Data Structures Quiz 1

Mihir Aggarwal

October 2023

1 Heap Insert

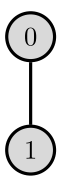
`arr1[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}`

1. `i = 0; arr1[i] = 1;`

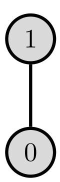
   Since $i = 0$, return.

2. `i = 1; arr[i] = 1;`
   `i != 0; parent = 0`
   `arr[parent] < arr[i]; swap`
   `i = parent = 0; return`

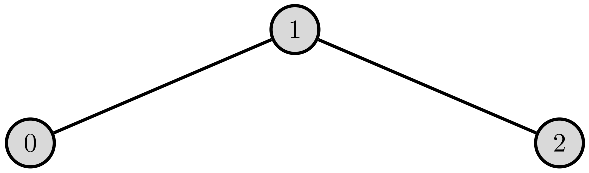
   

   $\downarrow$

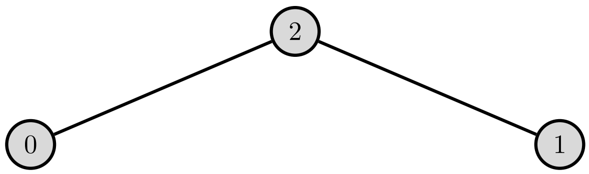
   

   cc = 1; sc = 1

3. `i = 2; arr[i] = 2;`
   `i != 0; parent = 0`
   `arr[parent] < arr[i]; swap`
   `i = parent = 0; return`

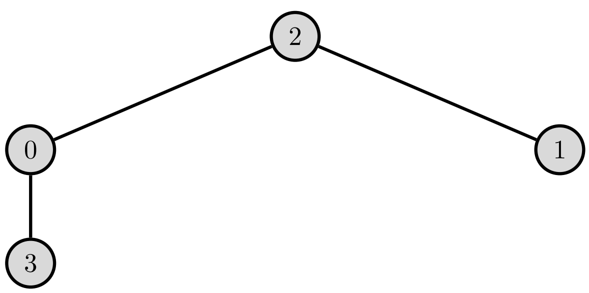


↓

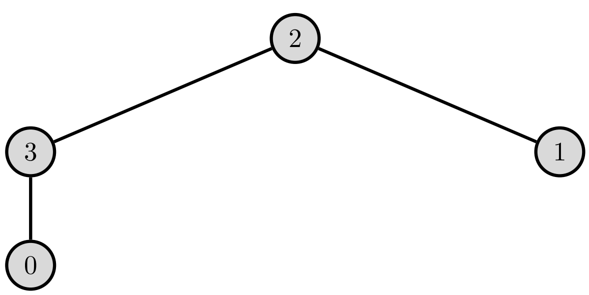


cc = 1+1=2; sc = 1+1=2

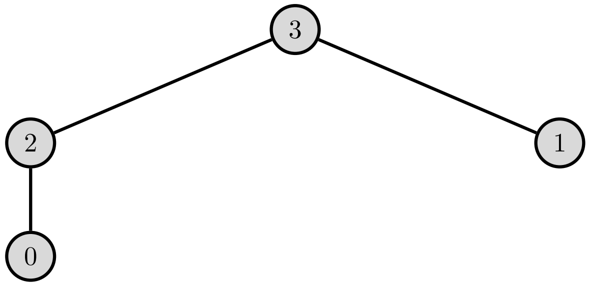
4. `i = 3; arr[i] = 3;`
   `i != 0; parent = 1`
   `arr[parent] < arr[i]; swap`
   `i != 0; parent = 0`
   `arr[parent] < arr[i]; swap`
   `i = parent = 0; return`



