Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Write the asymptotic worst-case running time of

finding Min and finding Max

operation performed on a Binary Search Tree.

COMP15: Data Structures

Week 8, Summer 2019

Admin

T7: **diff, grep, clear**

Due by 6pm on Wednesday, July 17

(a quick demo)

P4: **Course Registration System**

Project Due by 6pm on Sunday, July 21

(describing scenarios)

Questions about P4?

Feedbacks on Midterm from Matt

(small talks)

Bogosort
Sleep sort

Trees (cont.)

Tree:

- Terminologies (Week 6)
- Traversals (Week 6)

- Operations (Week 7)
- Rotations (Week 8)

Binary Search Tree (BST)

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

```hpp
//BinarySearchTree.hpp
#ifndef BINARYSEARCHTREE_HPP
#define BINARYSEARCHTREE_HPP

#include "BSTNode.hpp"

template<typename K, typename V>
class BinarySearchTree{
public:
  BinarySearchTree();
  BinarySearchTree(const BinarySearchTree& other);
  BinarySearchTree& operator=(const BinarySearchTree& other);
  ~BinarySearchTree();

  void insert(K key, V value);
  void deleteBy(K key);
  V search(K key) const;

  bool isEmpty() const;

private:
  BSTNode<K, V>* root;
};

#endif
```

```hpp
//BSTNode.hpp
#ifndef BSTNODE_HPP
#define BSTNODE_HPP

template<typename K, typename V>
class BSTNode{
public:
  //BSTNode();
  BSTNode(K key, V value);
  //copy constructor
  //assignment operator
  //~BSTNode();

  V getData() const;
  void setLeft(BSTNode<K, V>* left);
  BSTNode<K, V>* getLeft() const;
  void setRight(BSTNode<K, V>* right);
  BSTNode<K, V>* getRight() const;

private:
  K key;
  V value;
  BSTNode<K, V>* left;
  BSTNode<K, V>* right;
};

#endif
```

Asymptotic running time of

search(), insert(), delete()?

$O(h)$ where h is the height to the tree

Height of Balanced Binary (Search) Tree

Tree of height 0 has 2^0 = 1 node. => $2^{0+1} - 1$

Height of Balanced Binary (Search) Tree

Tree of height 0 has 2^0 = 1 node. => $2^{0+1} - 1$

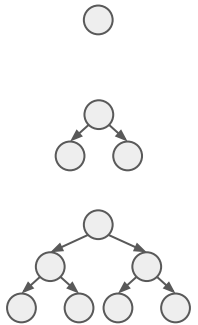
Tree of height 1 has $2^0 + 2^1$ = 3 nodes. => $2^{1+1} - 1$

Height of Balanced Binary (Search) Tree

Tree of height 0 has 2^0 = 1 node. => $2^{0+1} - 1$

Tree of height 1 has $2^0 + 2^1$ = 3 nodes. => $2^{1+1} - 1$

Tree of height 2 has $2^0 + 2^1 + 2^2$ = 7 nodes. => $2^{2+1} - 1$



Height of Balanced Binary (Search) Tree

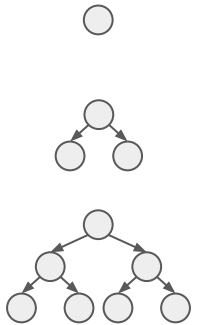
Tree of height 0 has 2^0 = 1 node. => $2^{0+1} - 1$

Tree of height 1 has $2^0 + 2^1$ = 3 nodes. => $2^{1+1} - 1$

Tree of height 2 has $2^0 + 2^1 + 2^2$ = 7 nodes. => $2^{2+1} - 1$

...

Tree of height h has $2^0 + 2^1 + 2^2 + \ldots + 2^{h-1} + 2^h$ = ? nodes. => $2^{h+1} - 1$

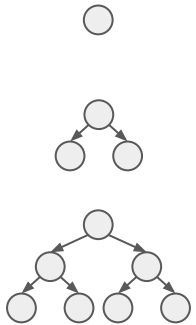


Height of Balanced Binary (Search) Tree

Tree of height 0 has 2^0 = 1 node. => $2^{0+1} - 1$

Tree of height 1 has $2^0 + 2^1$ = 3 nodes. => $2^{1+1} - 1$

Tree of height 2 has $2^0 + 2^1 + 2^2$ = 7 nodes. => $2^{2+1} - 1$



...

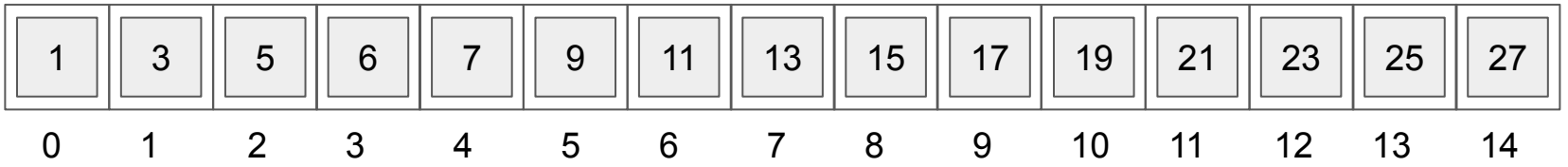
Tree of height h has $2^0 + 2^1 + 2^2 + ... + 2^{h-1} + 2^h$ = ? nodes. => $2^{h+1} - 1$

So $2^{h+1} - 1 = n$, $h = log_2(n+1) - 1$, then $h = O(log_2(n))$

Binary Search

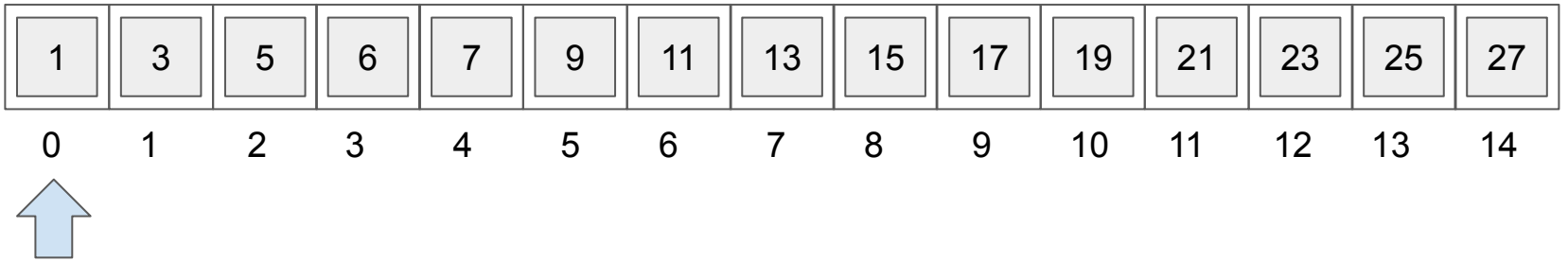
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



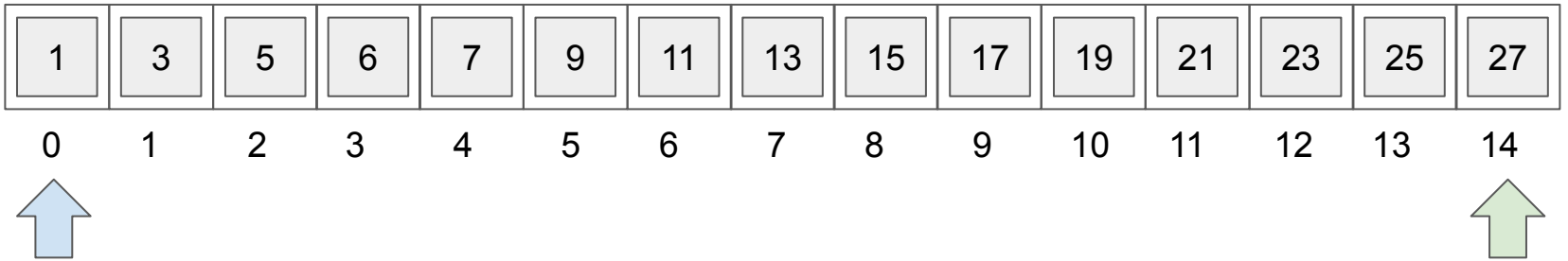
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



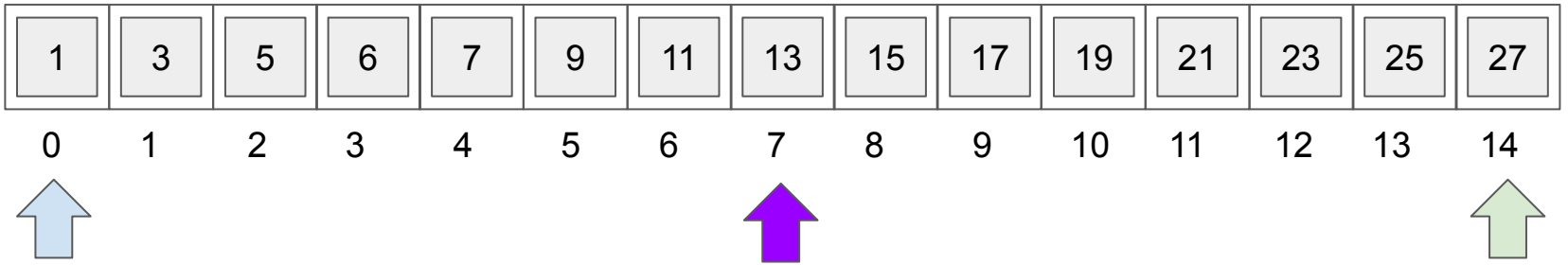
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



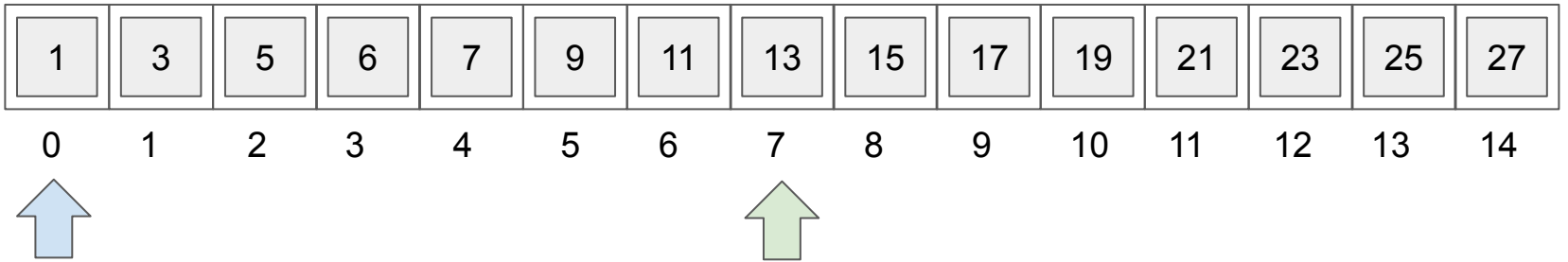
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



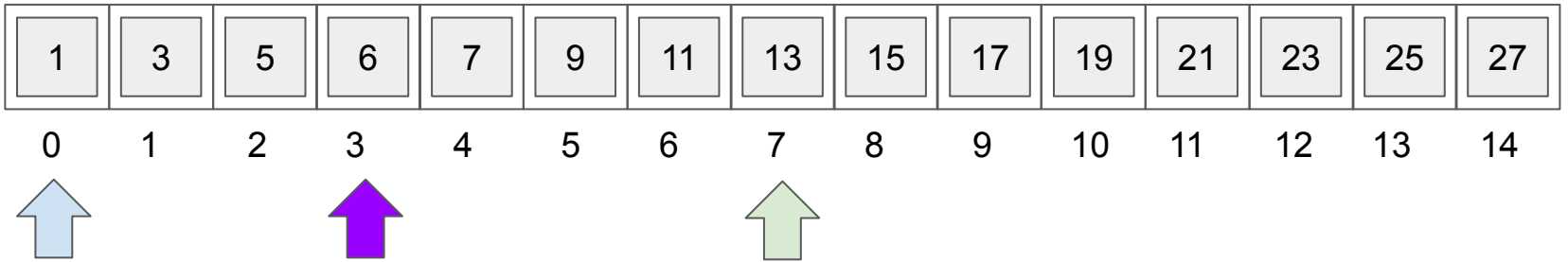
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



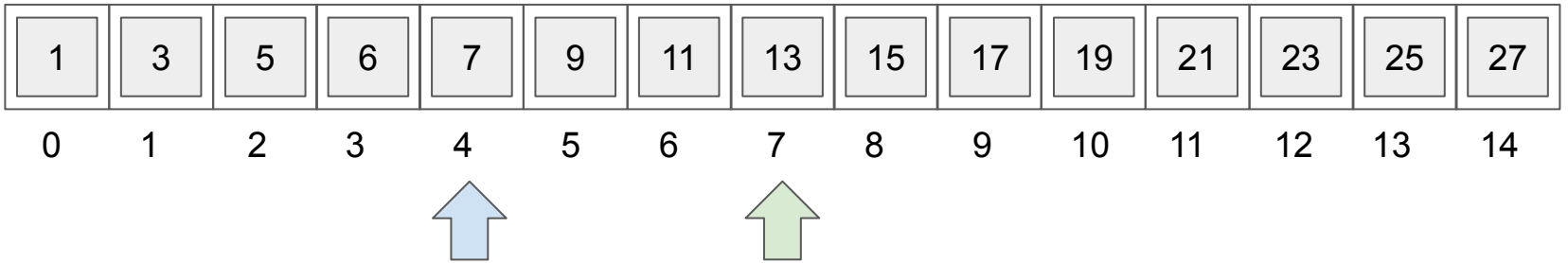
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



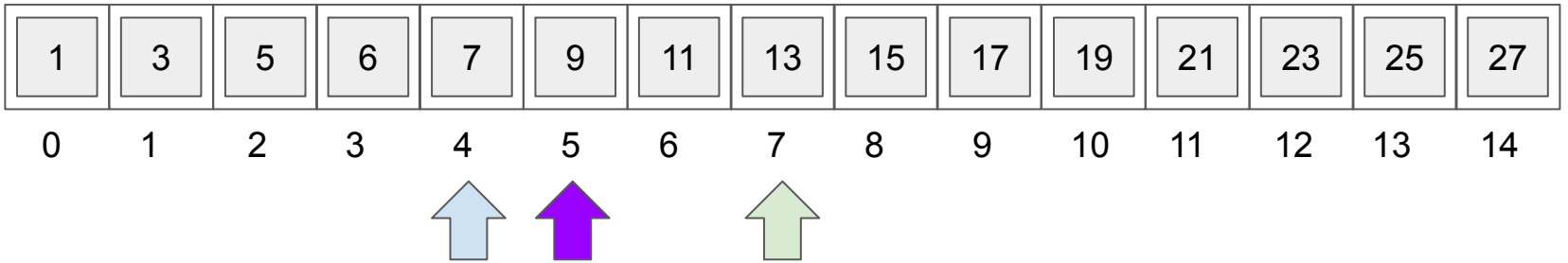
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



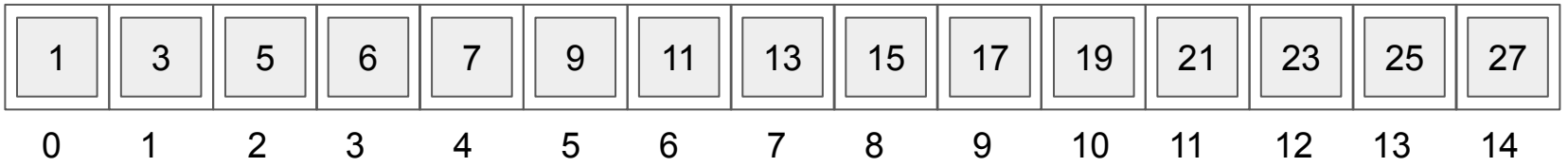
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(9)



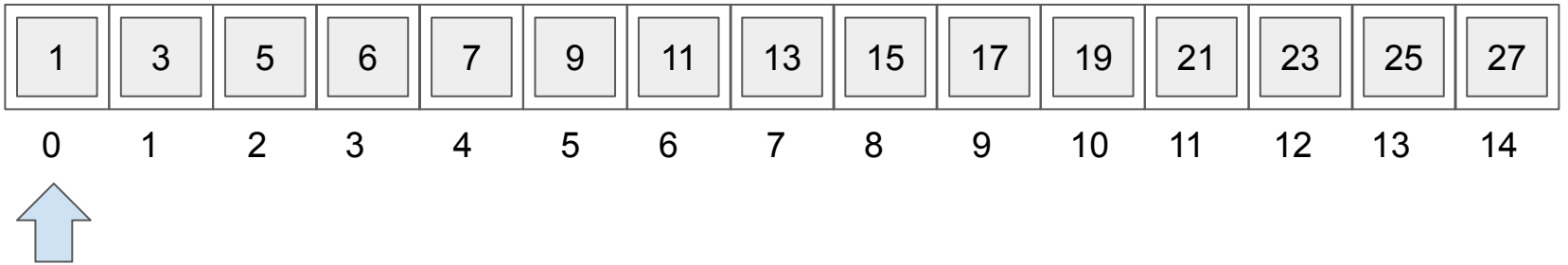
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



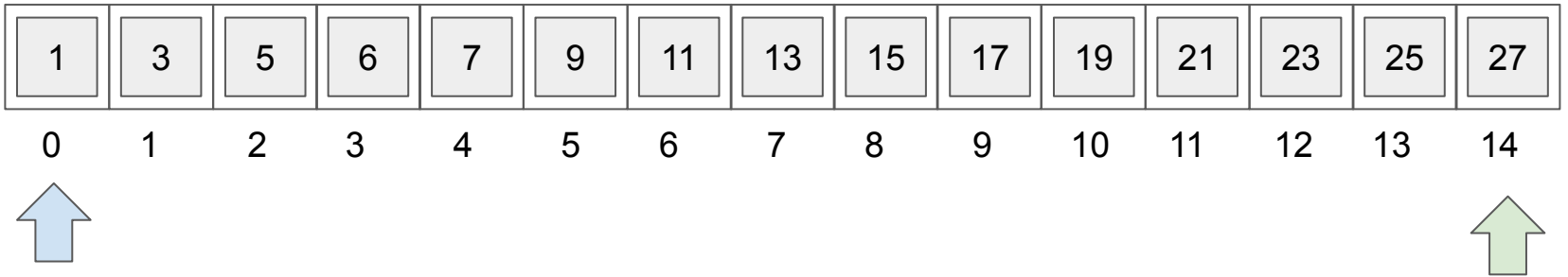
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



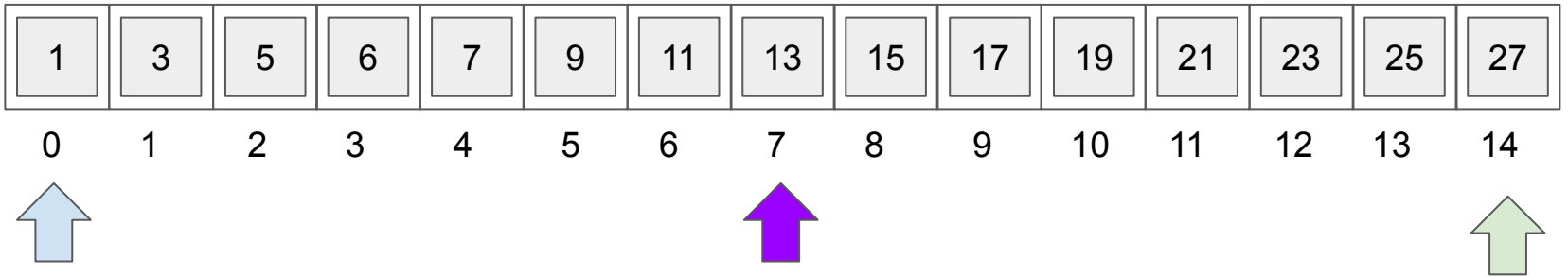
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



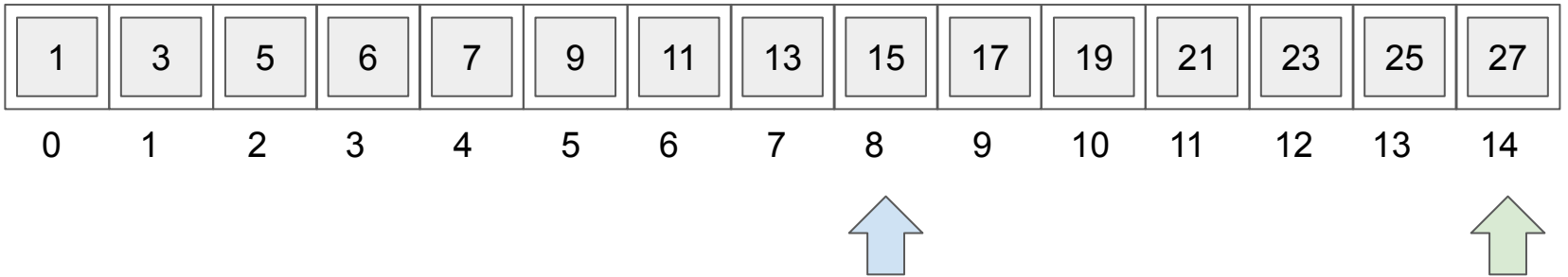
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



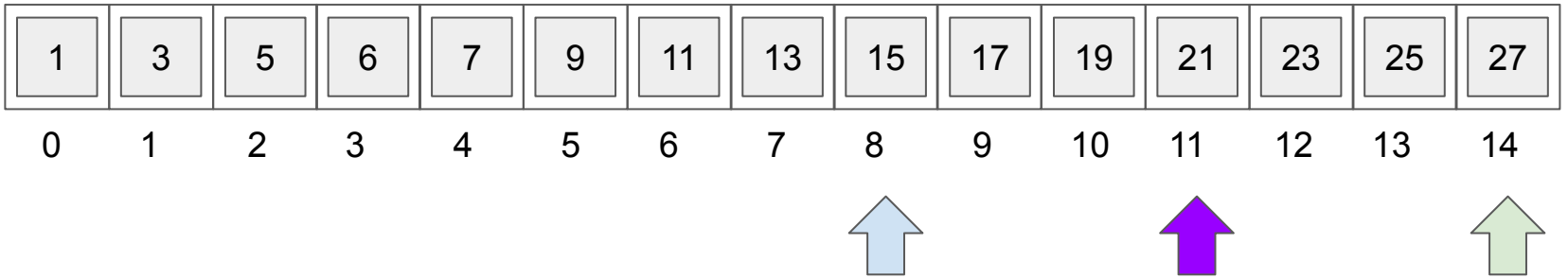
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



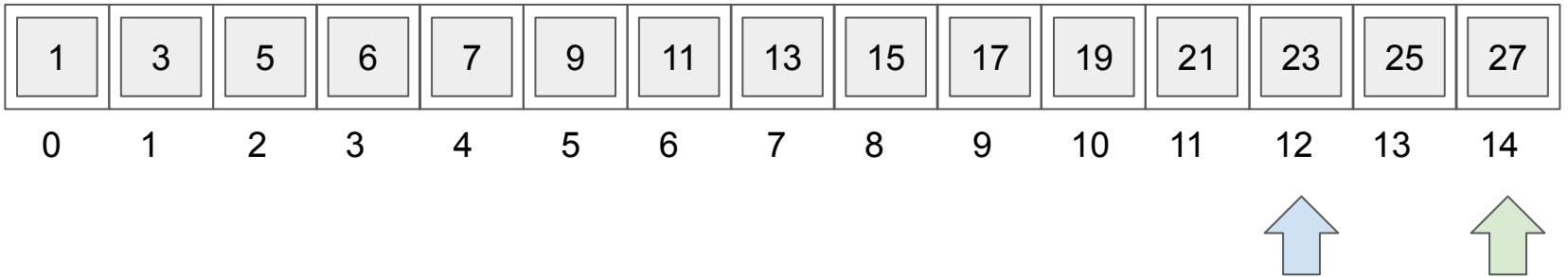
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



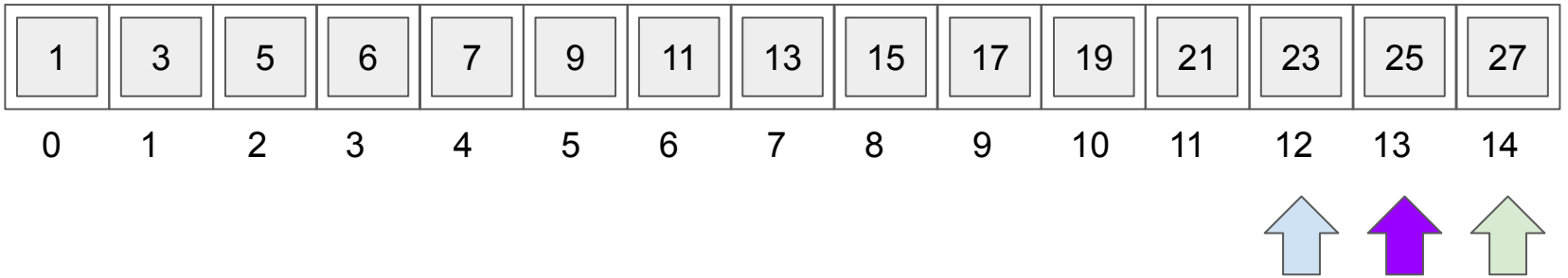
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



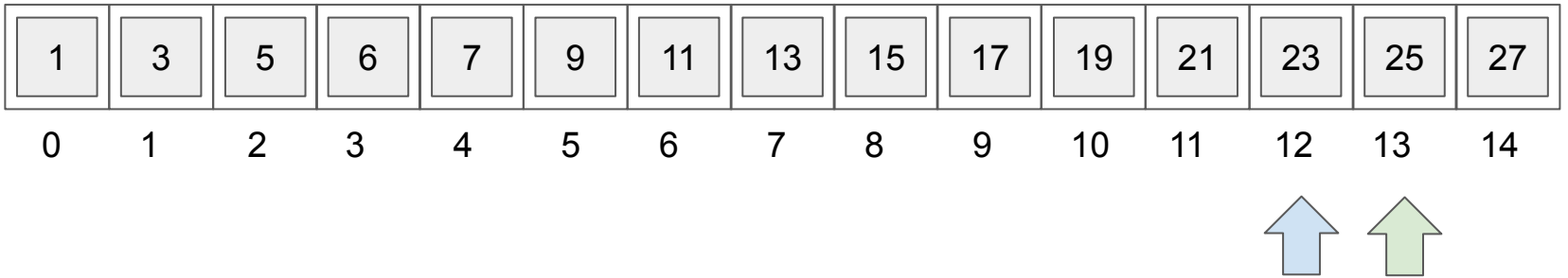
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



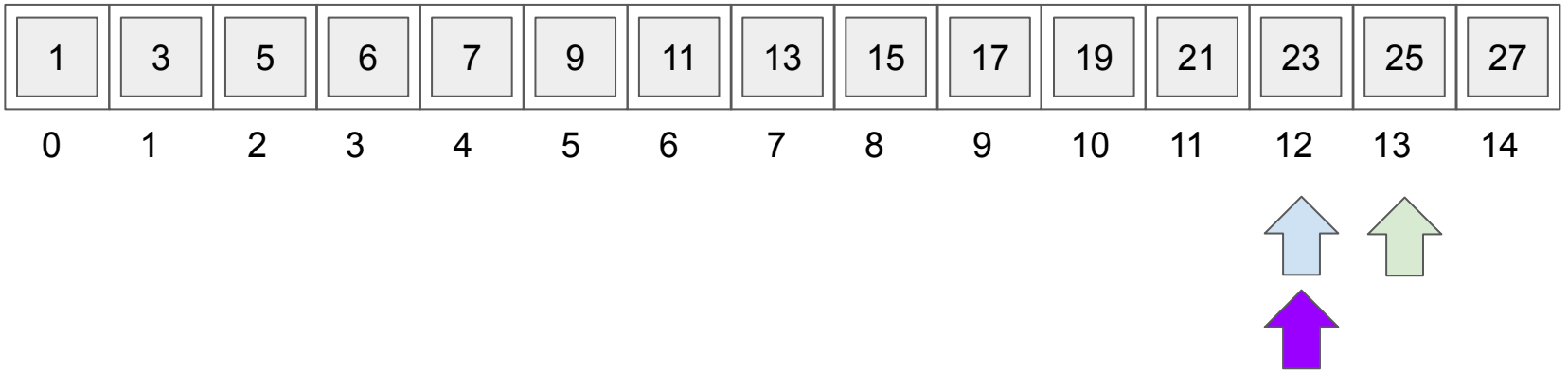
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



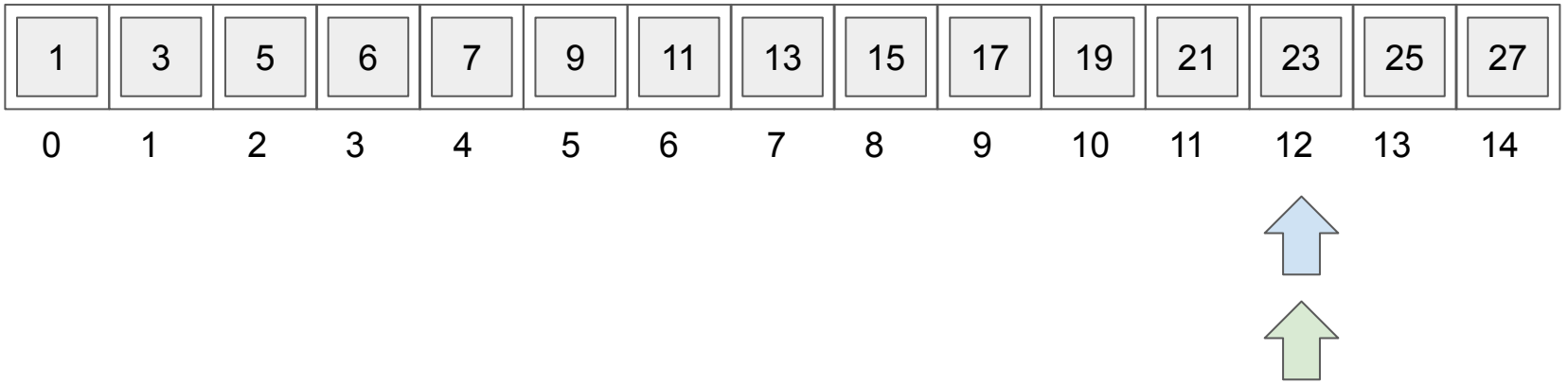
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

search(22)



(Node: We also discussed when we can perform binary search on an array.)

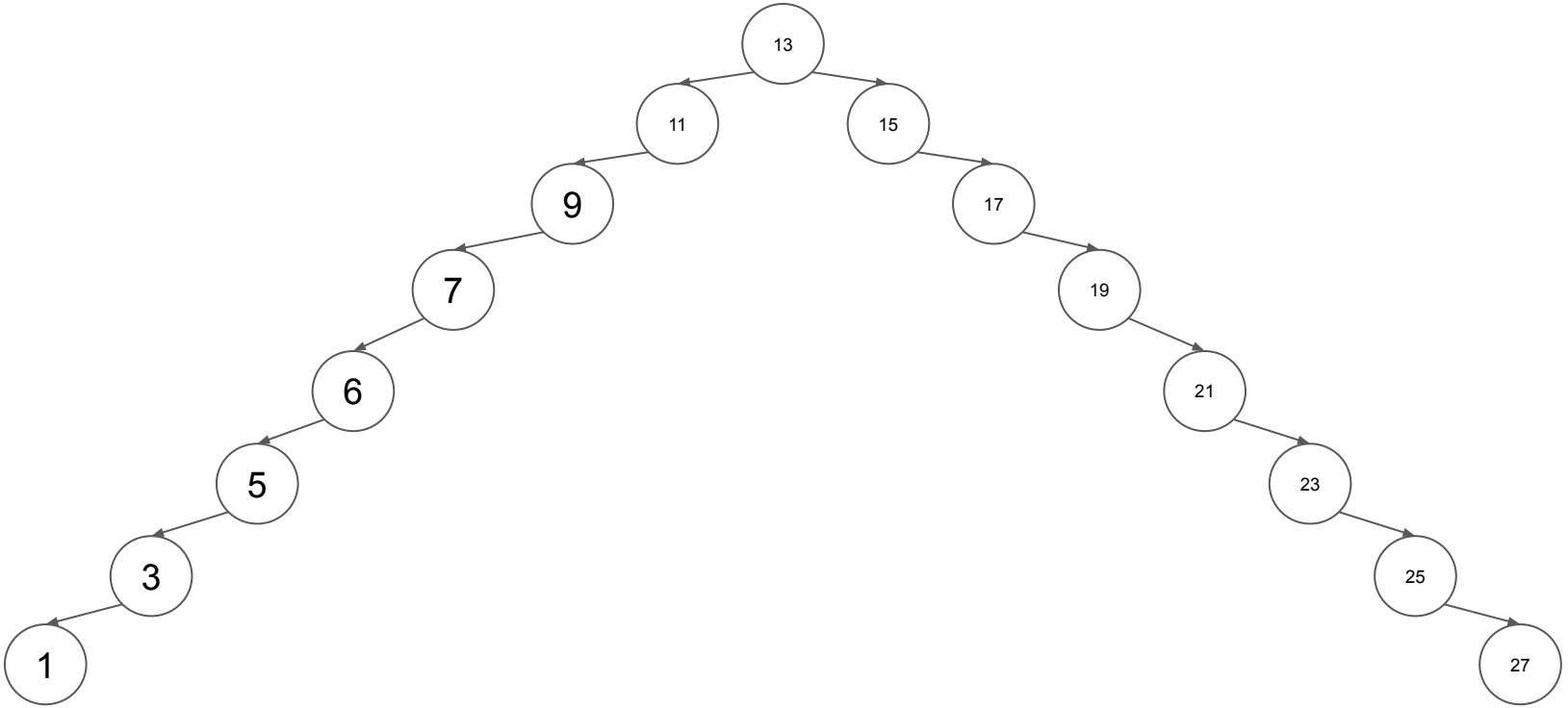
Asymptotic running time of a binary search performed on an array?

