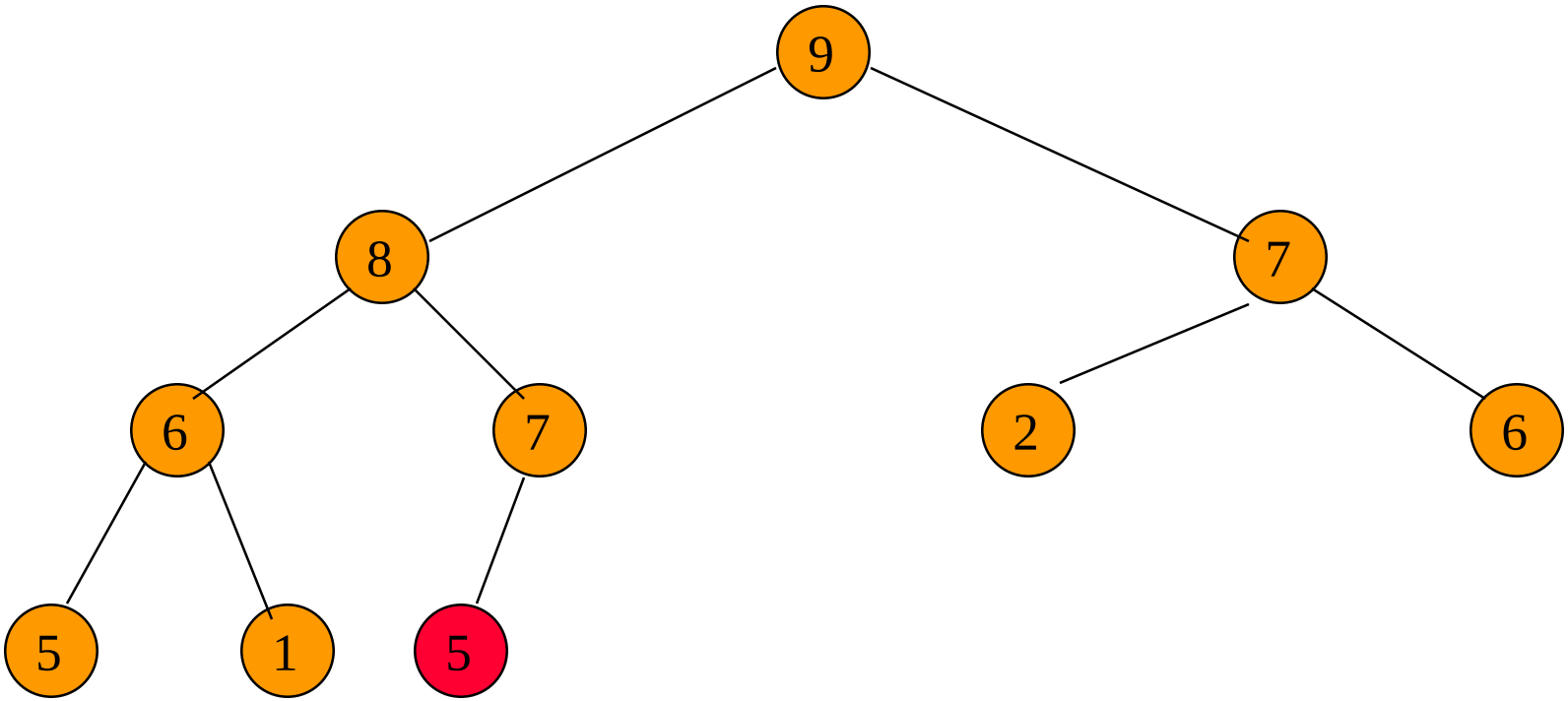
Heap Operations

Data structures

Spring 2017