↓



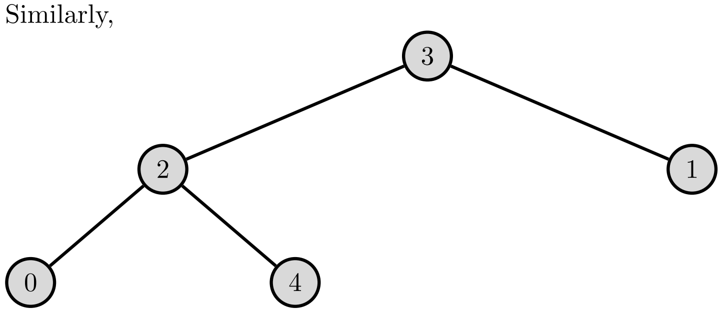
↓



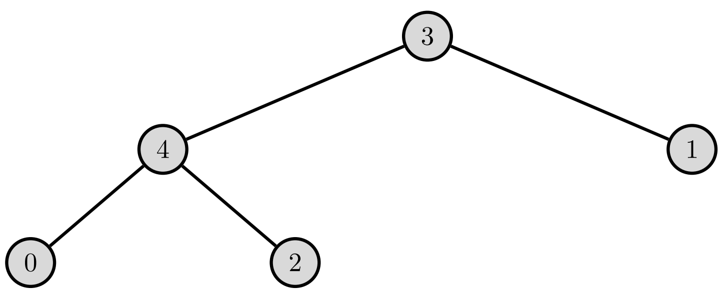
cc = 2+2=4; sc = 2+2=4

5. `i = 4`

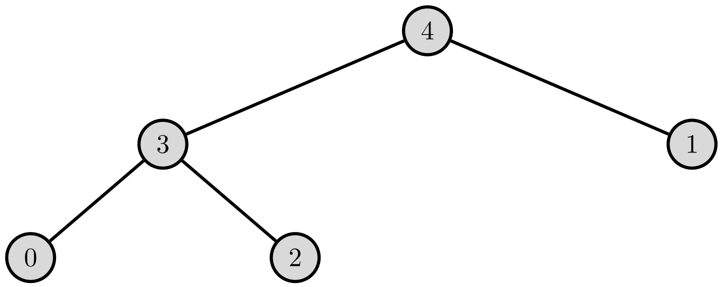
   Similarly,



↓



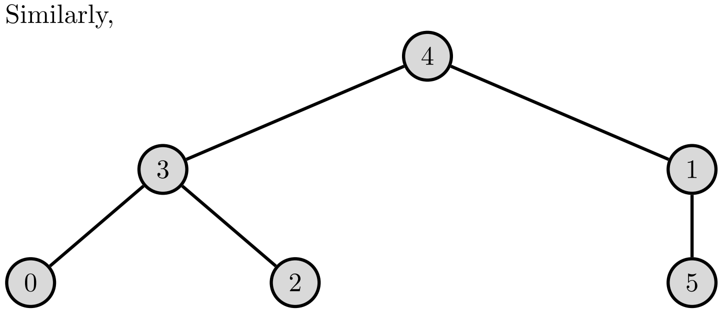
↓



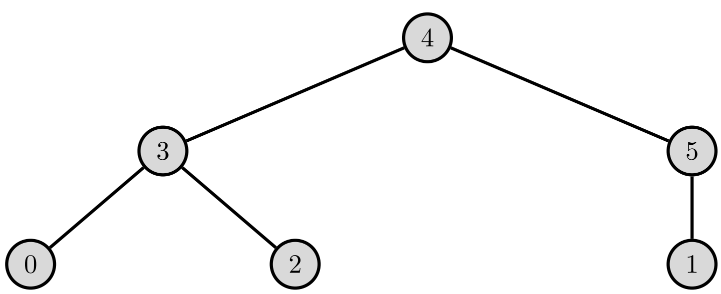
cc = 4+2=6; sc = 4+2=6

6. `i = 5`

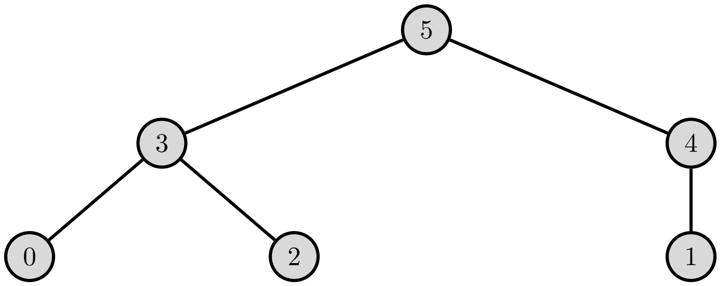
   Similarly,



↓



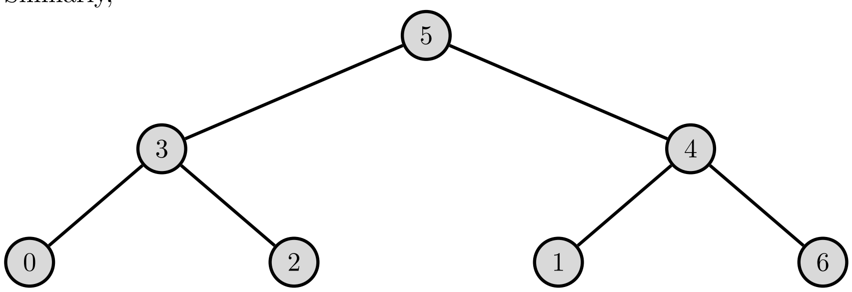
↓



