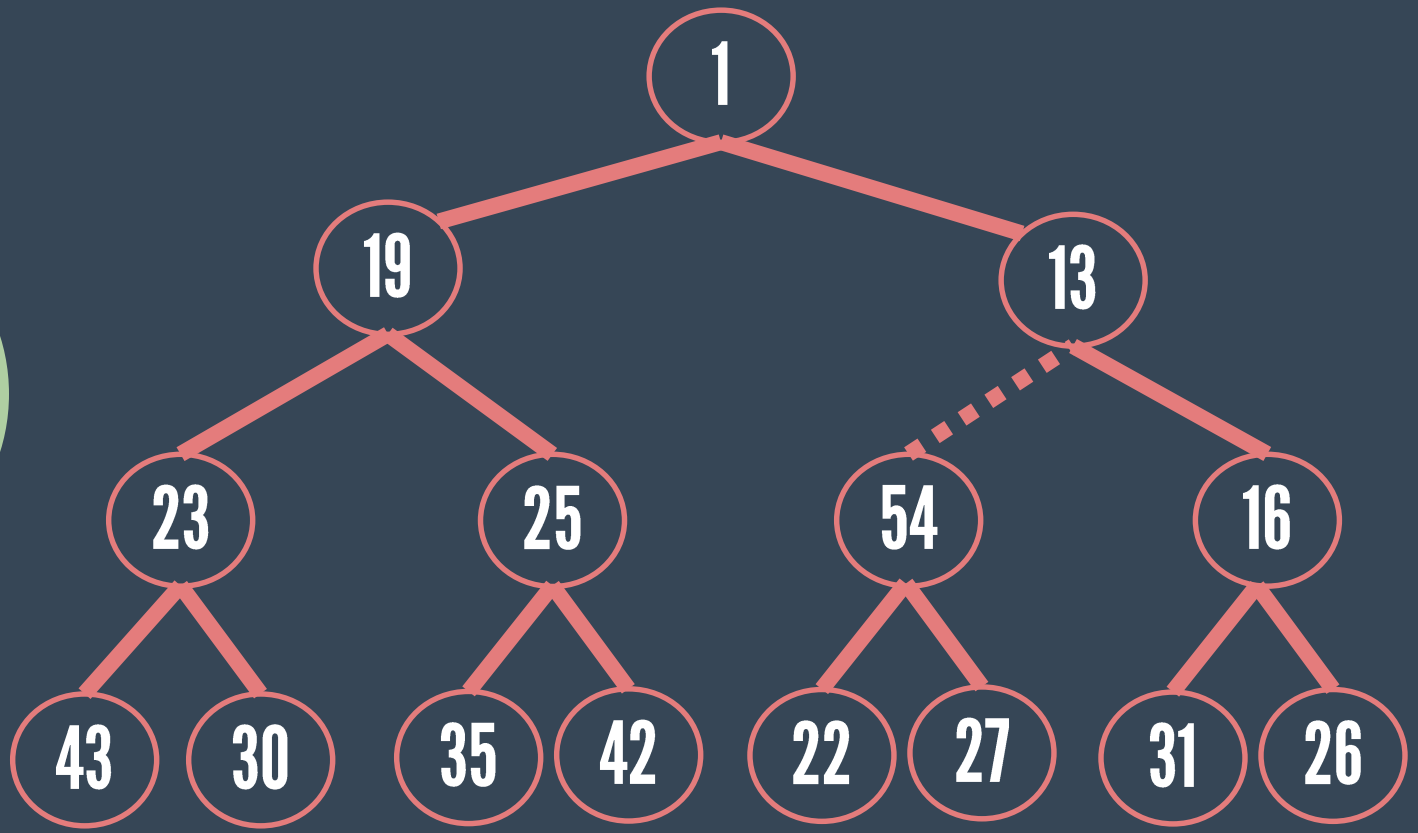
`percolate_down(2)`

build heap



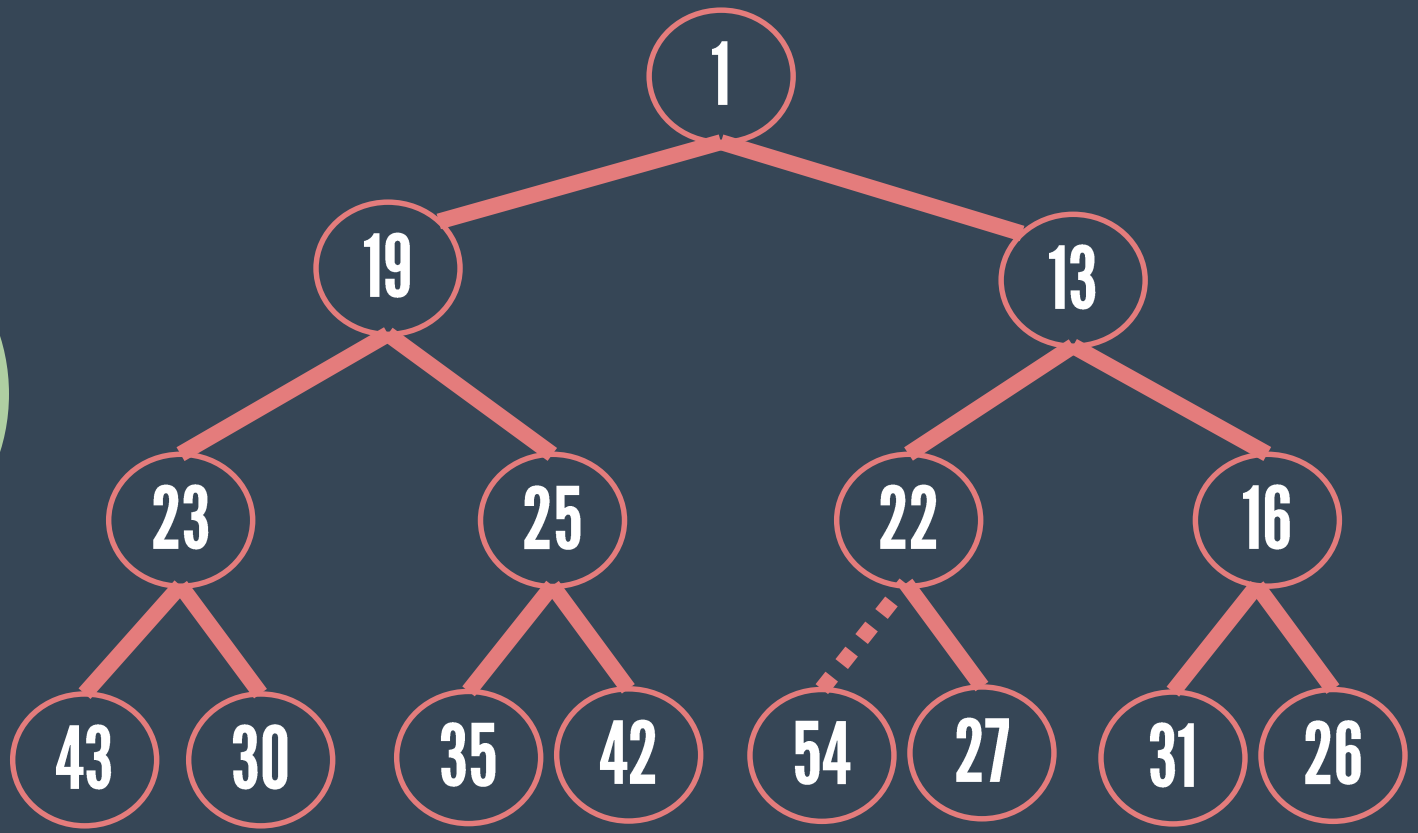
`percolate_down(1)`

build heap




```
percolate_down(1)
```

build heap




```
percolate_down(1)
```

Complete Binary Trees

build heap

height: h

# of nodes: $2^{h+1} - 1$

or

height: $\approx \log(n)$

# of nodes: n

Running Time

Compute the sum of the heights of all the nodes in the heap.

Running Time

build heap

$$\sum_{i=0}^{h} 2^i(h-i)$$

$$= 2^{h+1} - 1 - (h+1)$$

$$\approx 2^h$$

$$\approx n$$

$$O(n)$$

HEAPSORT

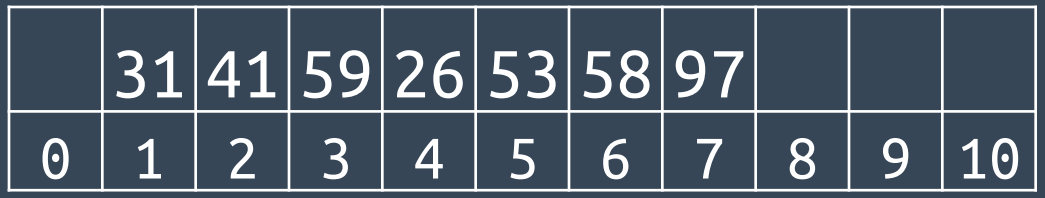
(APPLICATION)