Building a balanced tree

An idea

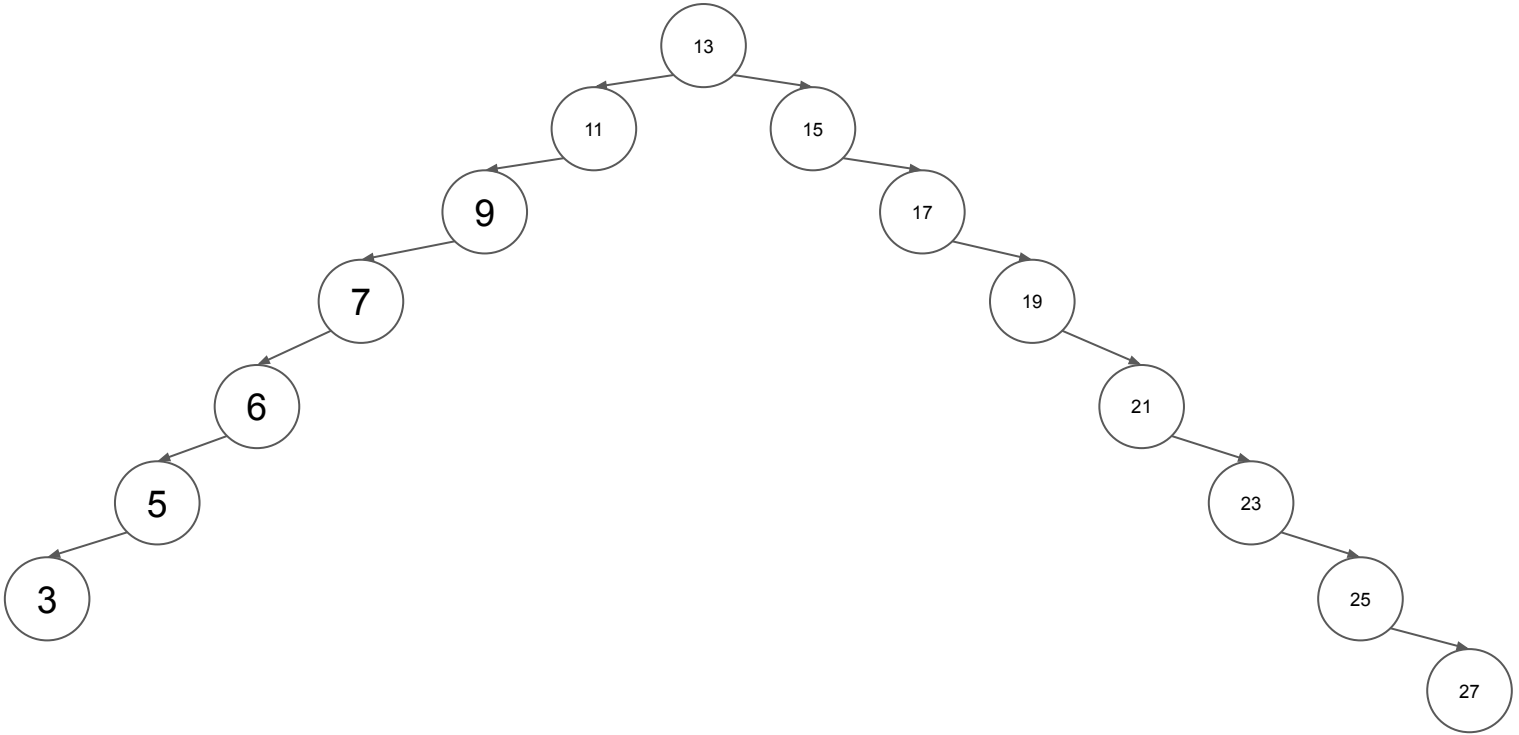
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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



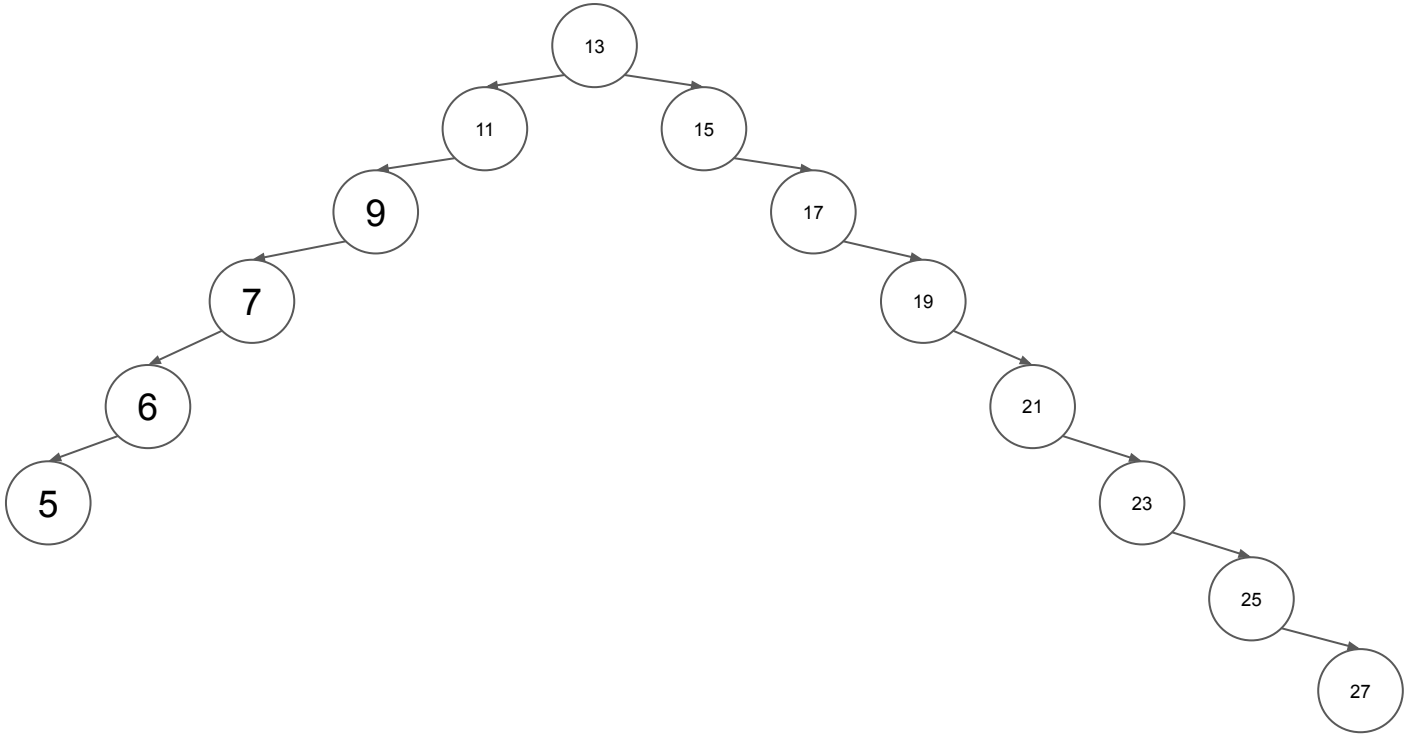
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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



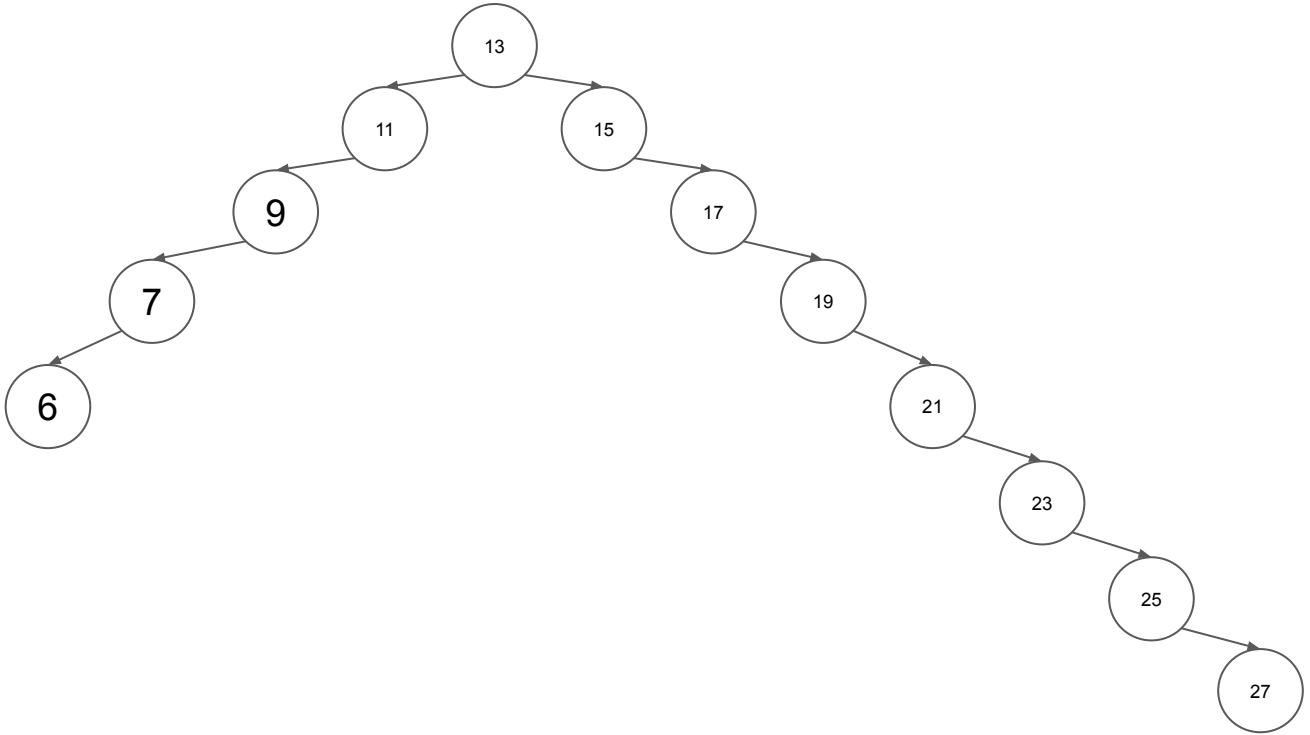
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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



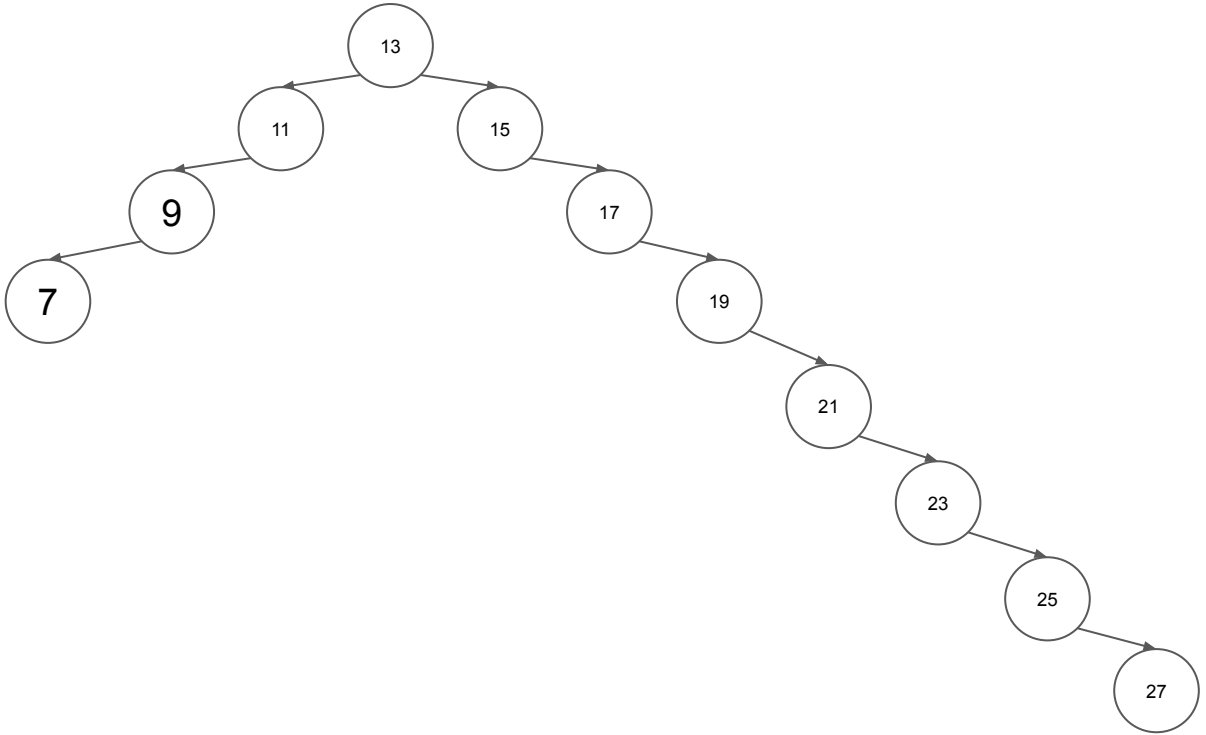
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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



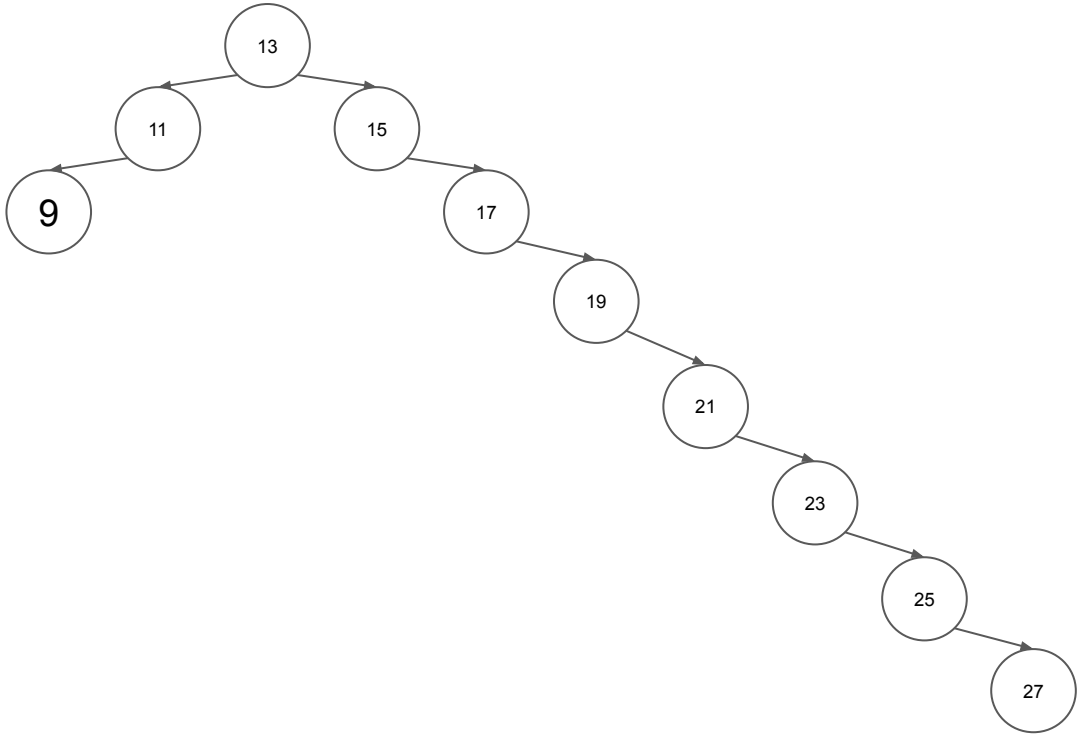
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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

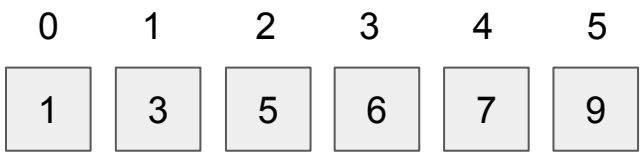


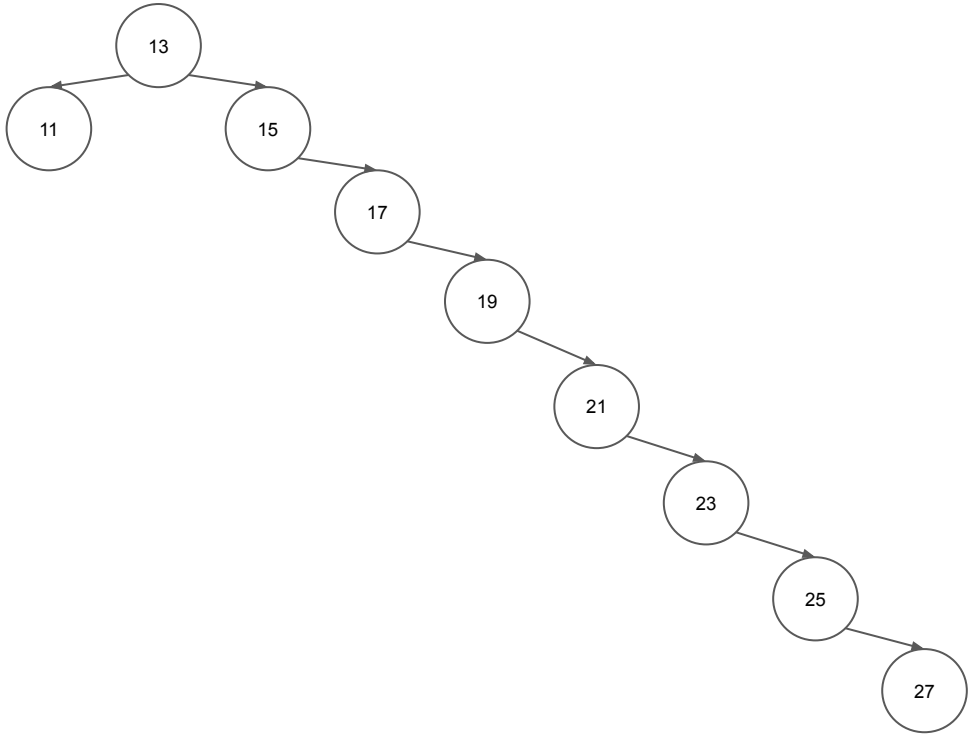
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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

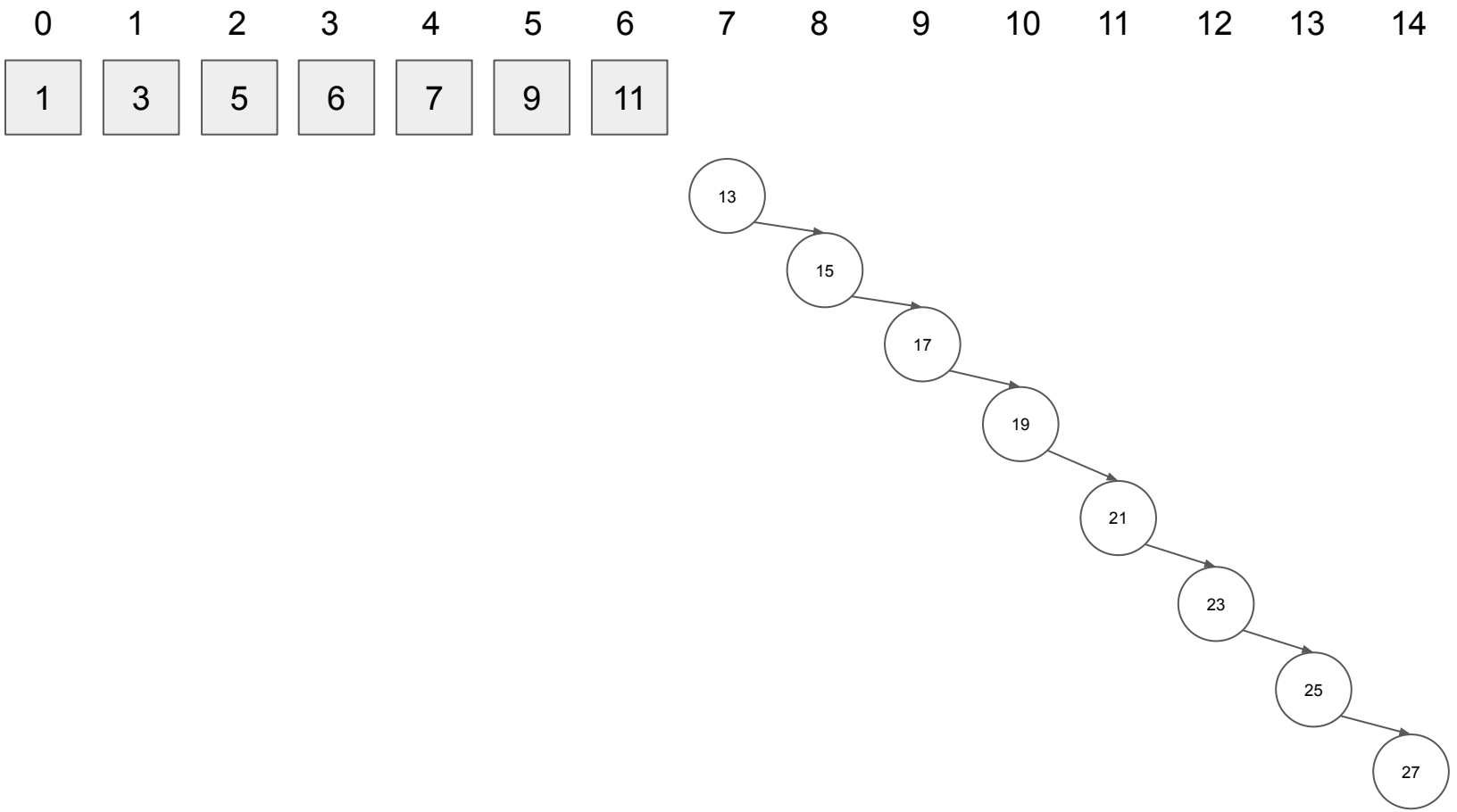


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



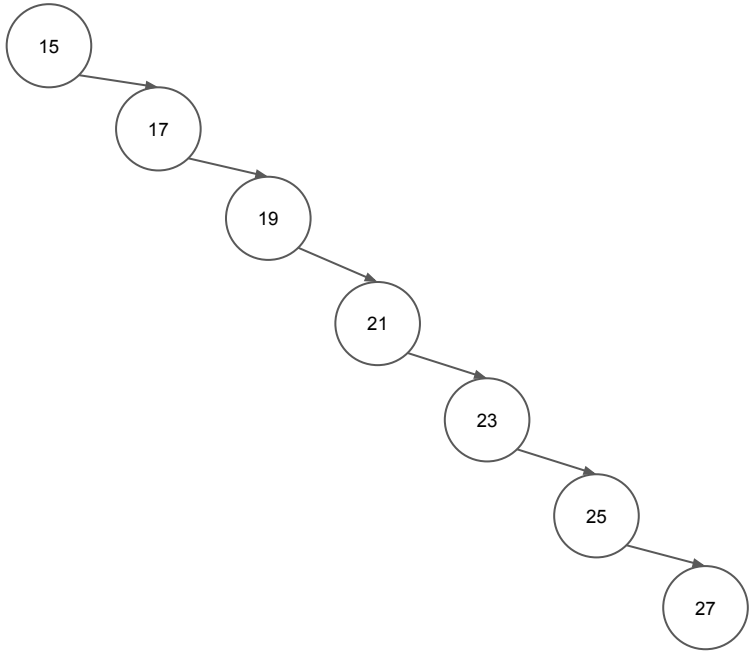


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



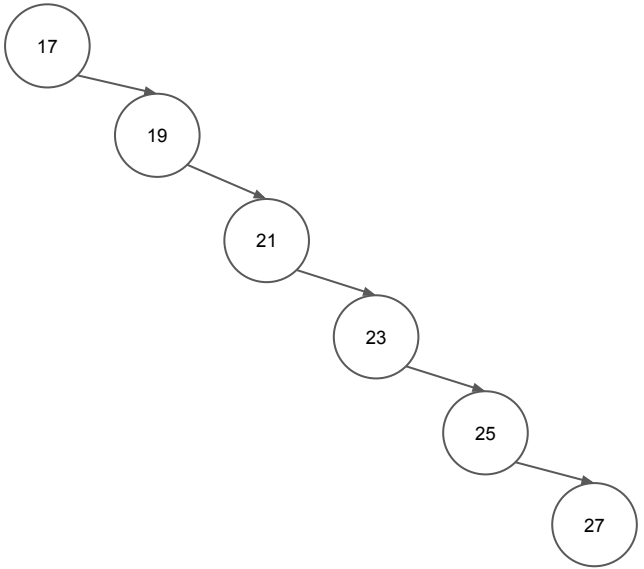
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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



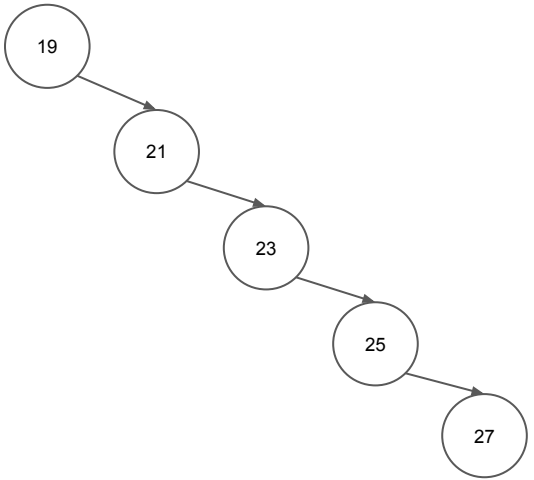
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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



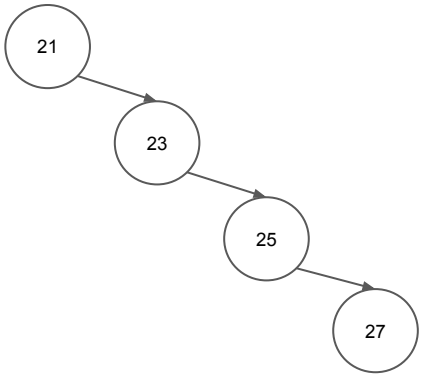
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	5	6	7	9	11	13	15	17					



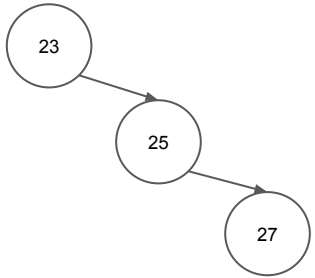
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	5	6	7	9	11	13	15	17	19				



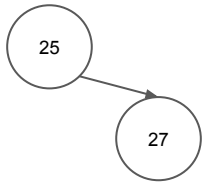
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	5	6	7	9	11	13	15	17	19	21			

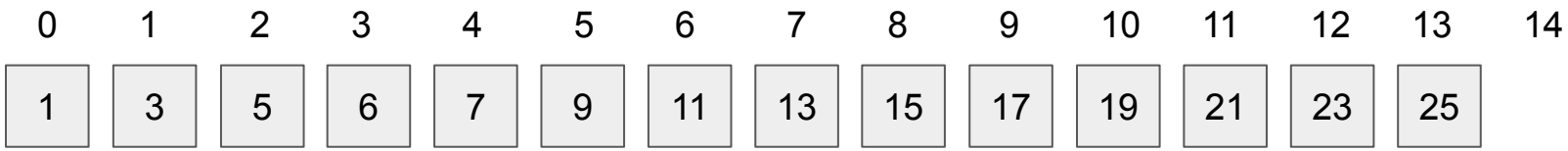


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	5	6	7	9	11	13	15	17	19	21	23		

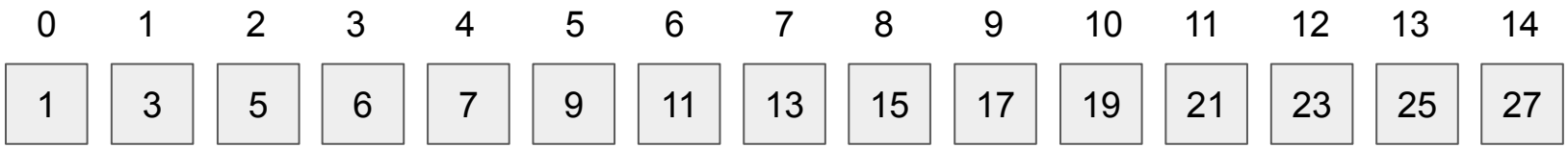


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

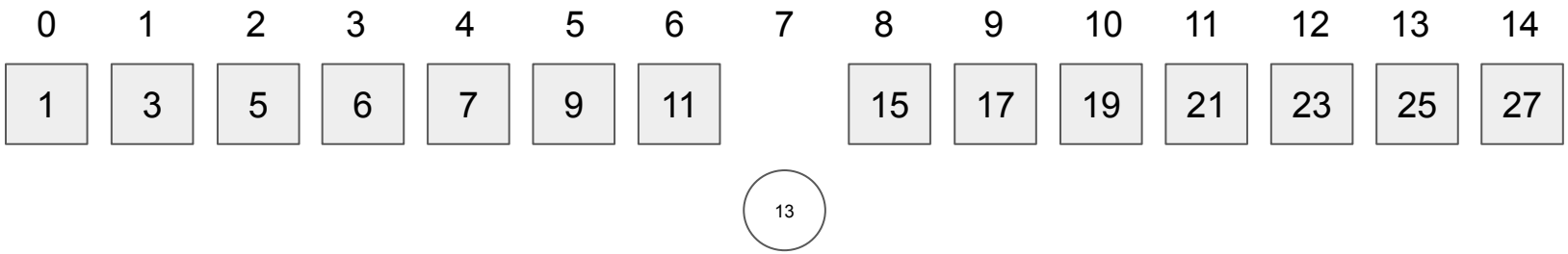


27

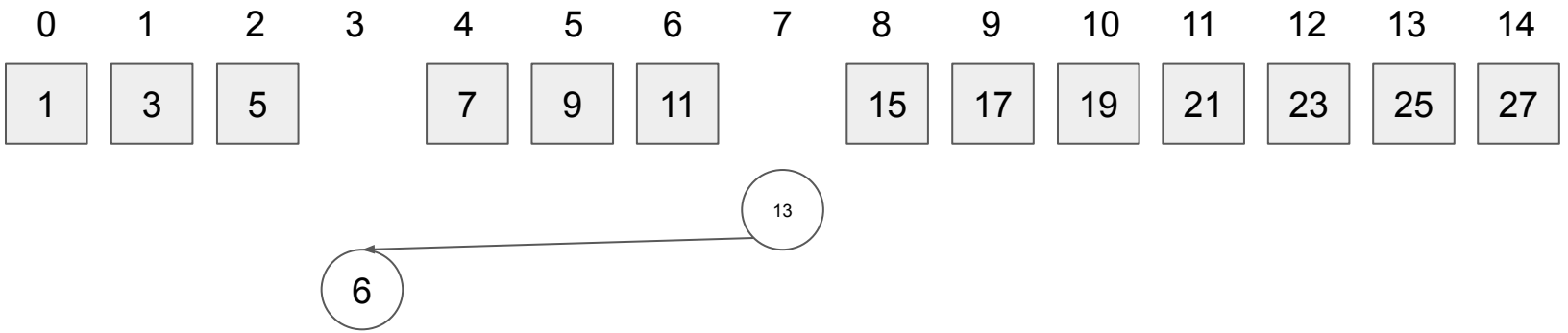
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



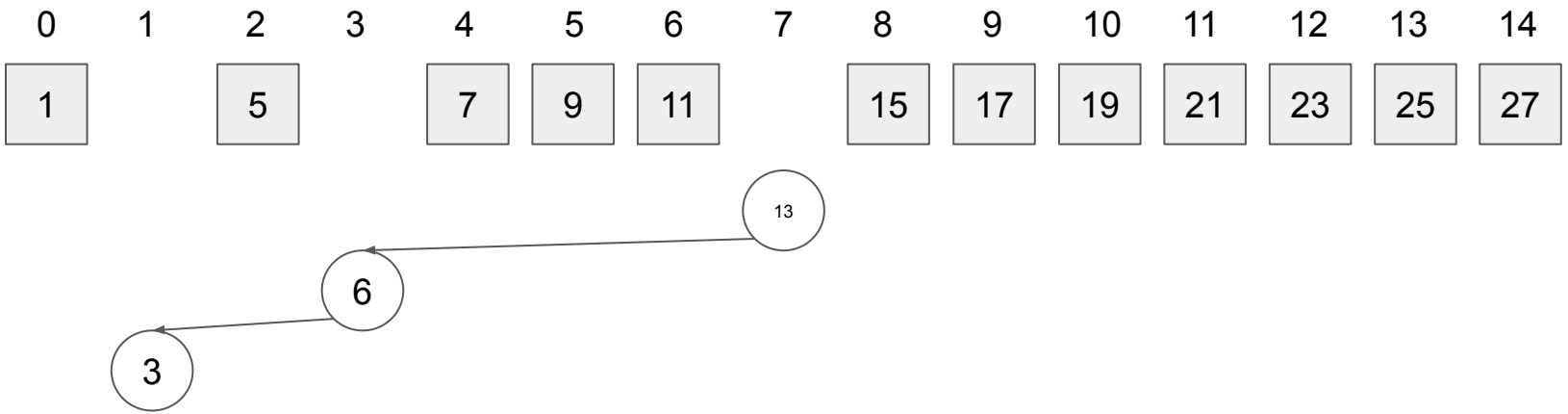
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



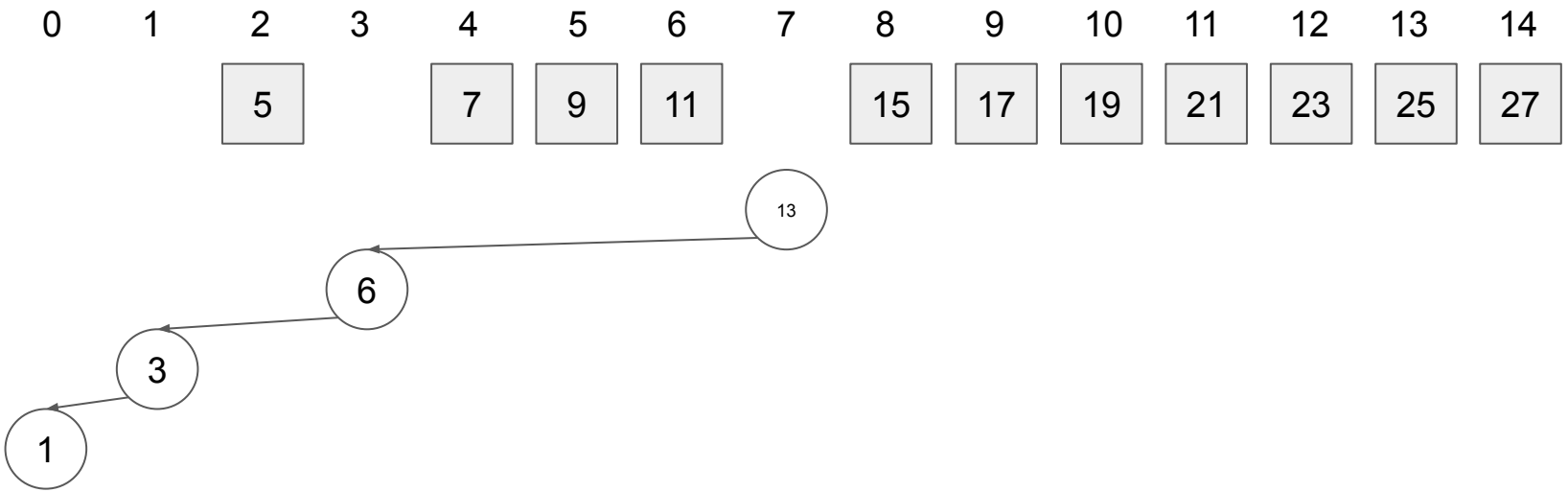
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



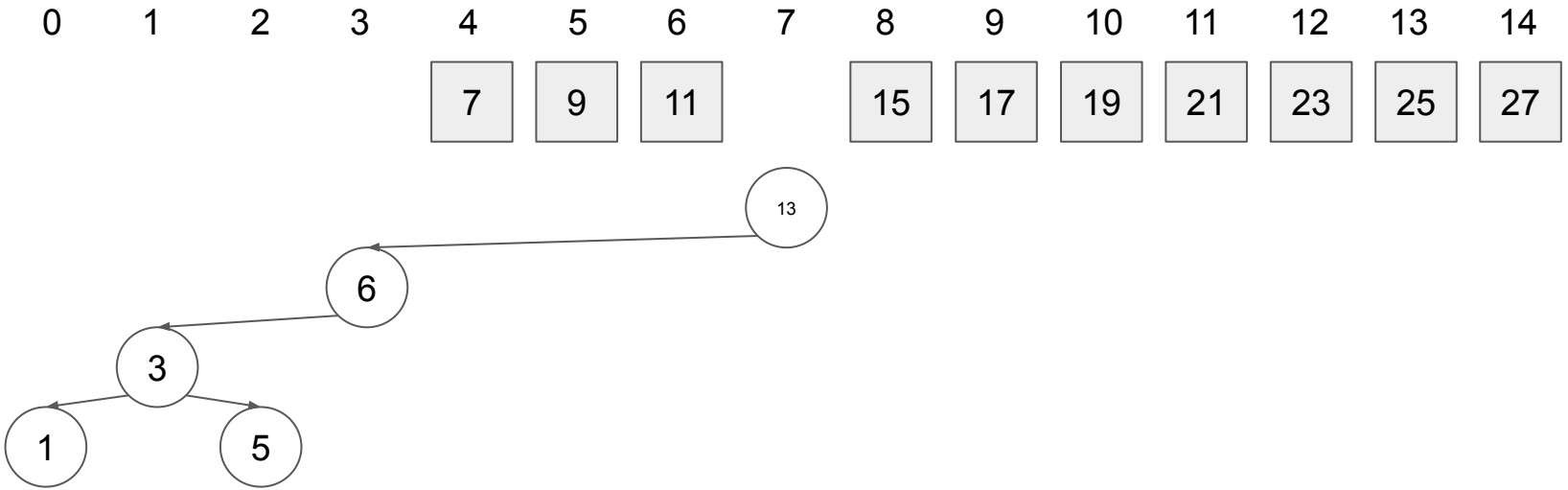
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



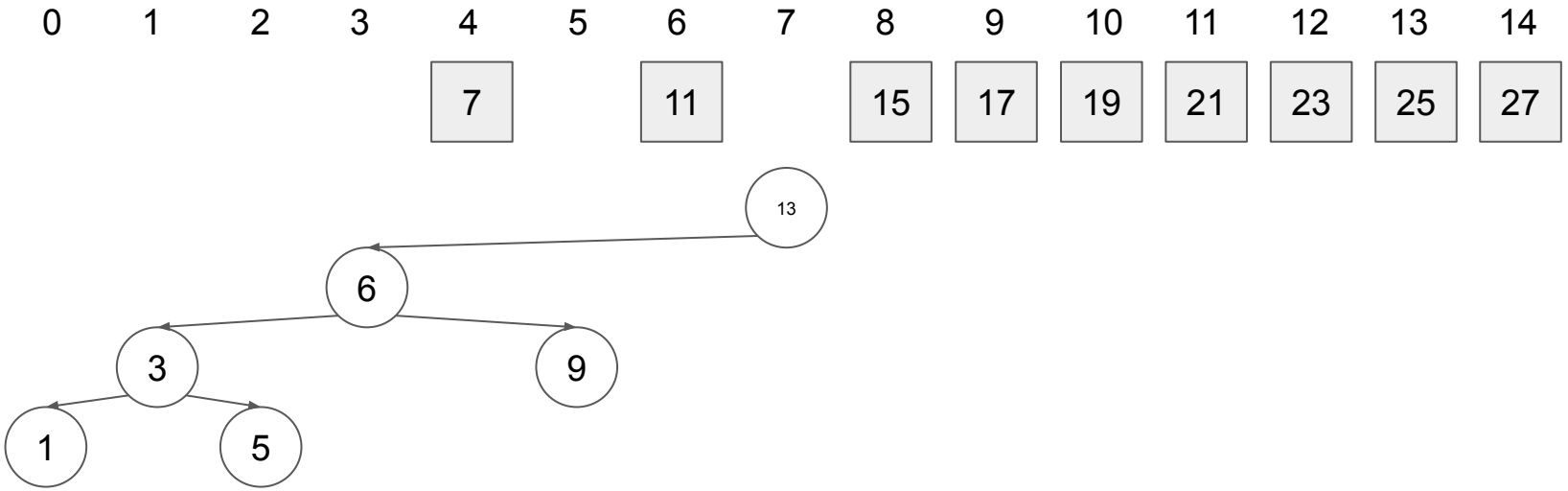
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



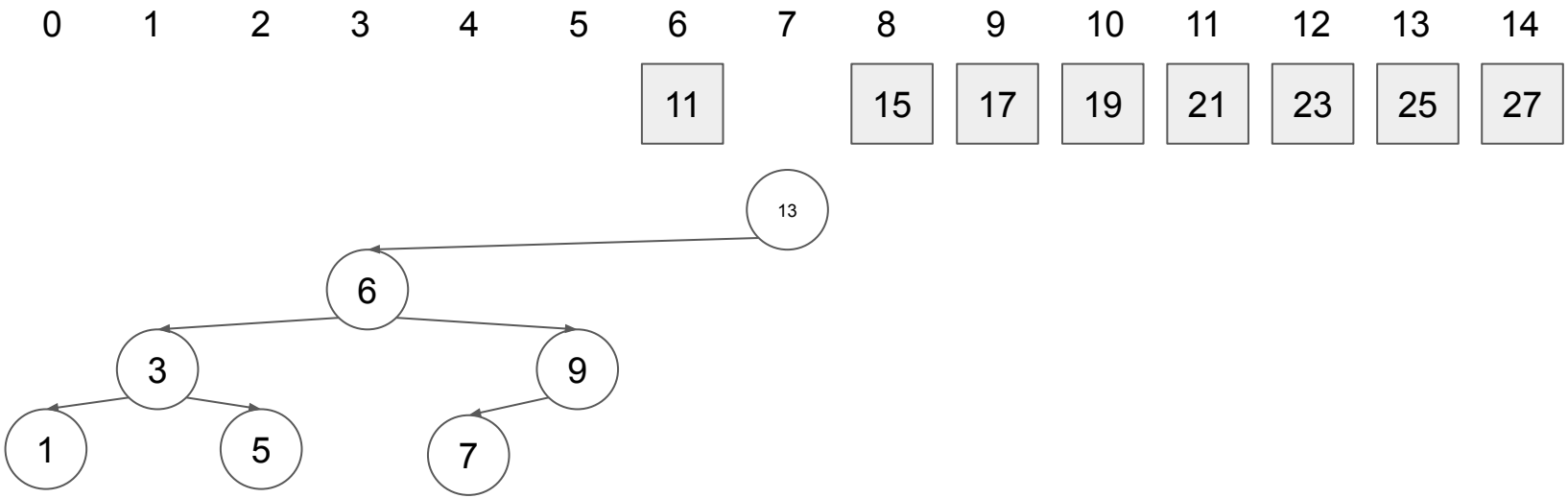
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