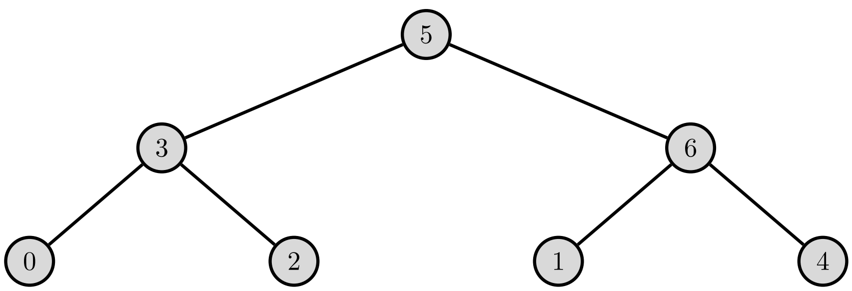
cc = 6+2=8; sc = 6+2=8

7. `i = 6`

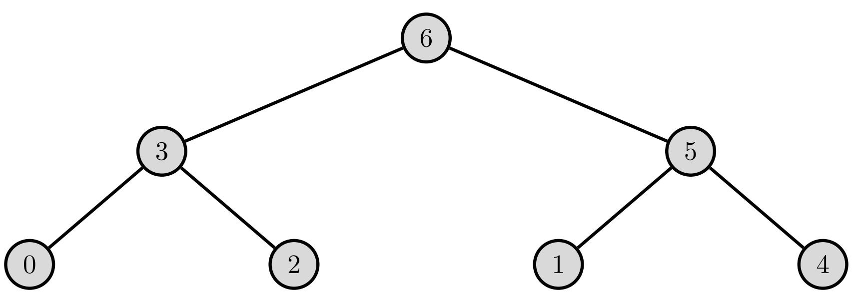
   Similarly,



$\downarrow$



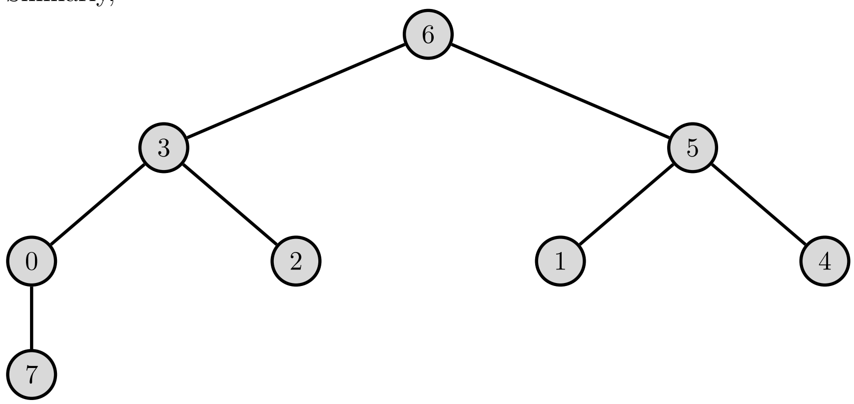
$\downarrow$



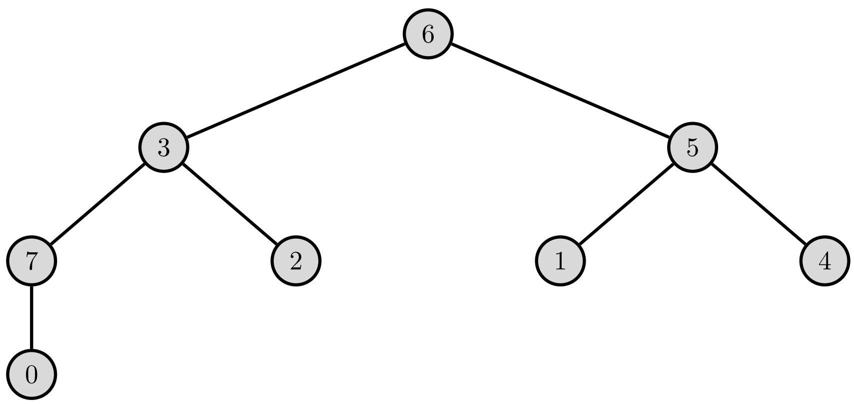
cc = 8+2=10; sc = 8+2=10

8. `i = 7`

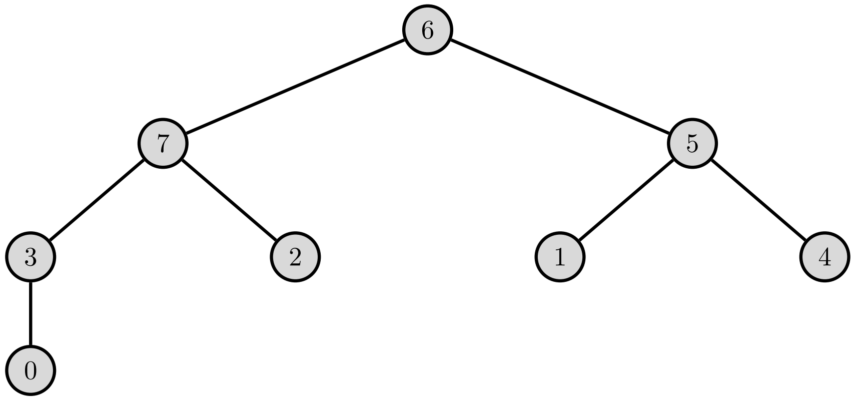
   Similarly,



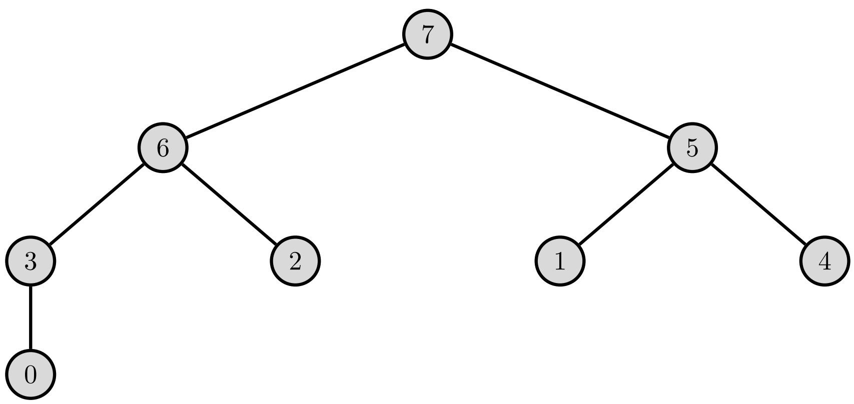
$\downarrow$



$\downarrow$



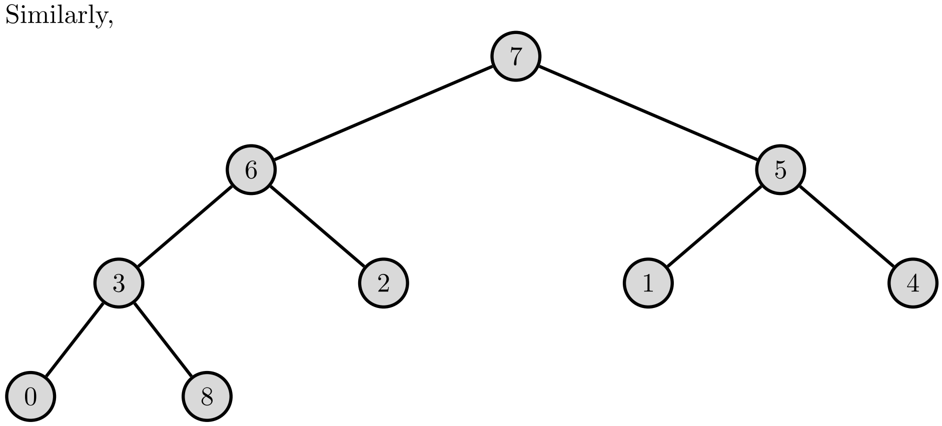
