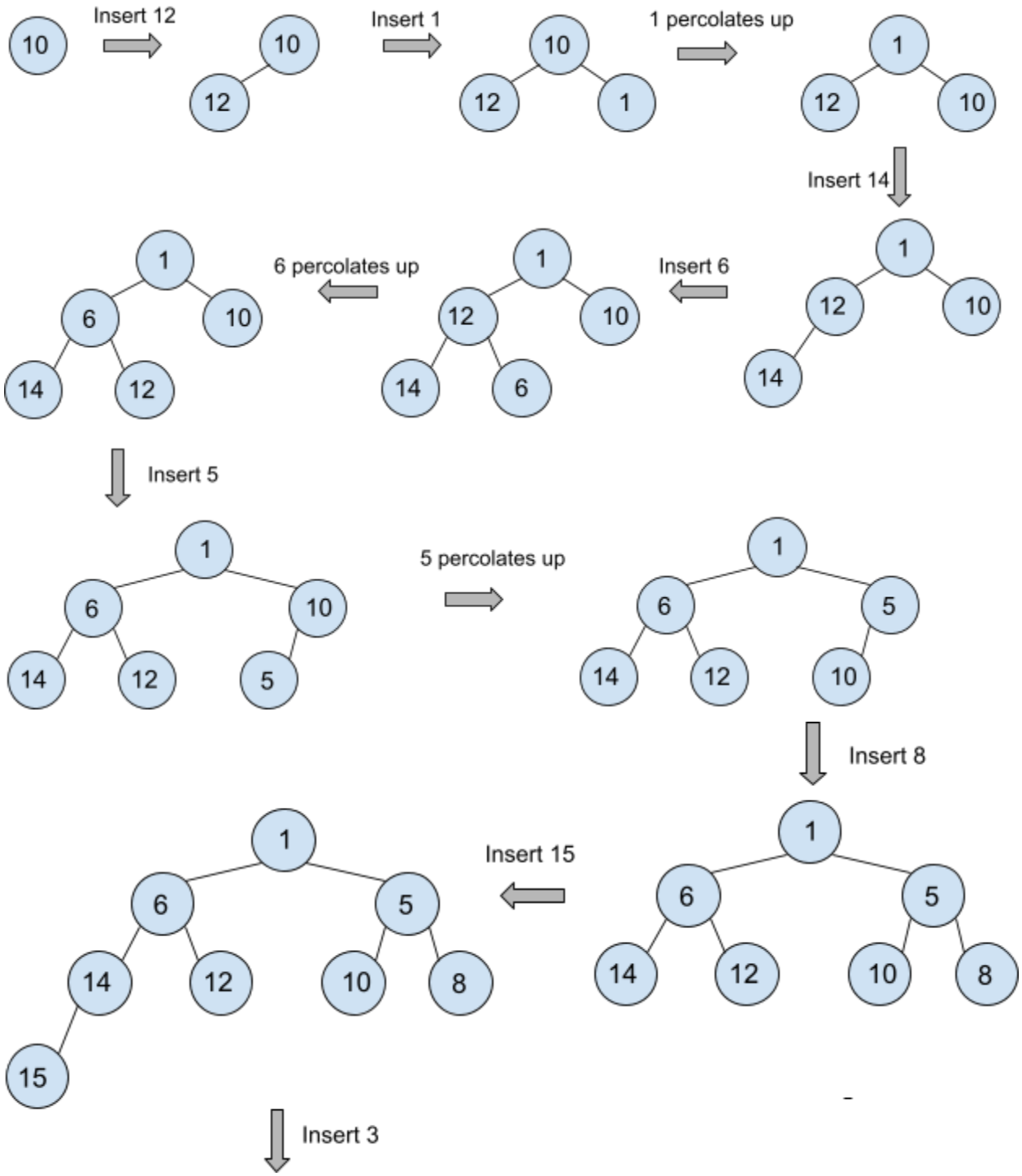
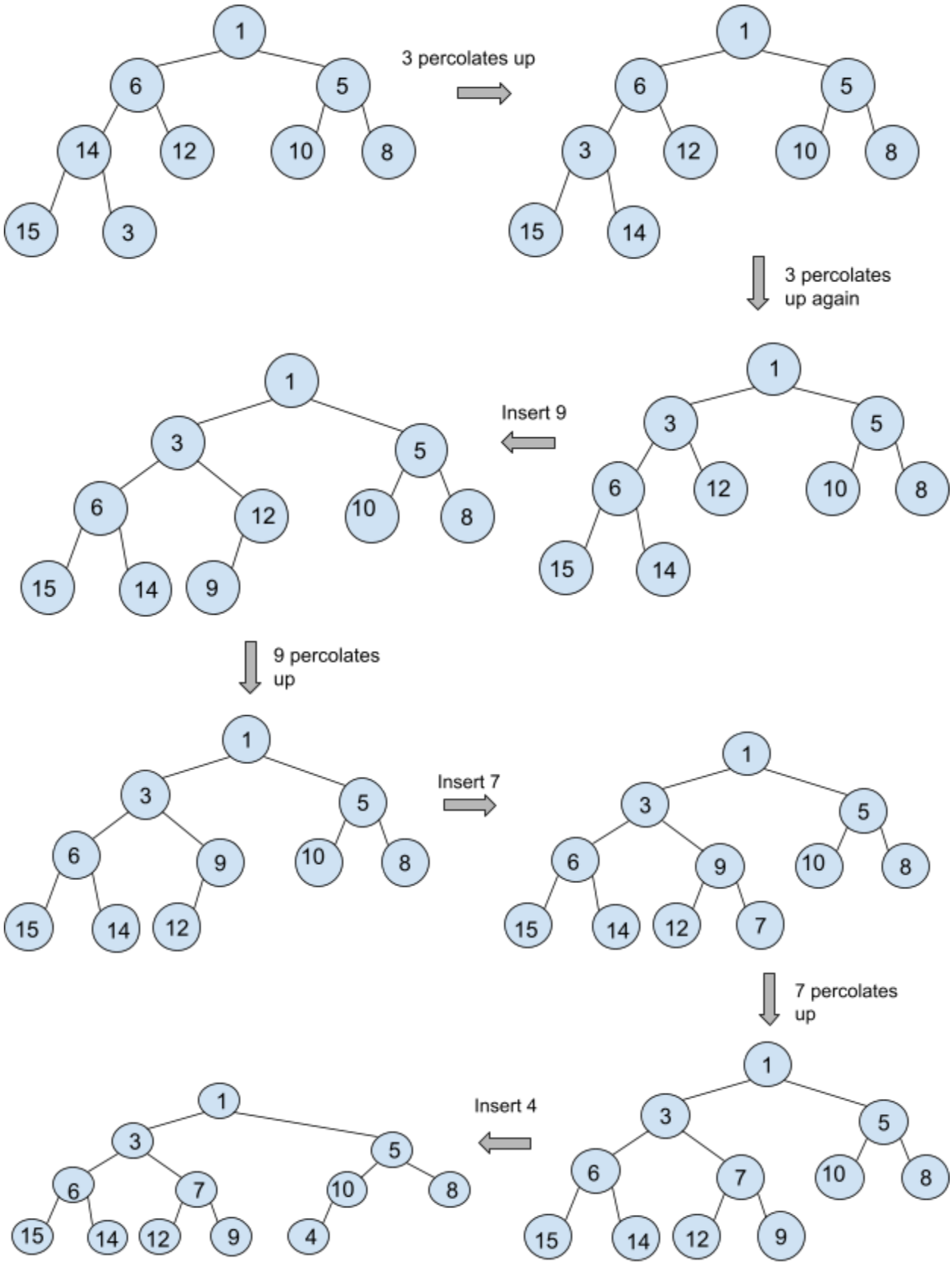