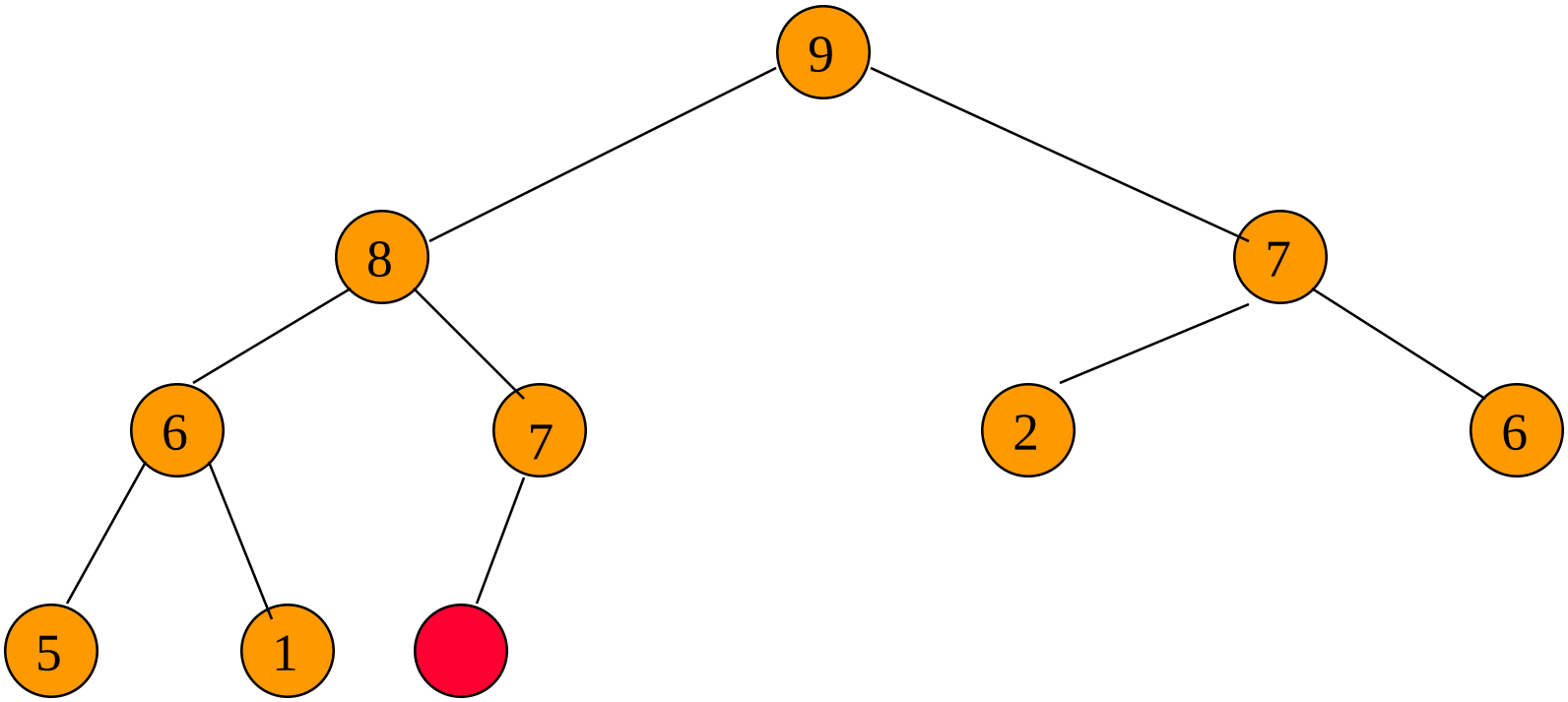
Putting An Element Into A Max Heap



Complete binary tree with 10 nodes.