↓



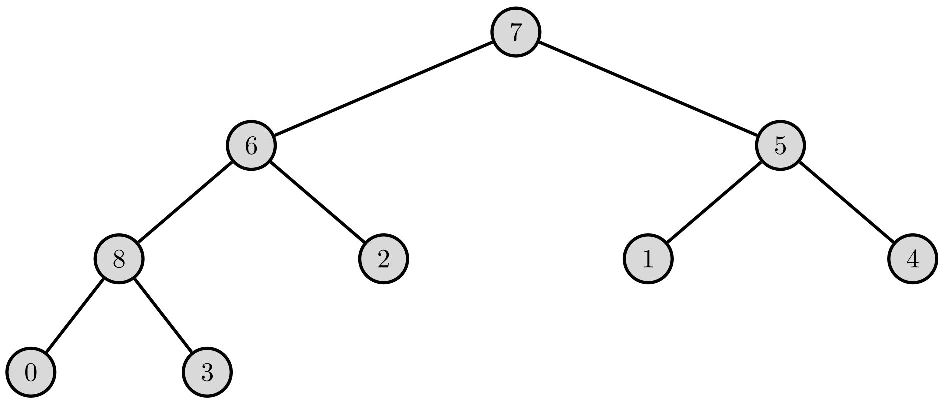
cc = 10+3=13; sc = 10+3=13

9. `i = 8`

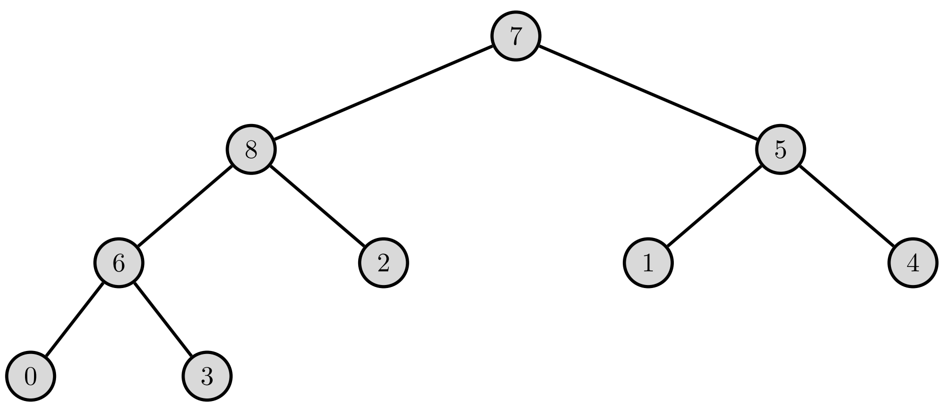
   Similarly,



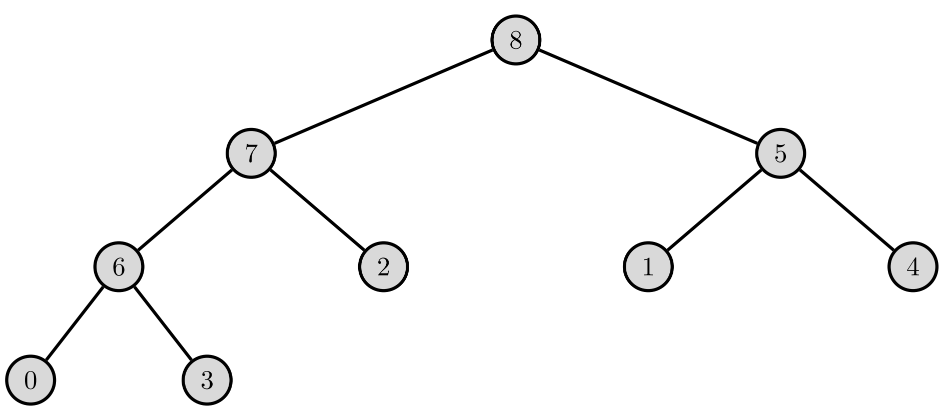
↓


7
6
5
8
2
1
4
0
3

↓


7
8
5
6
2
1
4
0
3

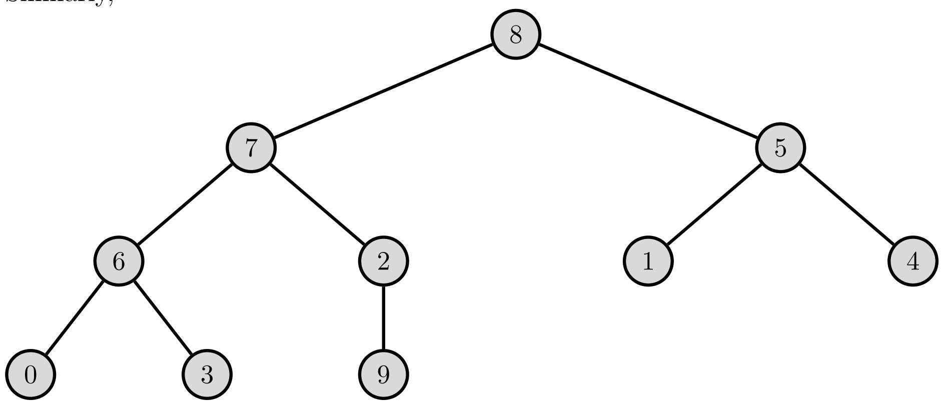
↓


8
7
5
6
2
1
4
0
3

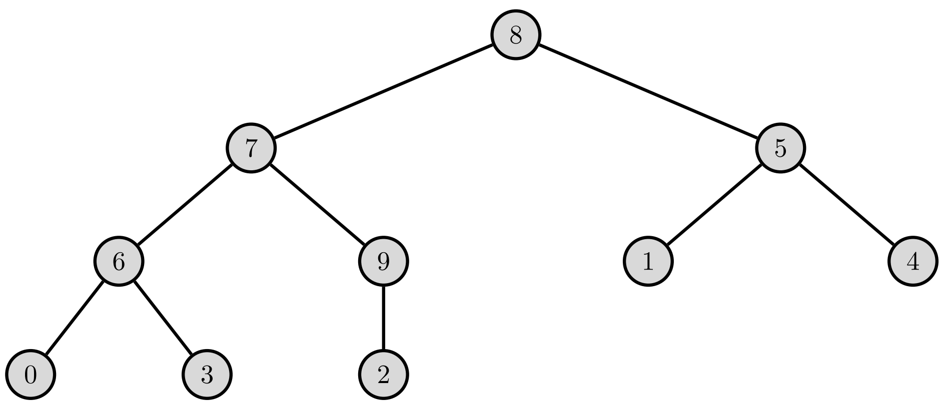
cc = 13+3=16; sc = 13+3=16

10. `i = 9`

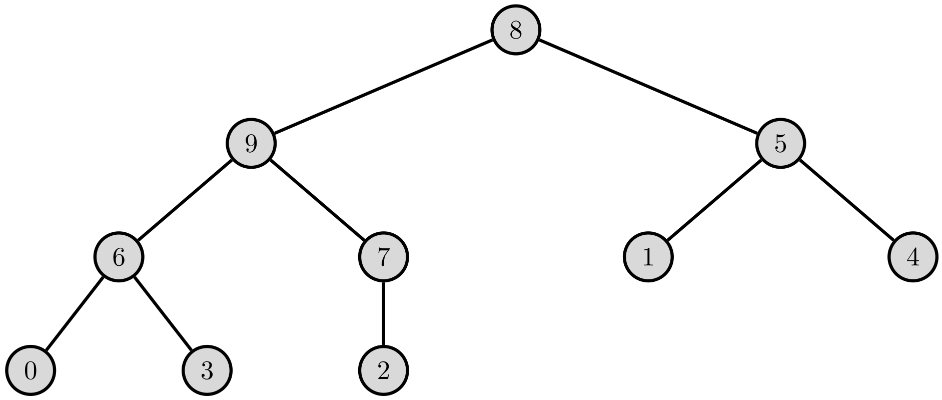
Similarly,