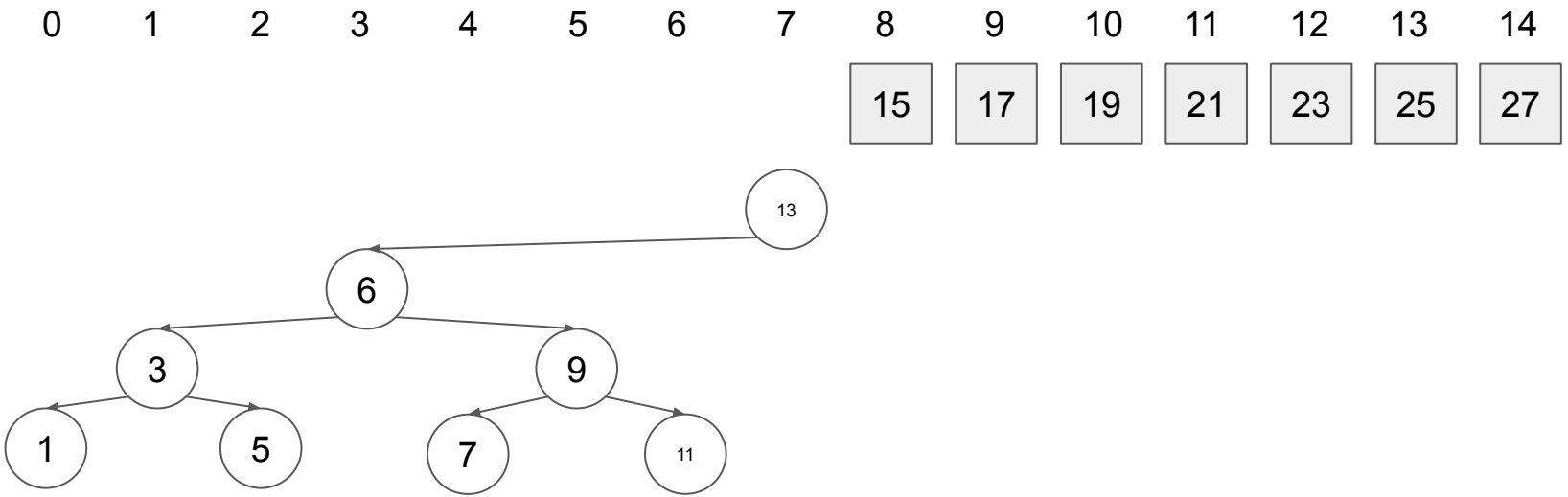
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



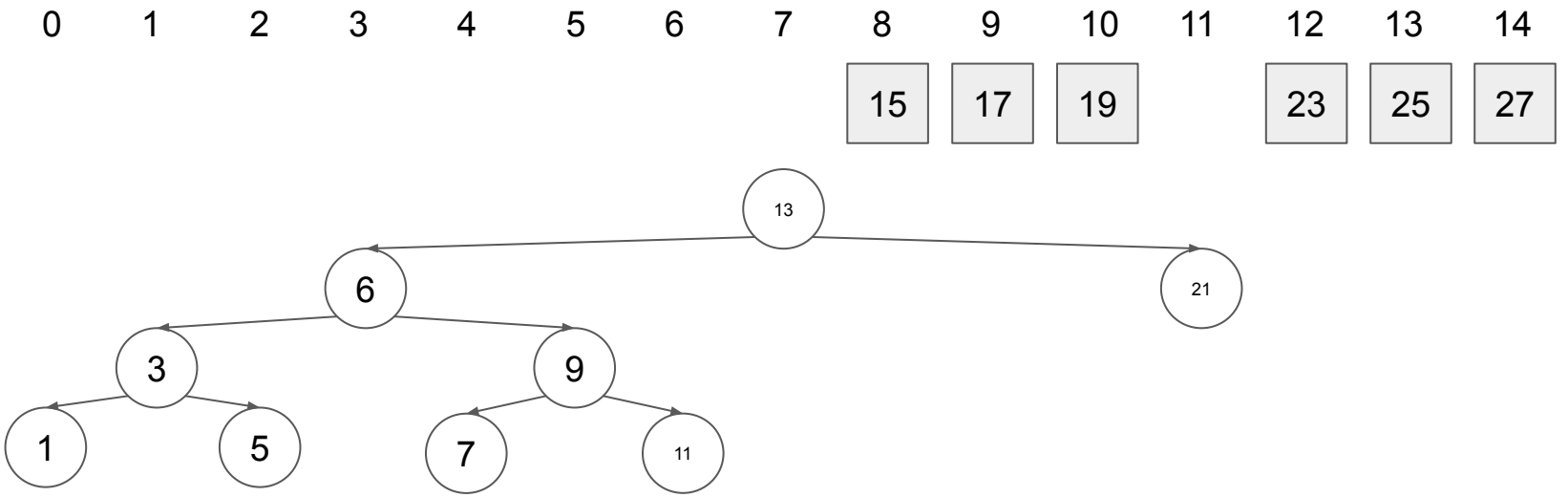
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



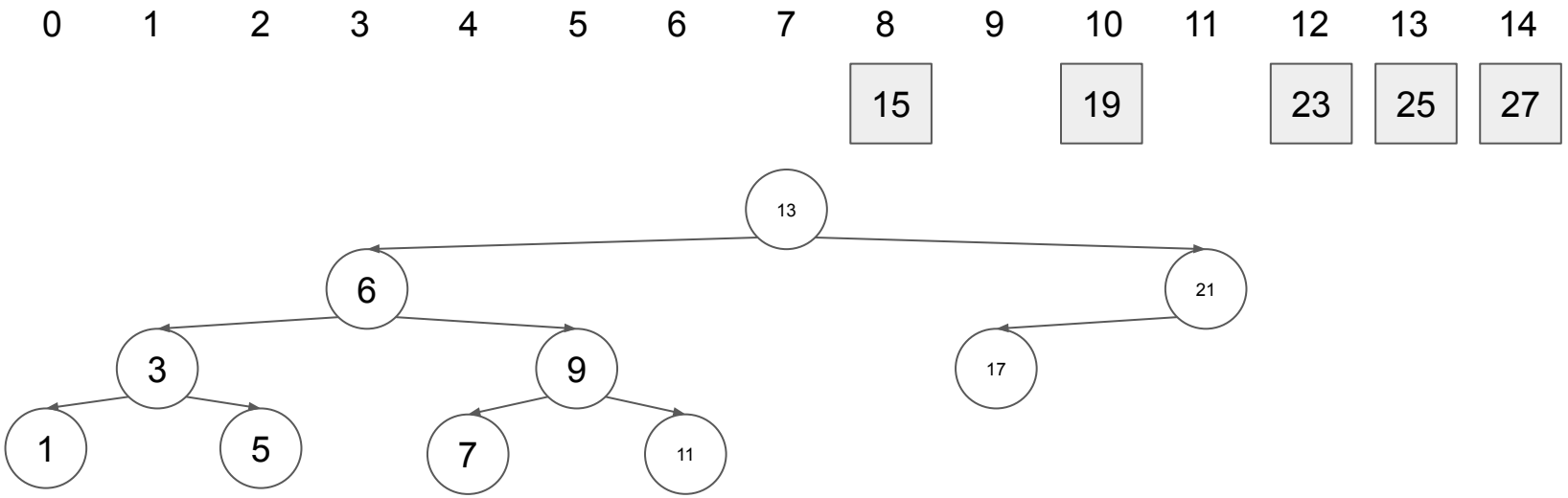
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



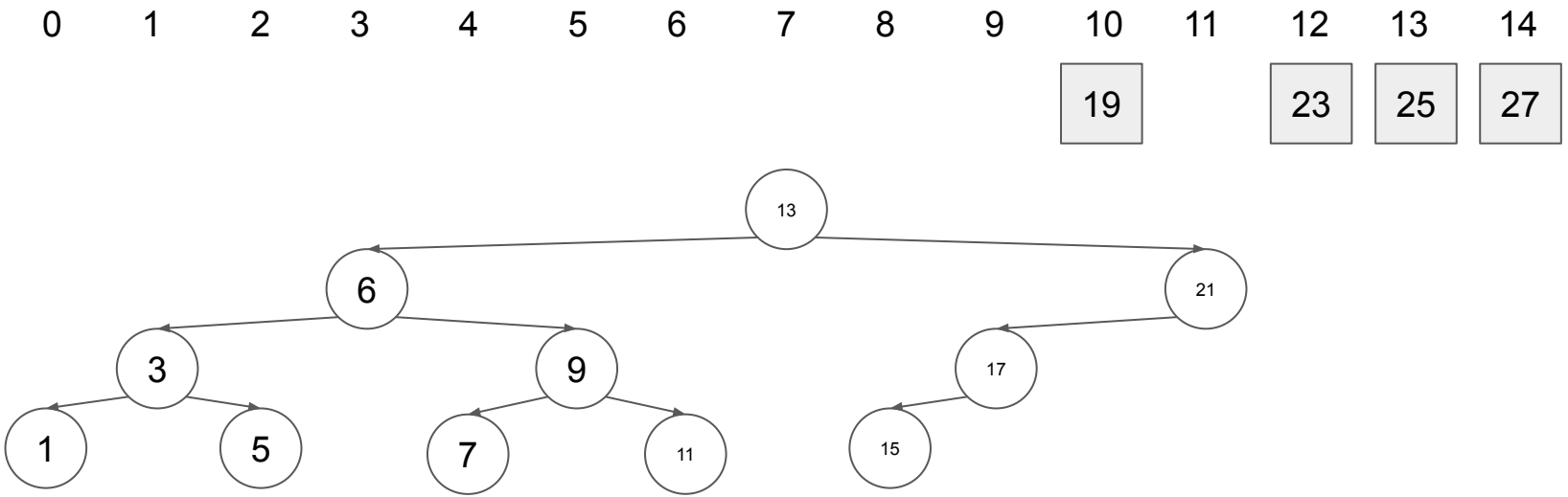
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

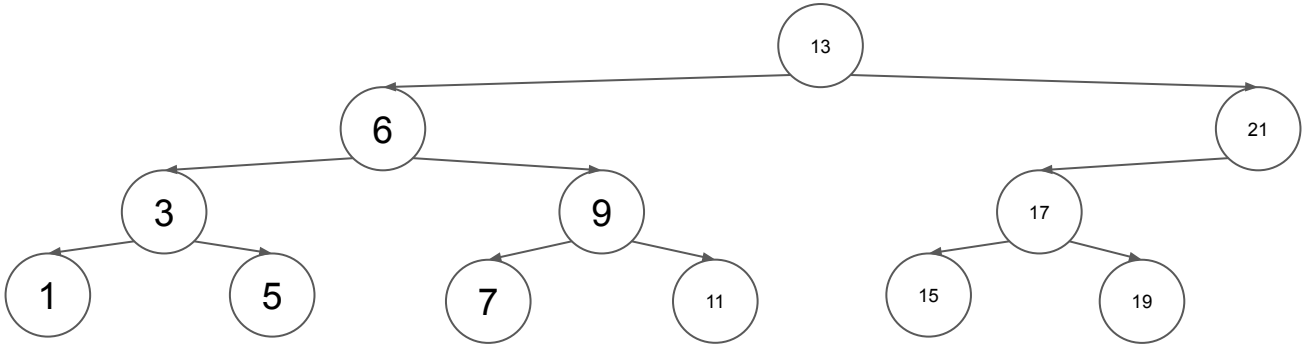


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

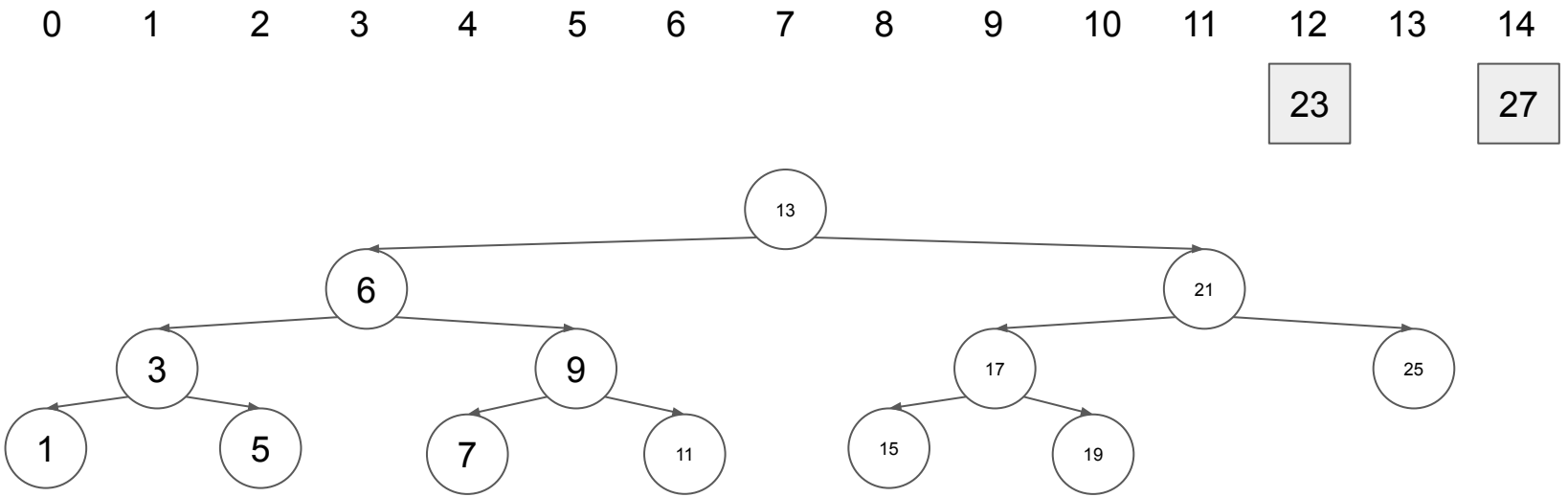


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
												23	25	27

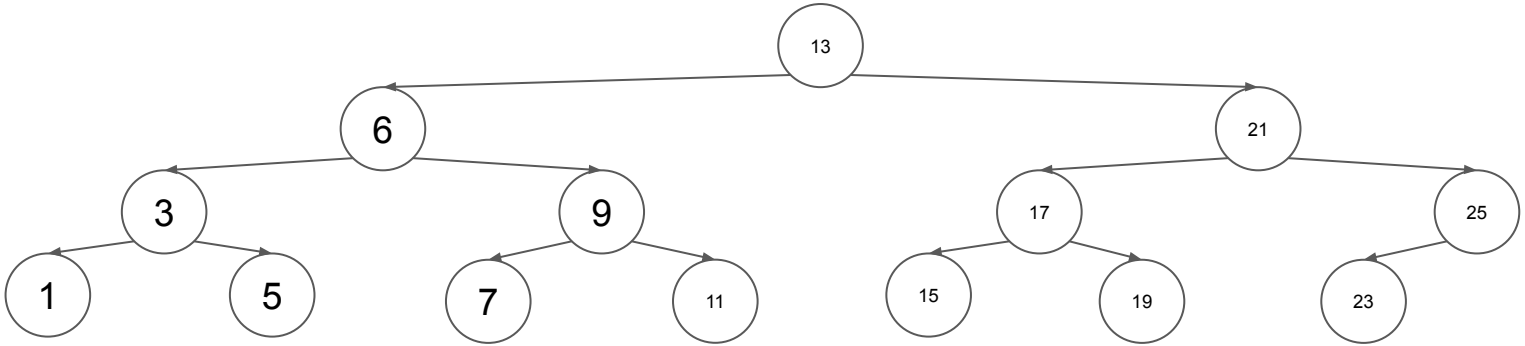


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



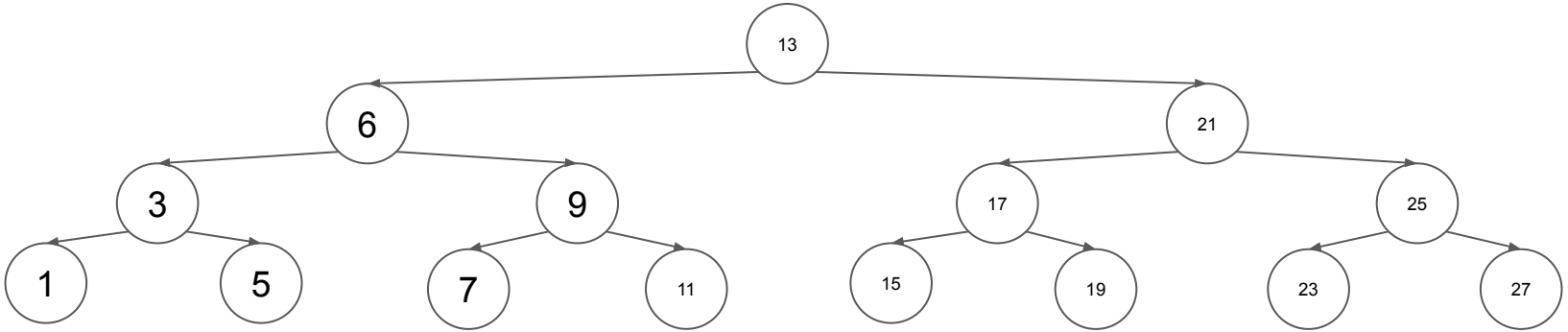
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
														27



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

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



Running time of this operation?

Const correctness

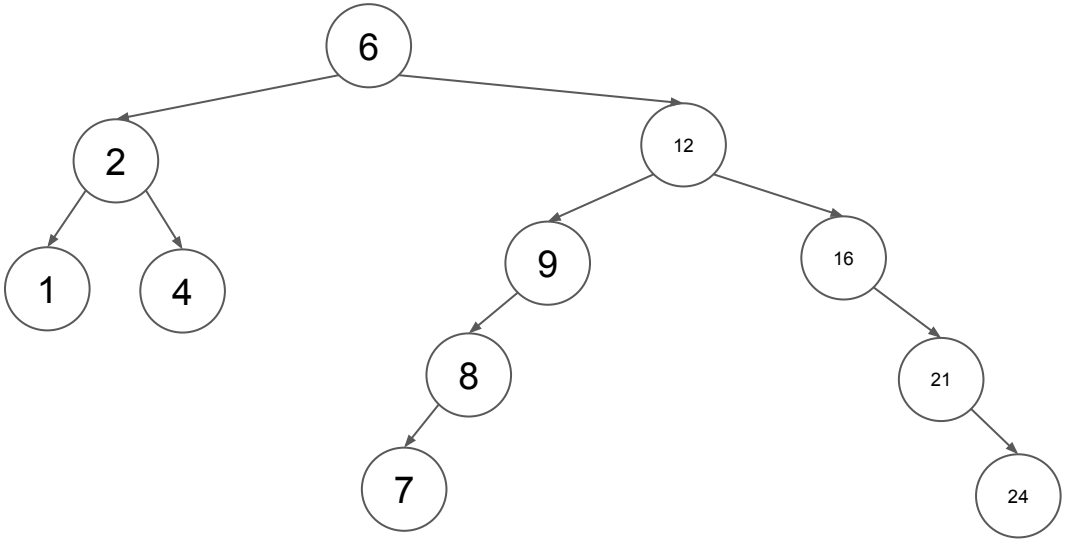
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

```
 1int main(){
 2  const int* a = new int(1);
 3  int* const b = new int(2);
 4  const int * const c = new int(3);
 5  int const * const d = new int(4);
 6
 7  delete a;
 8  delete b;
 9  delete c;
10  delete d;
11
12  return 0;
13}
```

(Note: We also took a look at the spec of our previous programming projects.)

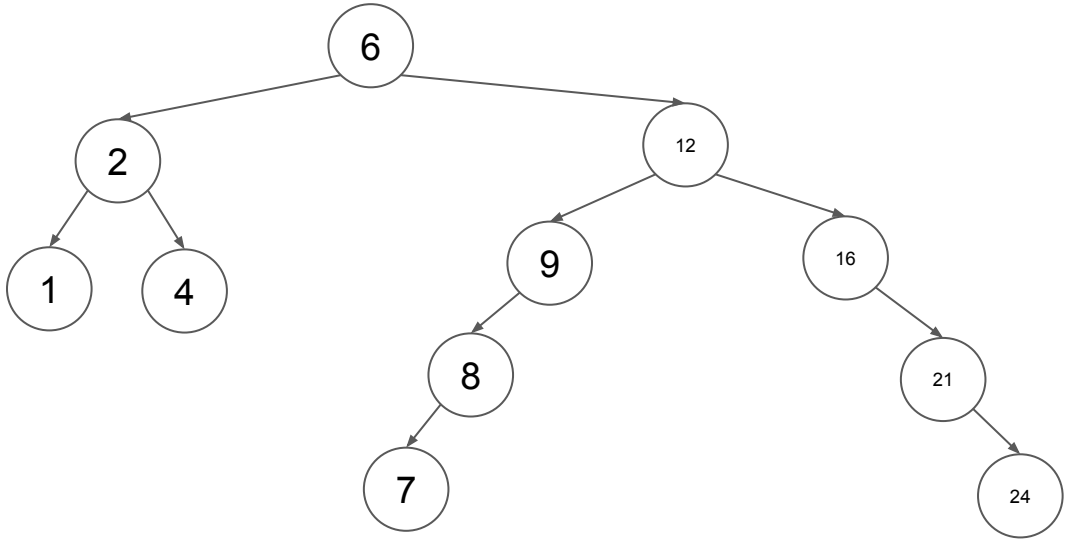
Rotations

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



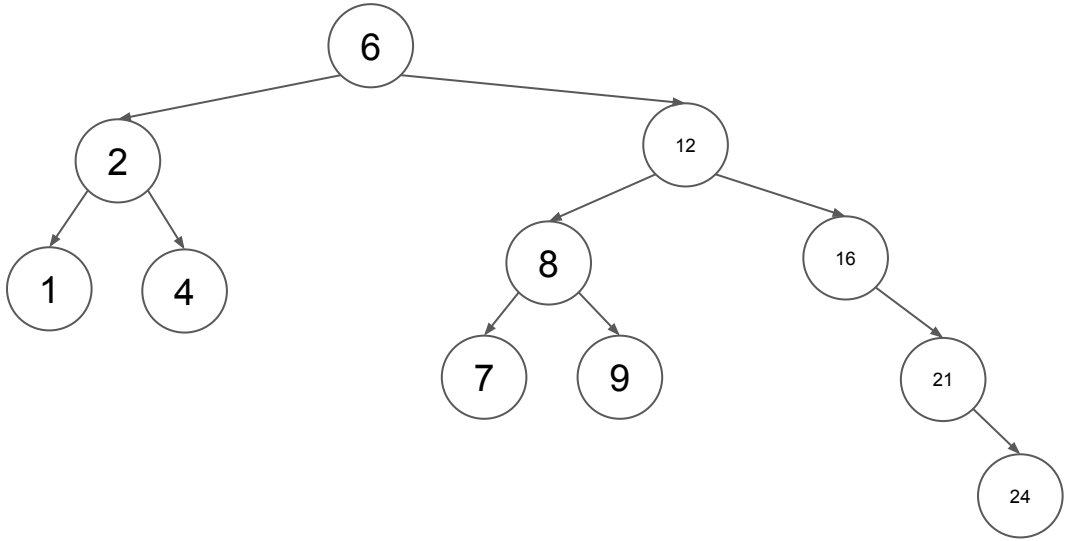
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around 9



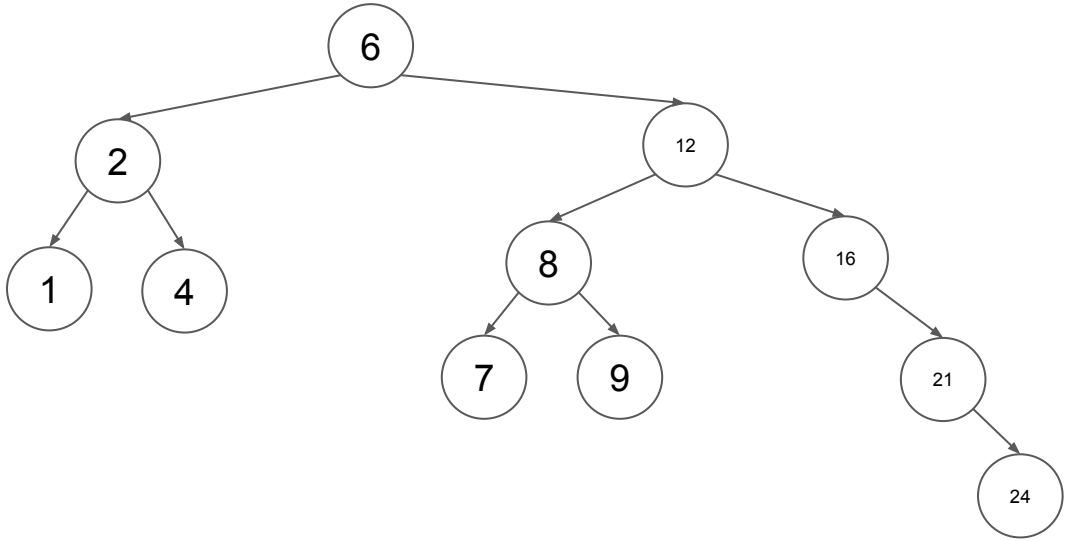
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around 9



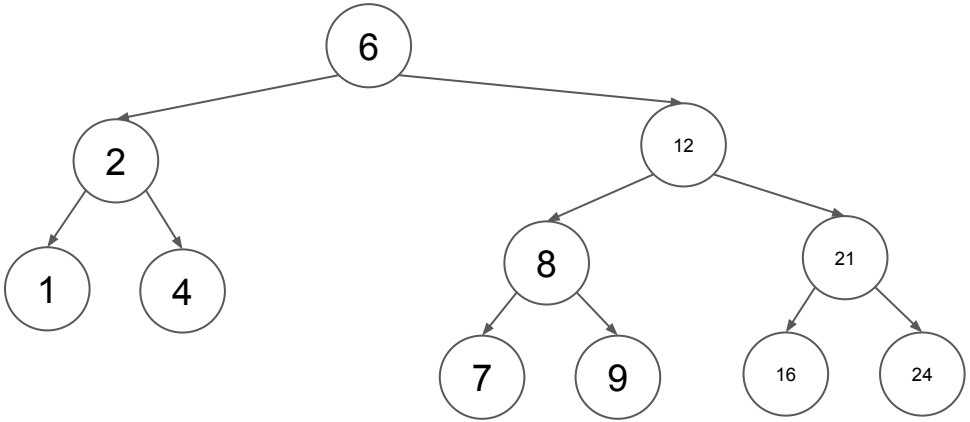
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around 16



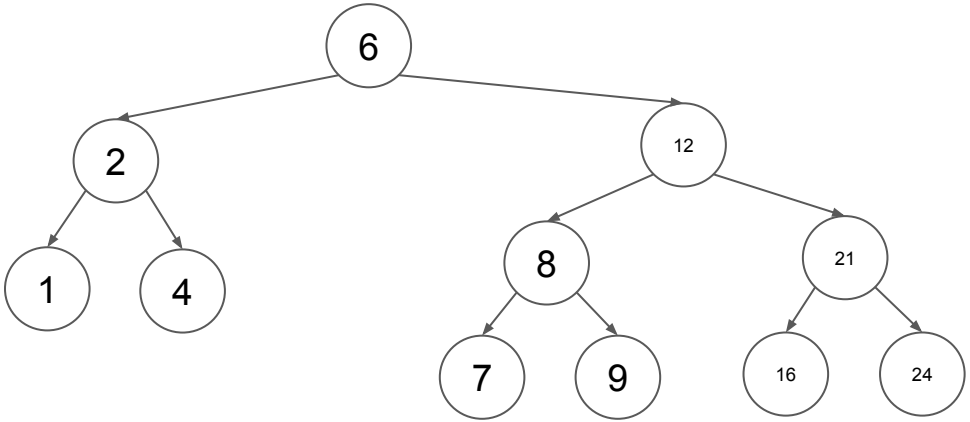
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around 16



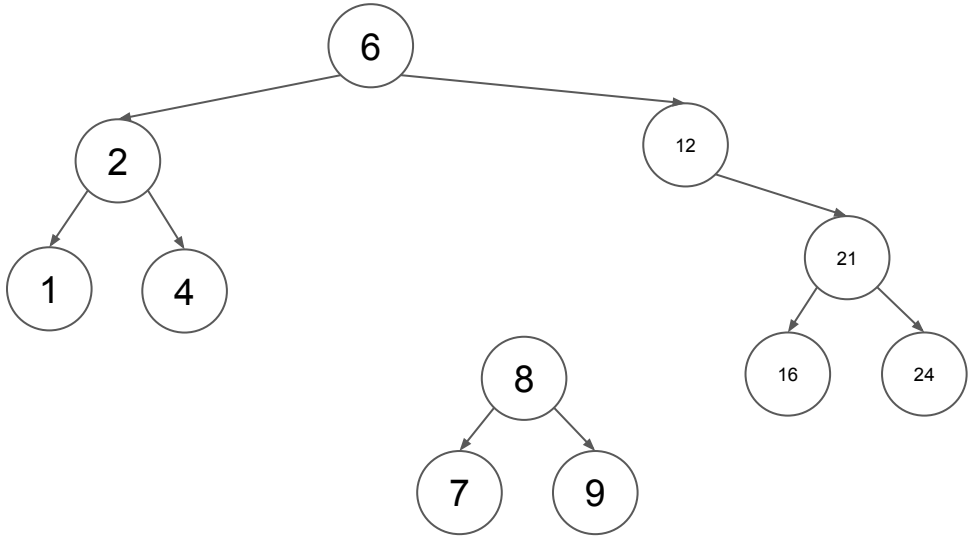
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around 6 (for demonstration purpose)



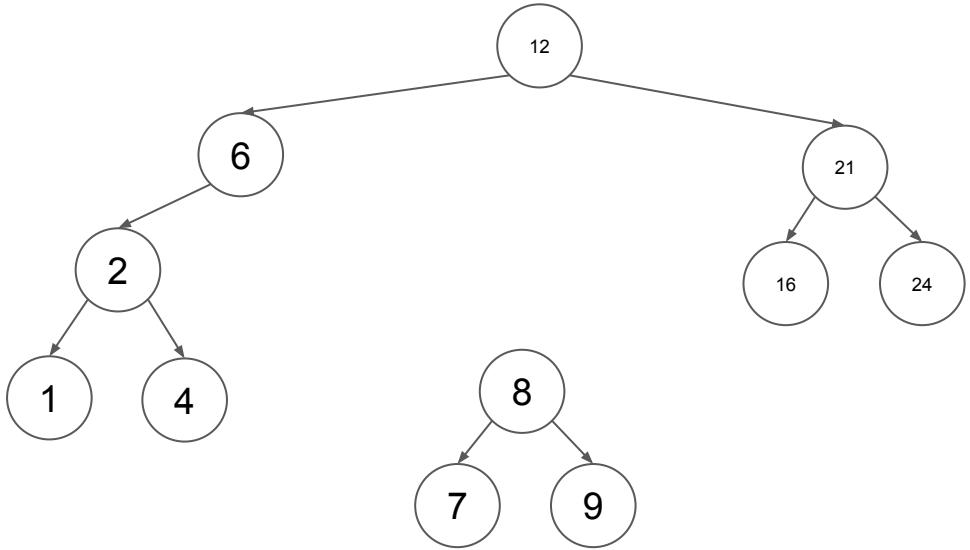
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around 6 (for demonstration purpose)



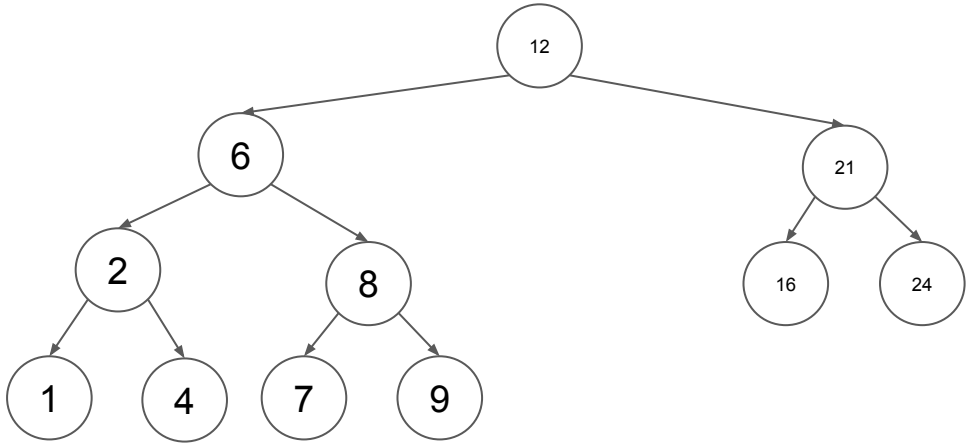
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around 6 (for demonstration purpose)



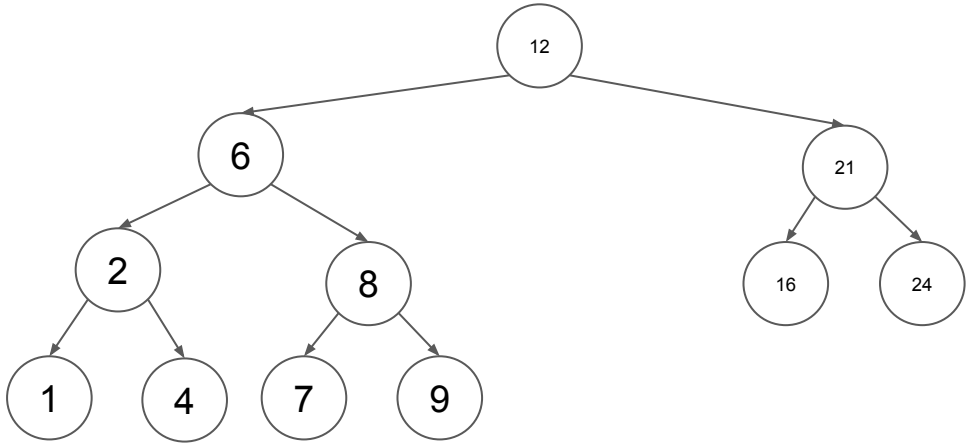
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around 6 (for demonstration purpose)



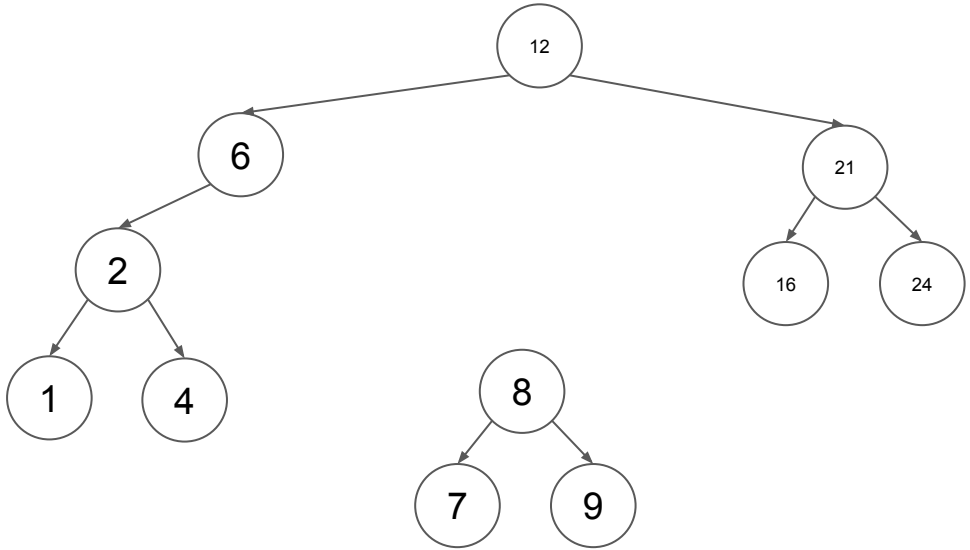
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around 12 (for demonstration purpose)



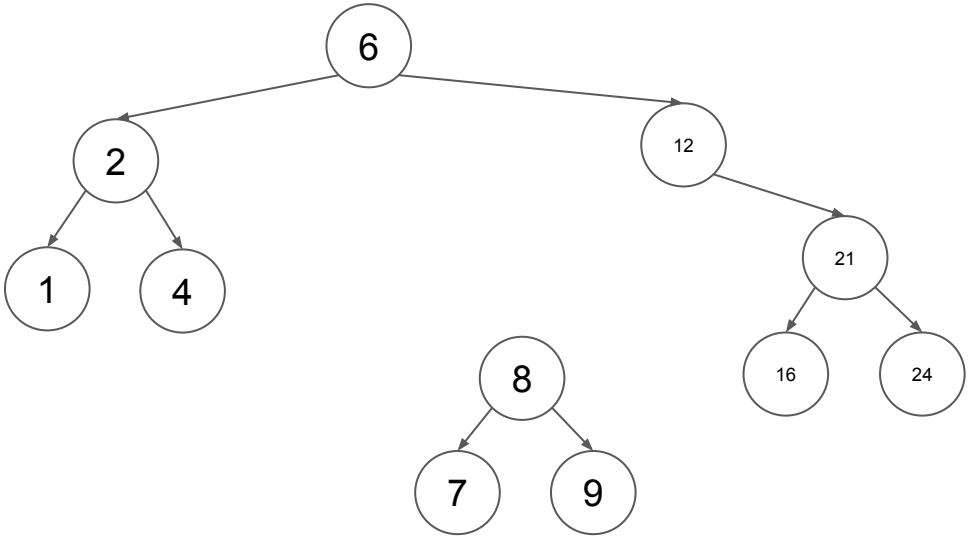
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around 12 (for demonstration purpose)



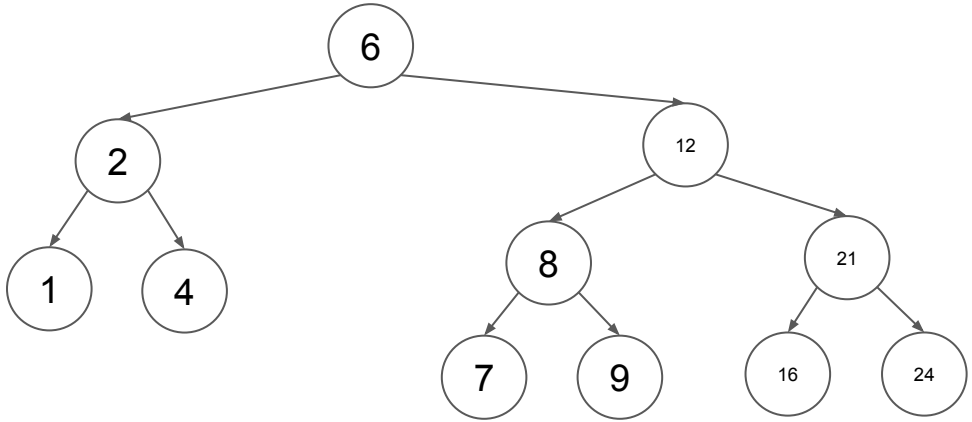
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around 12 (for demonstration purpose)