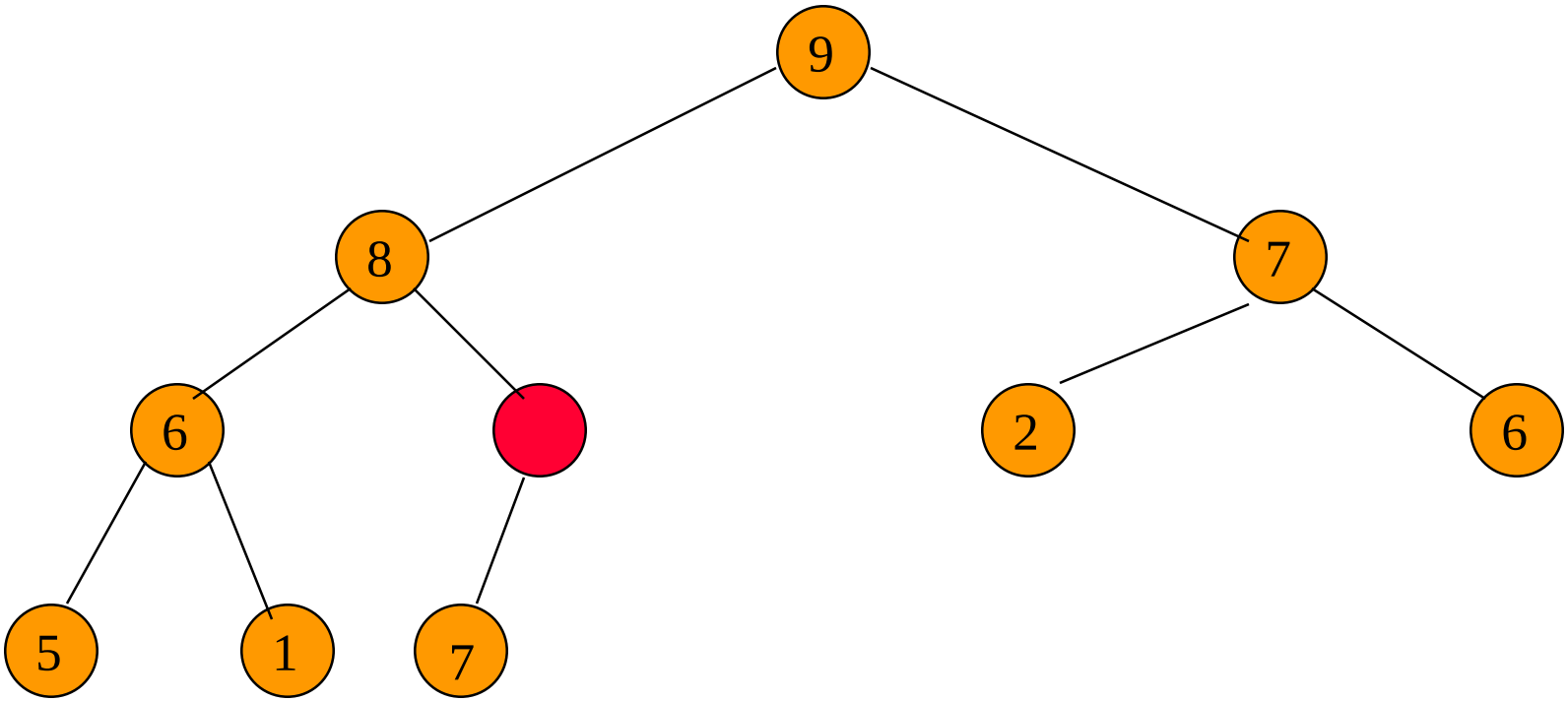
New element is 5.