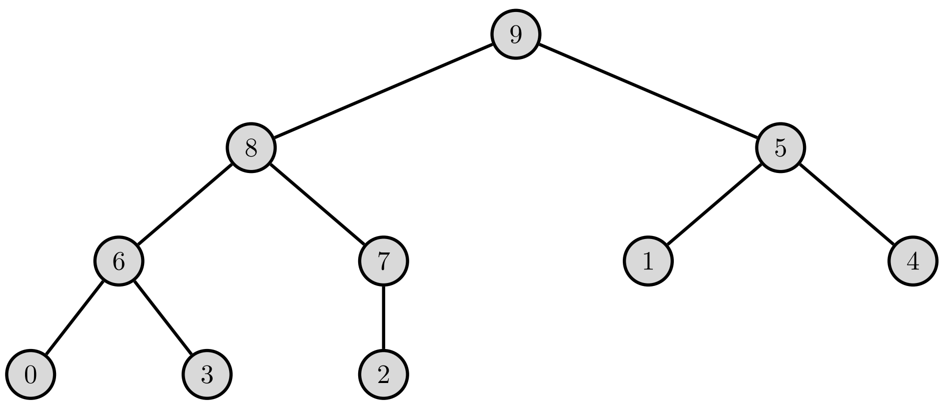
$\downarrow$



$\downarrow$



$\downarrow$



cc = 16+3=19; sc = 16+3=19

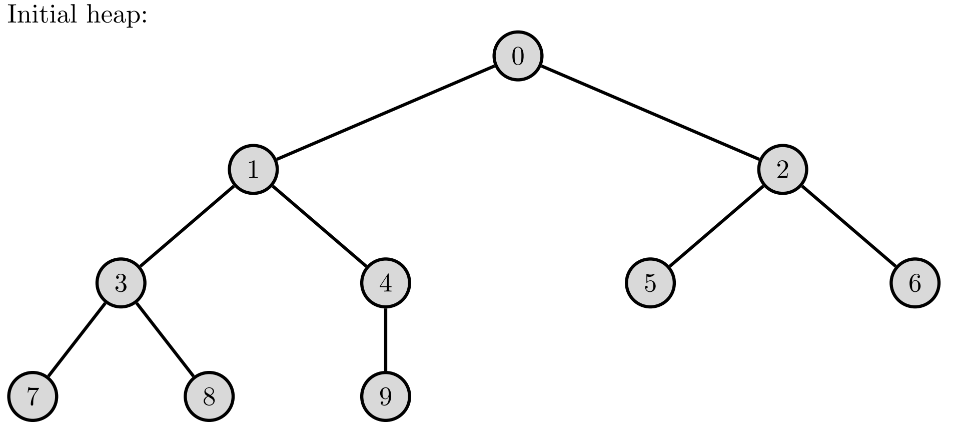
Heap Insert
Comparisons: 19
Swaps: 19
Since, after inserting each individual node, we have to go to the very top, thus, we have to traverse the height of the tree at each iteration.
Height of the tree = $O(\log n)$
Therefore, time complexity of heap insert = $O(n \log n)$