O(n log n)

HEAPSORT

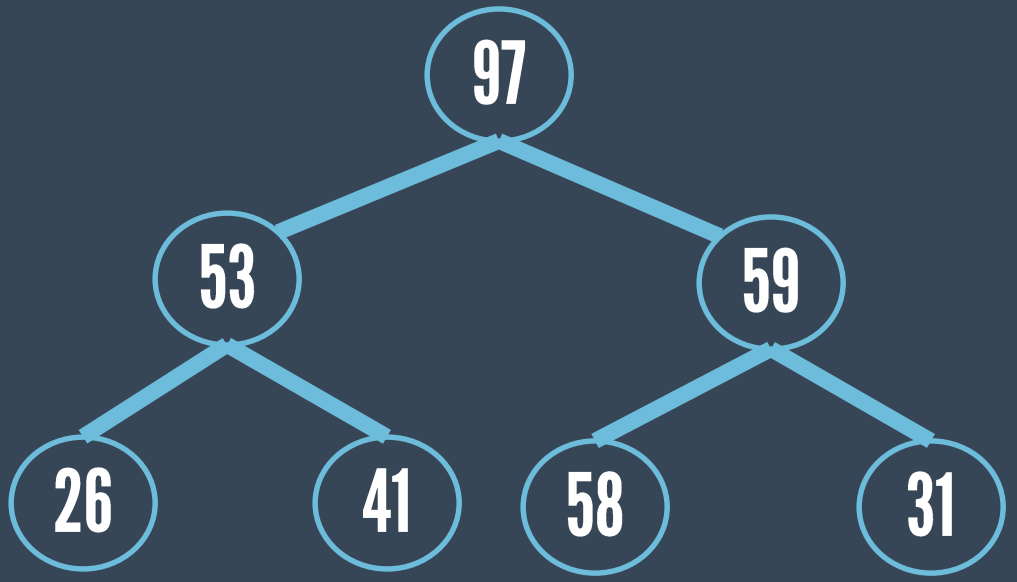
- build heap $O(n)$
- perform n delete_min operations $O(n \log n)$

HEAPSORT



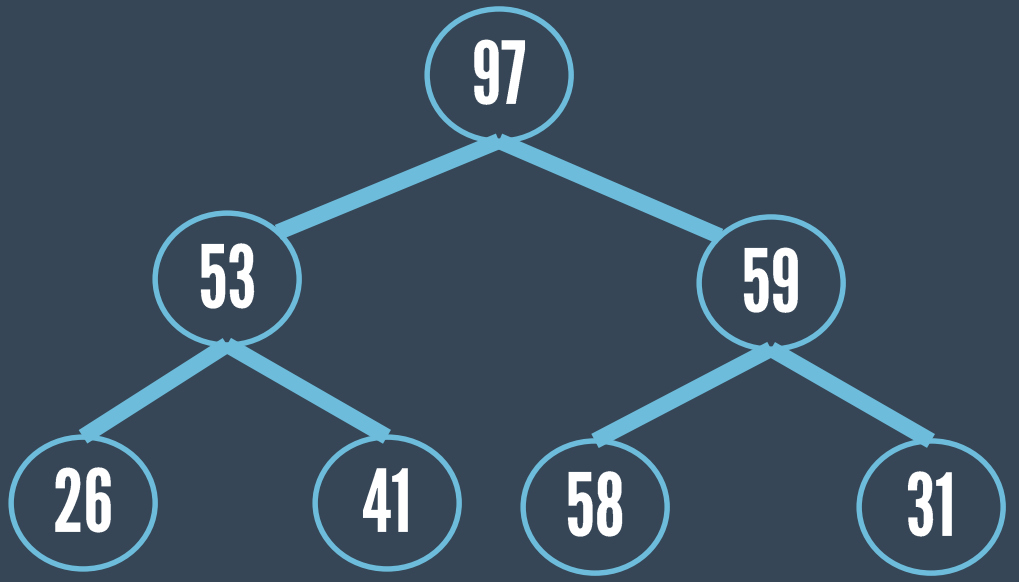
	31	41	59	26	53	58	97			
0	1	2	3	4	5	6	7	8	9	10