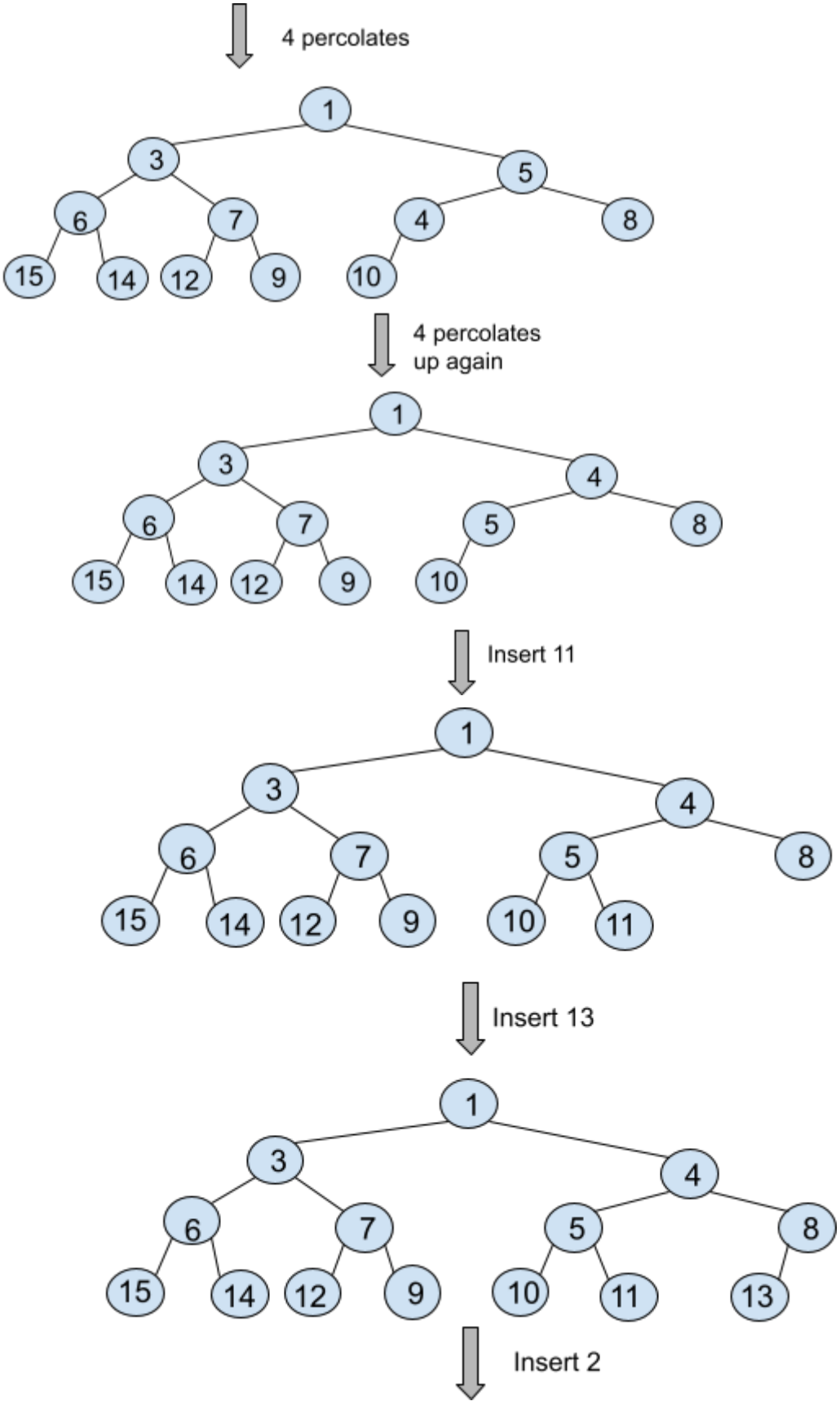