Putting An Element Into A Max Heap



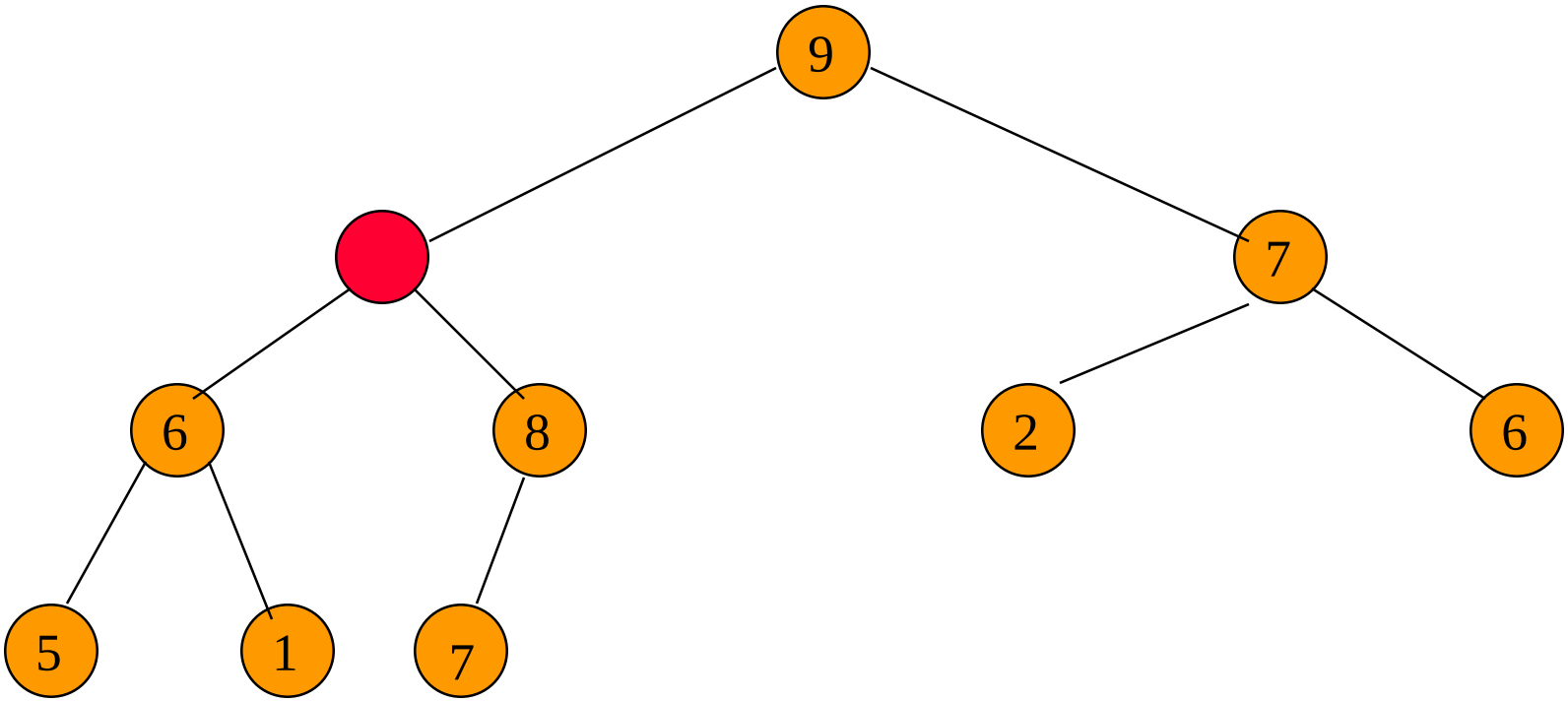
New element is 20.

Putting An Element Into A Max Heap



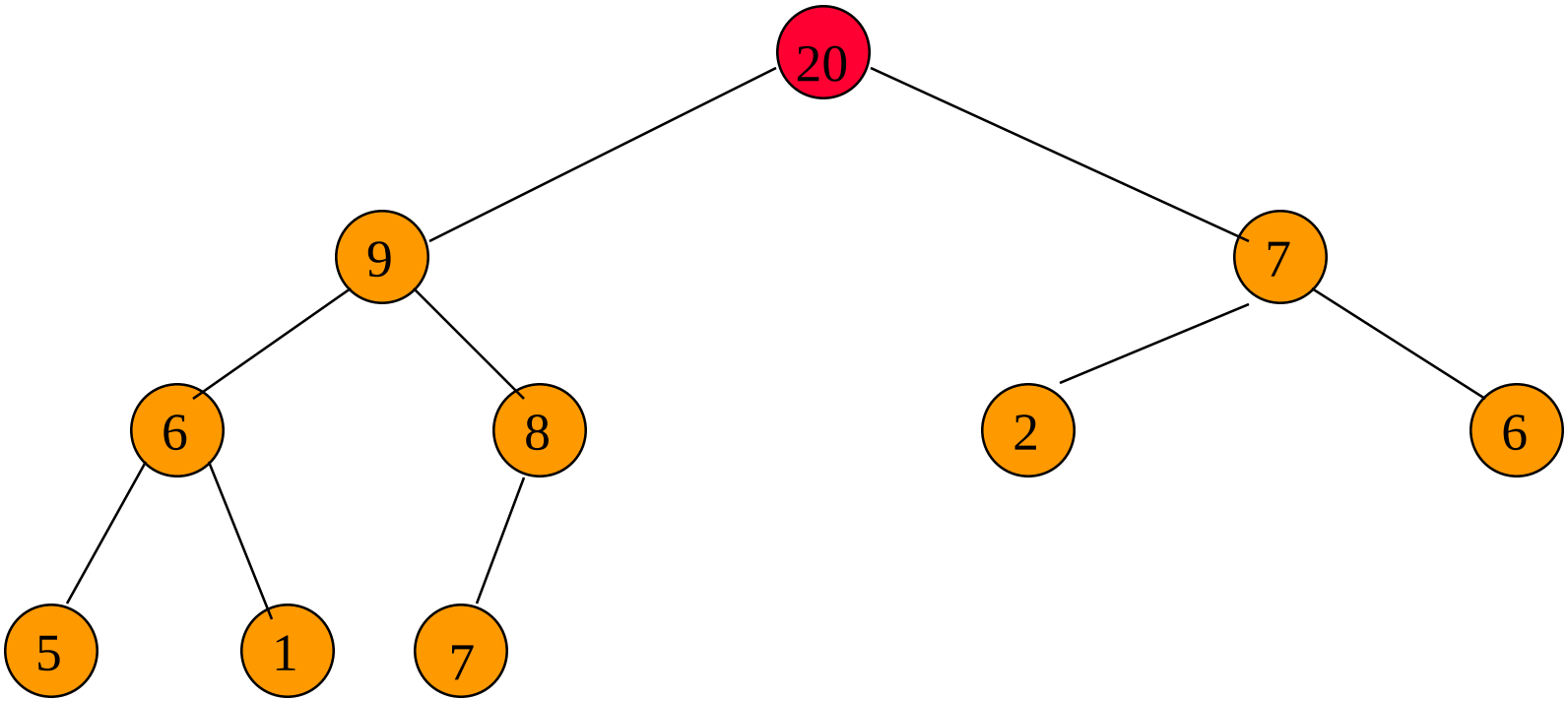
New element is 20.