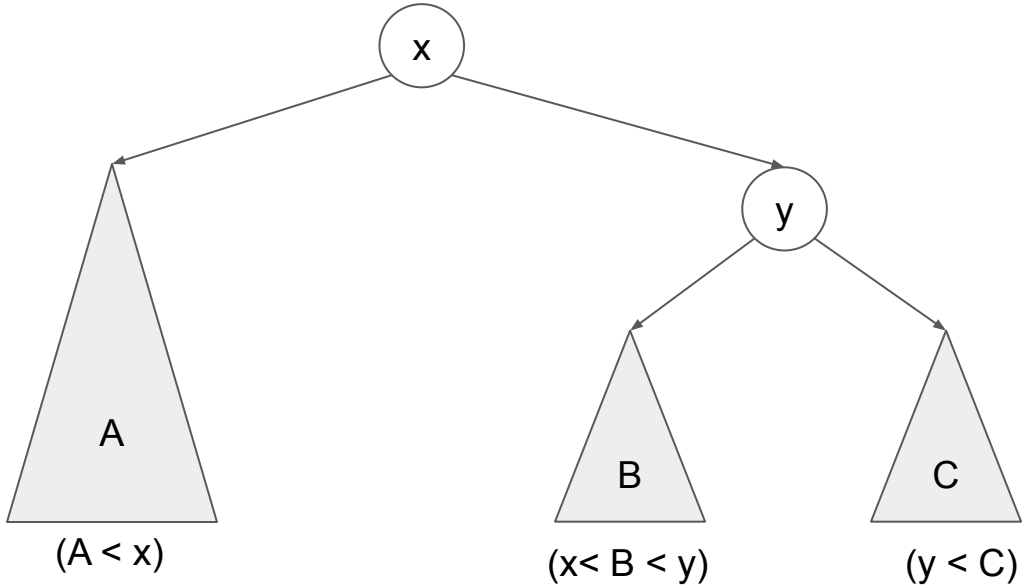
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around 12 (for demonstration purpose)



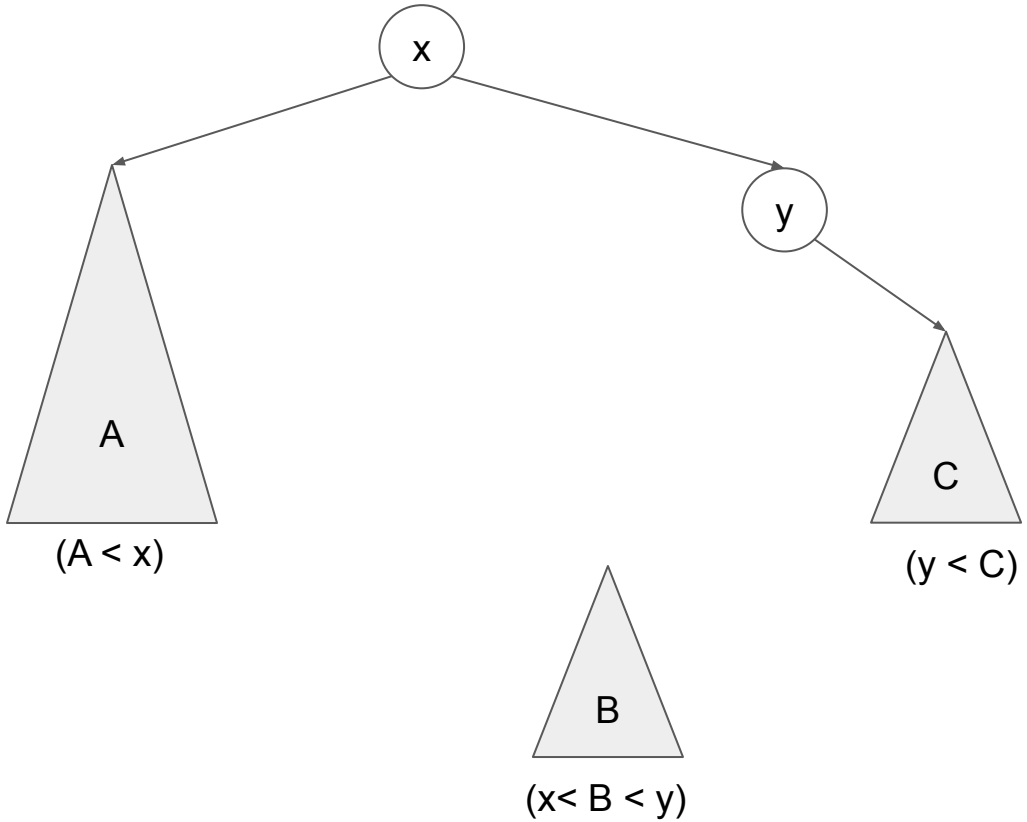
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around x



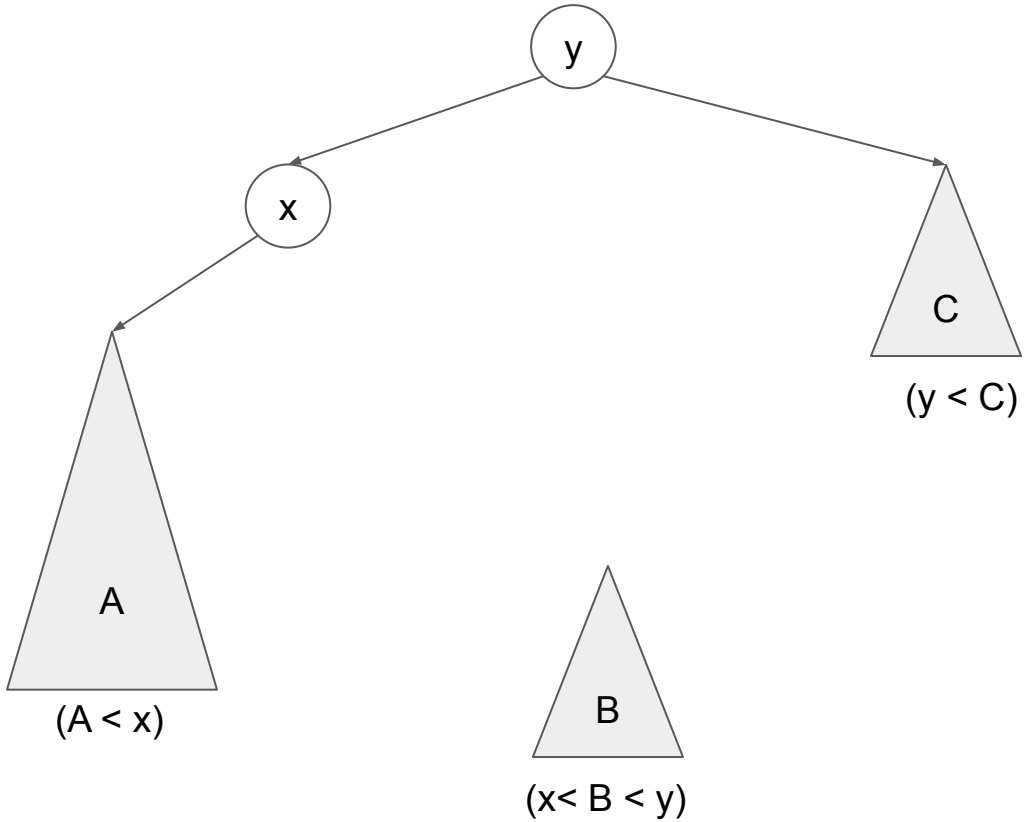
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around x



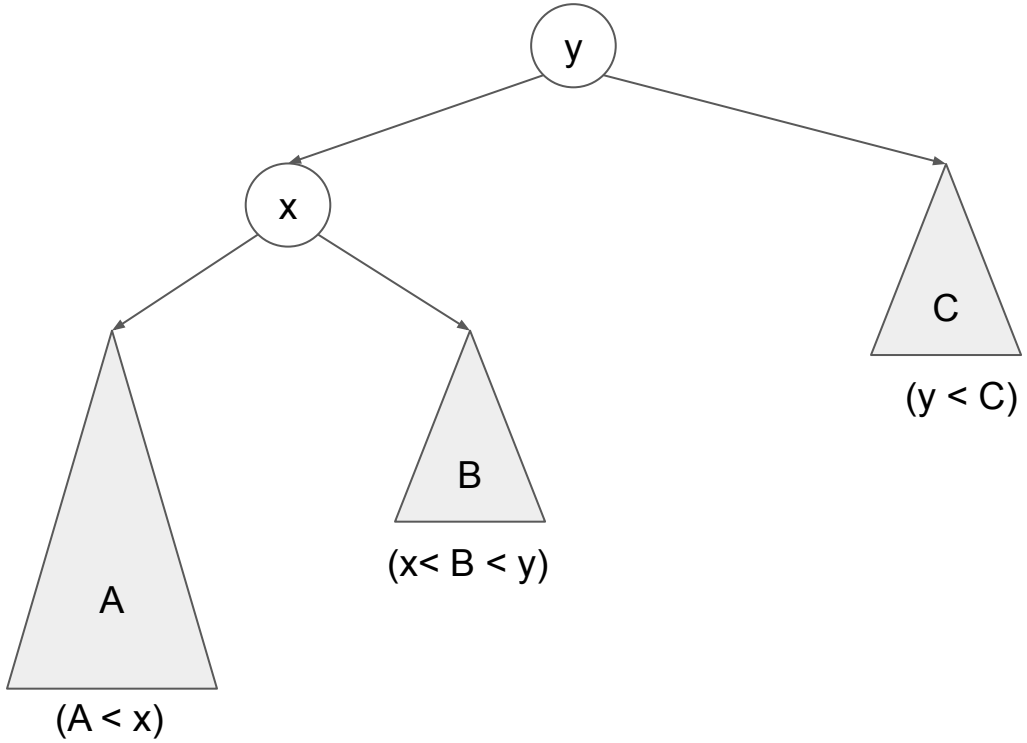
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around x



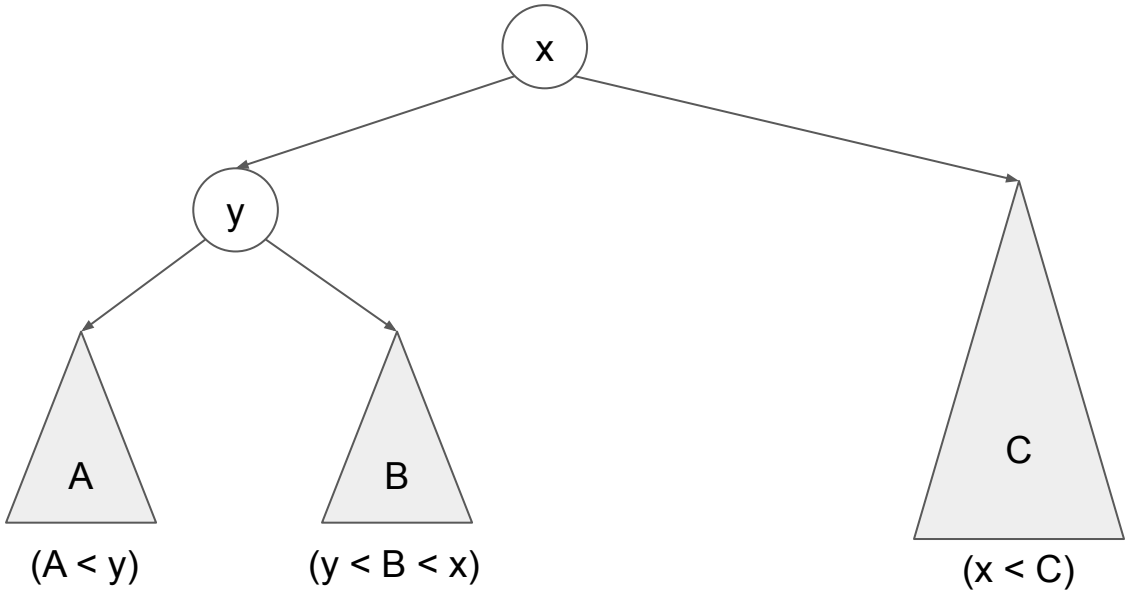
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate LEFT around x



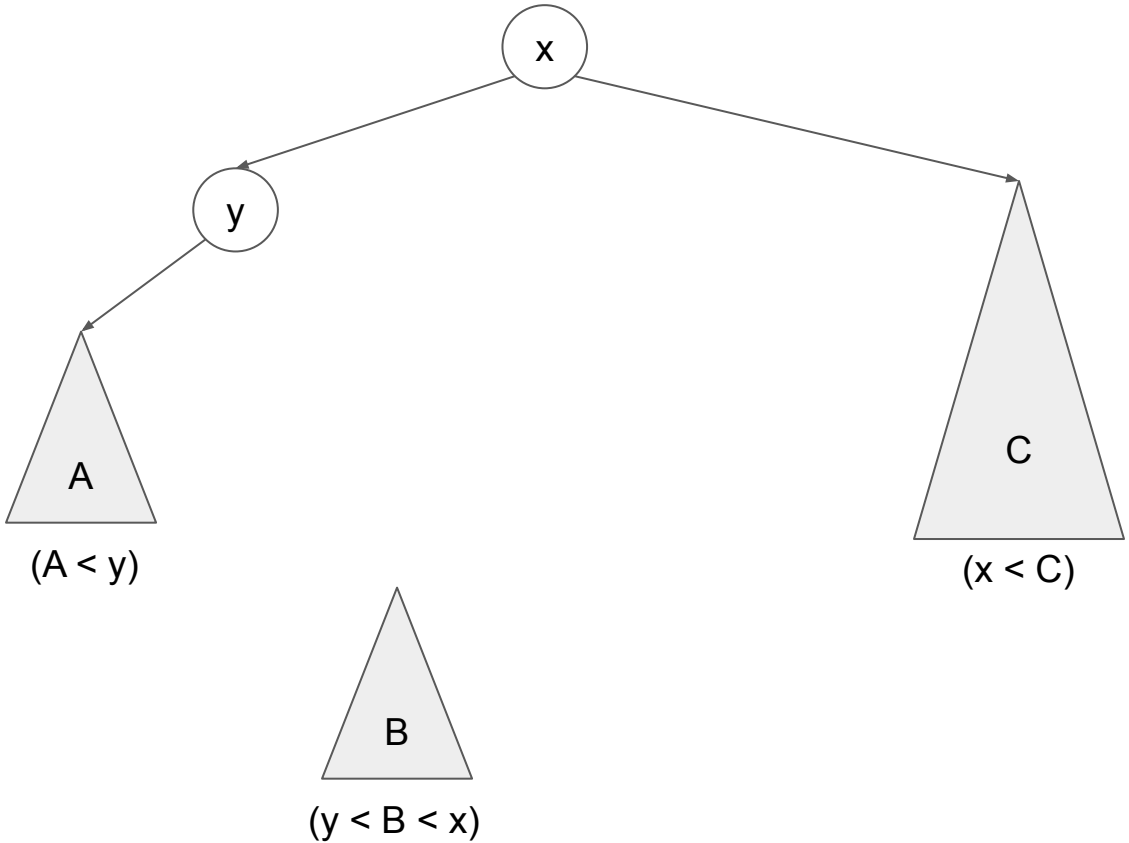
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around x



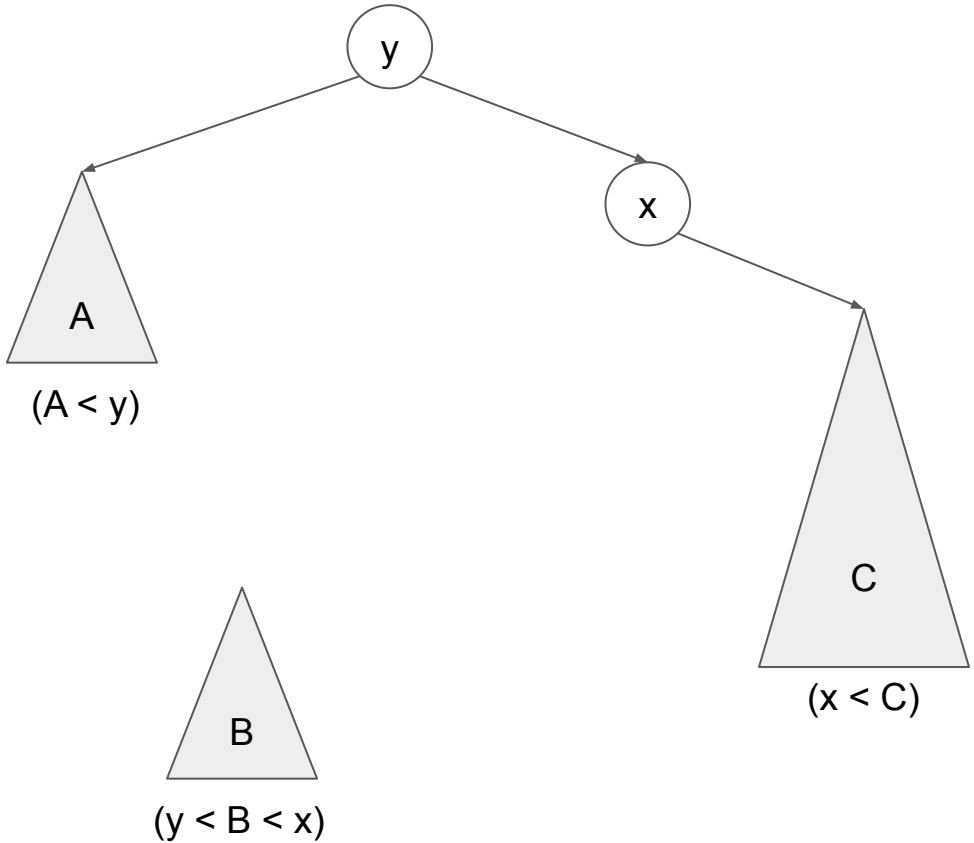
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around x



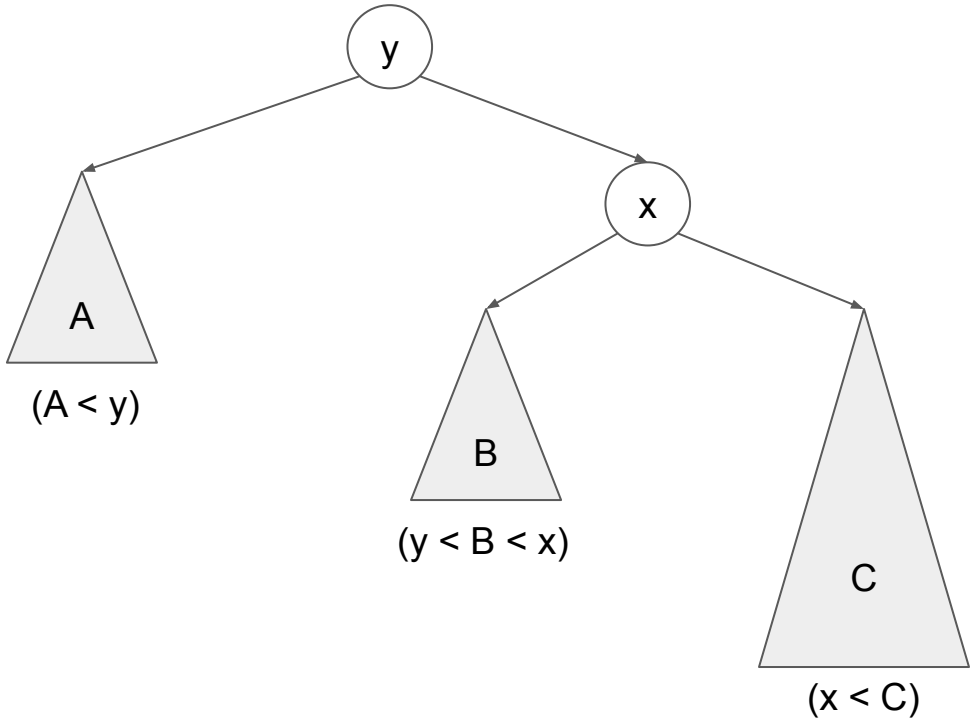
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around x



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

Rotate RIGHT around x



In-Class Activity

Self-balancing binary search trees

- AVL tree
- Red-black tree
- (and more)

Self-adjusting binary search trees

- Splay tree

Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Write the asymptotic worst-case running time of

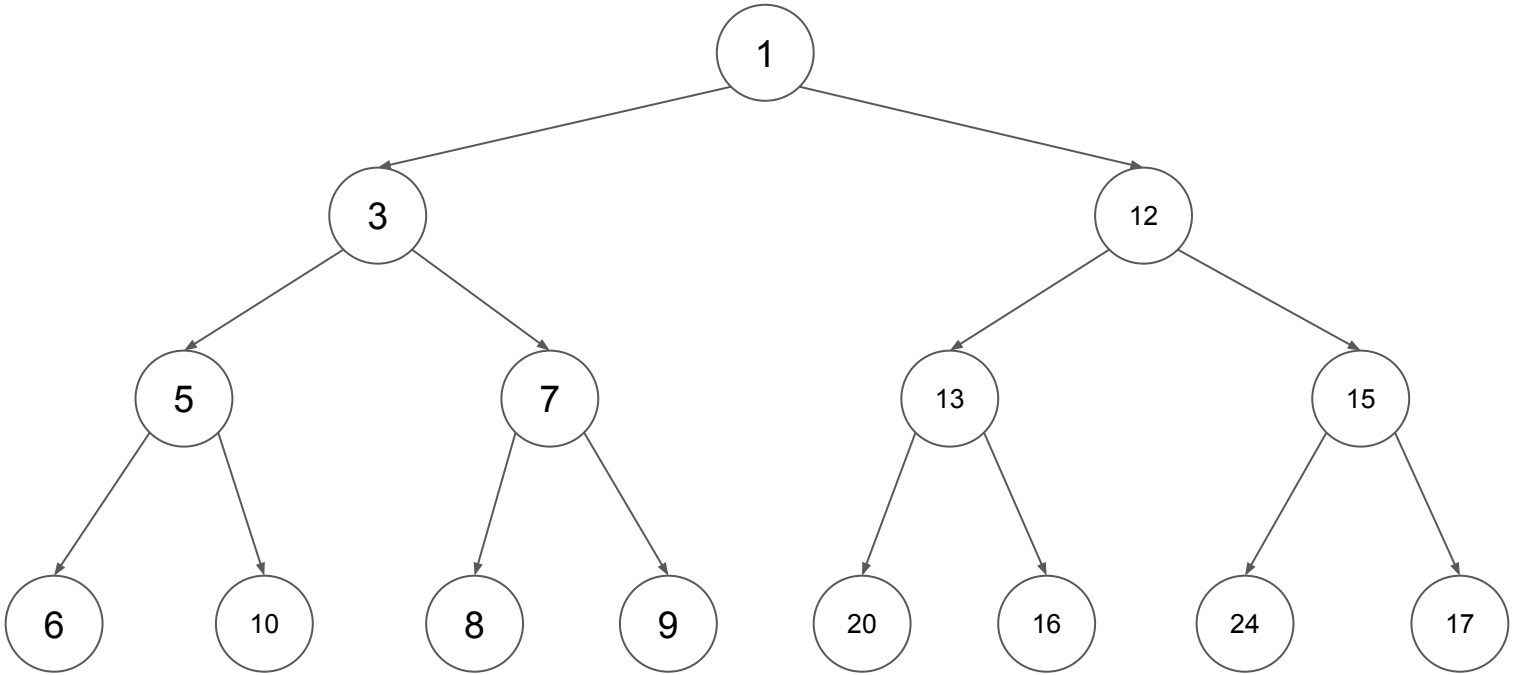
finding Min and finding Max

operation performed on a Binary Search Tree.

Do Now Exercise

Students' answers:

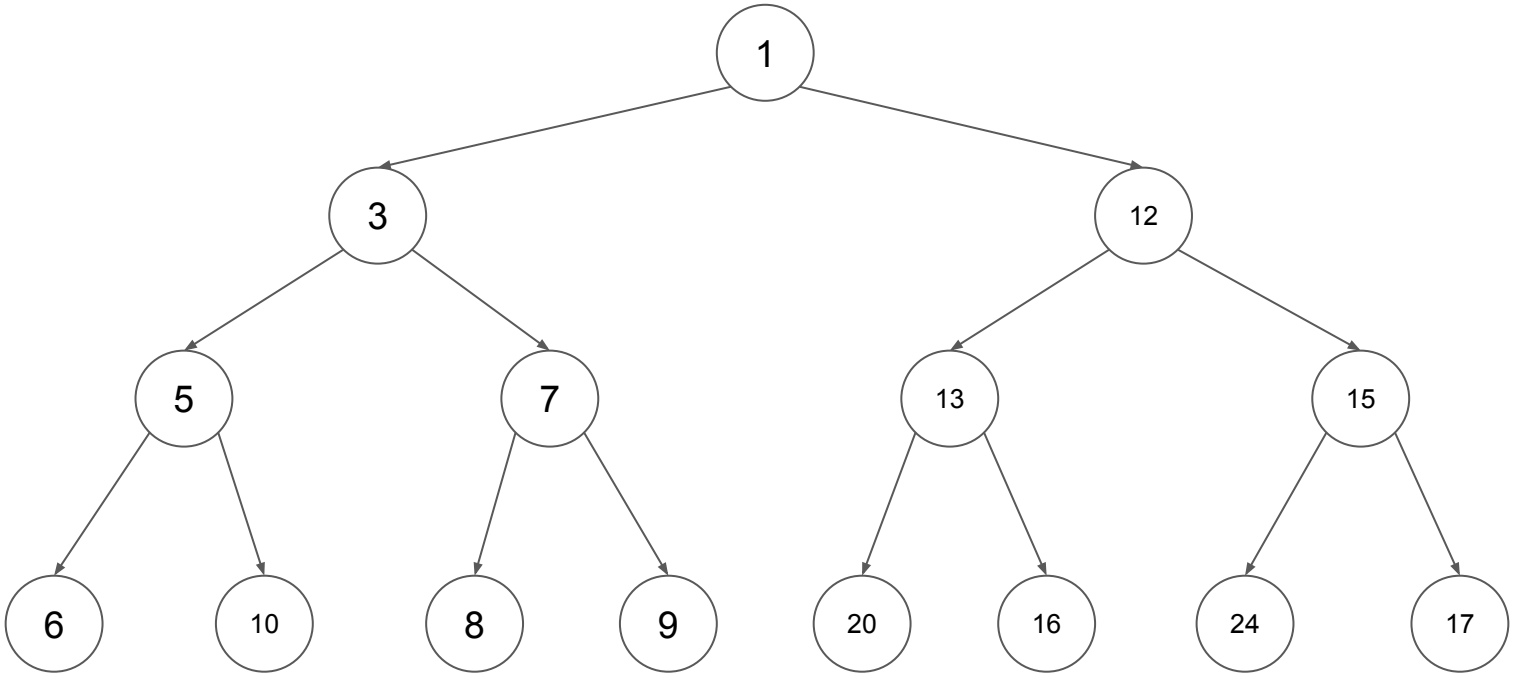
Heaps


1
3
12
5
7
13
15
6
10
8
9
20
16
24
17

(Binary) Heap

Min-Heap

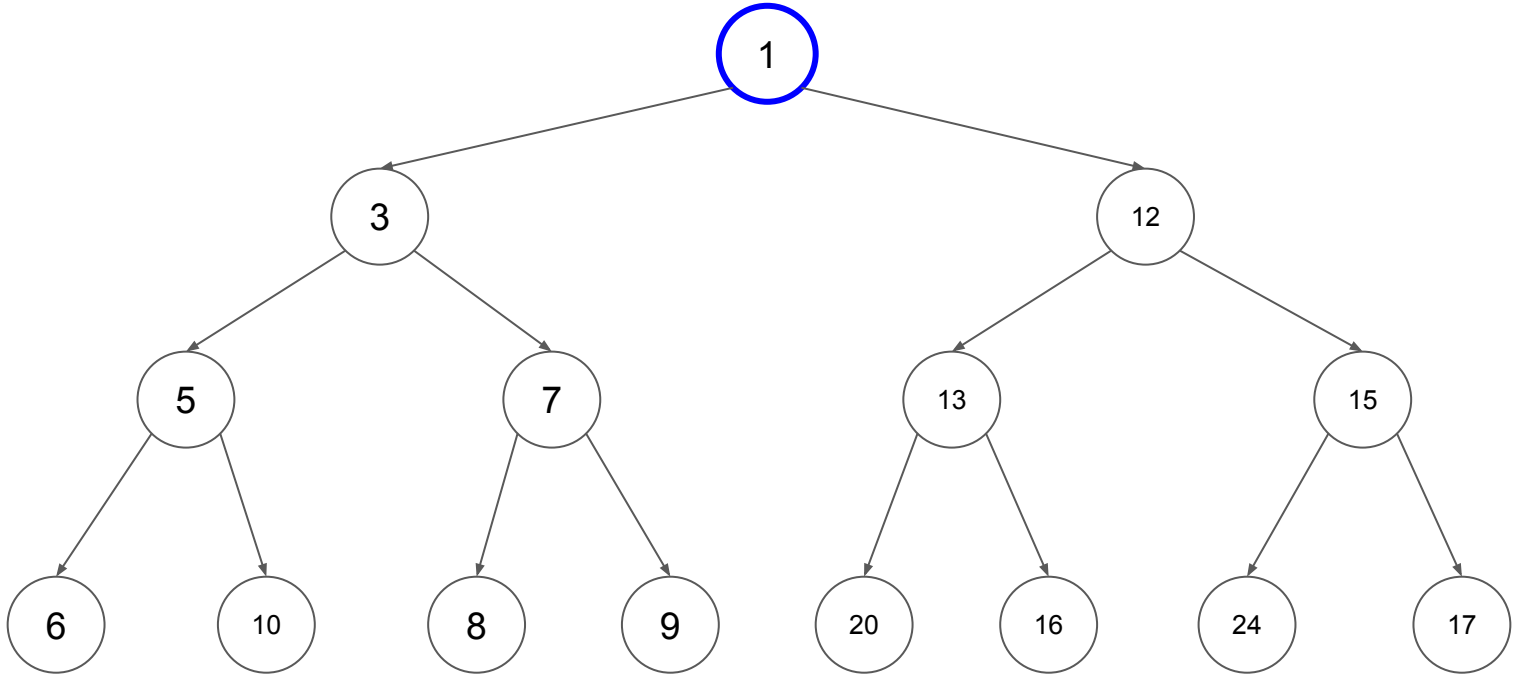
Min-Heap



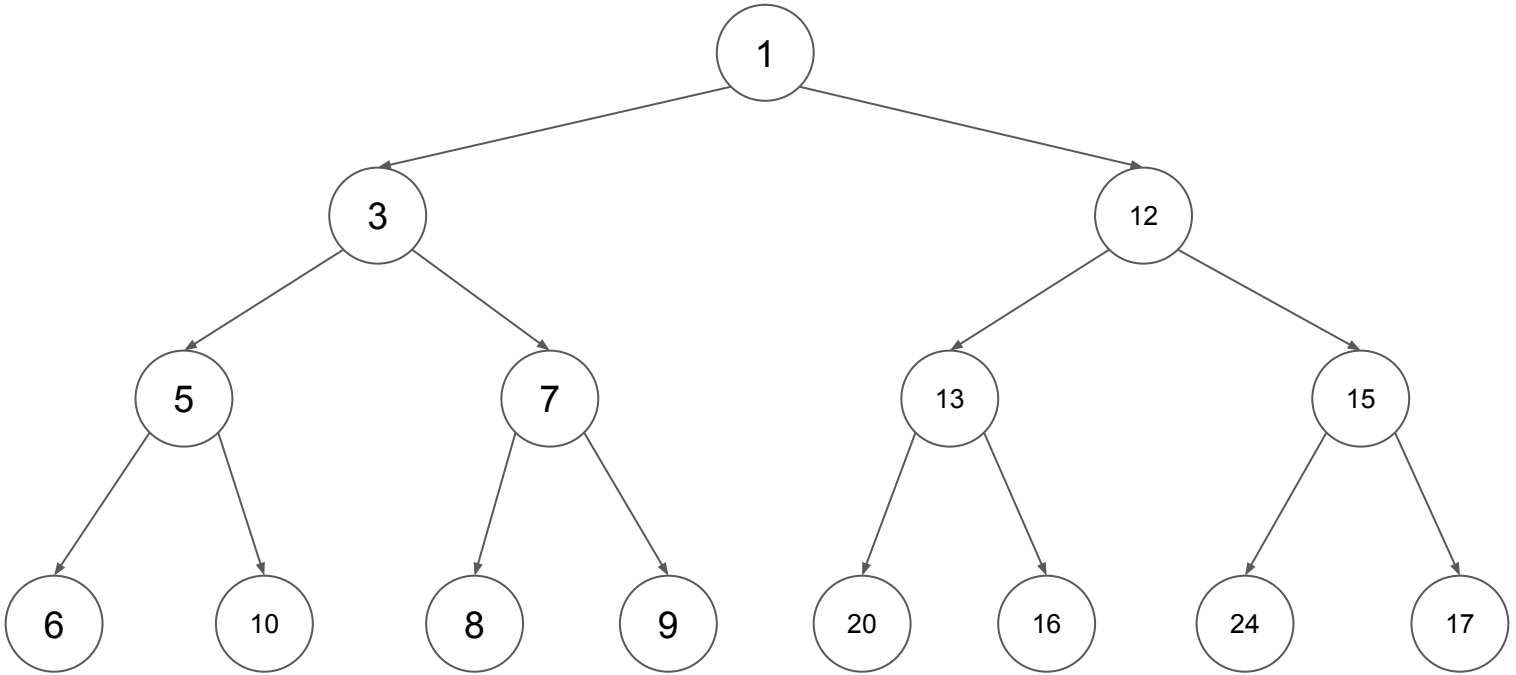
Min-Heap

- insert(item)
- findMin()
- extractMin()

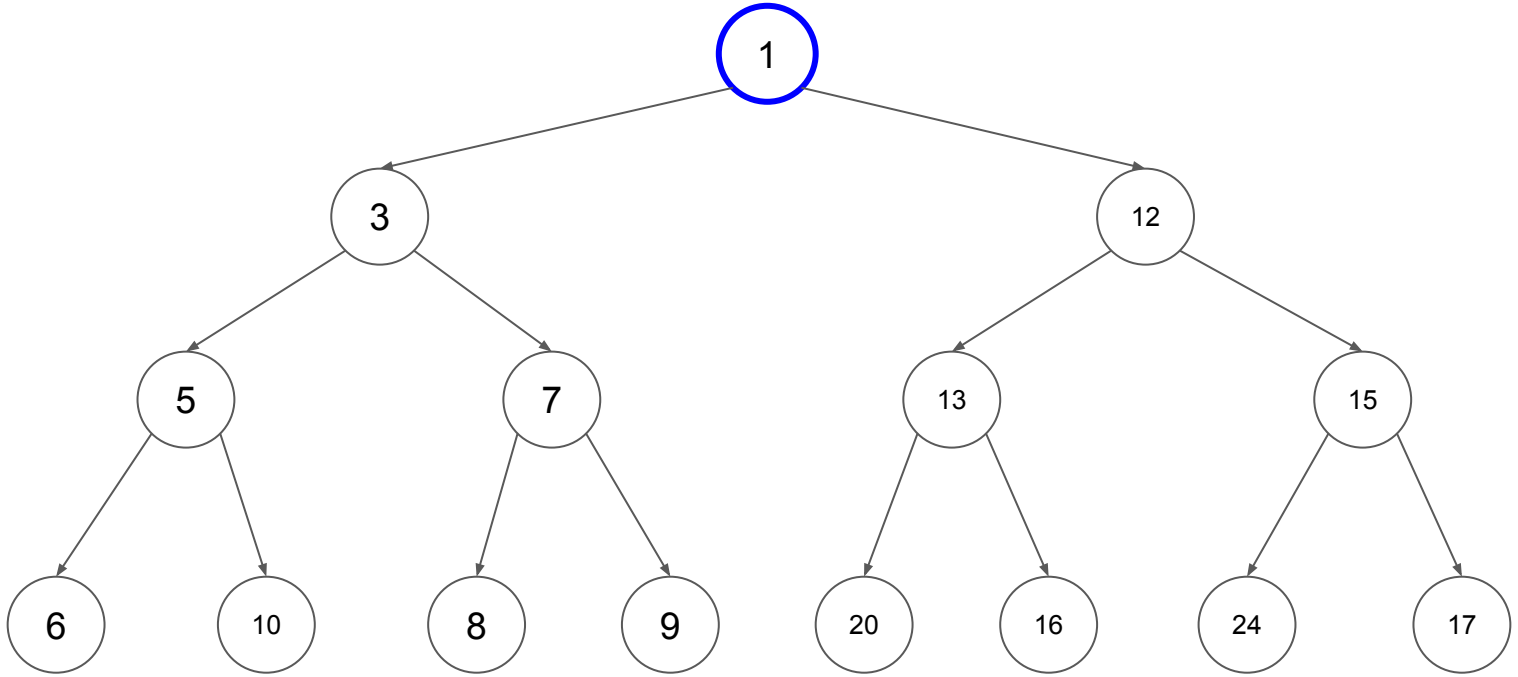
Min-Heap
findMin()



Min-Heap
extractMin()

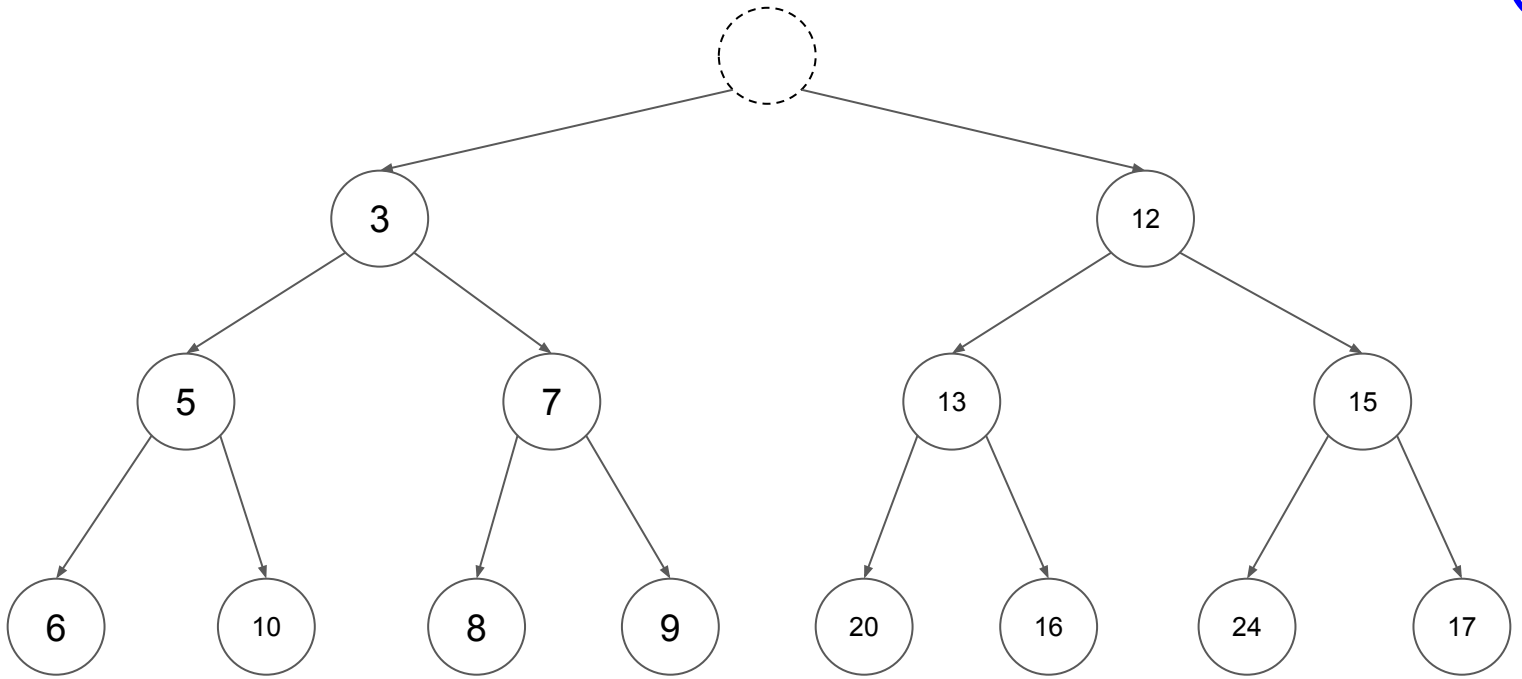


Min-Heap
extractMin()