We can see that our experimental result holds as:
$n = 10$
$\log 10 \sim 3$
$10 \log 10 \sim 30$
$19 < 30$

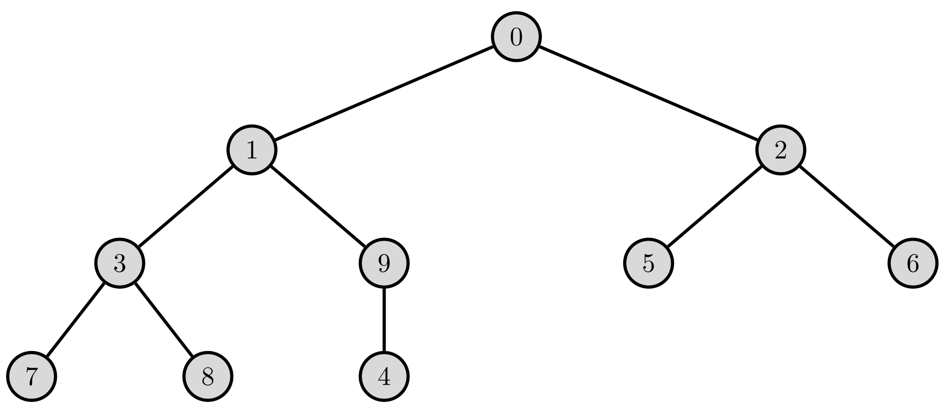
2 Heapify

`arr2[] = {0,1,2,3,4,5,6,7,8,9}`

Initial heap:

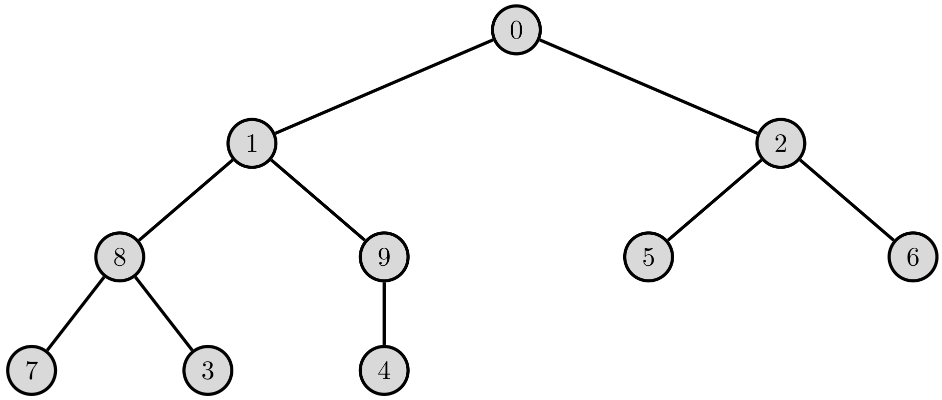


1. `i = 4; left = 9; right = 10; largest = 4; n = 10`
   `left < n; arr[left] > arr[largest]; largest = left = 9`
   `right = n`
   `largest != i; swap`
   `i = largest = 9; break`