Putting An Element Into A Max Heap



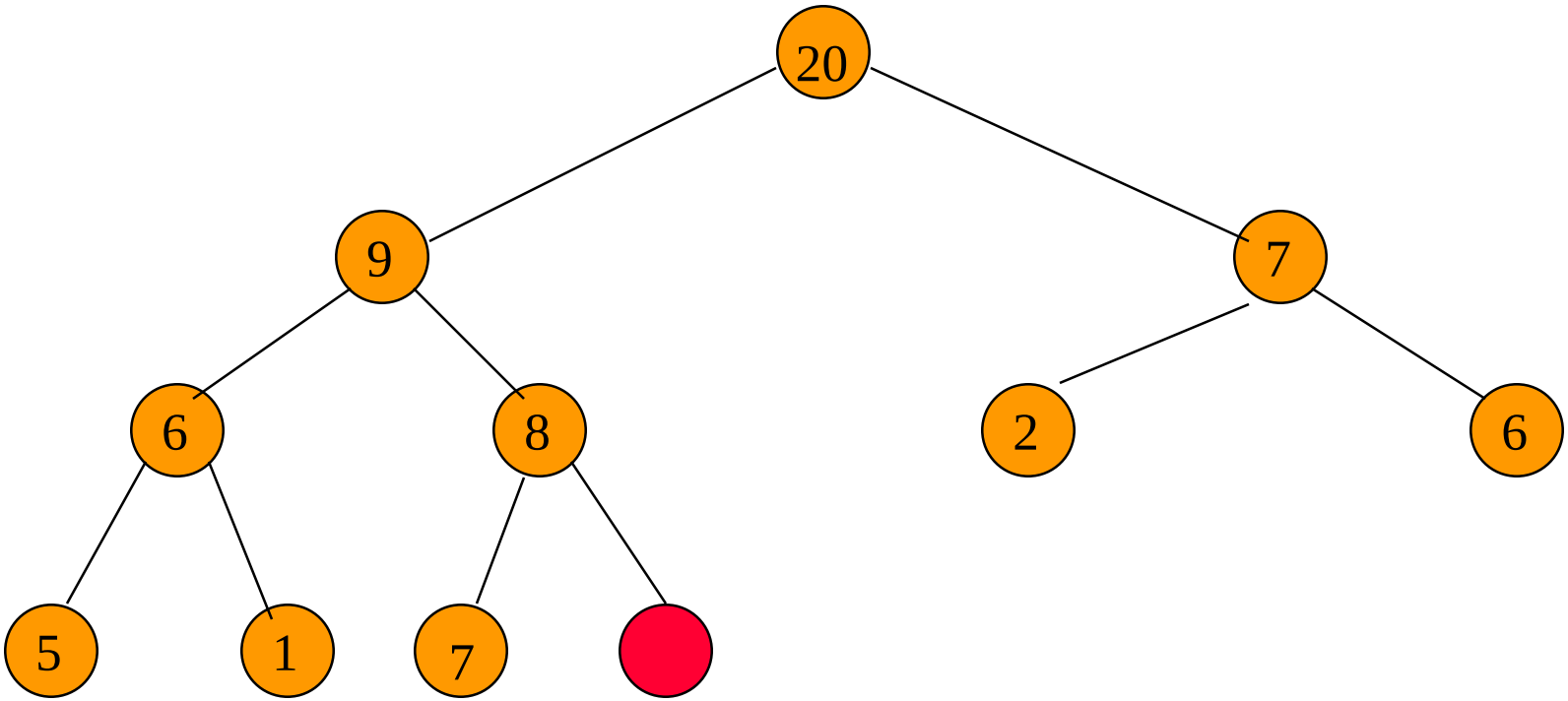
New element is 20.