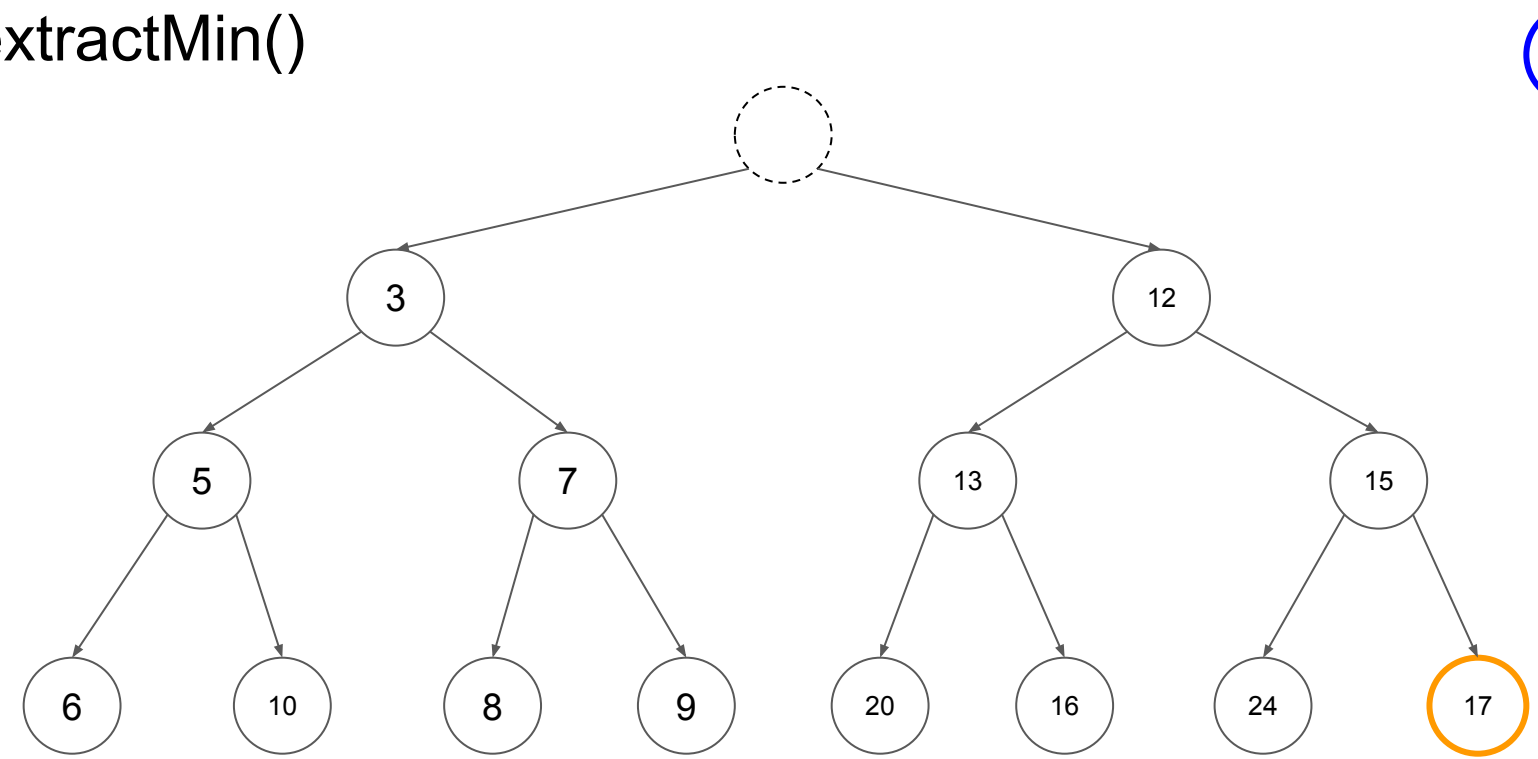


Min-Heap
extractMin()

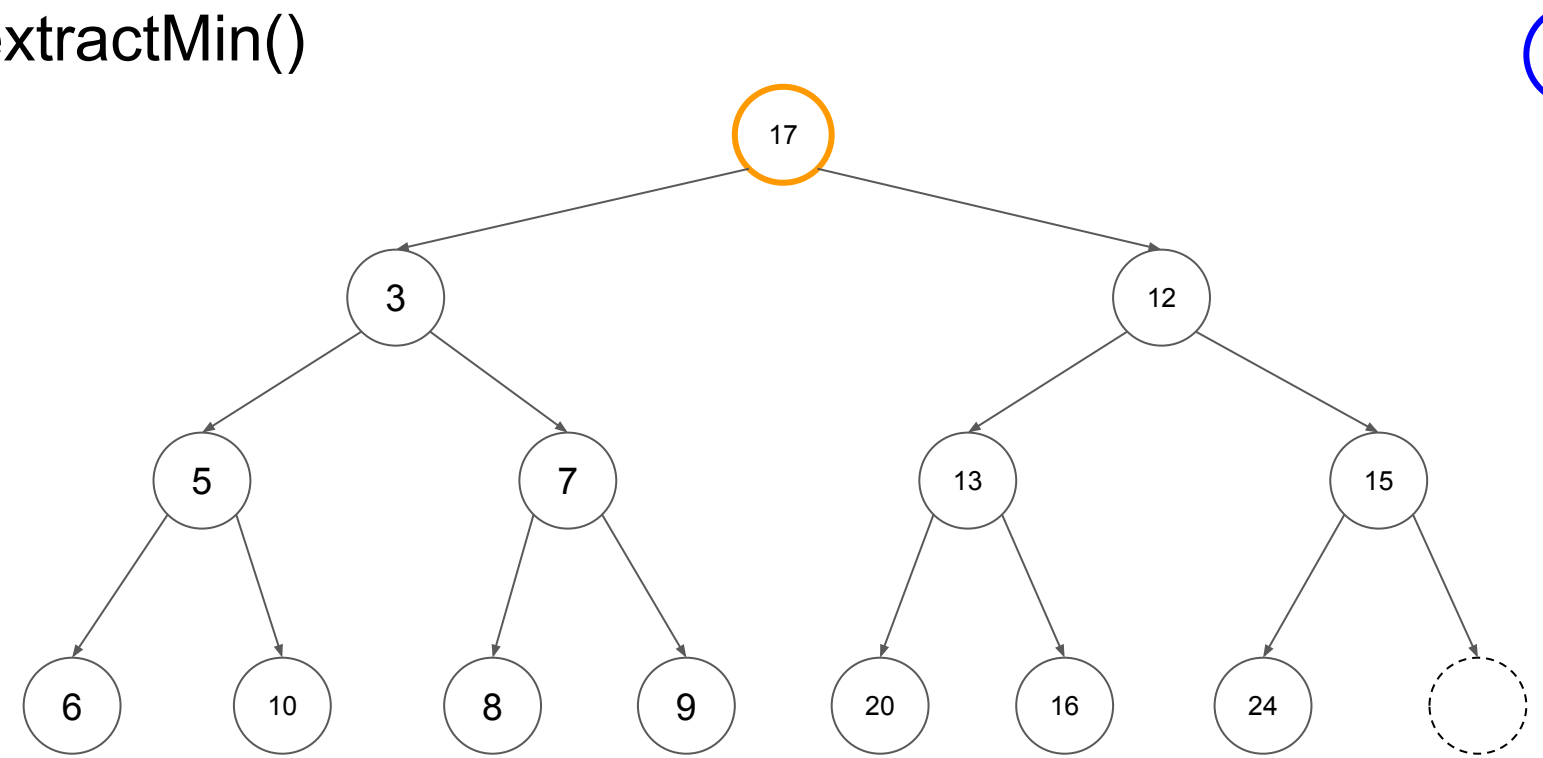
1



Min-Heap
extractMin()

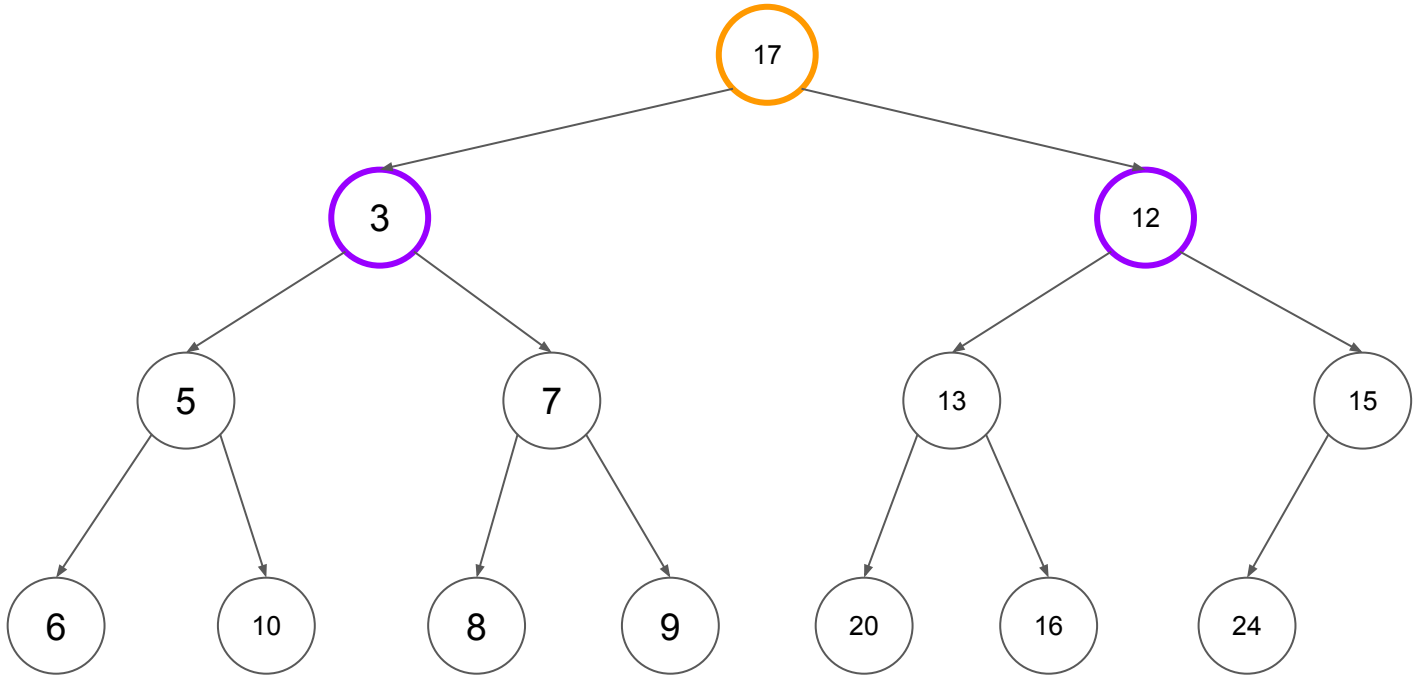


Min-Heap
extractMin()

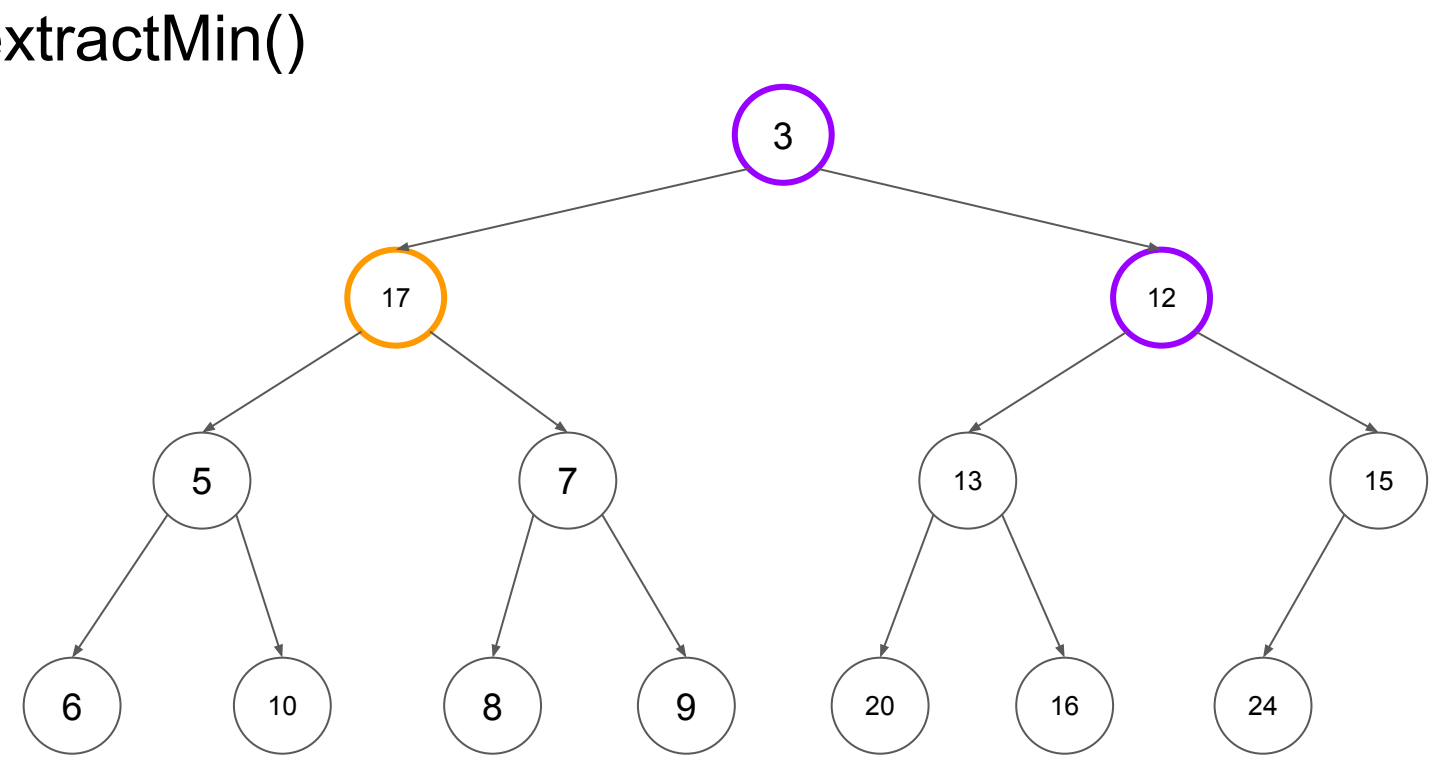


Min-Heap
extractMin()

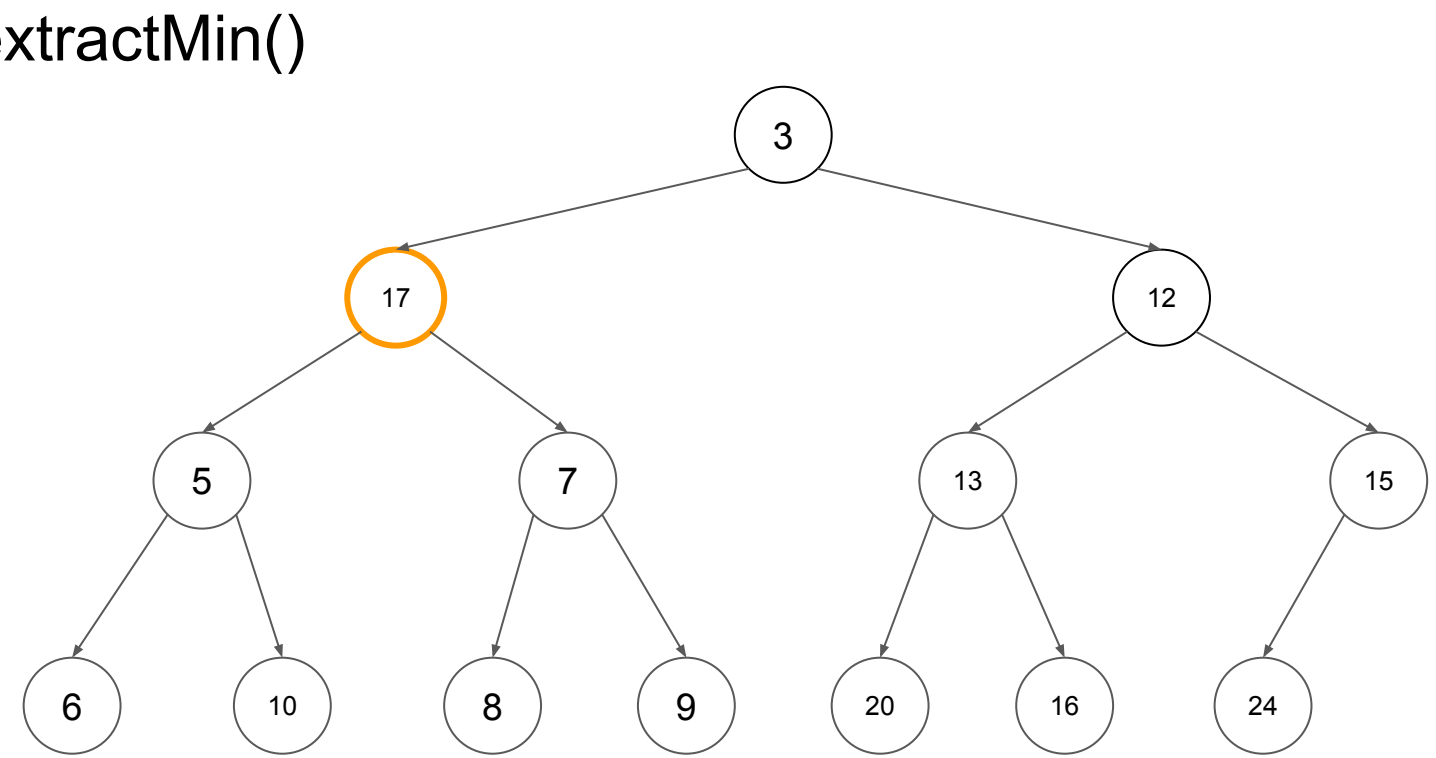
1



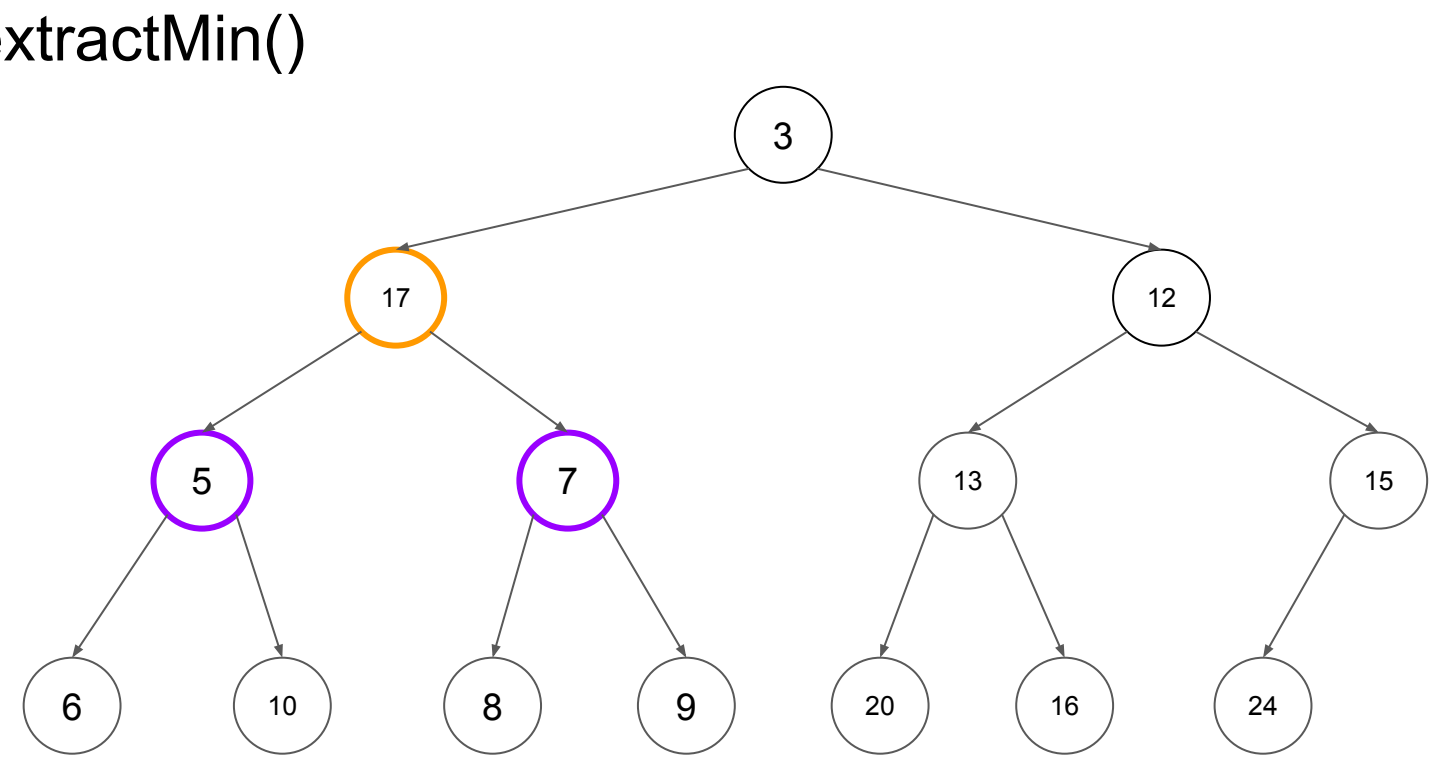
Min-Heap
extractMin()



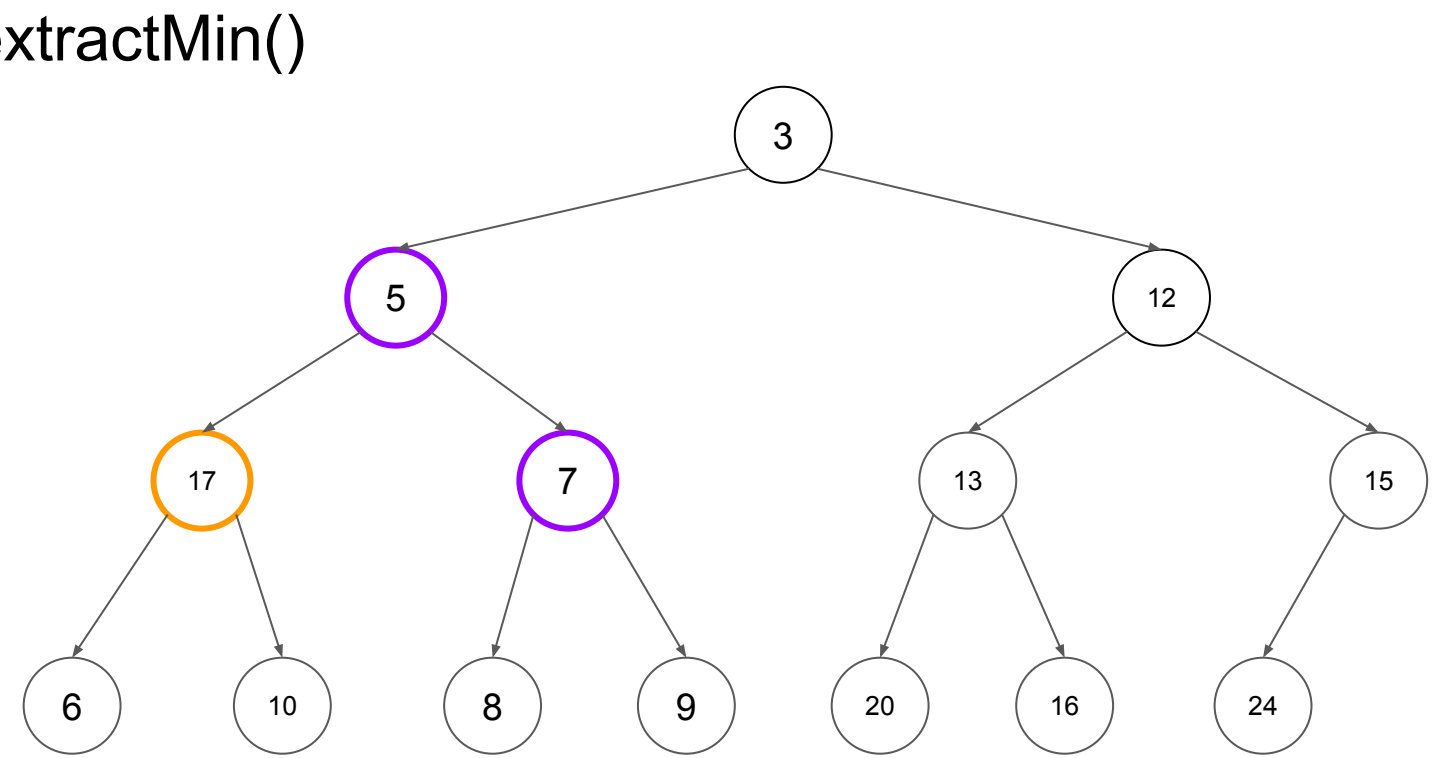
Min-Heap
extractMin()



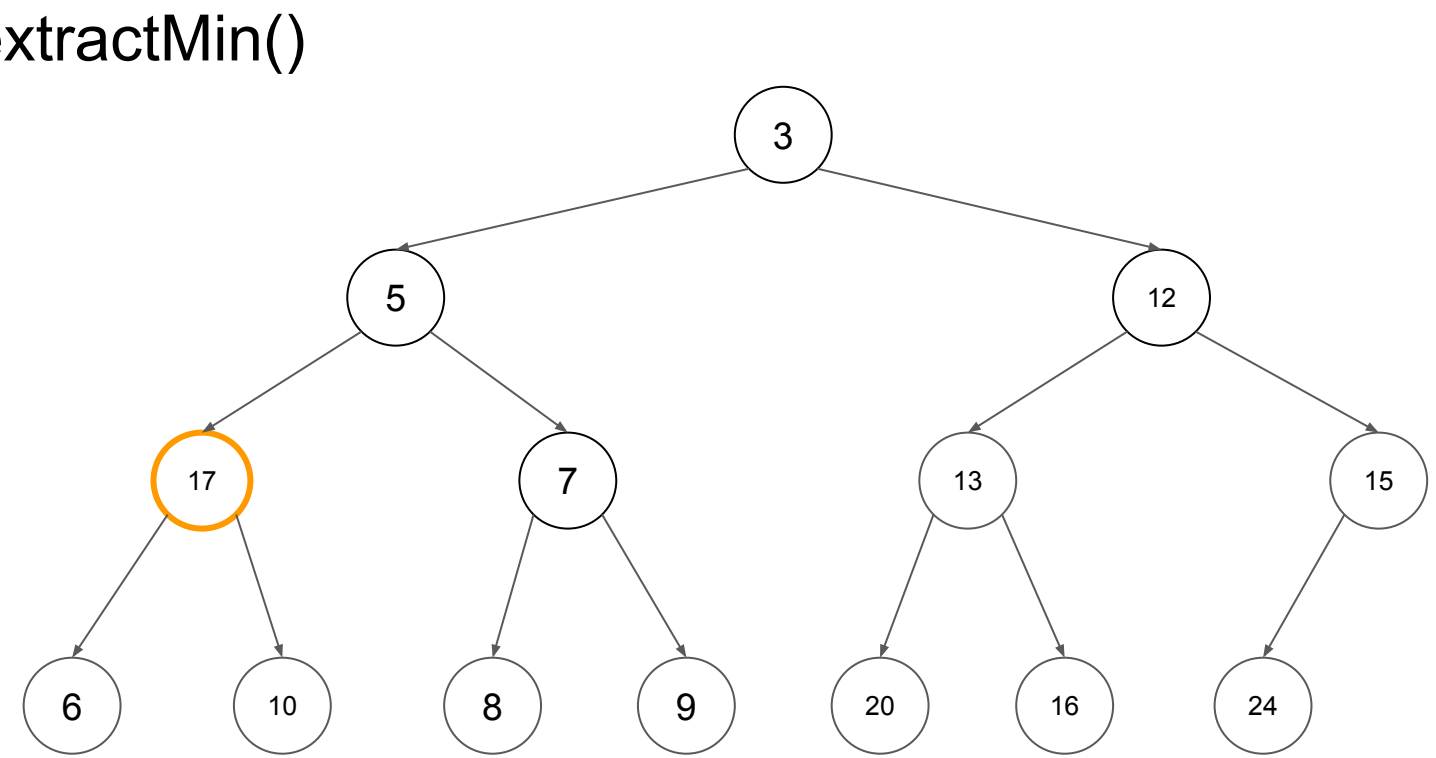
Min-Heap
extractMin()



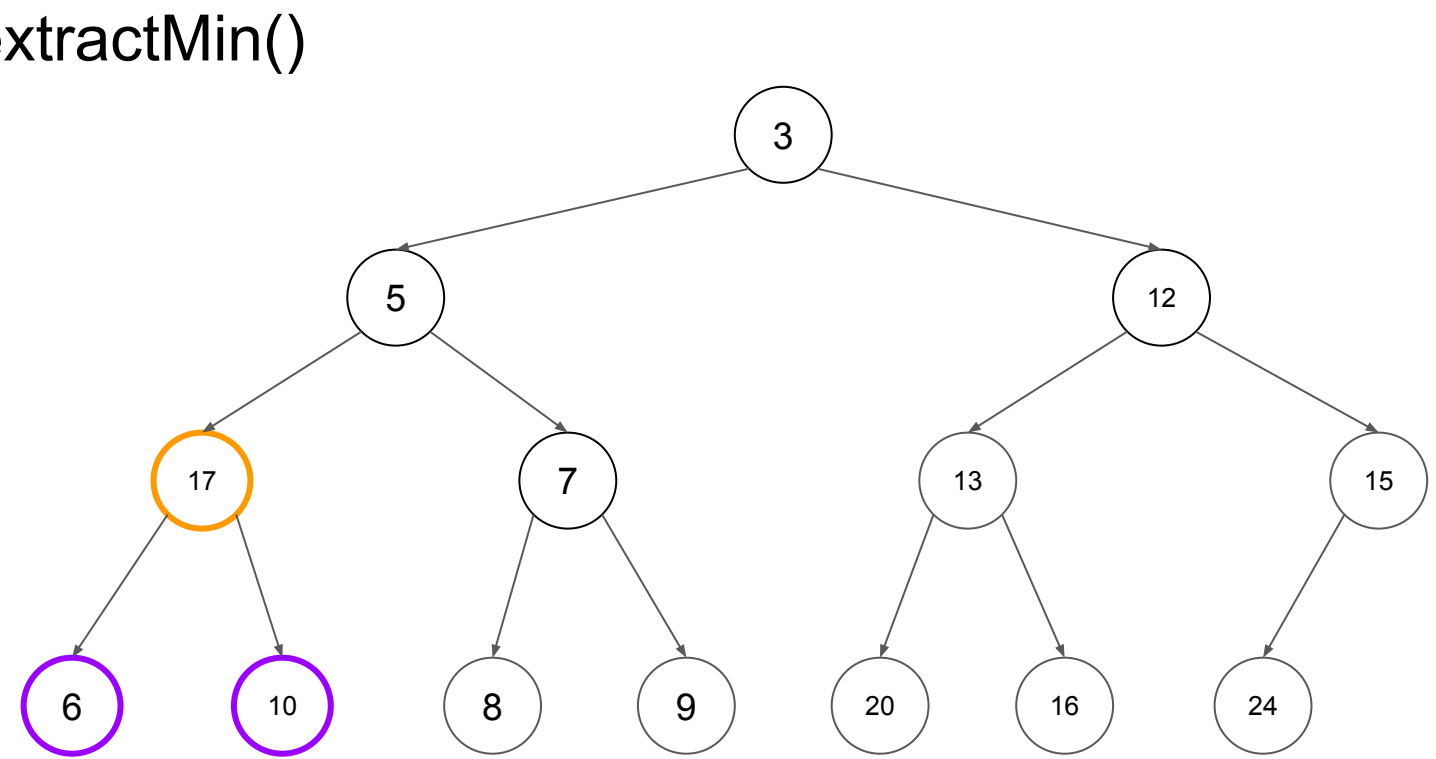
Min-Heap
extractMin()



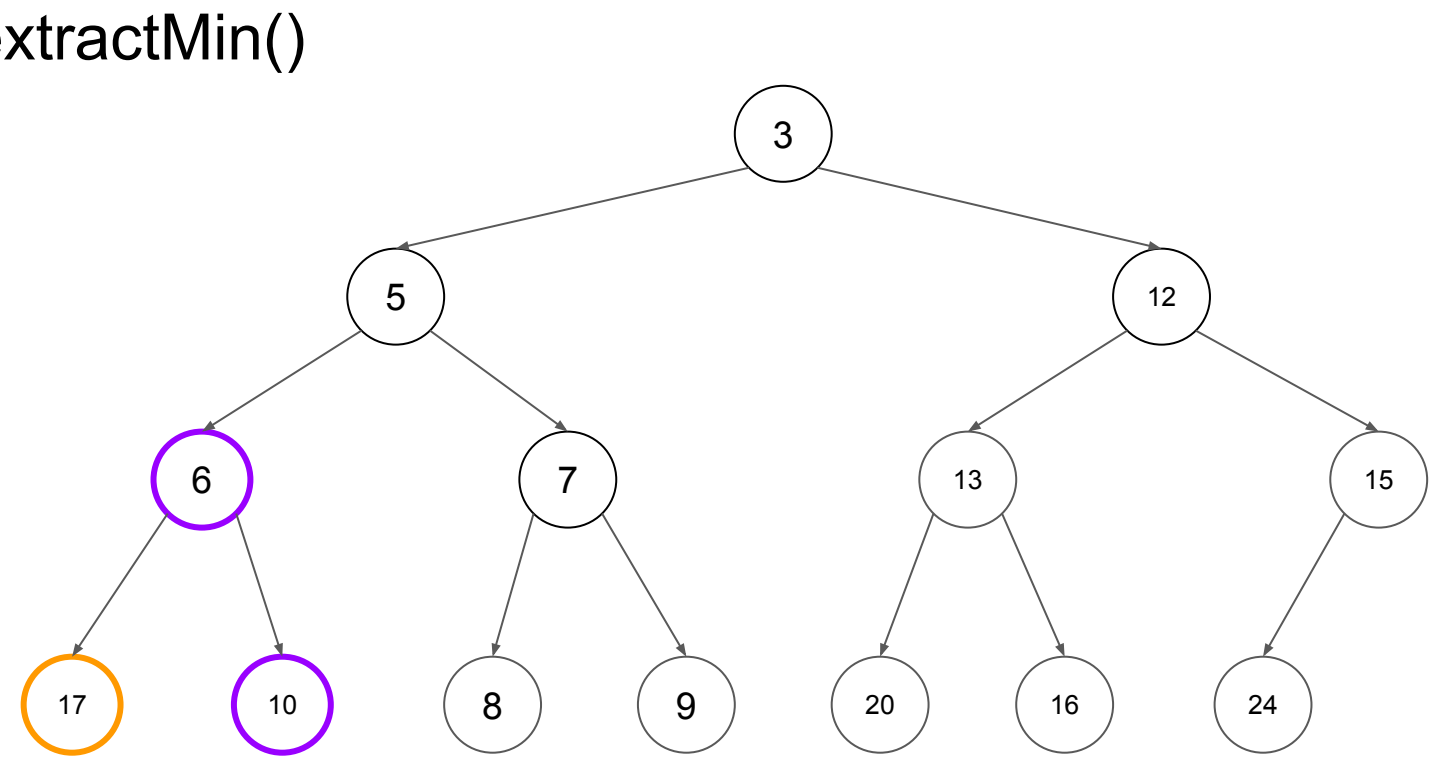
Min-Heap
extractMin()



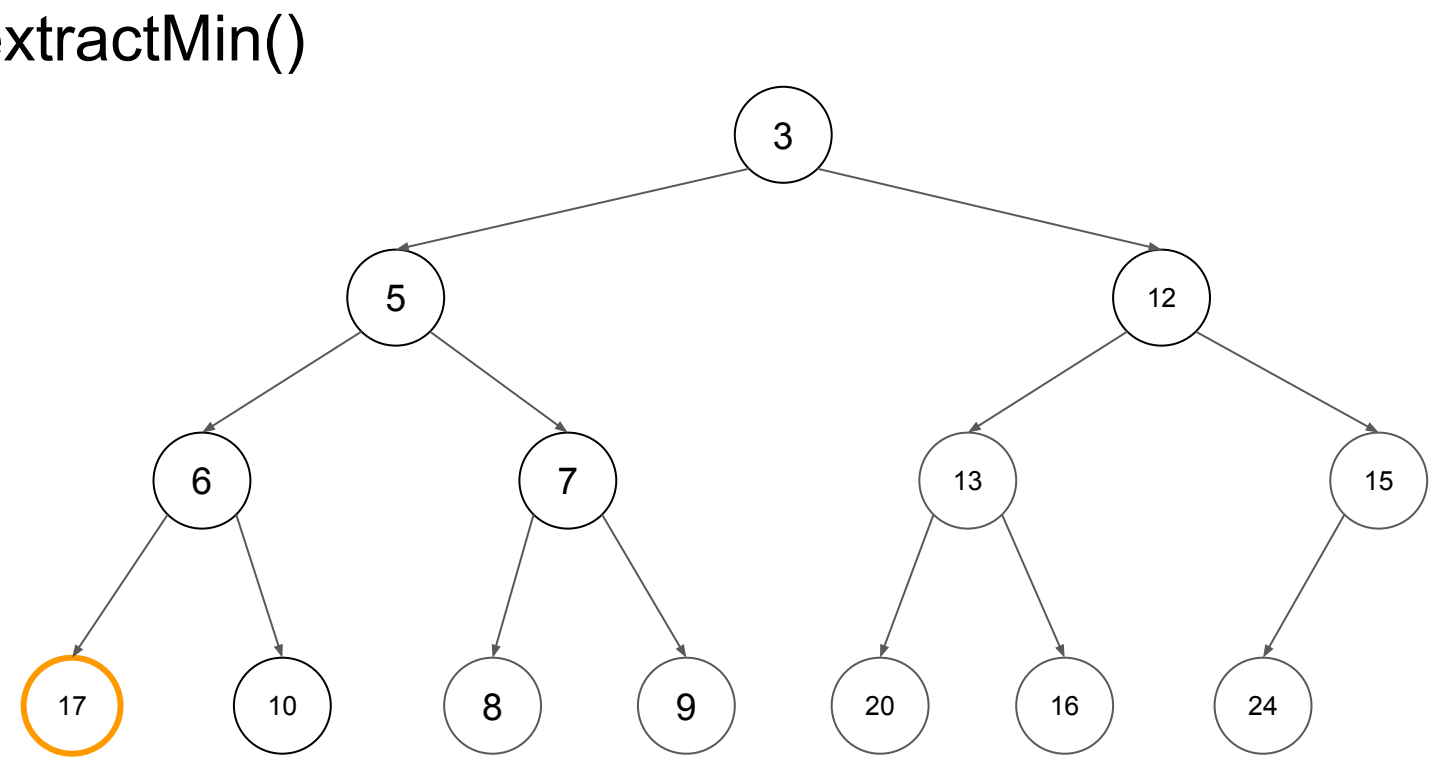
Min-Heap
extractMin()