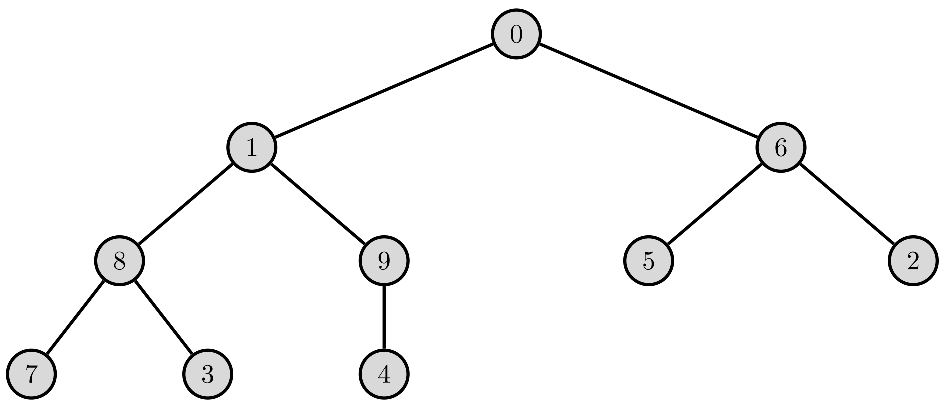
   cc = 1; sc = 1

2. `i = 3; left = 7; right = 8; largest = 3; n = 10`
   `left < n; arr[left] > arr[largest]; largest = left = 7`
   `right < n; arr[right] > arr[largest]; largest = right = 8`
   `largest != i; swap`
   `i = largest = 8; break`



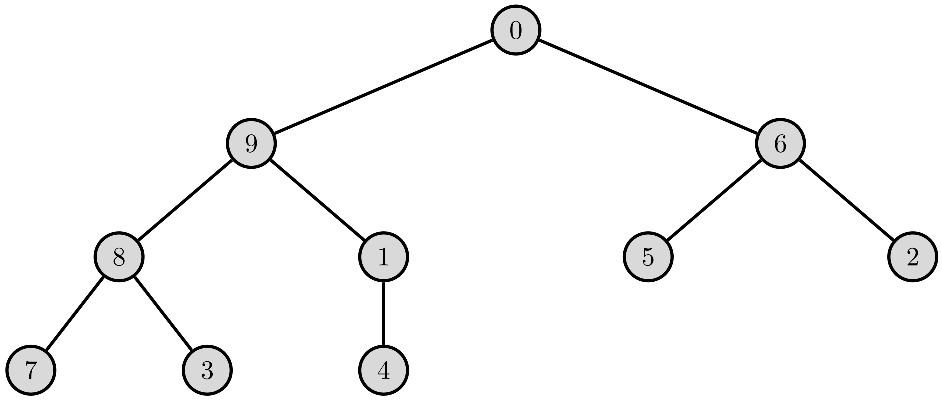
cc = 1+2=3; sc = 1+1=2

3.
```
i = 2; left = 5; right = 6; largest = 2; n = 10
left < n; arr[left] > arr[largest]; largest = left = 5
right < n; arr[right] > arr[largest]; largest = right = 6
largest != i; swap
i = largest = 6; break
```

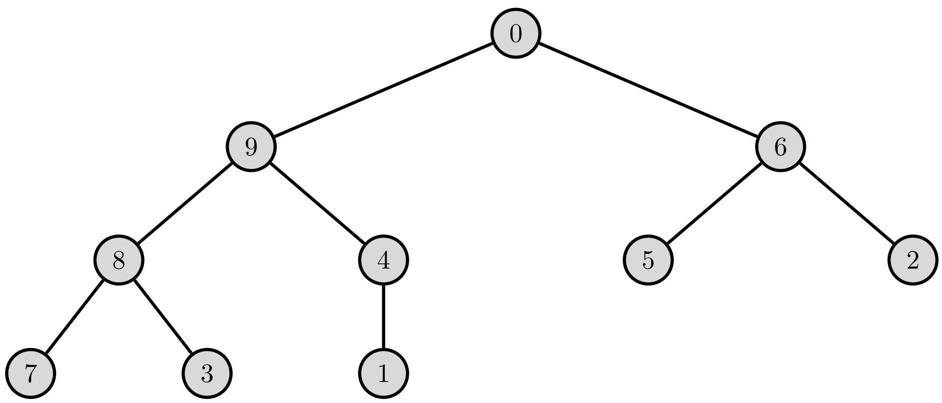


cc = 3+2=5; sc = 2+1=3

4.
```
i = 1; left = 3; right = 4; largest = 1; n = 10
left < n; arr[left] > arr[largest]; largest = left = 3
right < n; arr[right] > arr[largest]; largest = right = 4
largest != i; swap
i = largest = 4

i = 4; left = 9; right = 10; largest = 4; n = 10
left < n; arr[left] > arr[largest]; largest = left = 9
right = n
largest != i; swap
i = largest = 9; break
```