Putting An Element Into A Max Heap



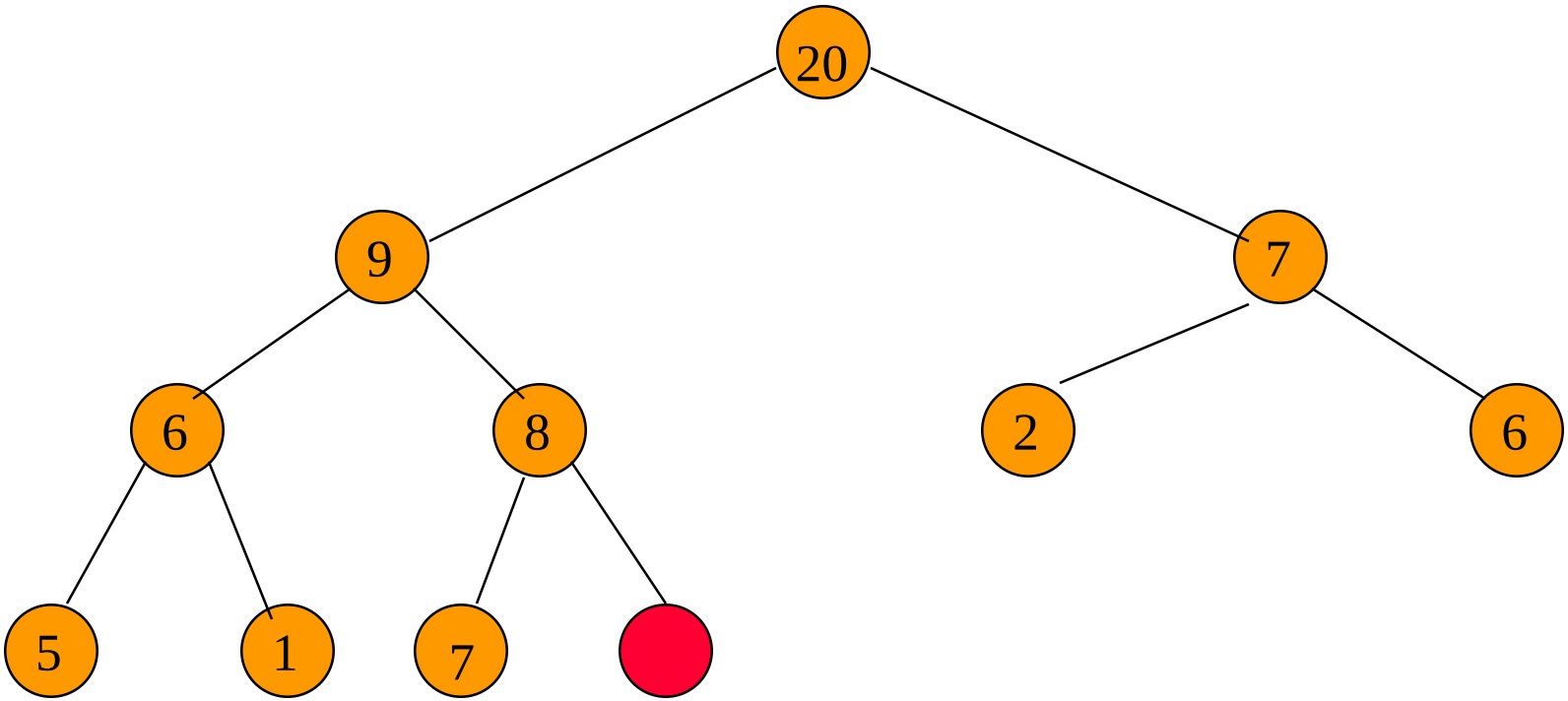
New element is 20.