Min-Heap
extractMin()

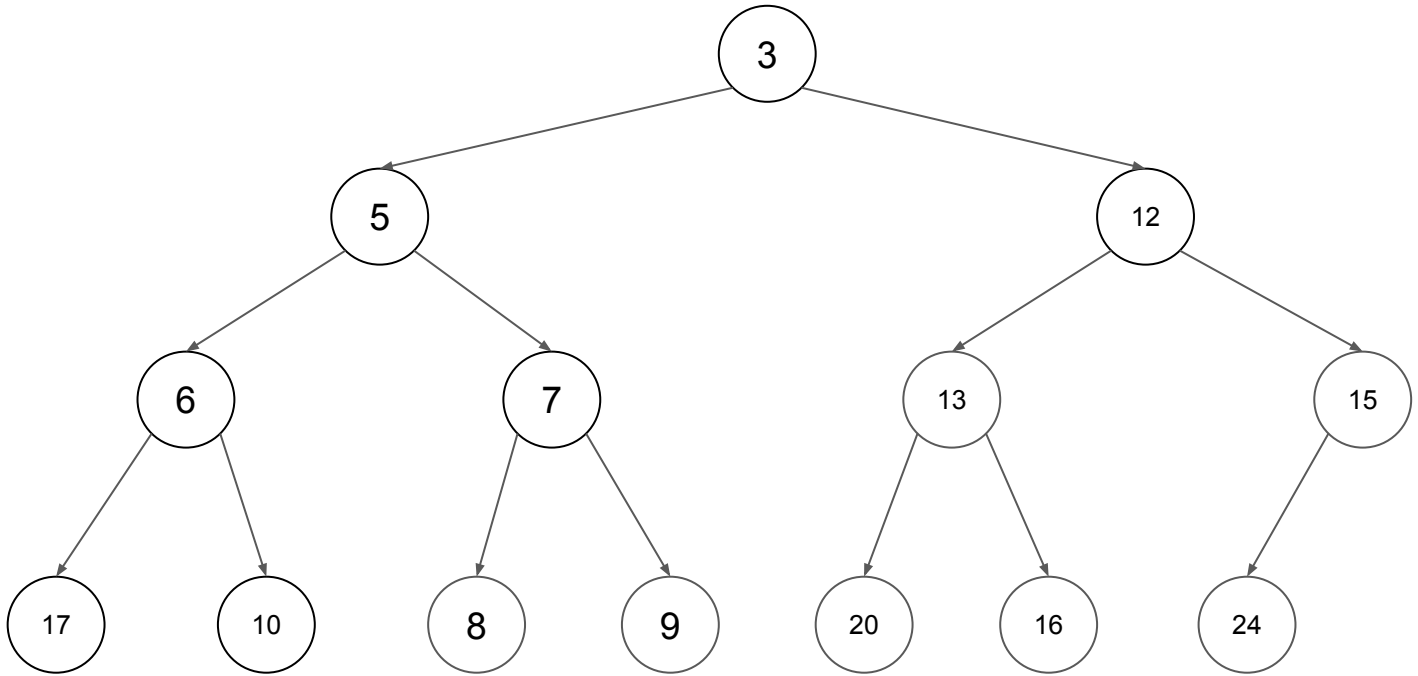


Min-Heap
extractMin()

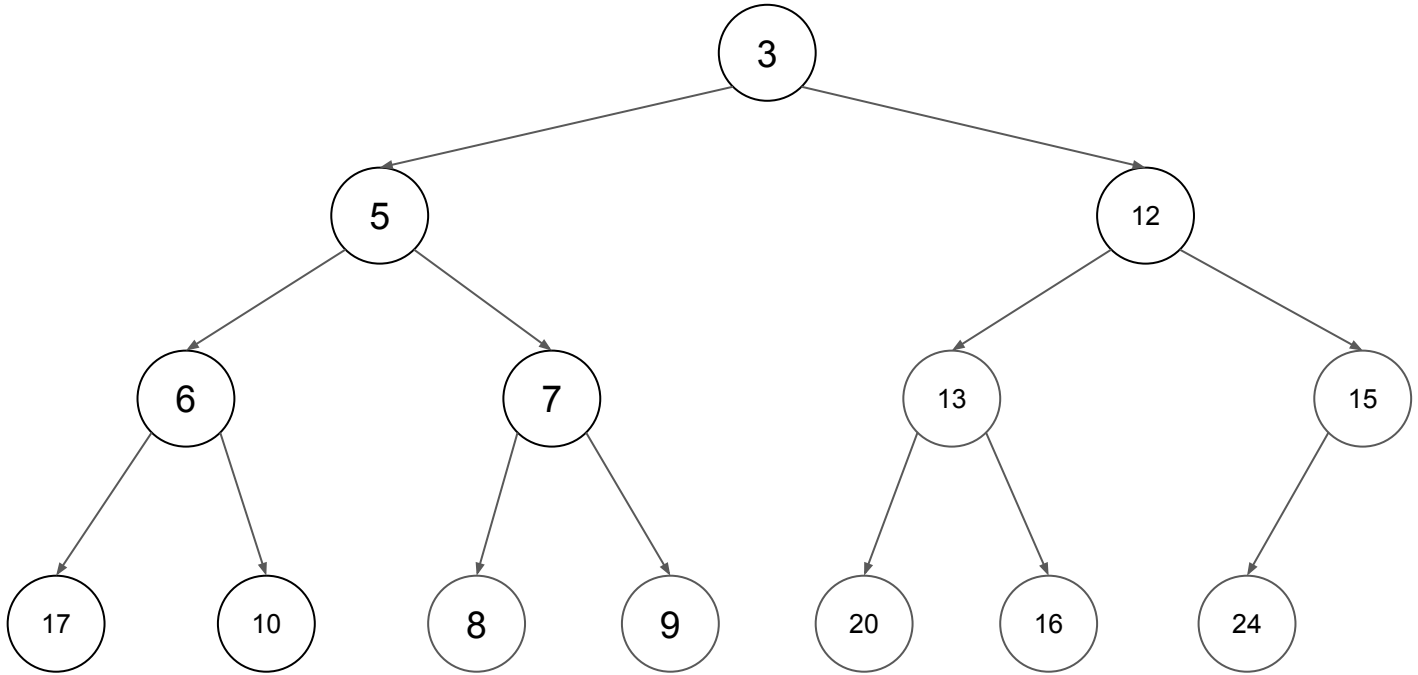


Min-Heap
extractMin()

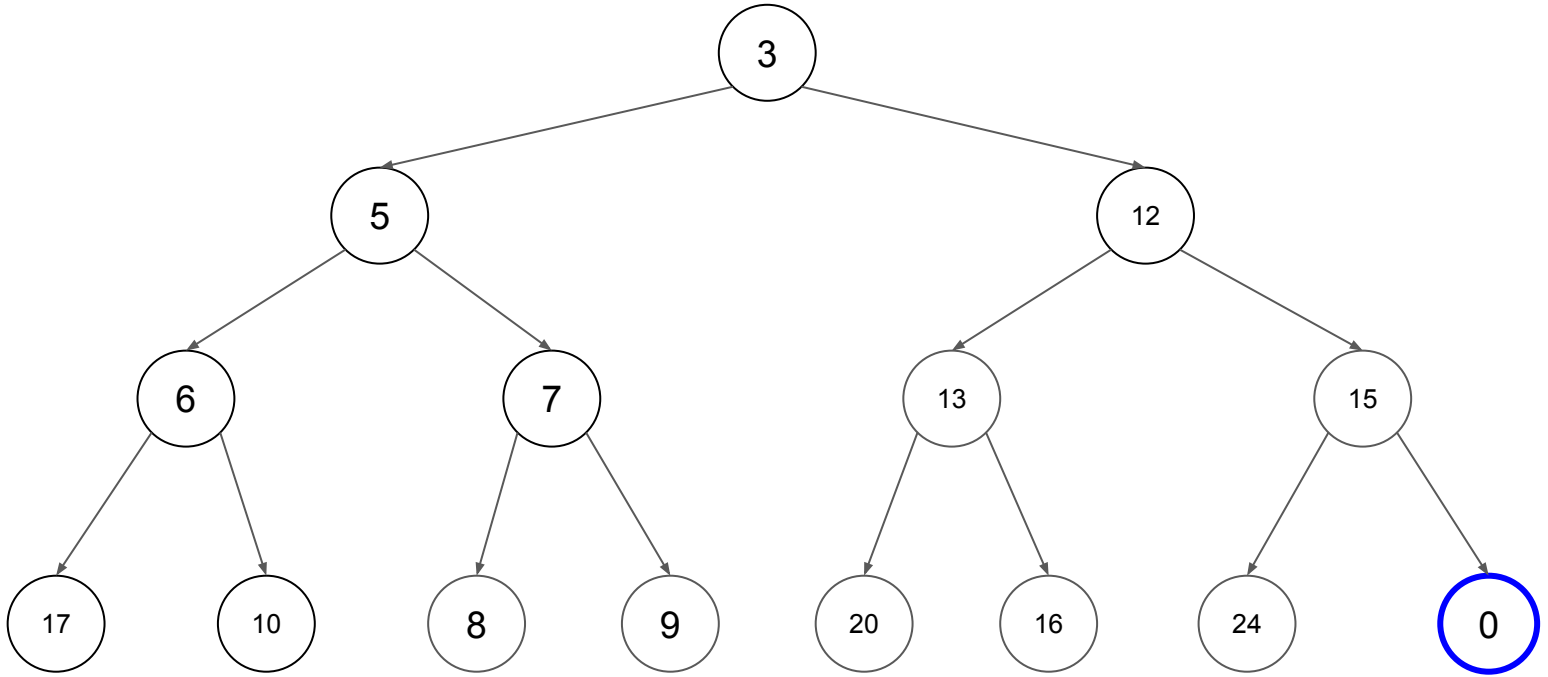
1



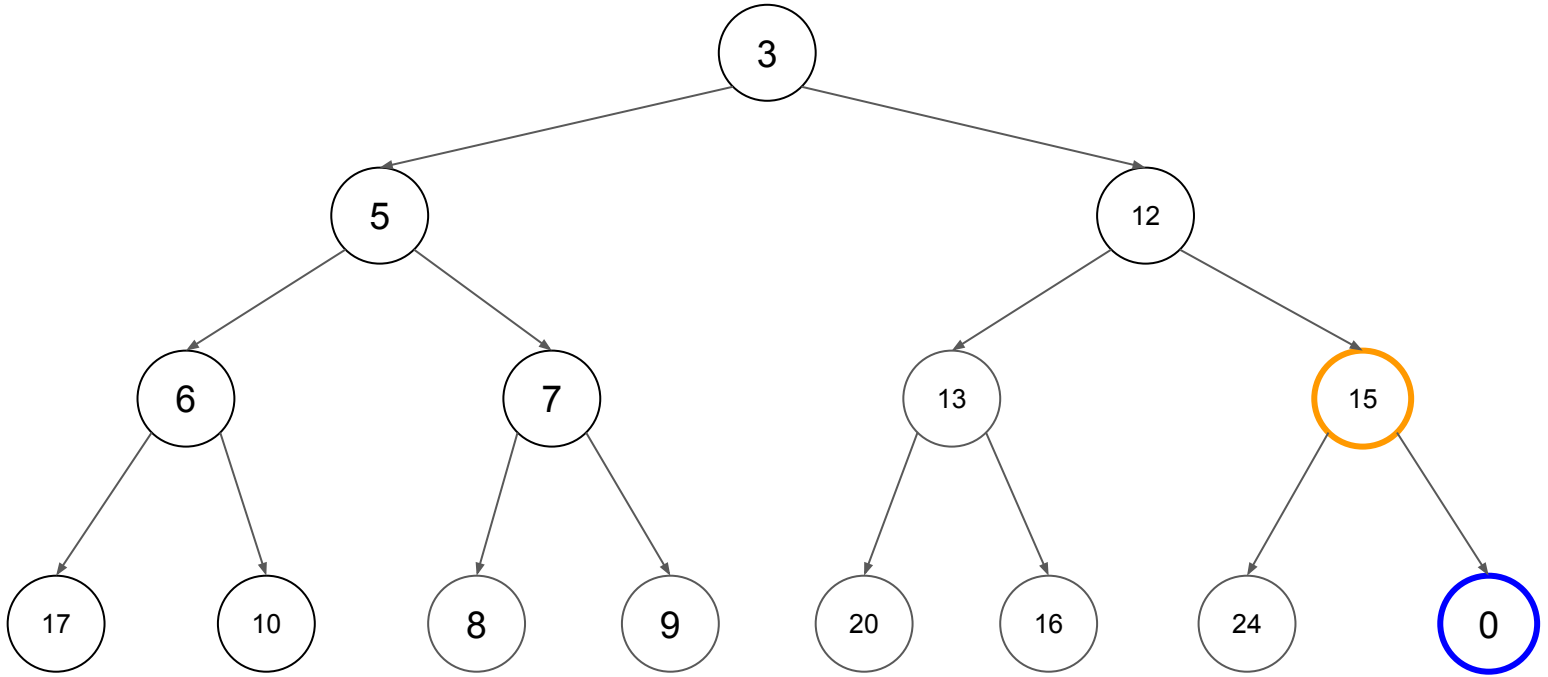
Min-Heap
insert(0)



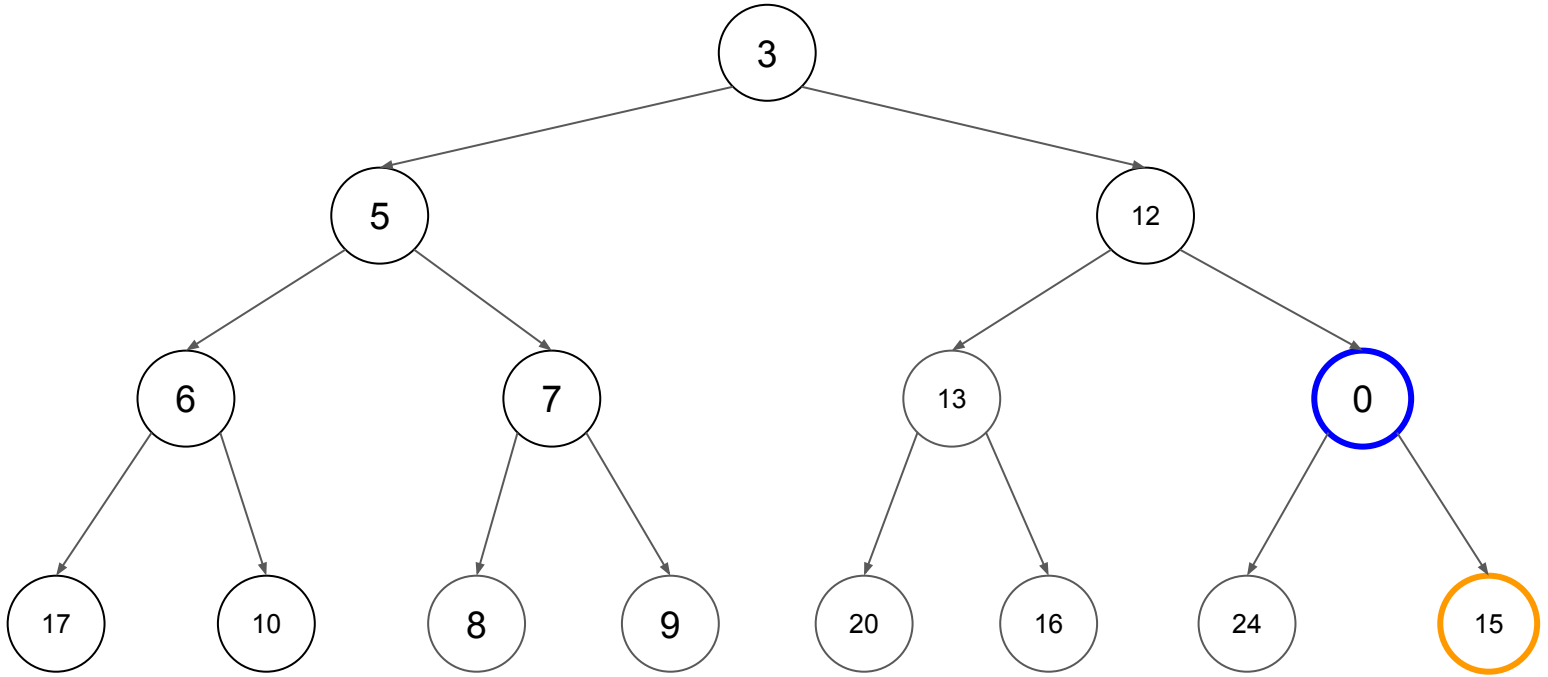
Min-Heap
insert(0)



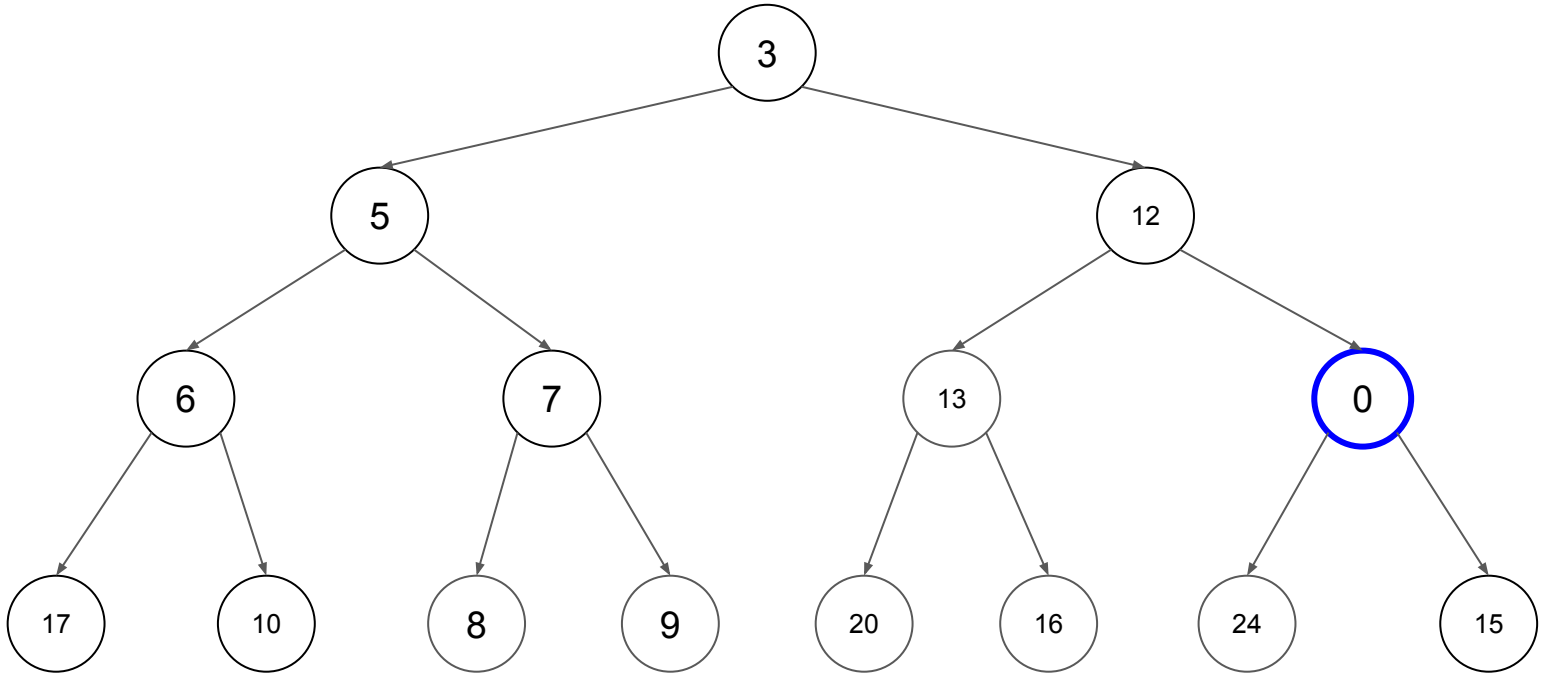
Min-Heap
insert(0)



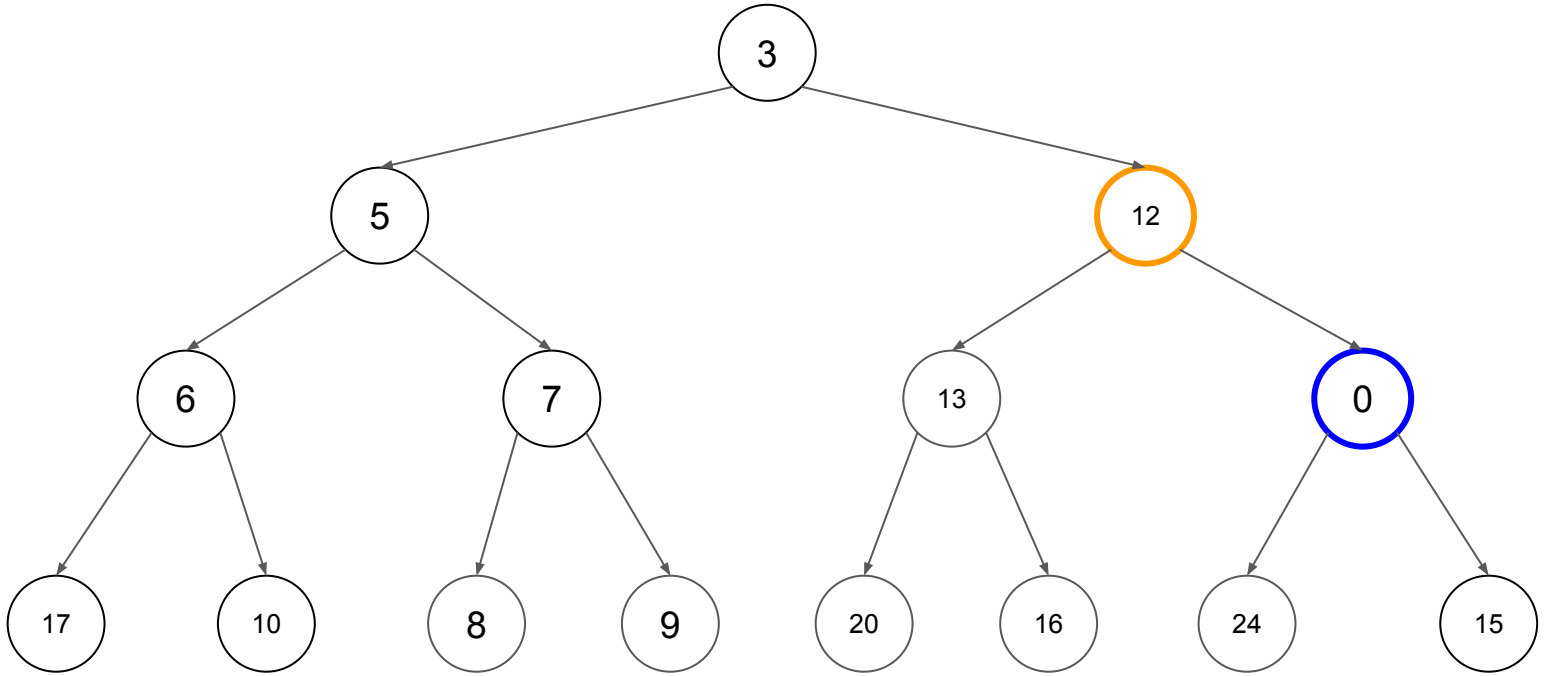
Min-Heap
insert(0)



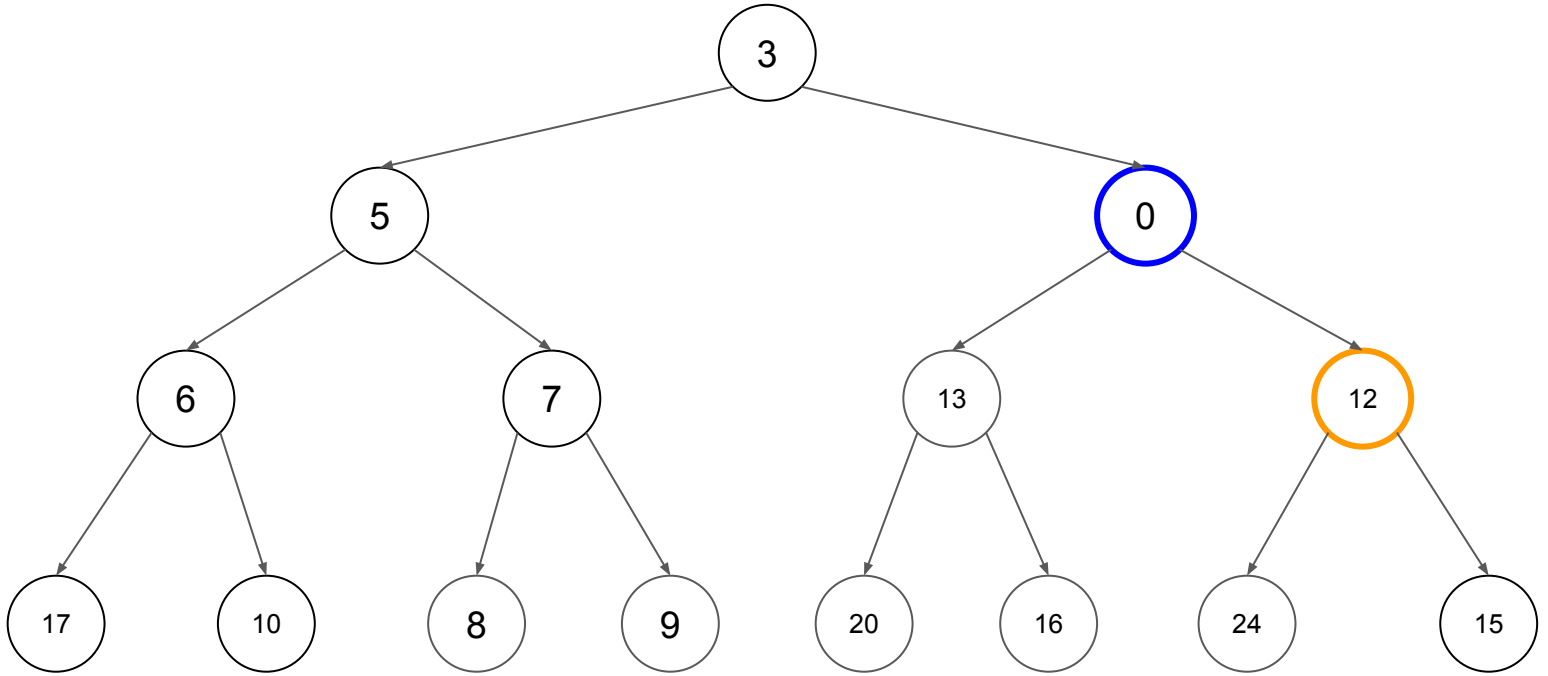
Min-Heap
insert(0)



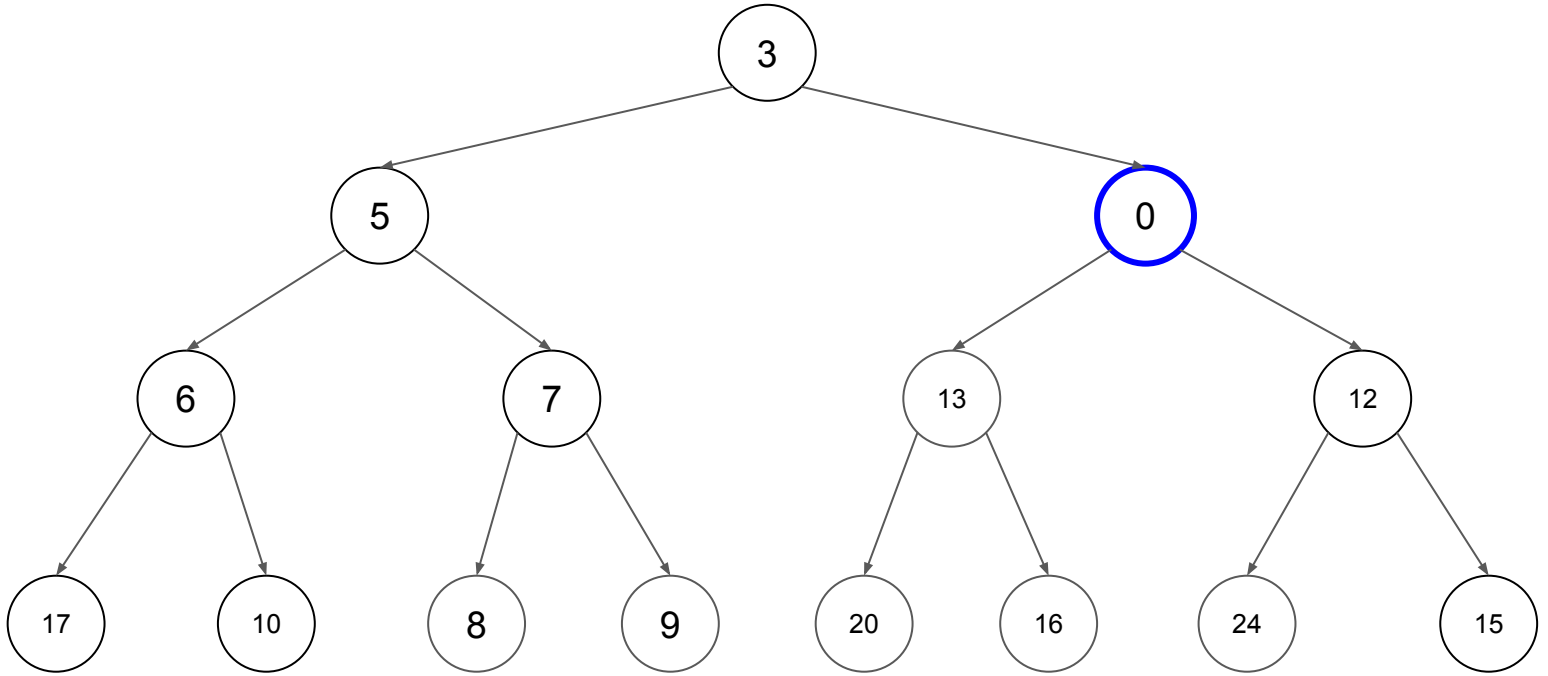
Min-Heap
insert(0)



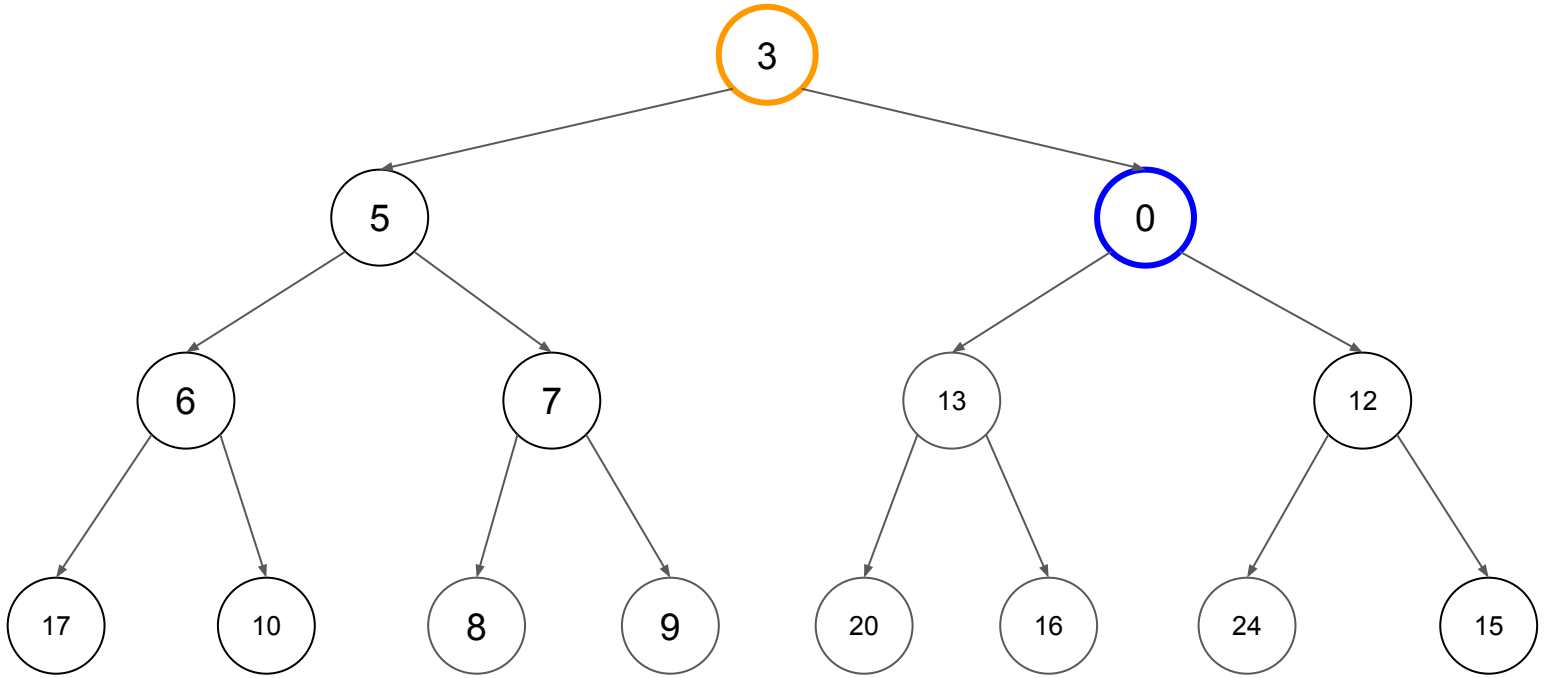
Min-Heap
insert(0)



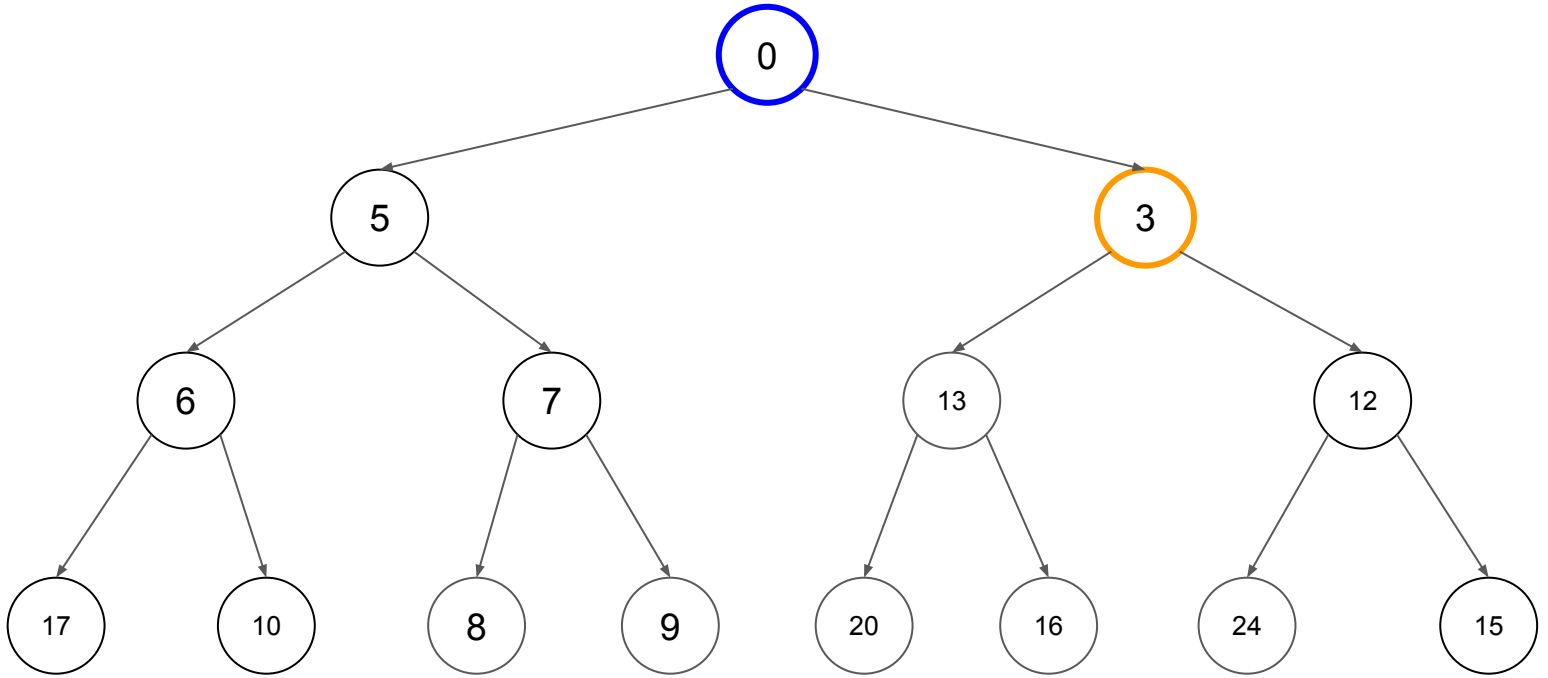
Min-Heap
insert(0)