build heap



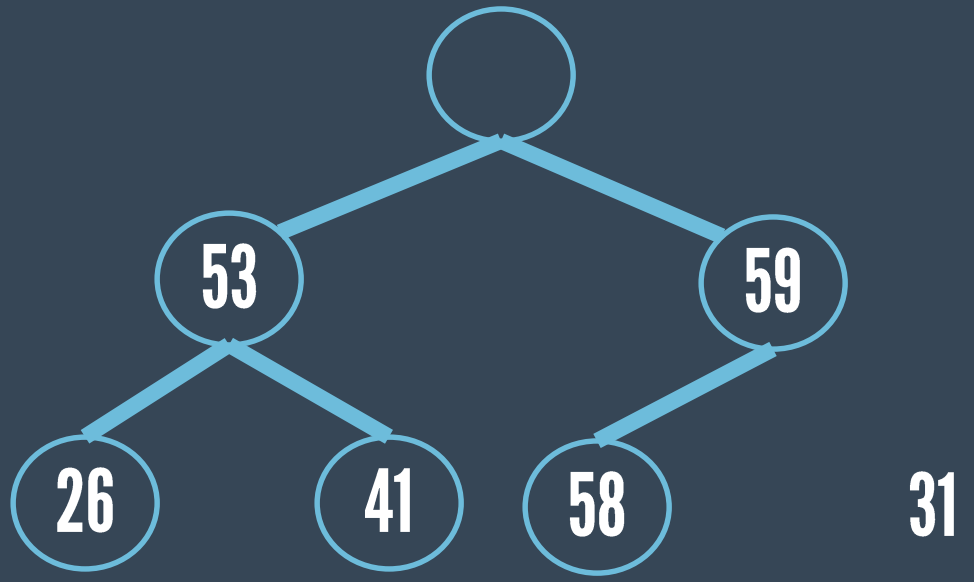
	97	53	59	26	41	58	31			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



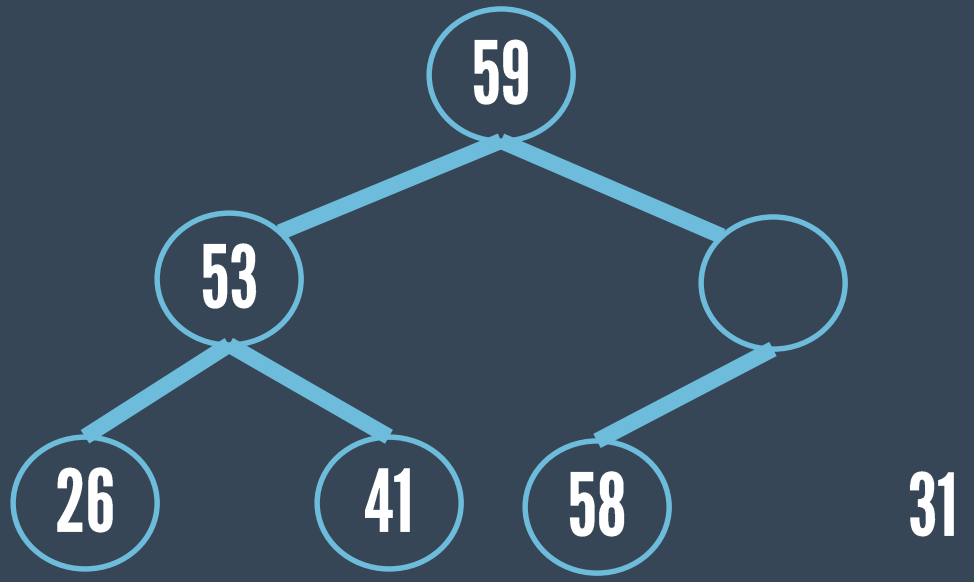
	97	53	59	26	41	58	31			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