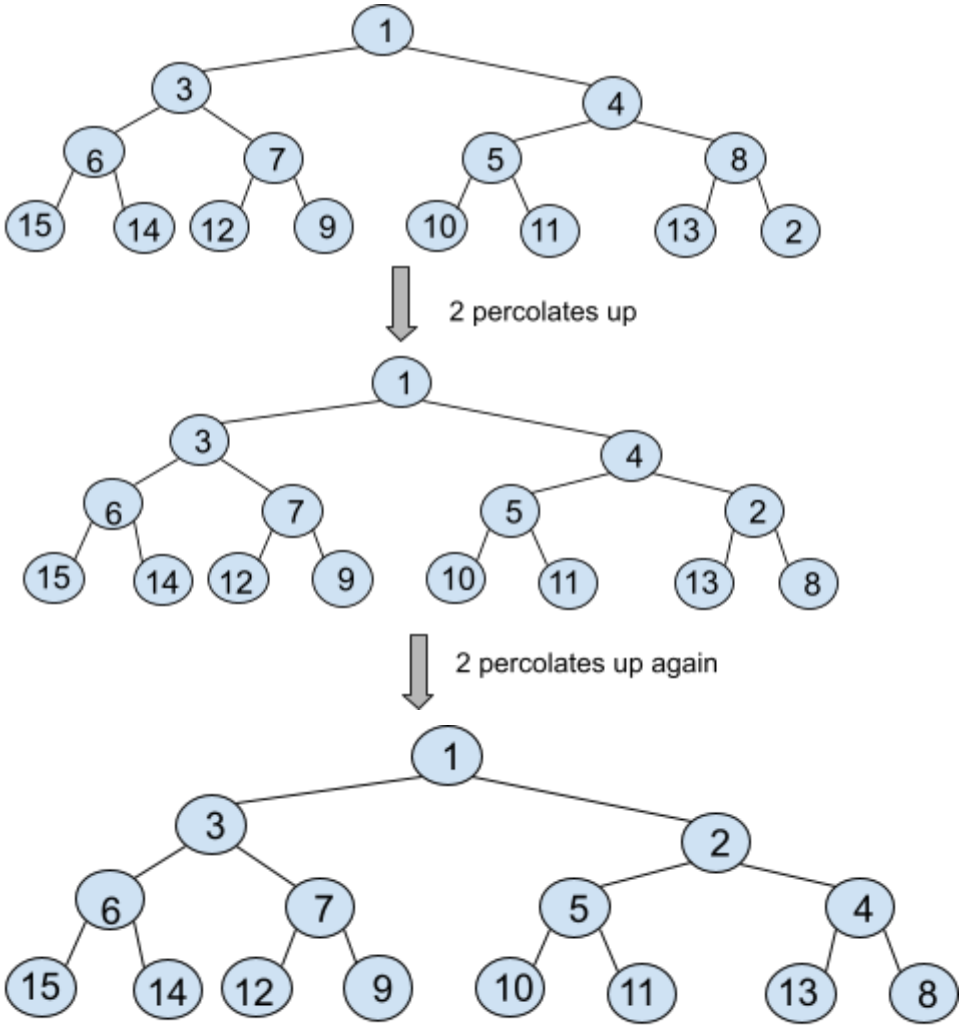
Problem 2: (45 pts)

A) Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, and 2 at one time and in the given order, into an initially empty **binary min heap**.

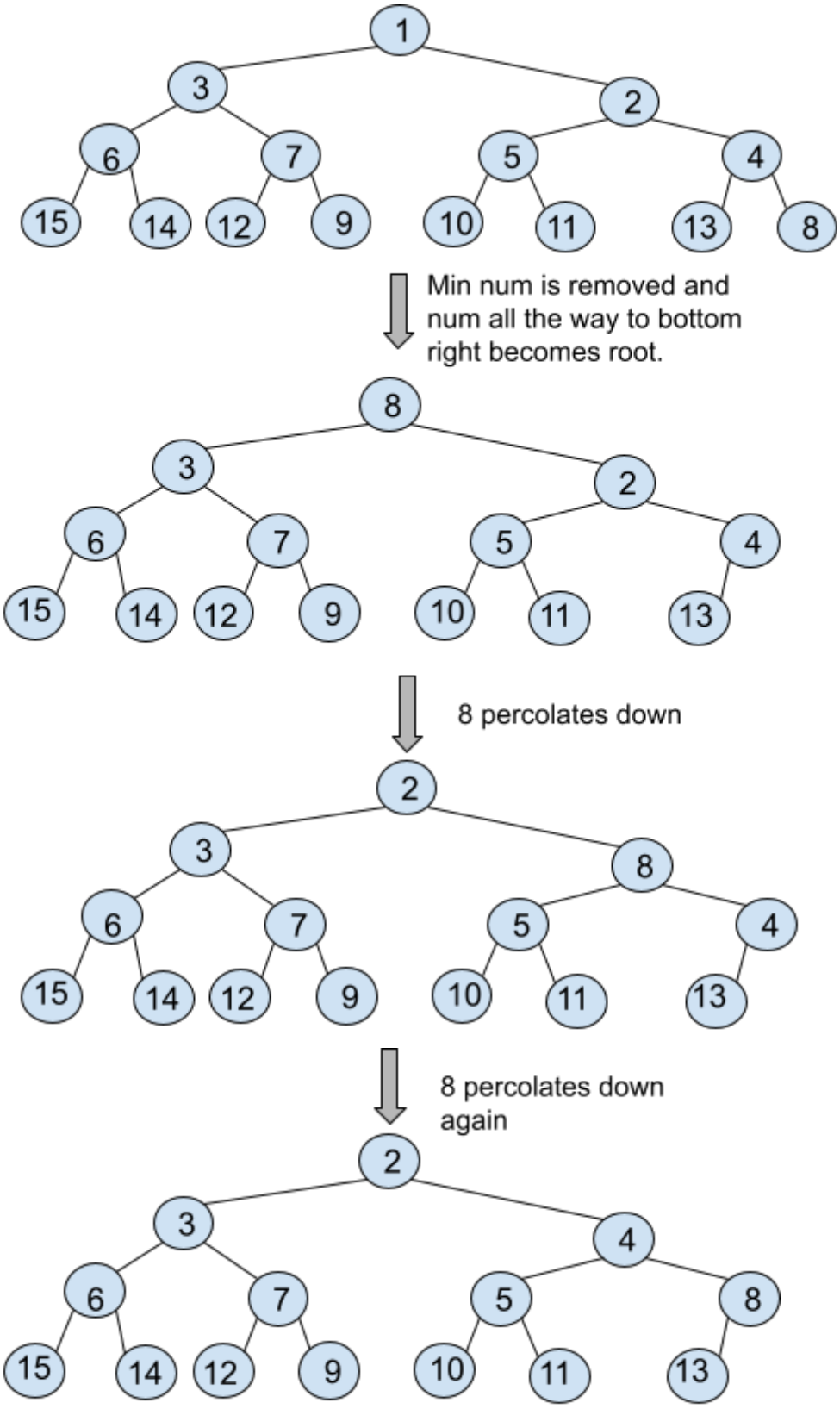


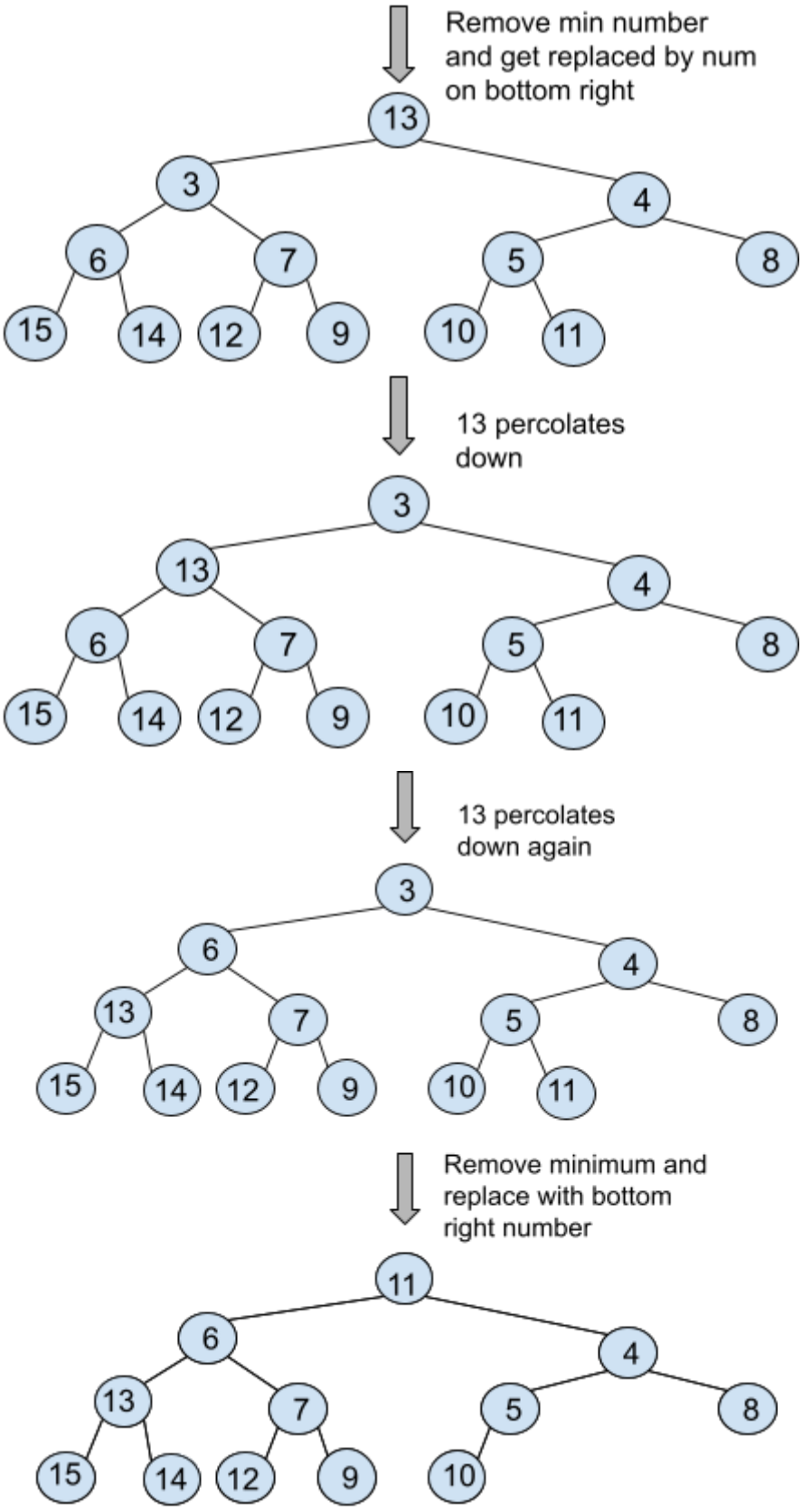

3 percolates up
3 percolates up again
Insert 9
9 percolates up
Insert 7
7 percolates up
Insert 4