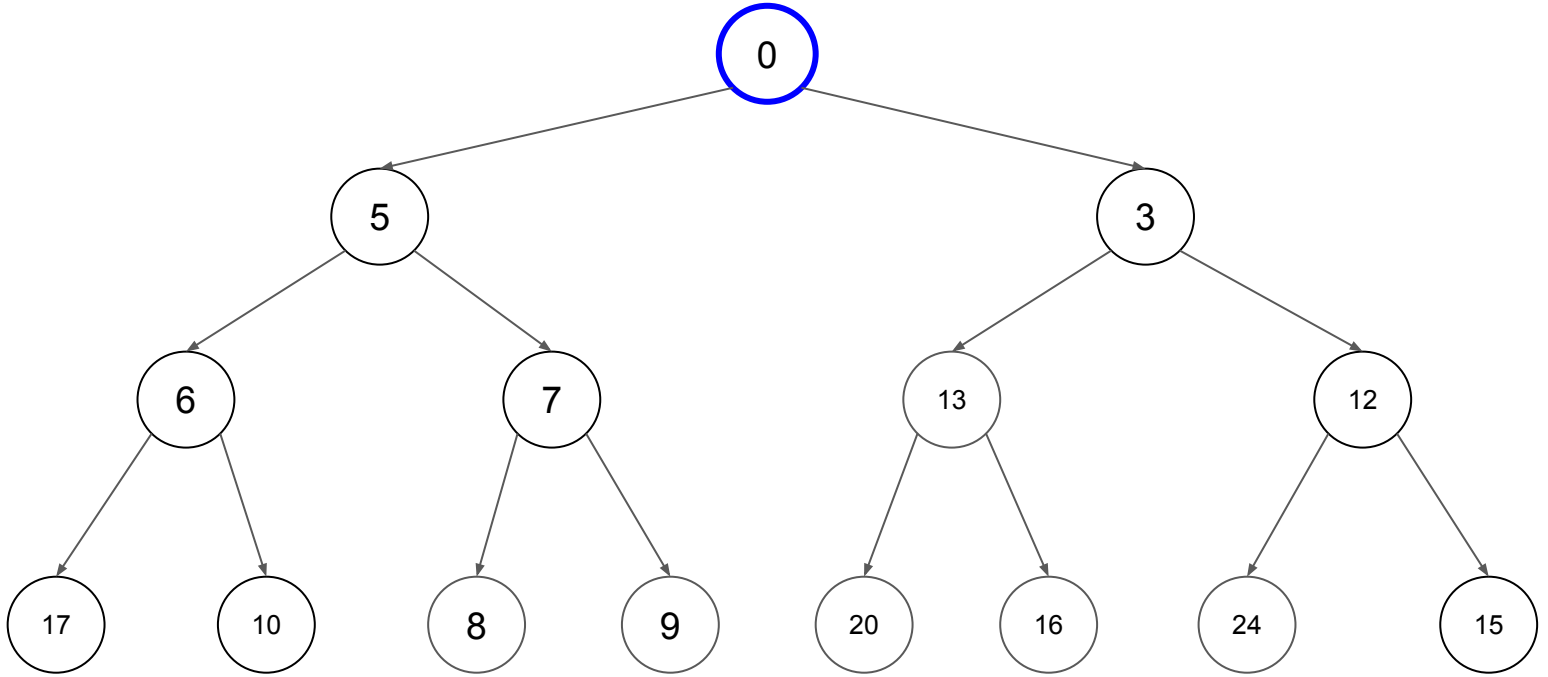
Min-Heap
insert(0)



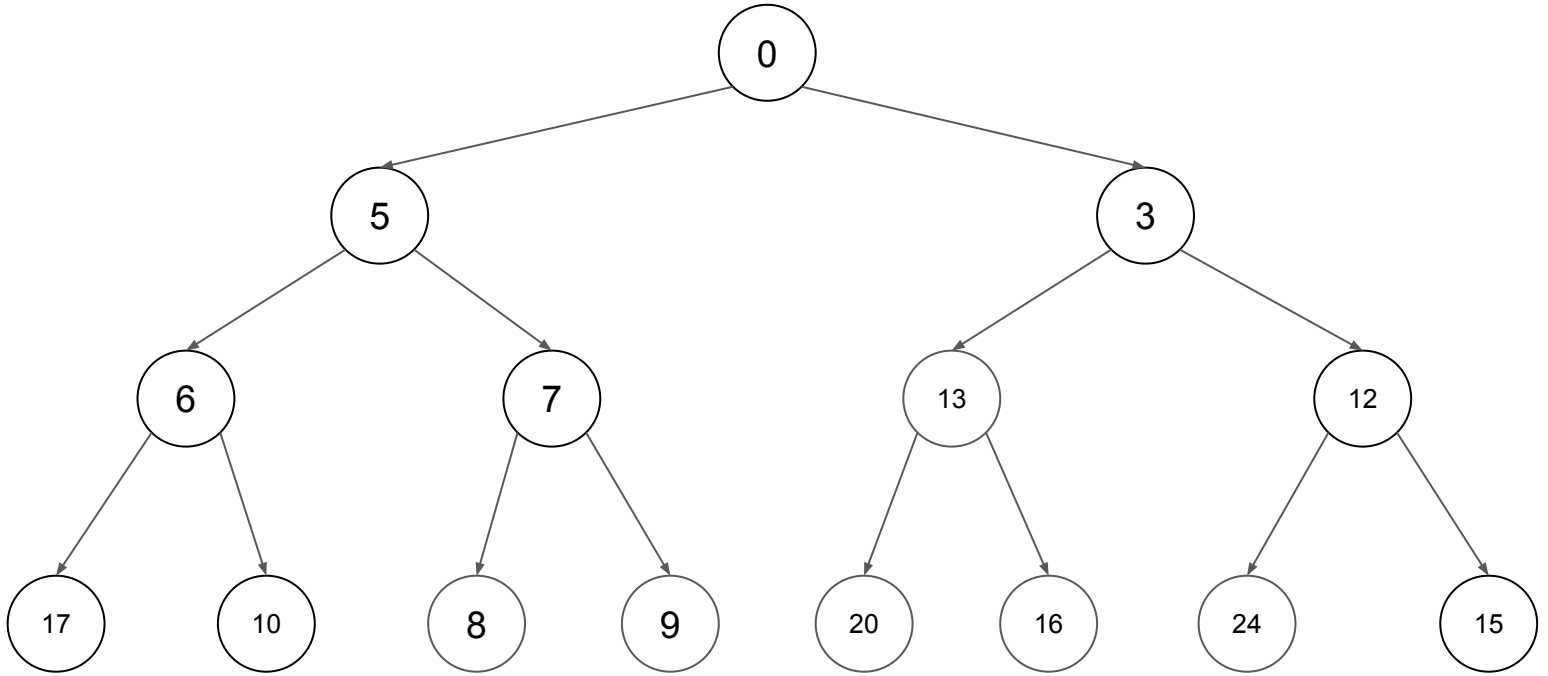
Min-Heap
insert(0)



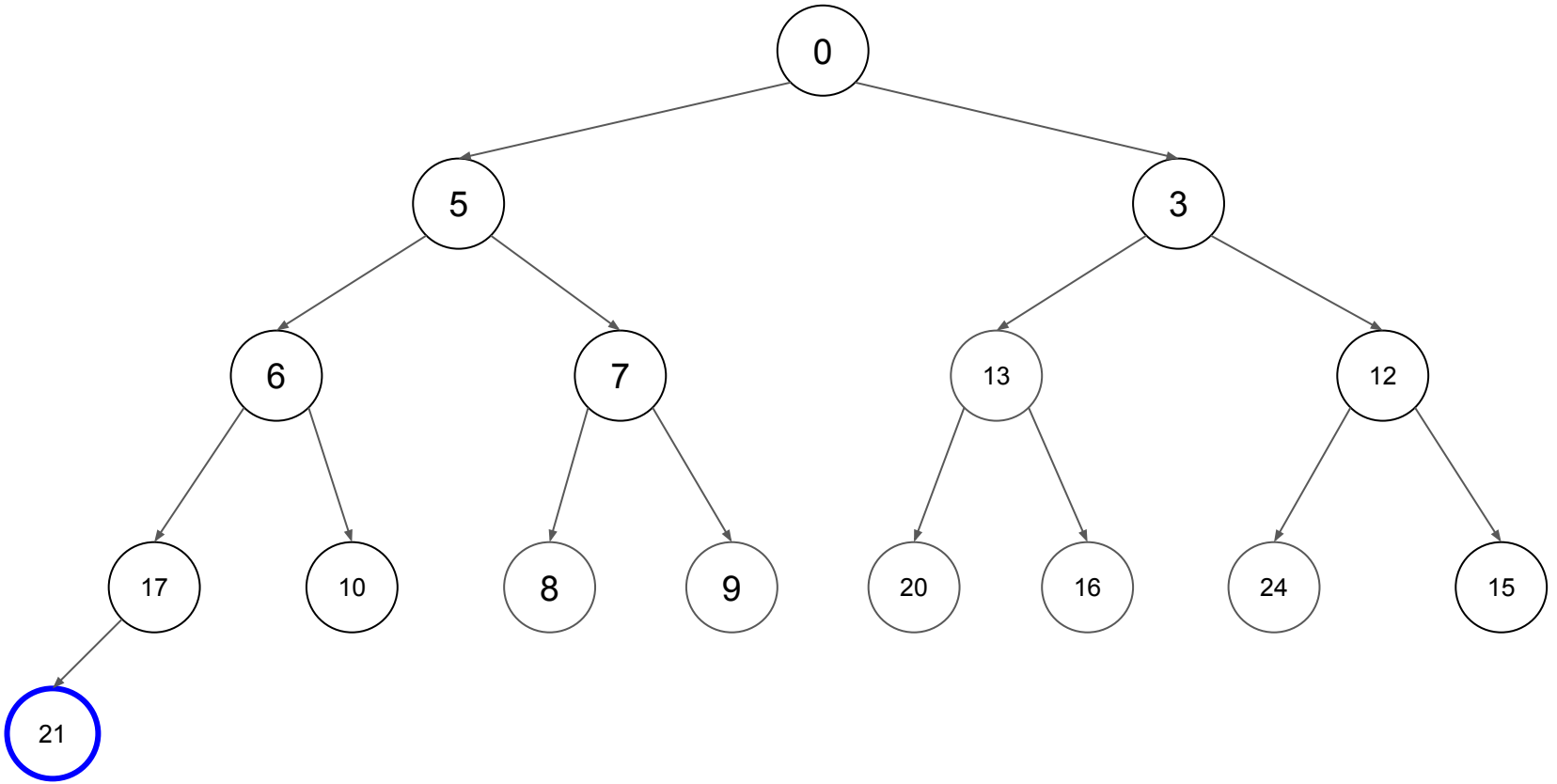
Min-Heap
insert(0)



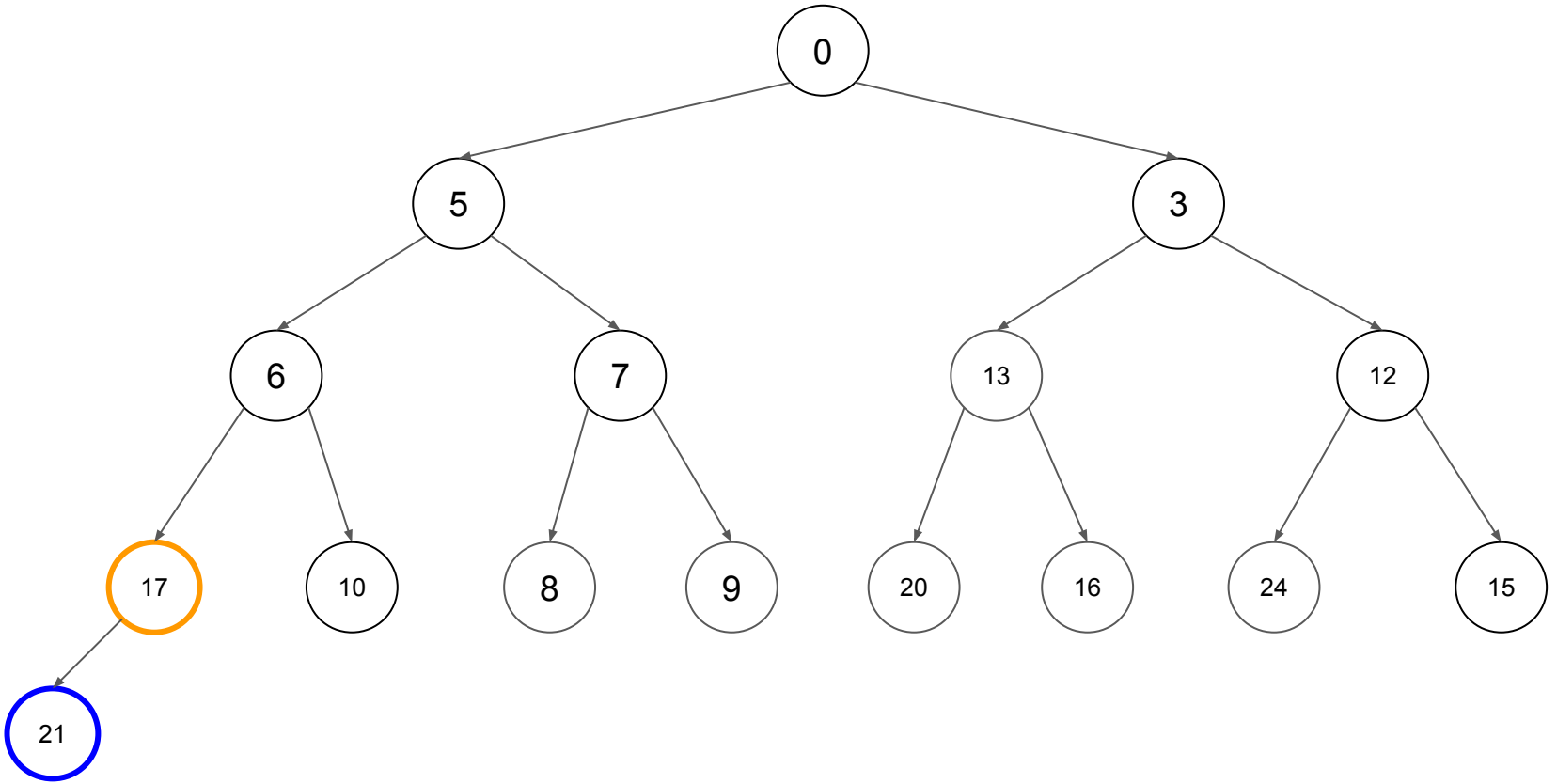
Min-Heap
insert(0)



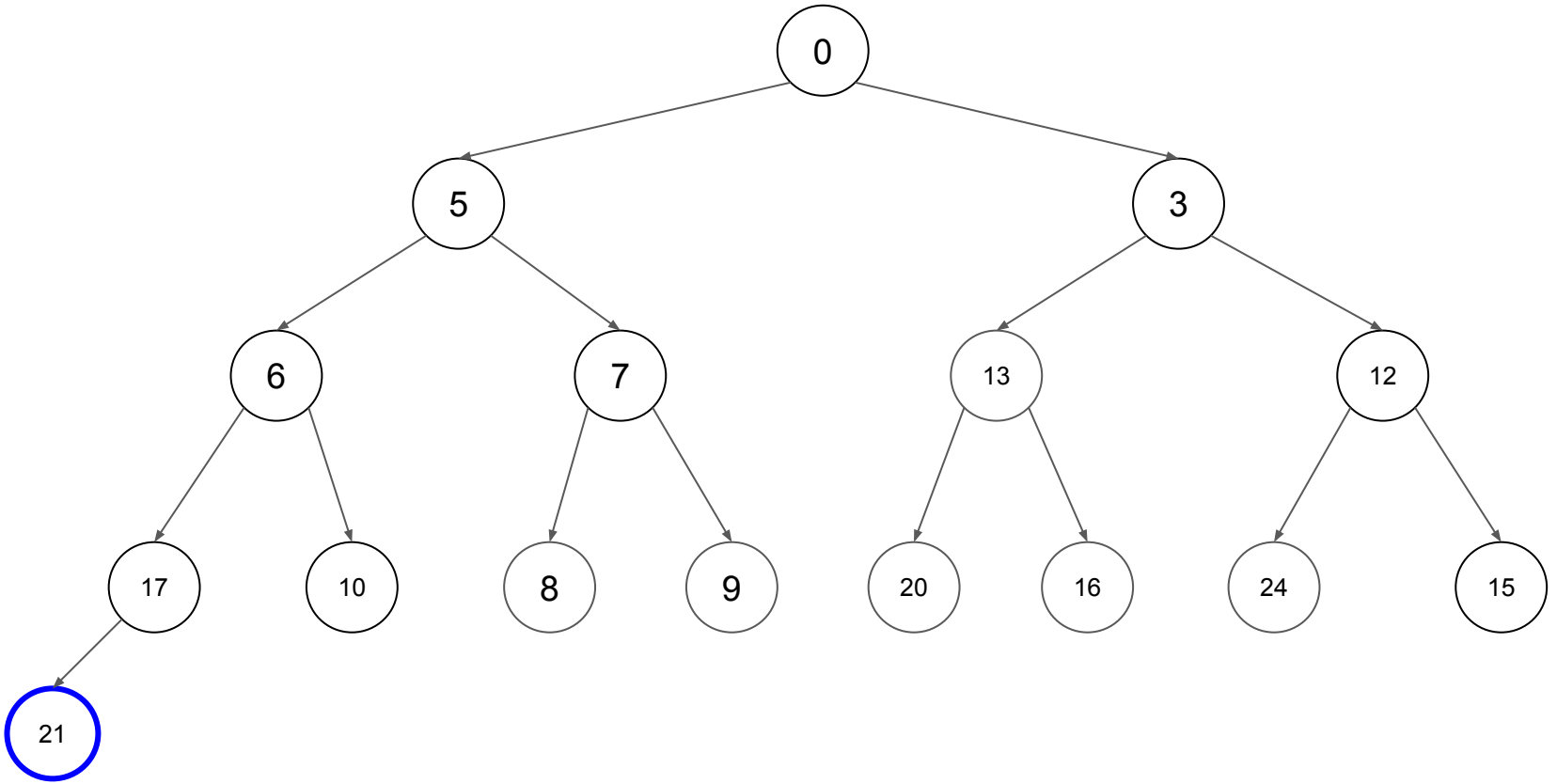
Min-Heap
insert(21)



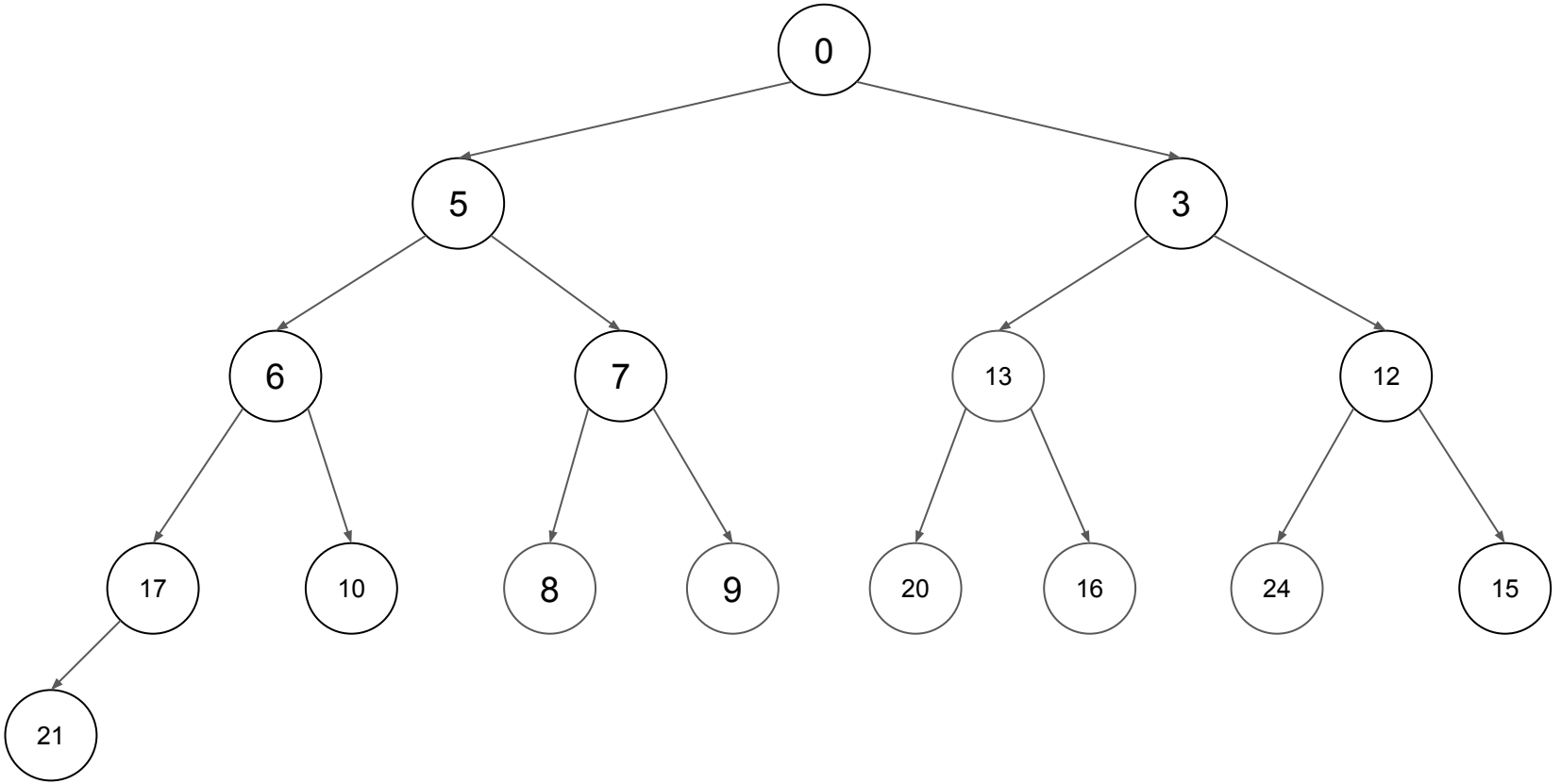
Min-Heap
insert(21)



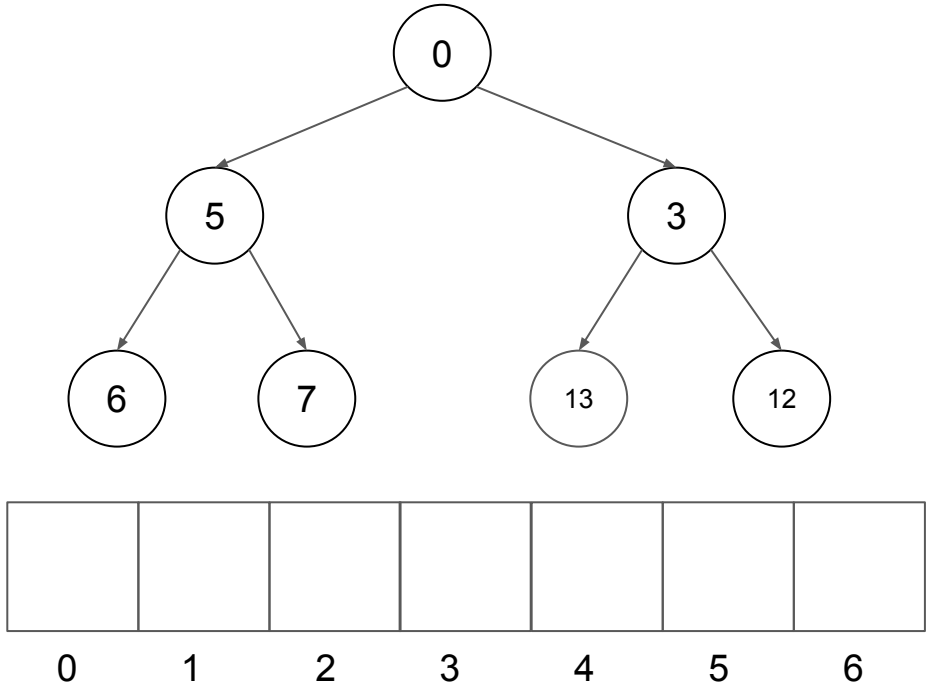
Min-Heap
insert(21)

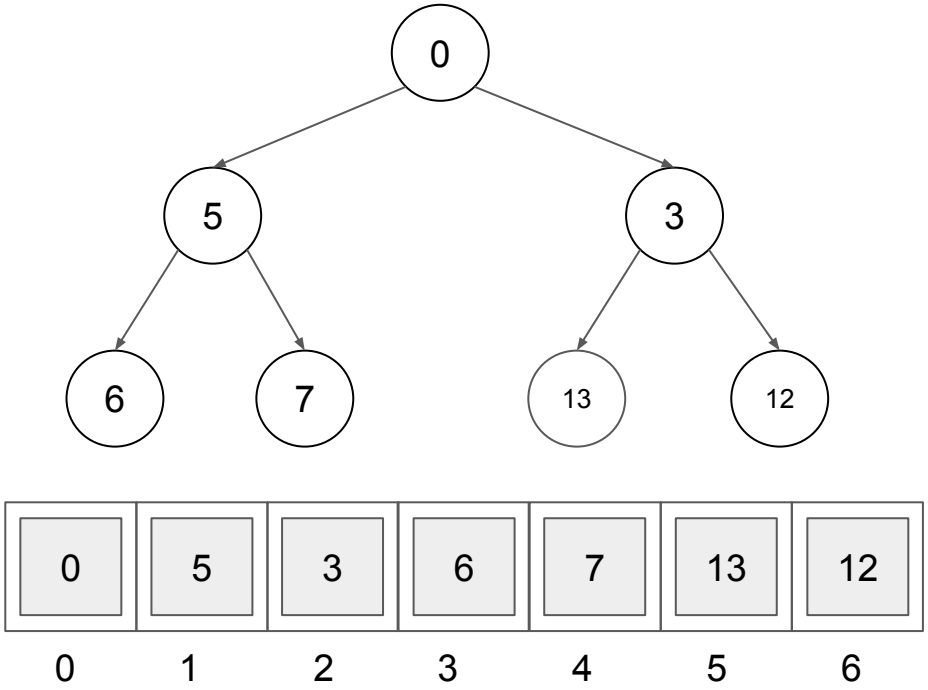


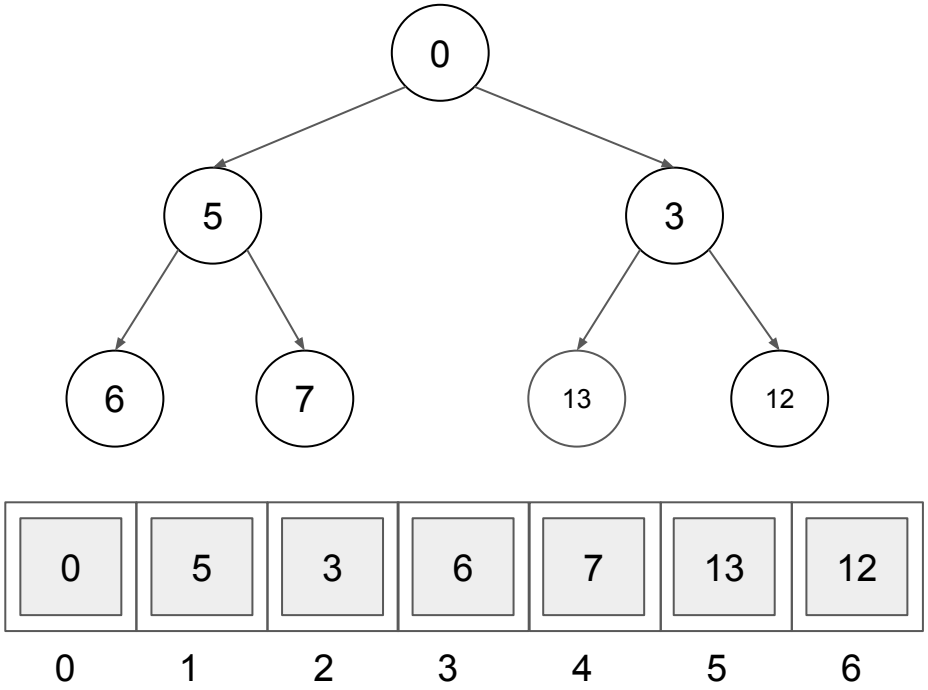
Min-Heap
insert(21)

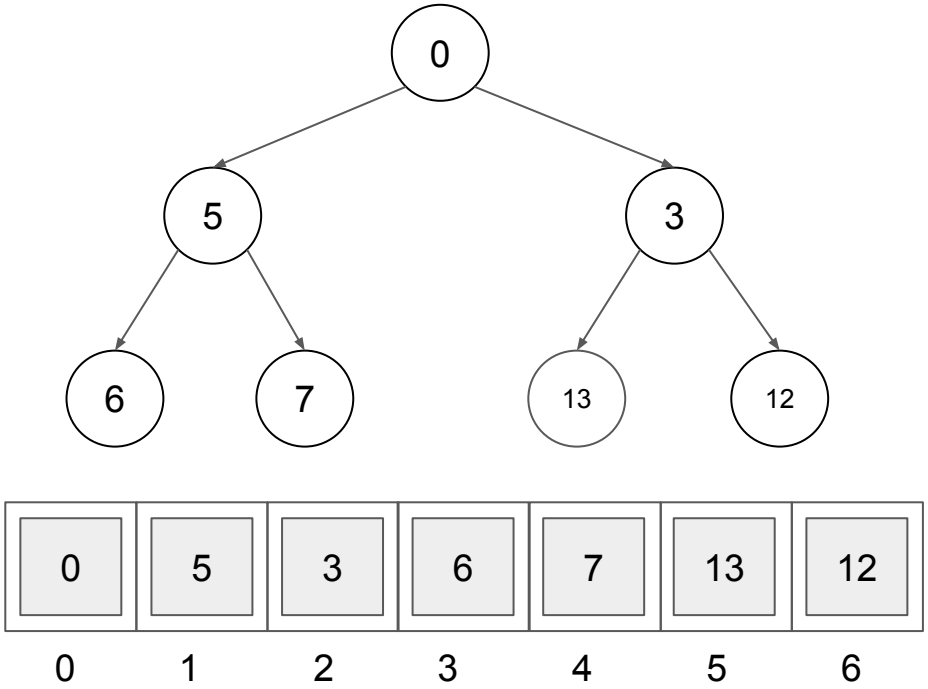


A min-heap in an array


0
5
3
6
7
13
12
0
1
2
3
4
5
6


0
5
3
6
7
13
12
0
5
3
6
7
13
12
0
1
2
3
4
5
6

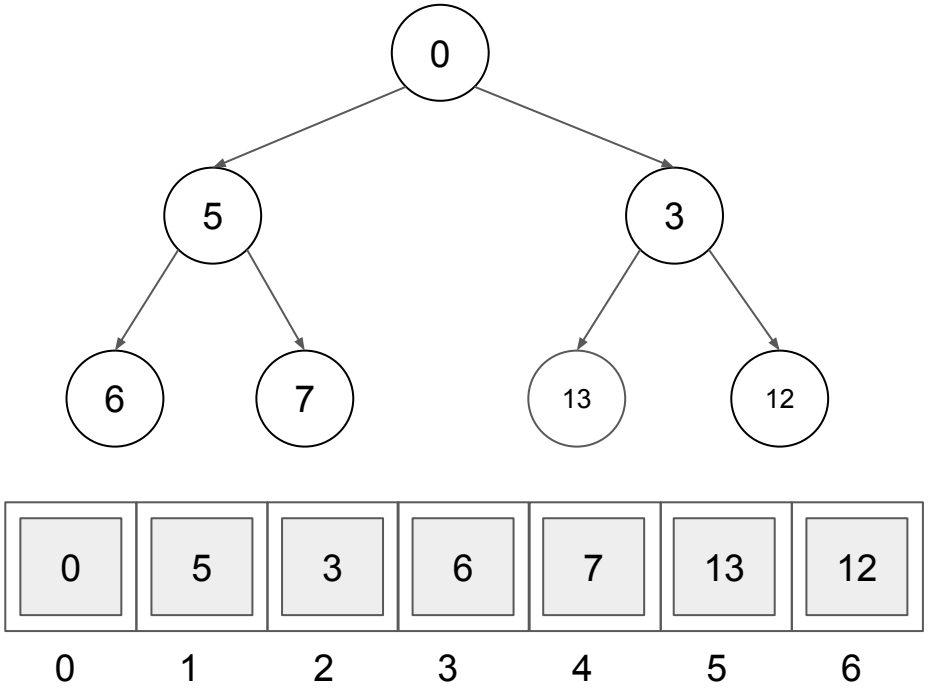

0
5
3
6
7
13
12
0
5
3
6
7
13
12
0
1
2
3
4
5
6



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

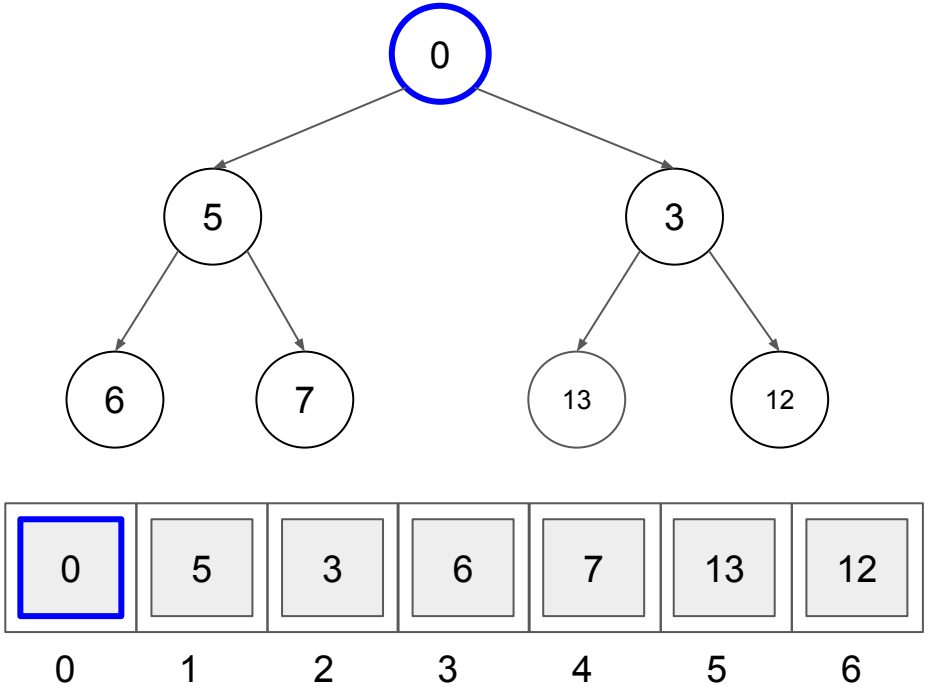
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

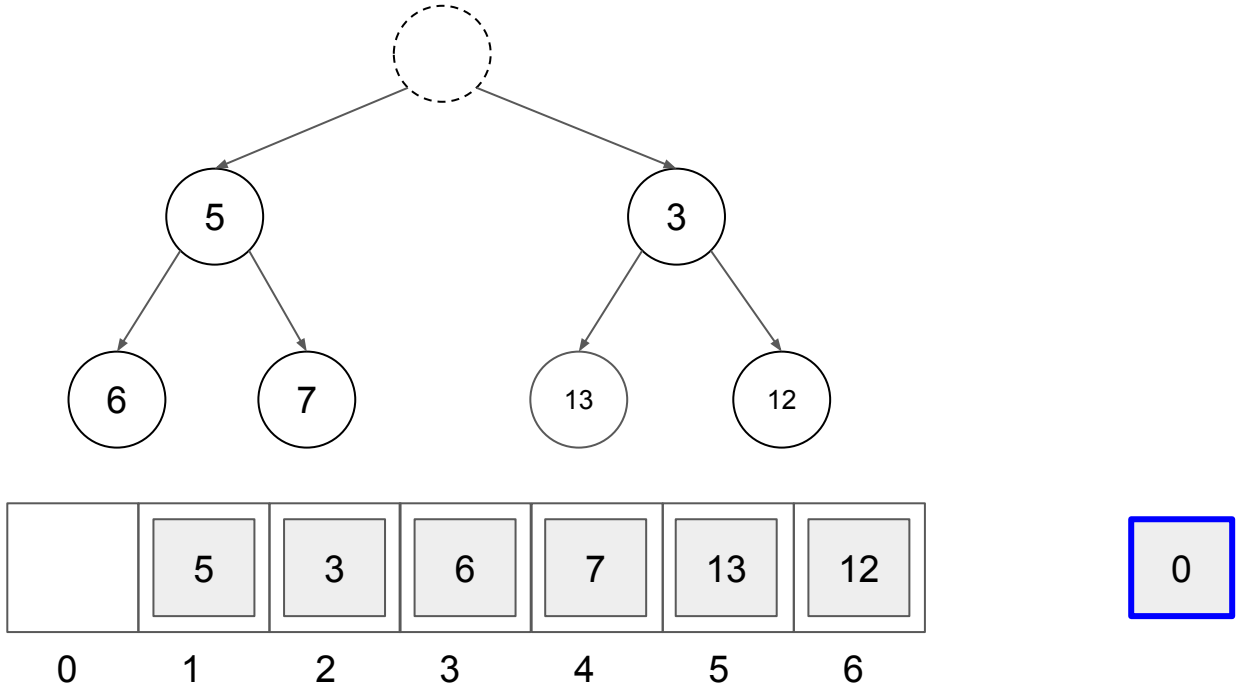
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

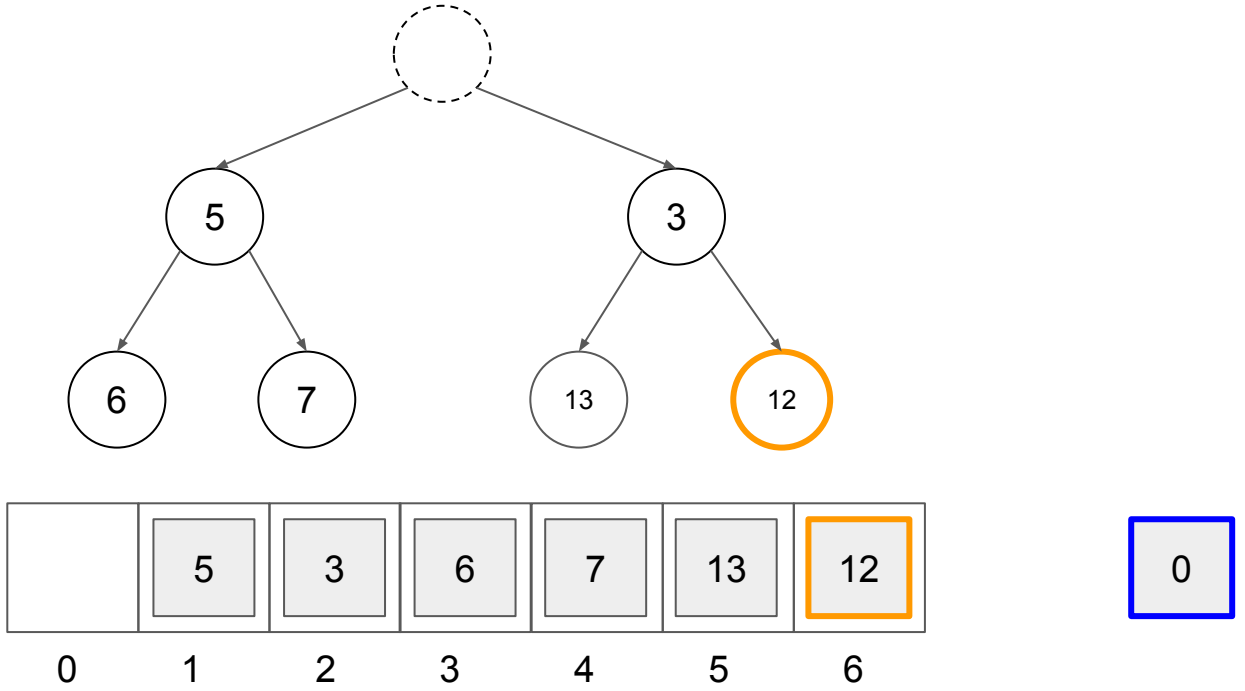
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