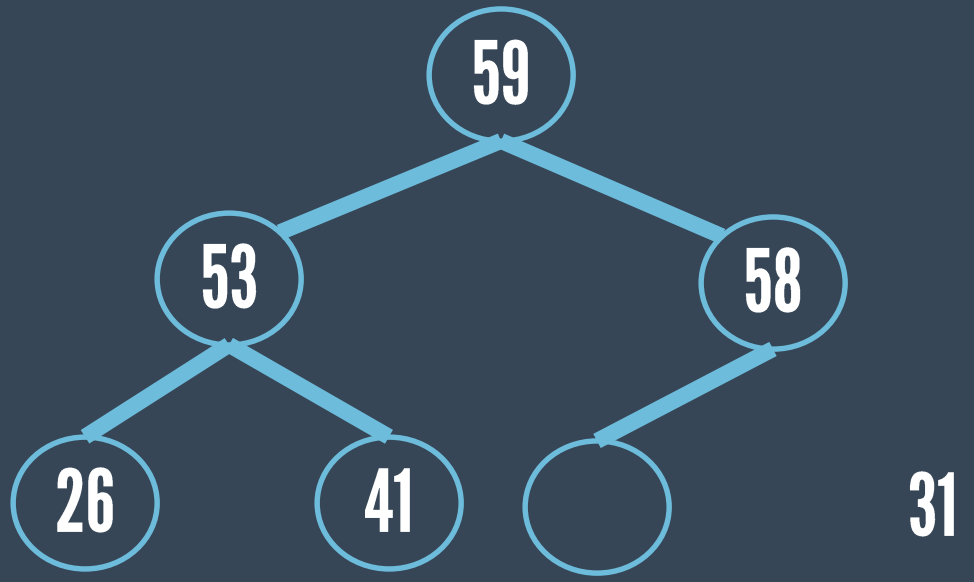
	97	53	59	26	41	58	31			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



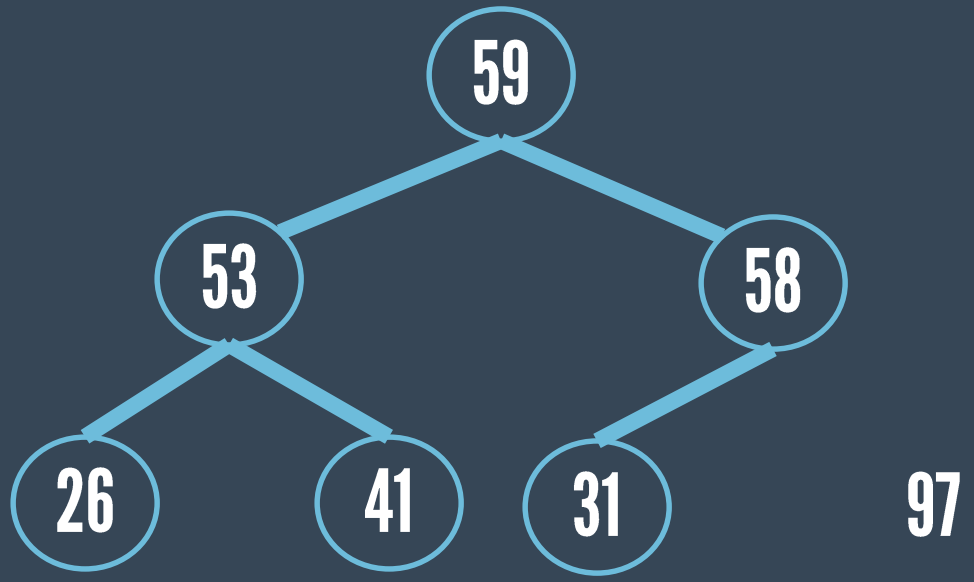
	97	53	59	26	41	58	31			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



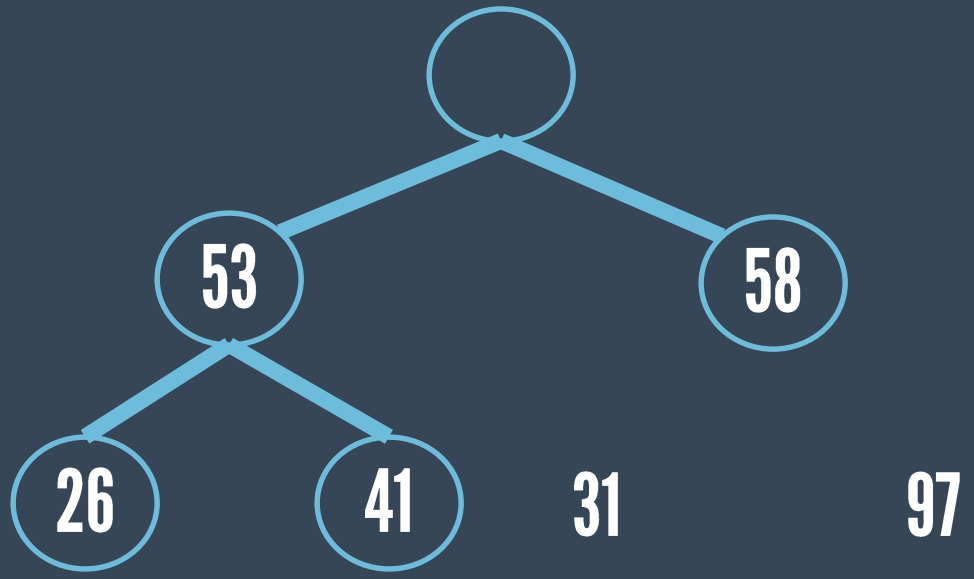
	97	53	59	26	41	58	31			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