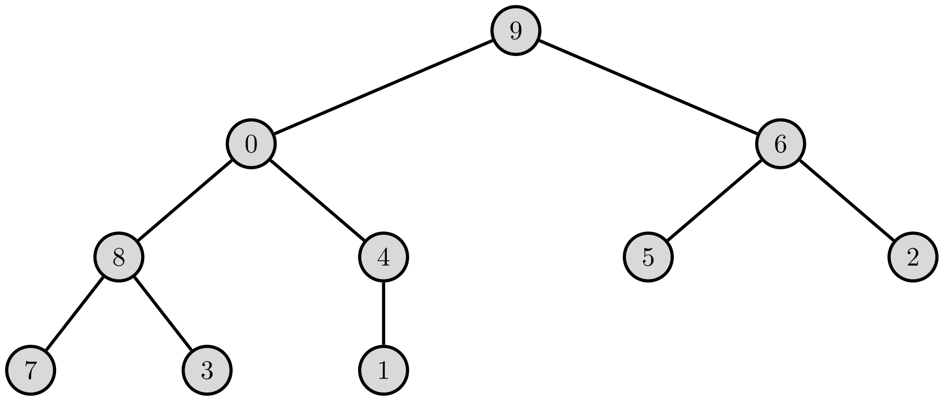
↓



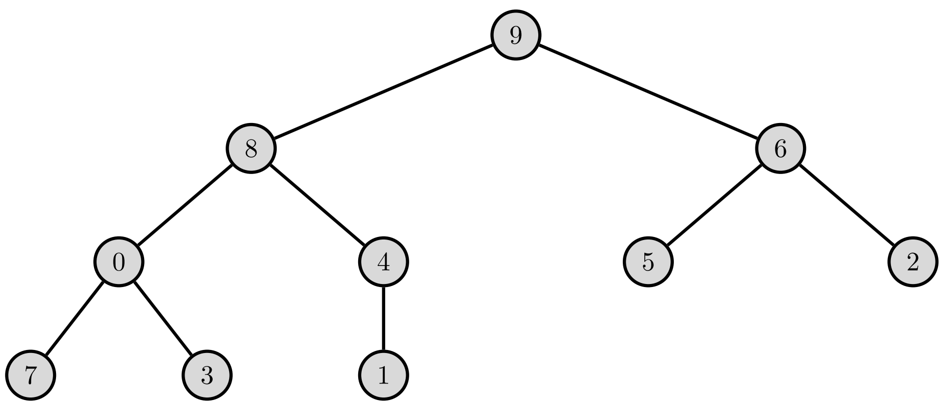
cc = 5+3=8; sc = 3+2=5

5.
```
i = 0; left = 1; right = 2; largest = 0; n = 10
left < n; arr[left] > arr[largest]; largest = left = 1
right < n; arr[right] < arr[largest]
largest != i; swap
i = largest = 1

i = 1; left = 3; right = 4; largest = 1; n = 10
left < n; arr[left] > arr[largest]; largest = left = 3
right < n; arr[right] < arr[largest]
largest != i; swap
i = largest = 3

i = 3; left = 7; right = 8; largest = 3; n = 10
left < n; arr[left] > arr[largest]; largest = left = 7
right < n; arr[left] < arr[largest]
largest != i; swap
```

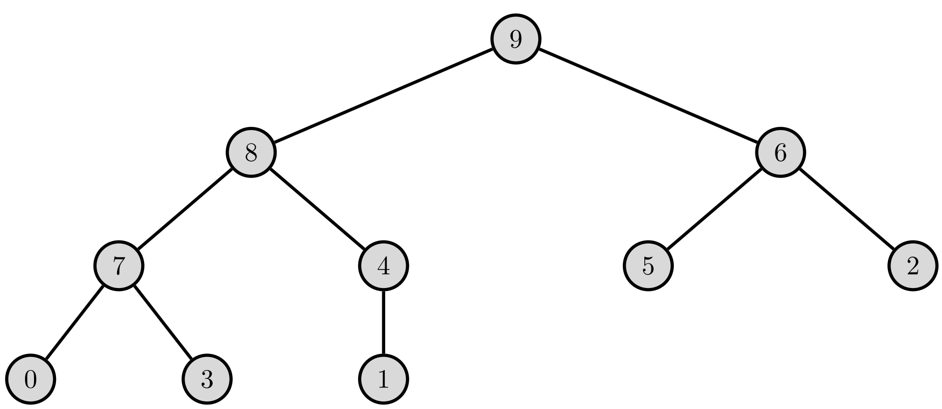
`i = largest = 7; break`



↓



↓



cc = 8+6=14; sc = 5+3=8

Heapify
Comparisons: 14
Swaps: 8
Since, we have to only check for $n/2$ nodes, and the number of swaps only increase as we go higher, we can show that the time complexity is $O(n)$.
We can see that our experimental result holds as:
$n = 10$
$8 < 10$