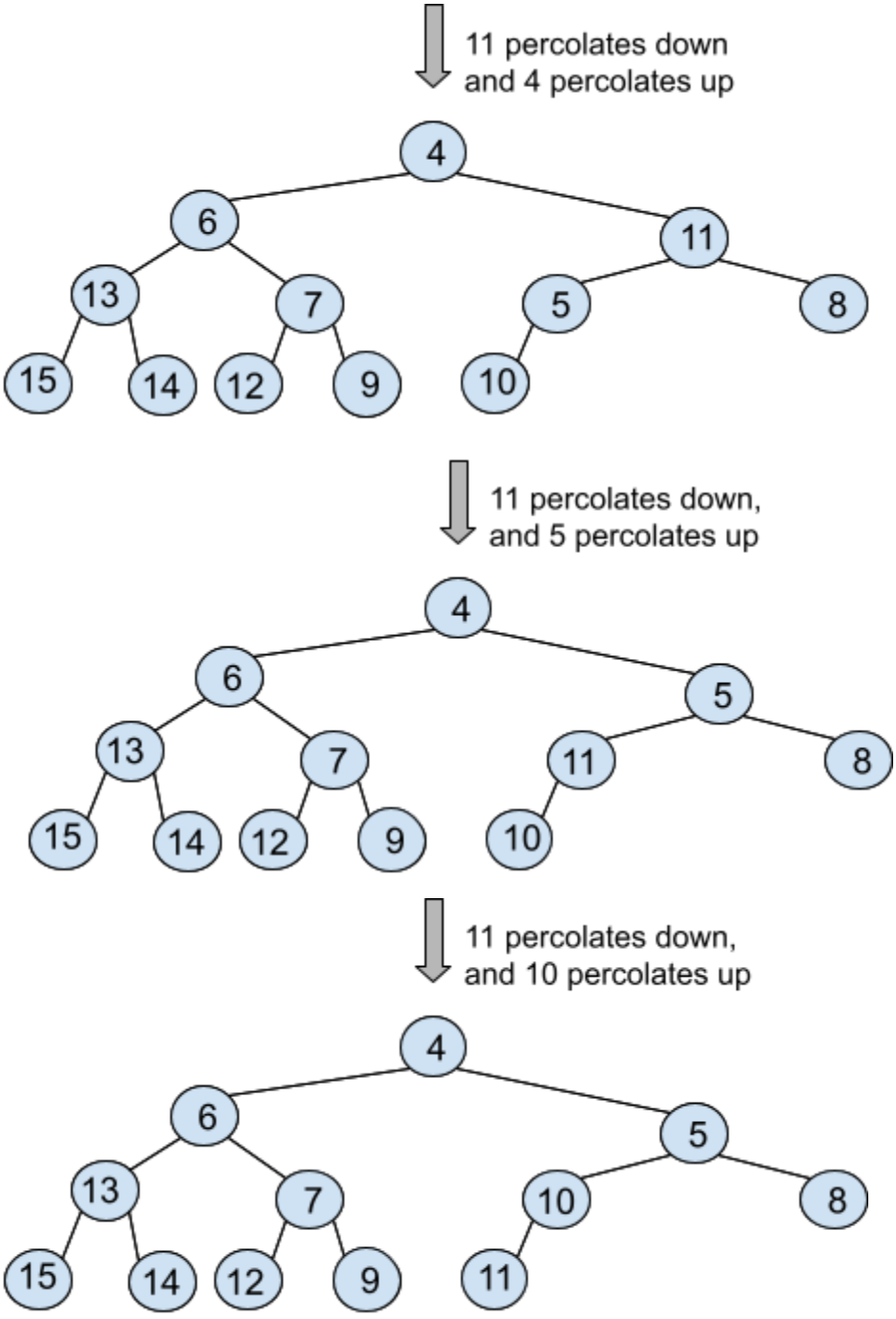

4 percolates
1
3
5
6
7
4
8
15
14
12
9
10
4 percolates
up again
1
3
4
6
7
5
8
15
14
12
9
10
Insert 11
1
3
4
6
7
5
8
15
14
12
9
10
11
Insert 13
1
3
4
6
7
5
8
15
14
12
9
10
11
13
Insert 2


1
3
4
6
7
5
8
15
14
12
9
10
11
13
2
2 percolates up
1
3
4
6
7
5
2
15
14
12
9
10
11
13
8
2 percolates up again
1
3
2
6
7
5
4
15
14
12
9
10
11
13
8

B) Show the result of performing three **deleteMin** operations in the heap of the previous **binary min heap.**



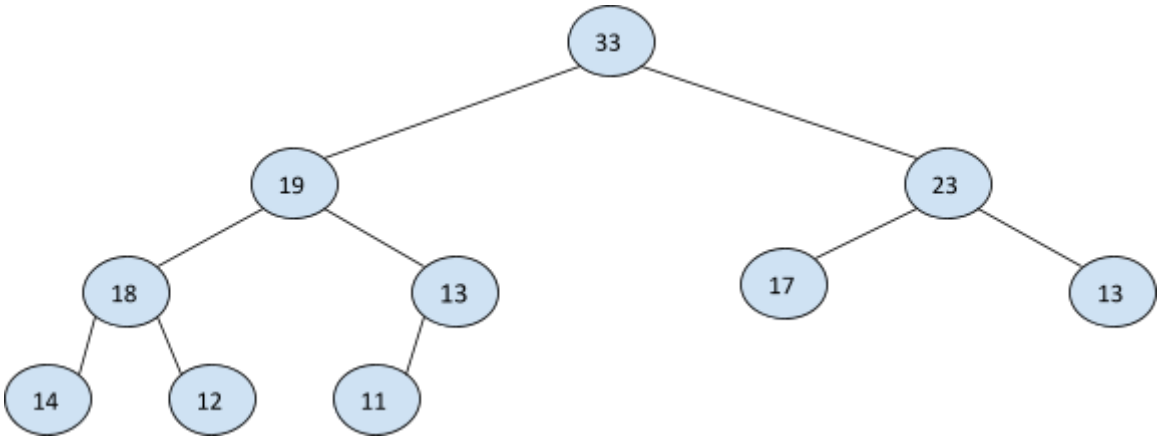

Remove min number
and get replaced by num
on bottom right
13
3
4
6
7
5
8
15
14
12
9
10
11
13 percolates
down
3
13
4
6
7
5
8
15
14
12
9
10
11
13 percolates
down again
3
6
4
13
7
5
8
15
14
12
9
10
11
Remove minimum and
replace with bottom
right number
11
6
4
13
7
5
8
15
14
12
9
10


11 percolates down
and 4 percolates up
4
6
11
13
7
5
8
15
14
12
9
10
11 percolates down,
and 5 percolates up
4
6
5
13
7
11
8
15
14
12
9
10
11 percolates down,
and 10 percolates up
4
6
5
13
7
10
8
15
14
12
9
11

D) What is the running time complexity of your replace key method?

- Worst case scenario is O(n) because you're looking for a key using a for loop iterating 1 by one and you don't know where the key is.

E) Consider an initially empty **max-heap**, where the following keys are to be inserted one at a time: 11, 19, 23, 12, 13, 14, 18, and 33. Draw a **tree** that results after building this **max-heap**.



F) Is it possible to find the maximum in a min heap in O(log(n)) time? Justify.

- It is not possible to find the maximum in a minimum heap at O(log(n)) time. Since the maximum number is stored at the leaf node, we would need to check each leaf node in the worst case scenario making it an O(n) time complexity.