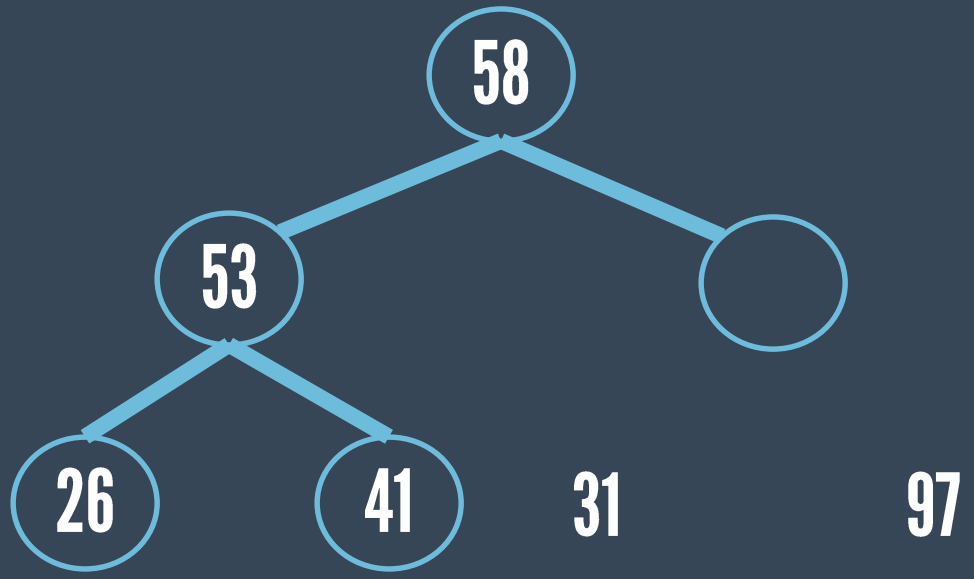
	59	53	58	26	41	31	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



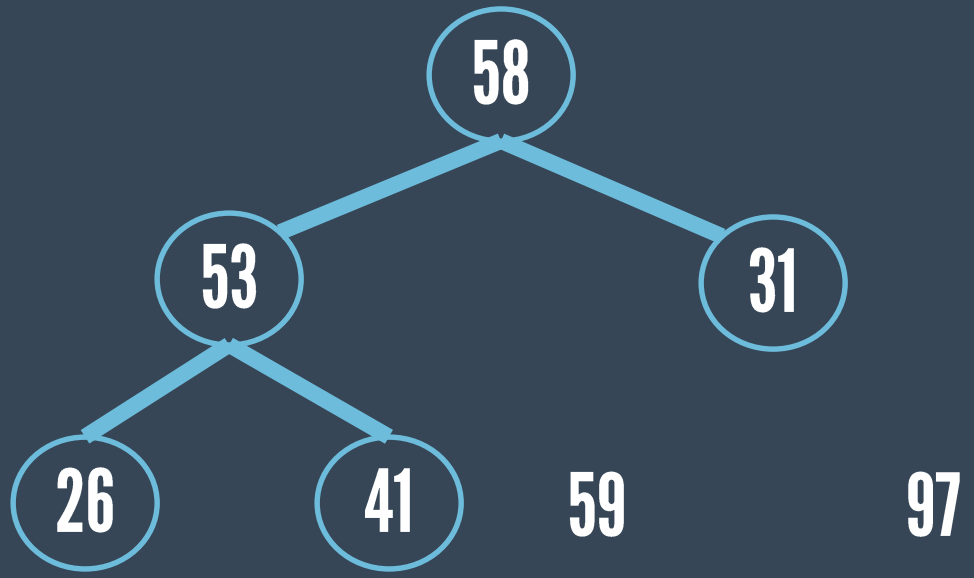
	97	53	59	26	41	58	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