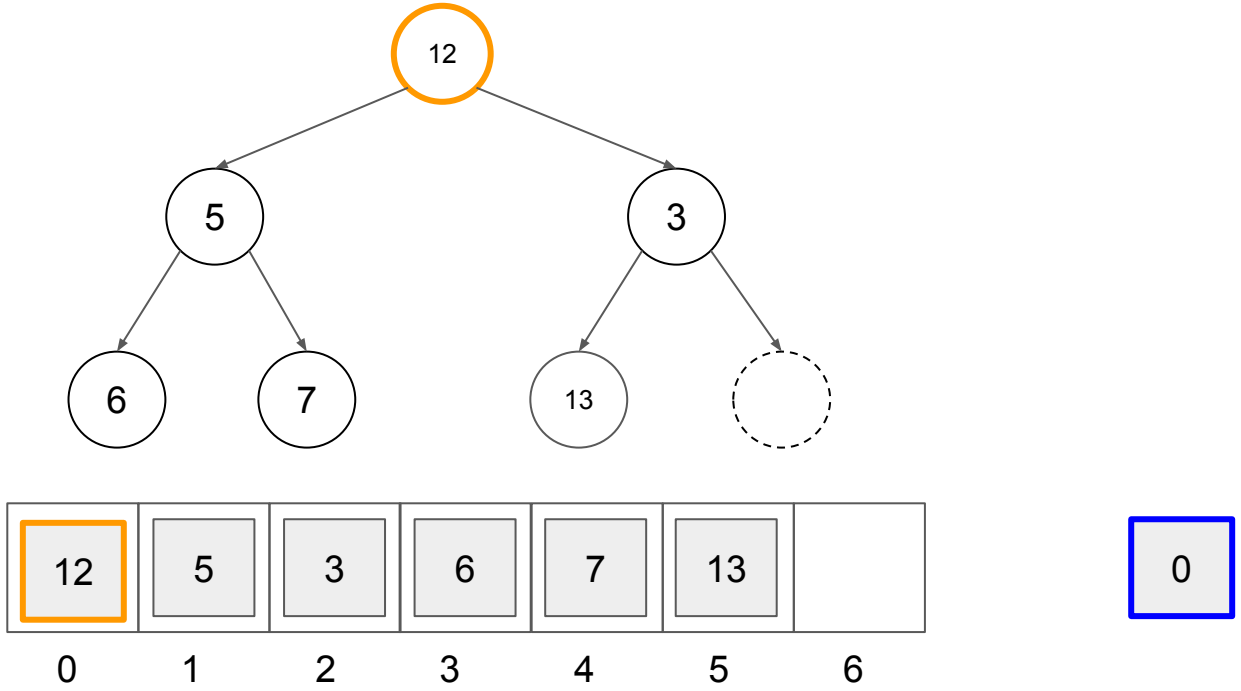
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

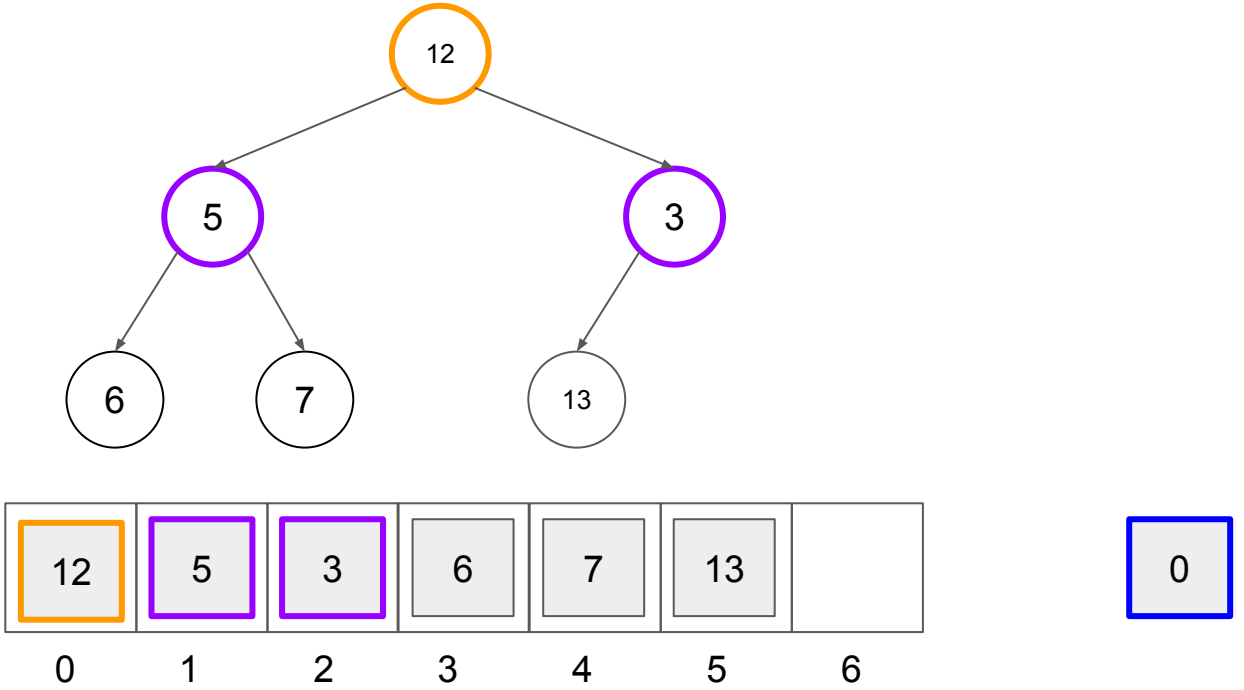
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

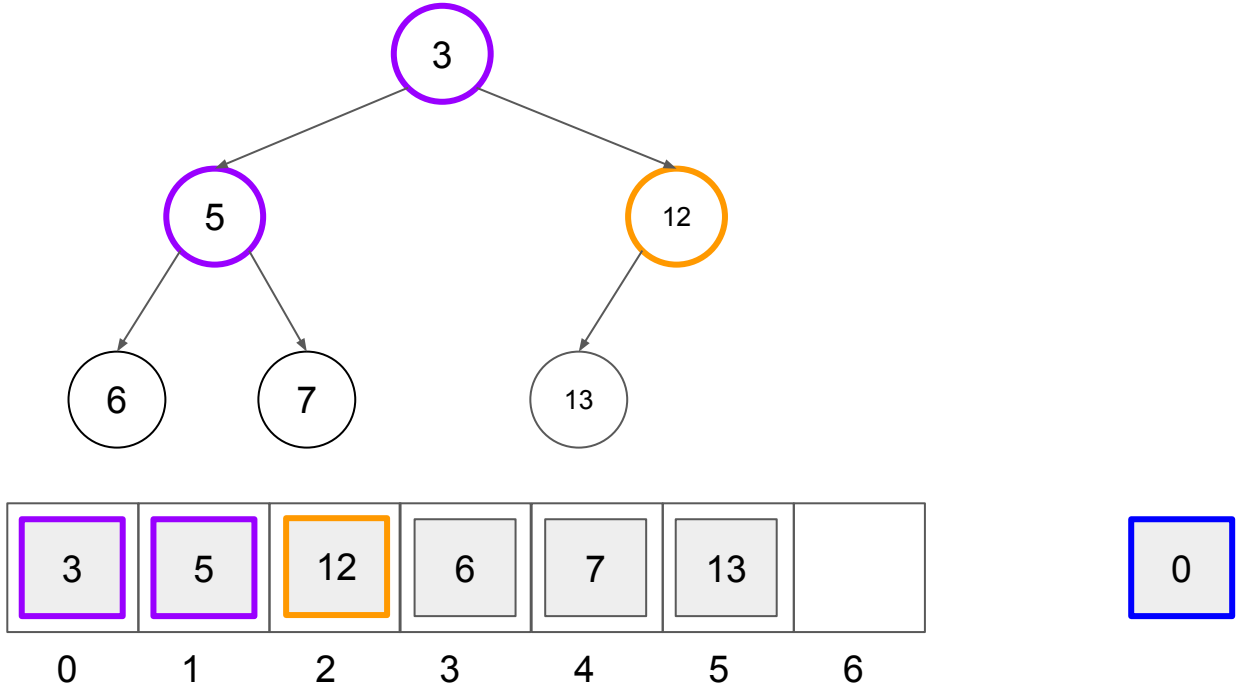
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

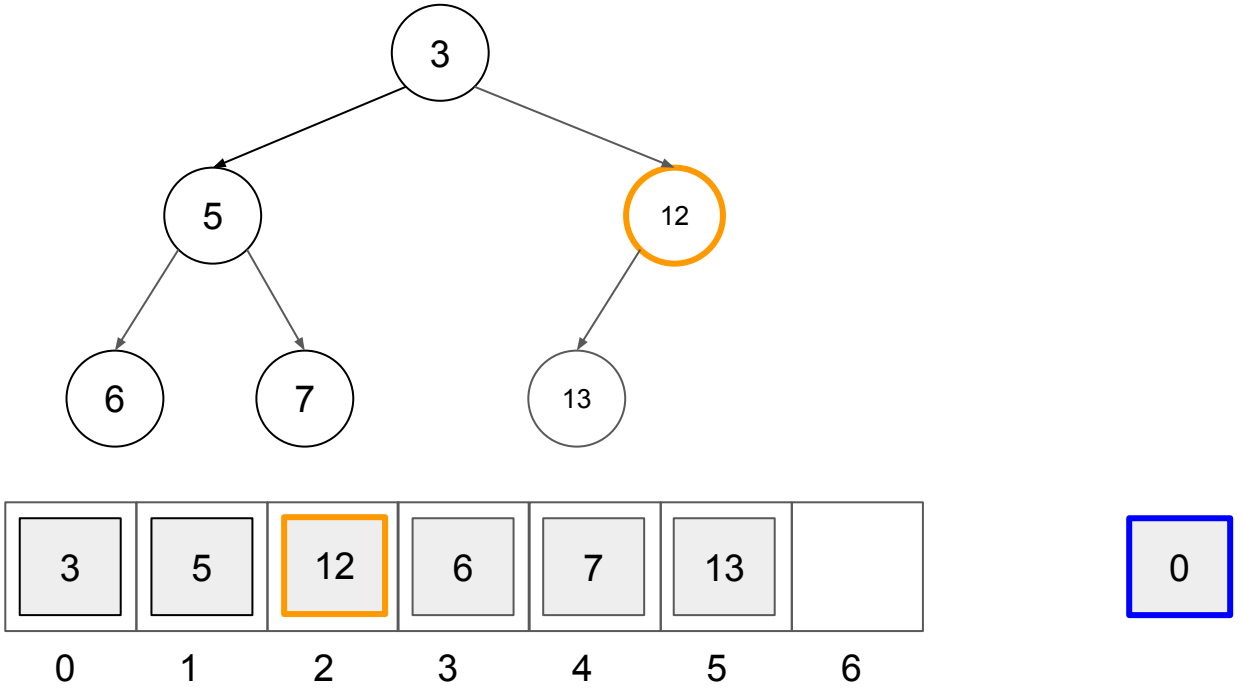
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

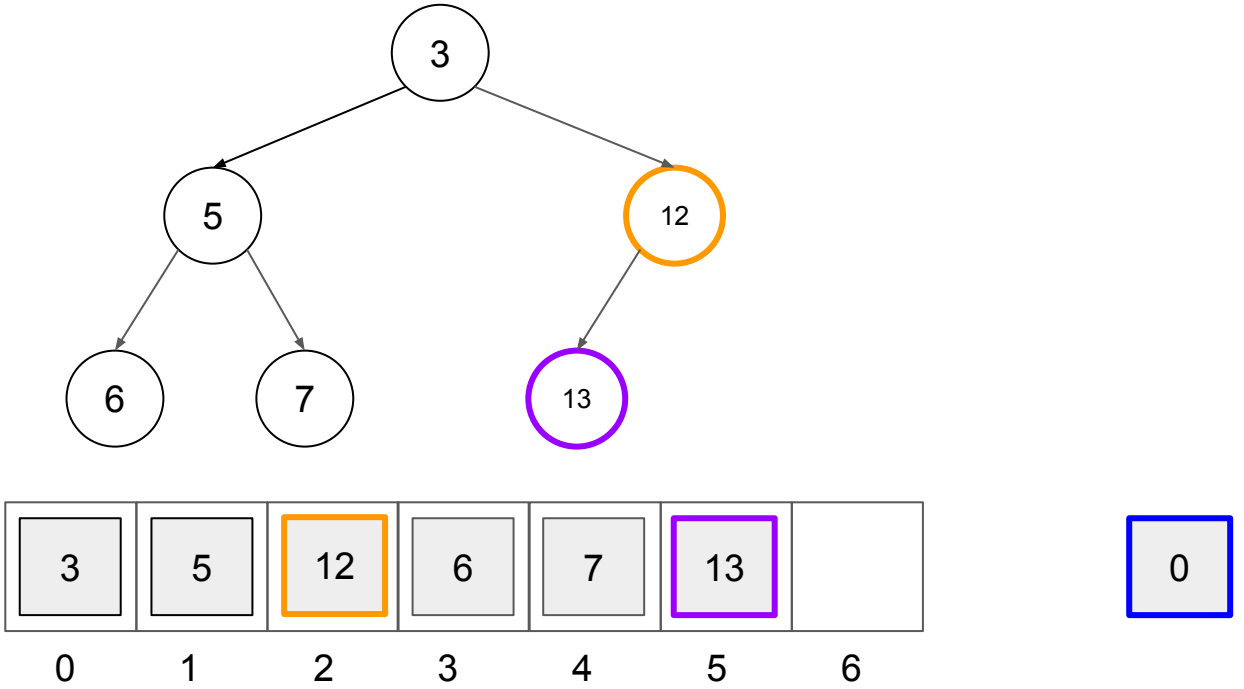
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

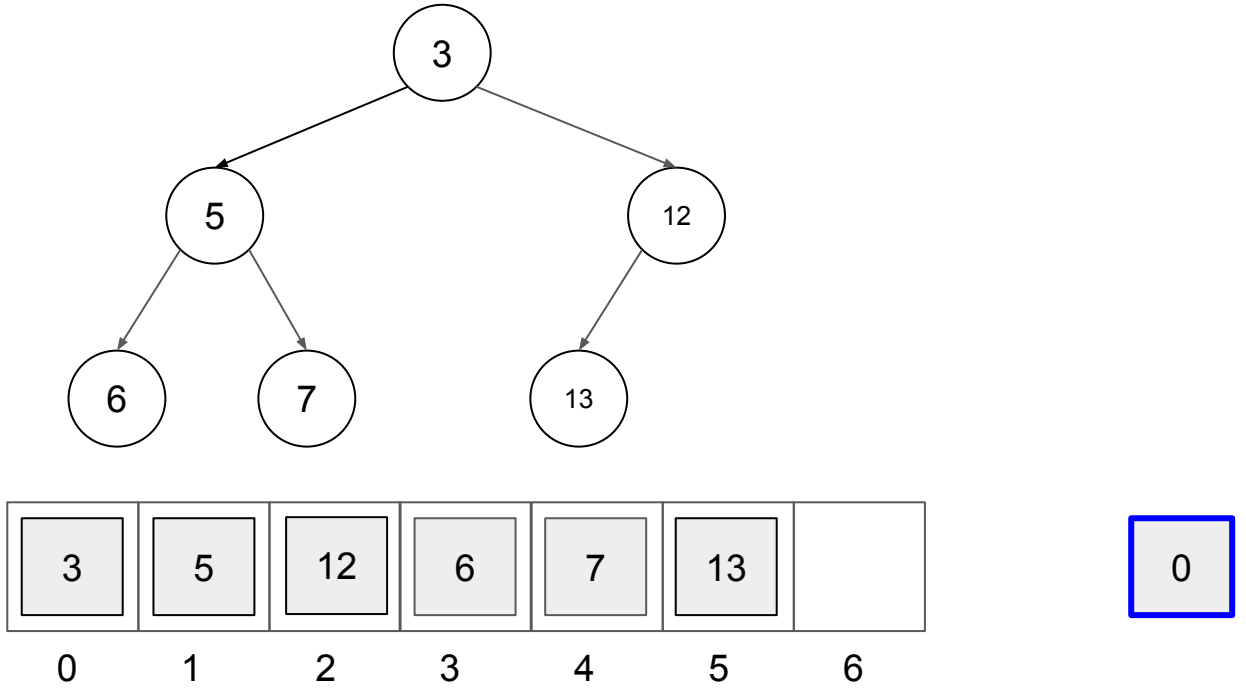
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

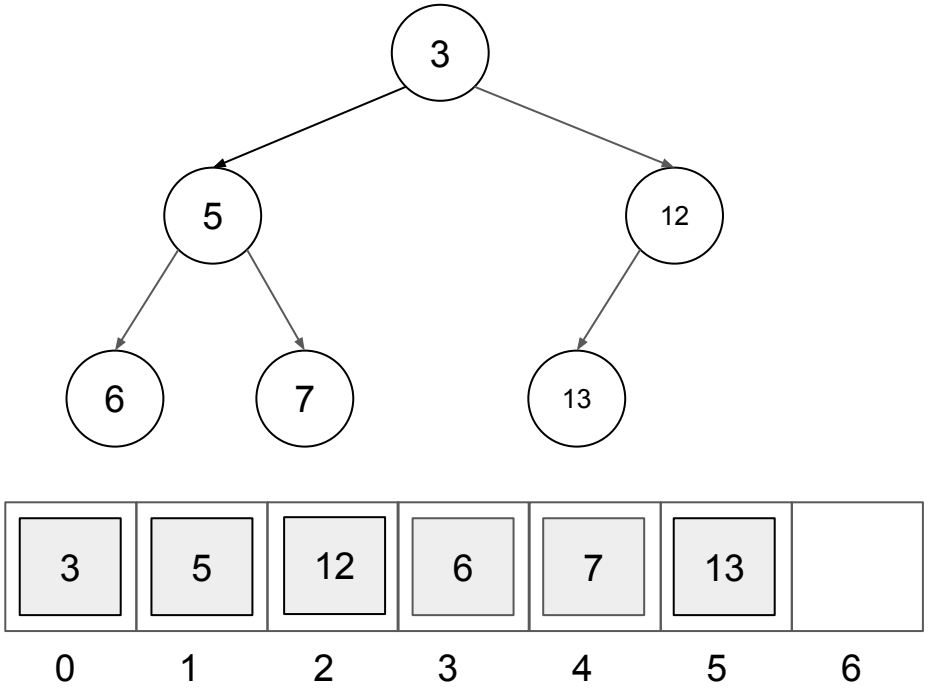
Min-Heap
extractMin()



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

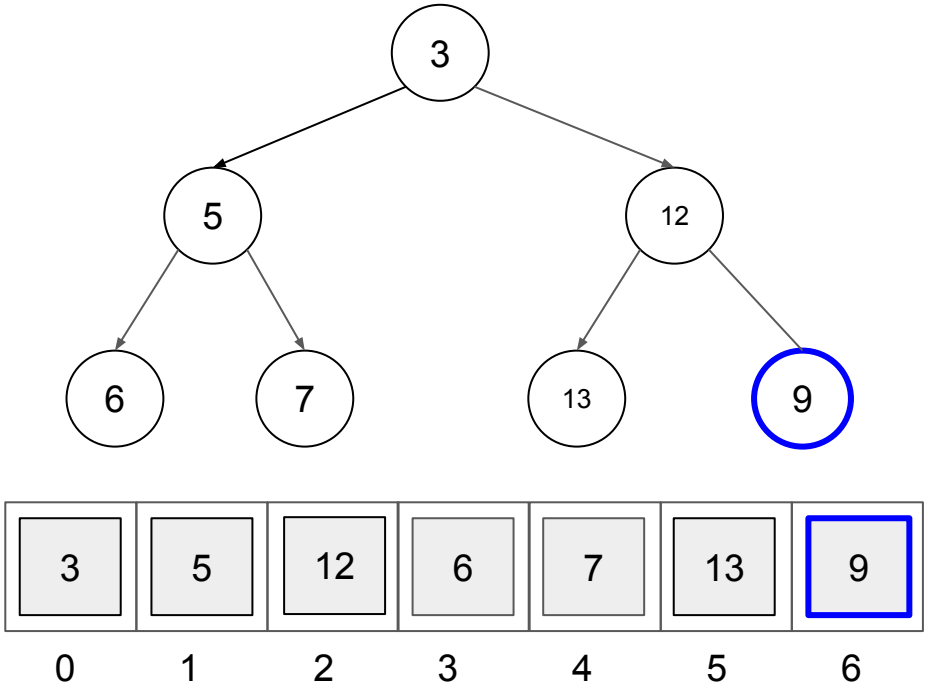
Min-Heap
insert(9)



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

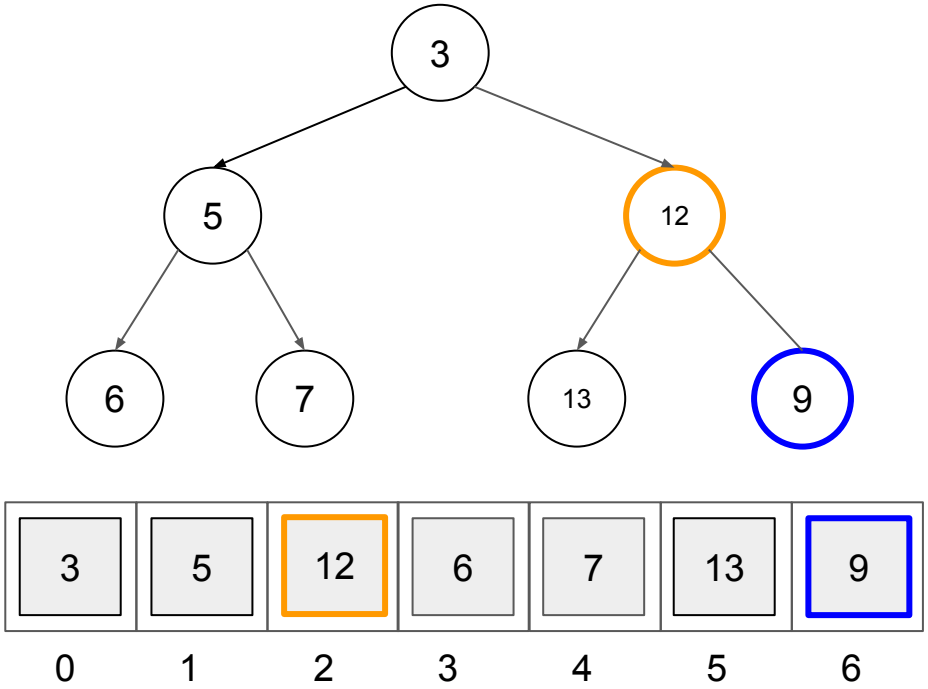
Min-Heap
insert(9)



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

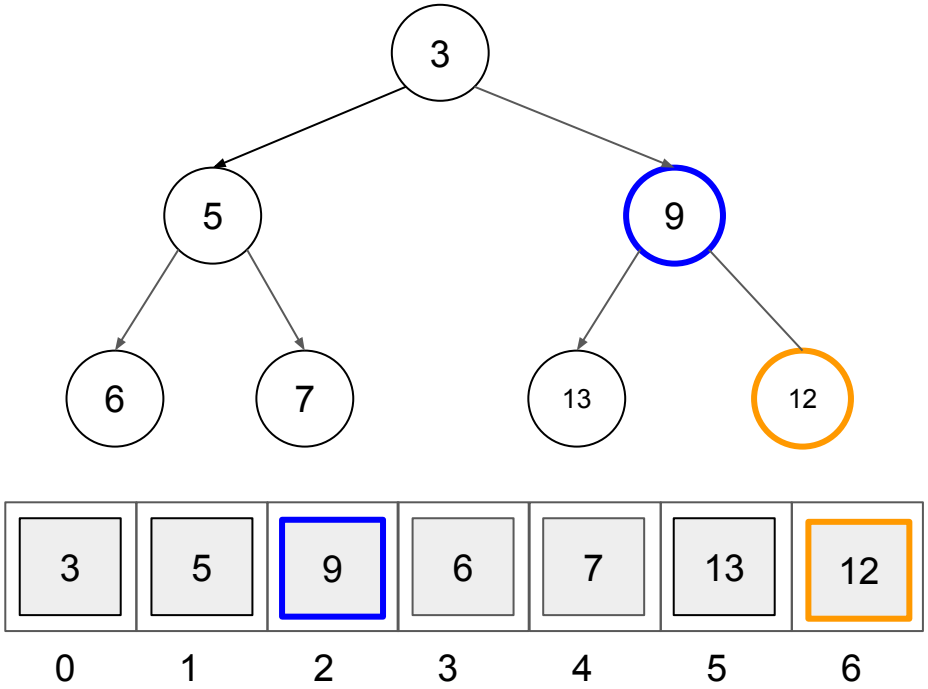
Min-Heap
insert(9)



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

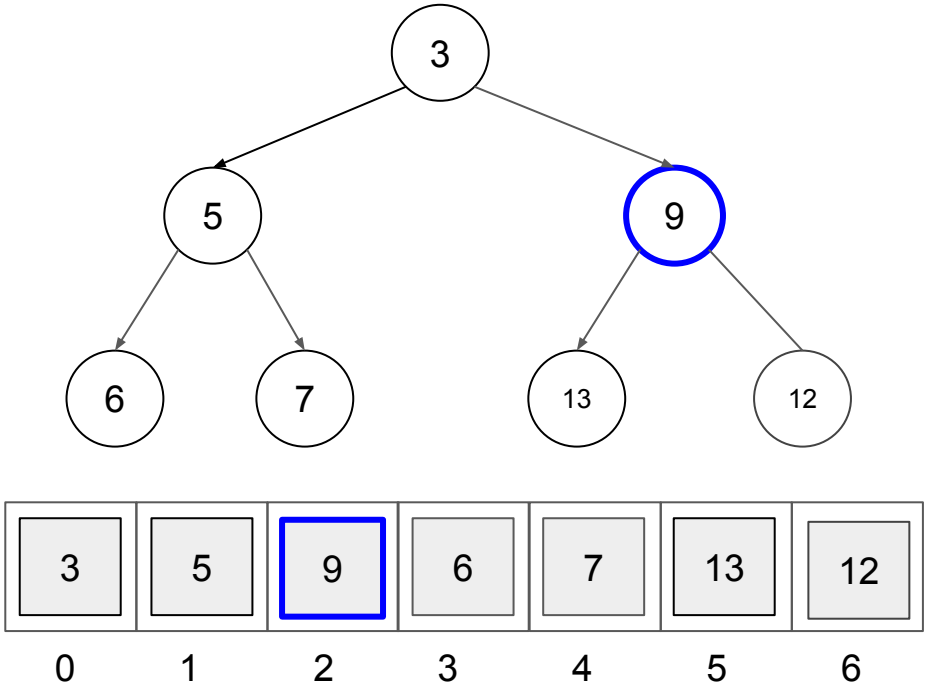
Min-Heap
insert(9)



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

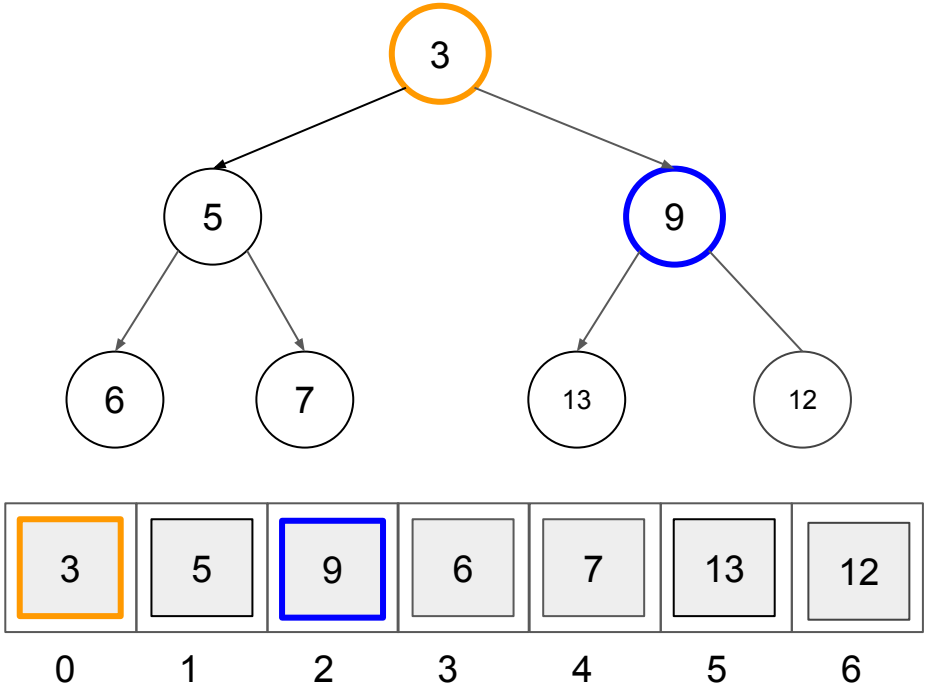
Min-Heap
insert(9)



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

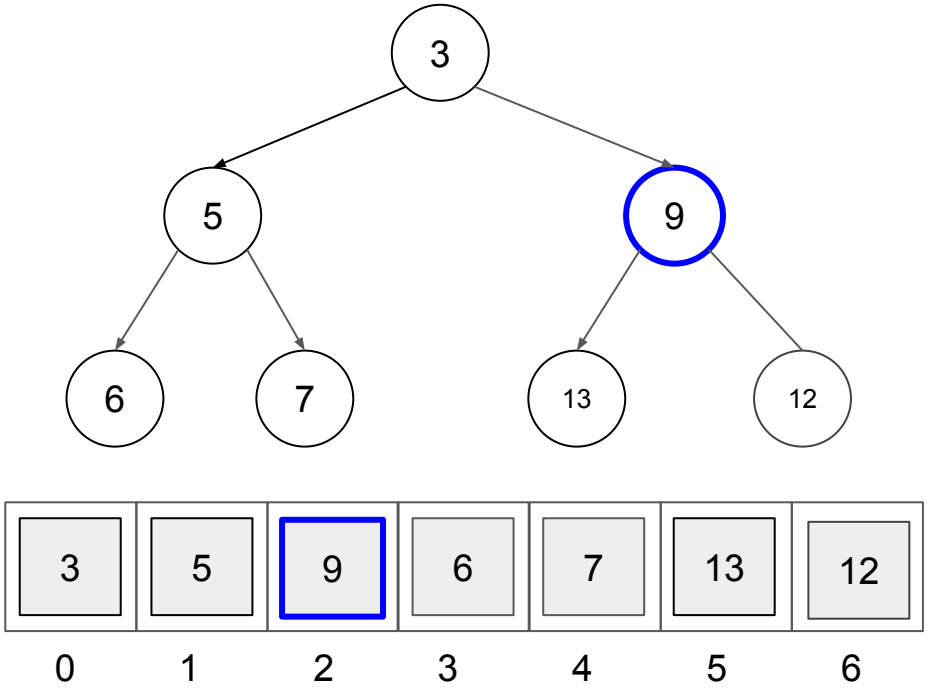
Min-Heap
insert(9)



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

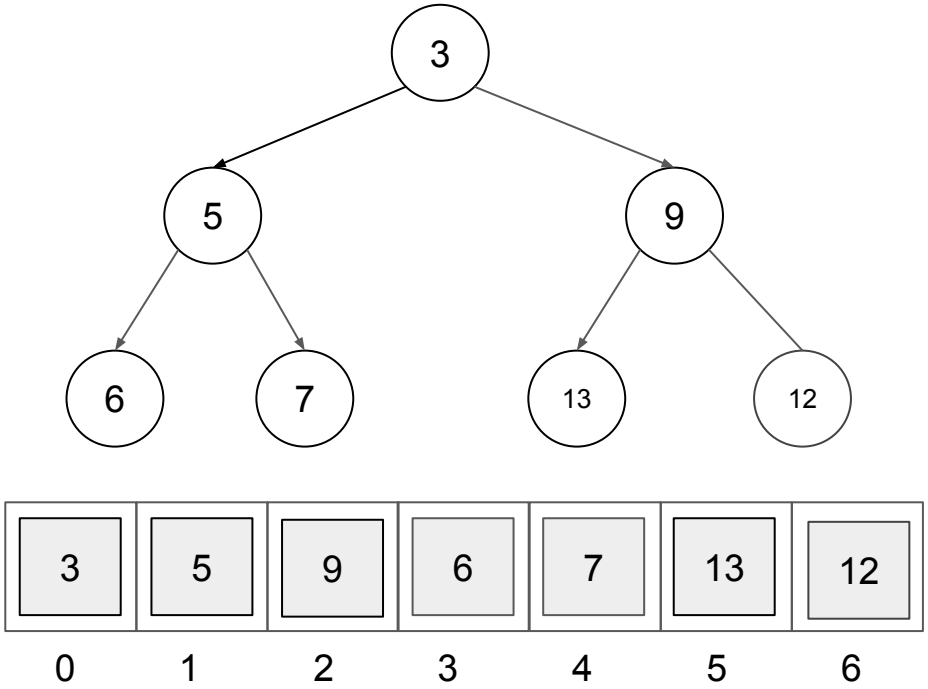
Min-Heap
insert(9)



Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

Min-Heap
insert(9)

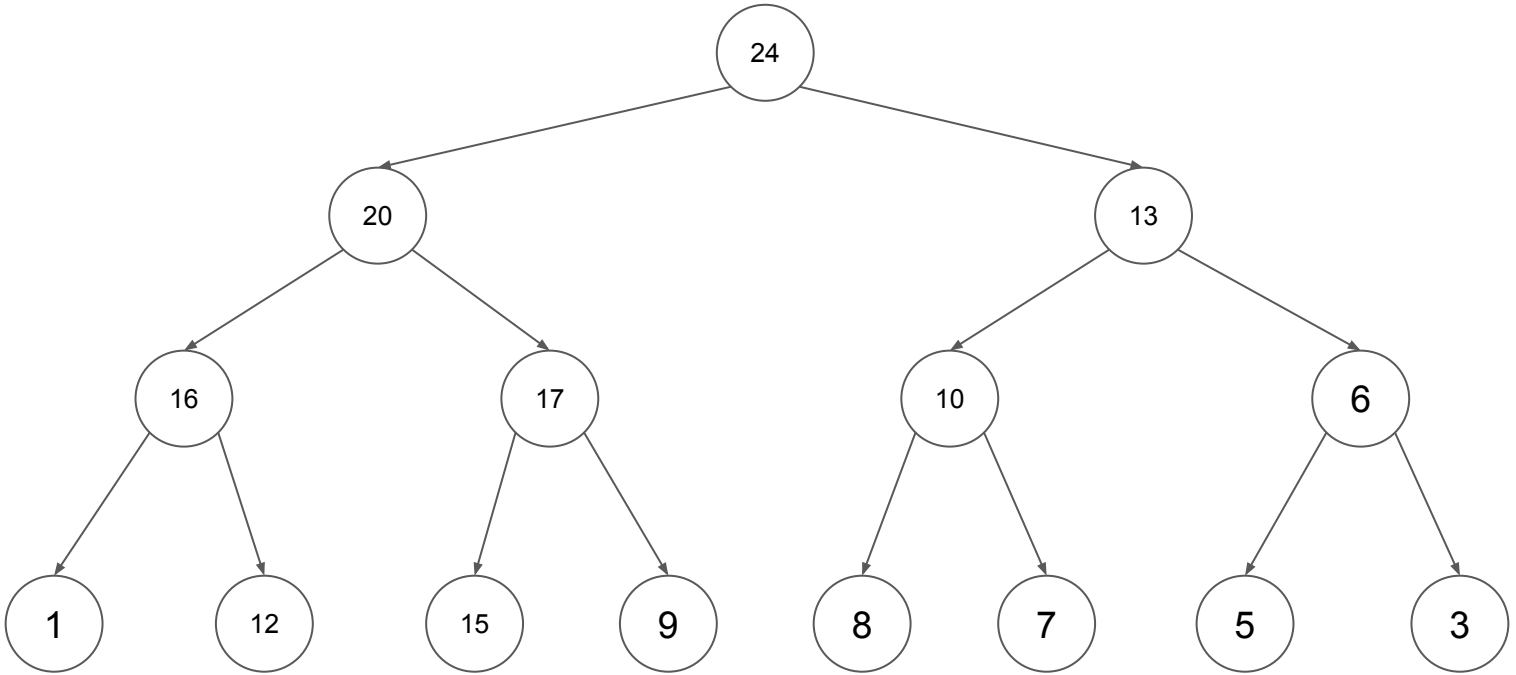


Parent: p
Children: 2p + 1, 2p + 2

Parent: (c - 1) / 2
Child: c

Max-Heap

Max-Heap



A sneak peek preview (Comp 160, Algorithms)

heapify

In Your Pocket

arrays
linked lists
stacks
queues
trees
heaps

man ssh exit pwd cd ls
valgrind touch mkdir cp rm
rmdir mv cat head tail less
redirect (>, >>, <) pipe (|)
(echo, sort, uniq, wc)

Sorting Algorithms
- Selection sort
- Insertion sort
- Merge sort
- Quicksort
- Counting sort

Some keywords from today's lecture:

- Bogosort, Sleep sort
- (C++) templates with two template parameters
- tree height
- binary search performed on a sorted array
- an idea to rebuild a balanced tree
- (C++) const correctness
- const keyword
- tree rotation, (single) left rotation, (single) right rotation
- self-balancing binary search trees, self-adjusting binary search trees
- (binary) heap, insert, min-heap (+ find/extract min), max-heap (+ find/extract max)
- bubble down, bubble up
- min-heap in array, parent index, child index

To the lab!