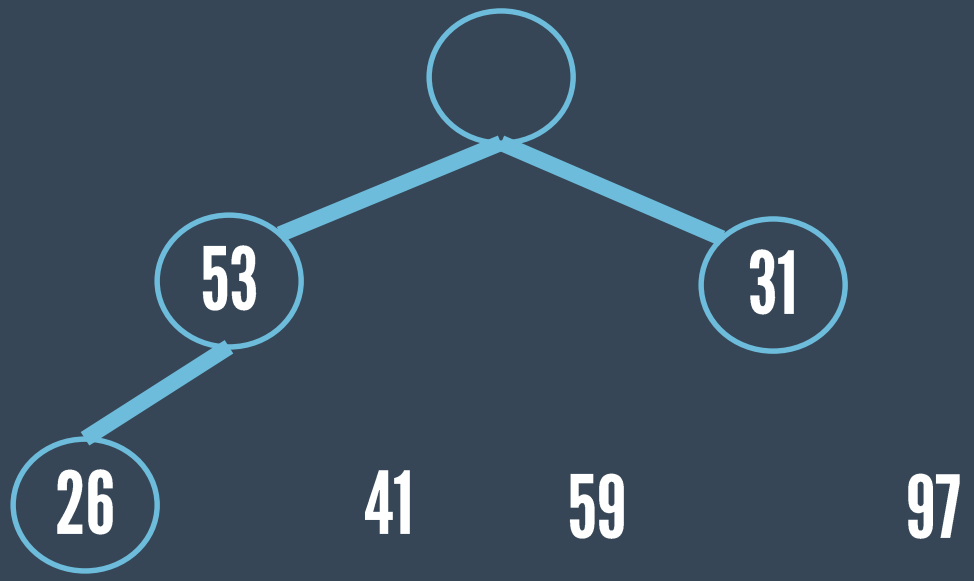
	59	53	58	26	41	31	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



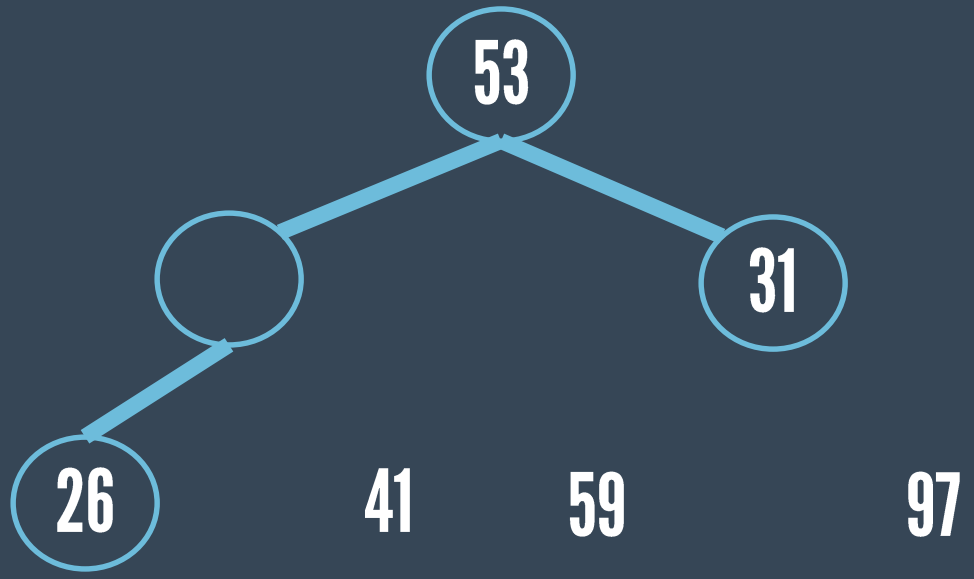
	58	53	31	26	41	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



	58	53	31	26	41	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



	58	53	31	26	41	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs

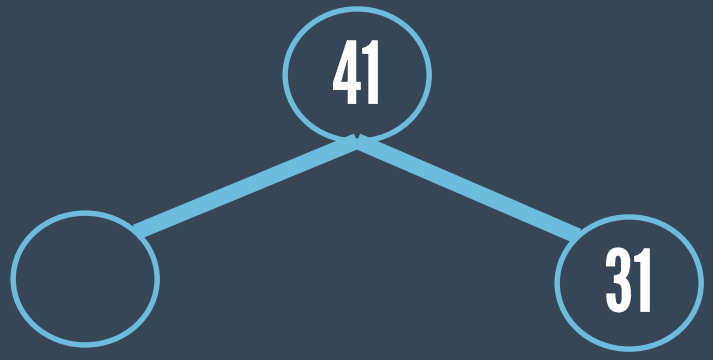
	53	41	31	26	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs

26 58 59 97

	53	41	31	26	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

n delete maxs



26 58 59 97

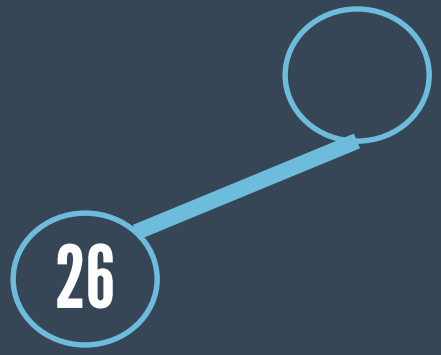
	53	41	31	26	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs

53 58 59 97

	41	26	31	53	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs

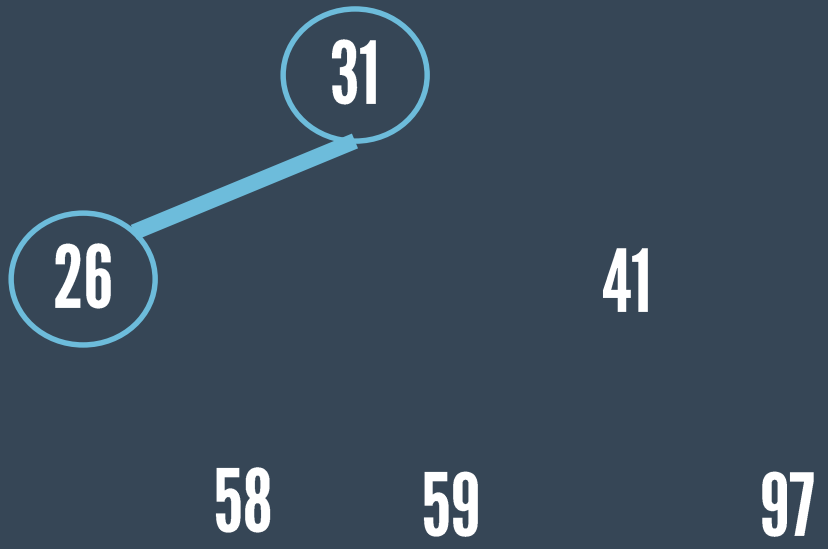


31

53 58 59 97

	41	26	31	53	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



	31	26	41	53	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

26

41

53

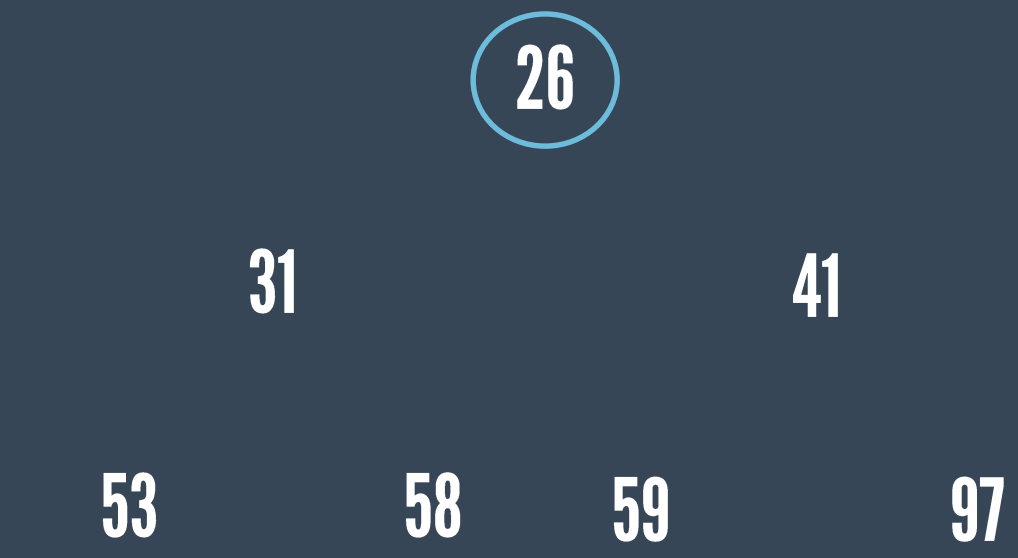
58

59

97

	31	26	41	53	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs



	26	31	41	53	58	59	97			
0	1	2	3	4	5	6	7	8	9	10

n
delete
maxs

	26	31	41	53	58	59	97			
0	1	2	3	4	5	6	7	8	9	10