Putting An Element Into A Max Heap



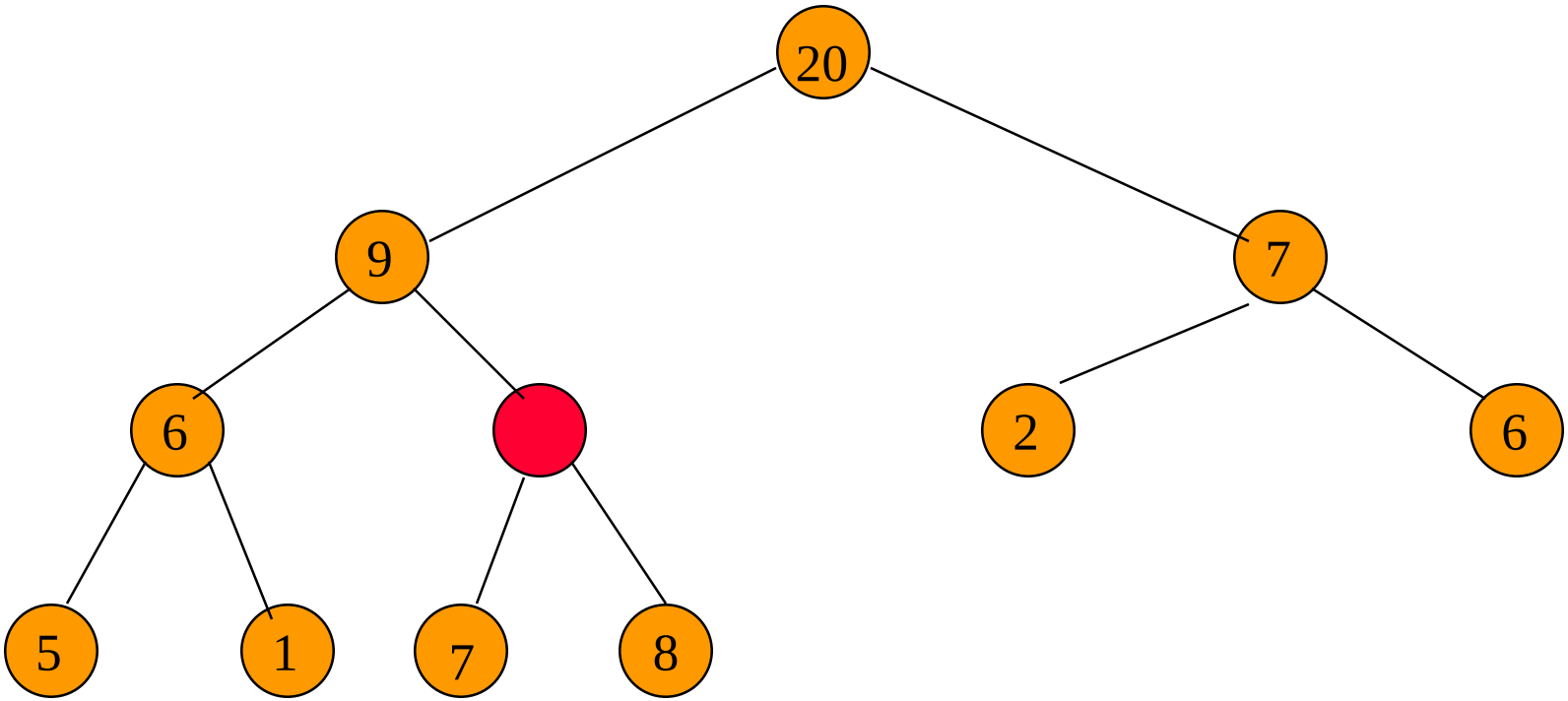
Complete binary tree with 11 nodes.

Putting An Element Into A Max Heap



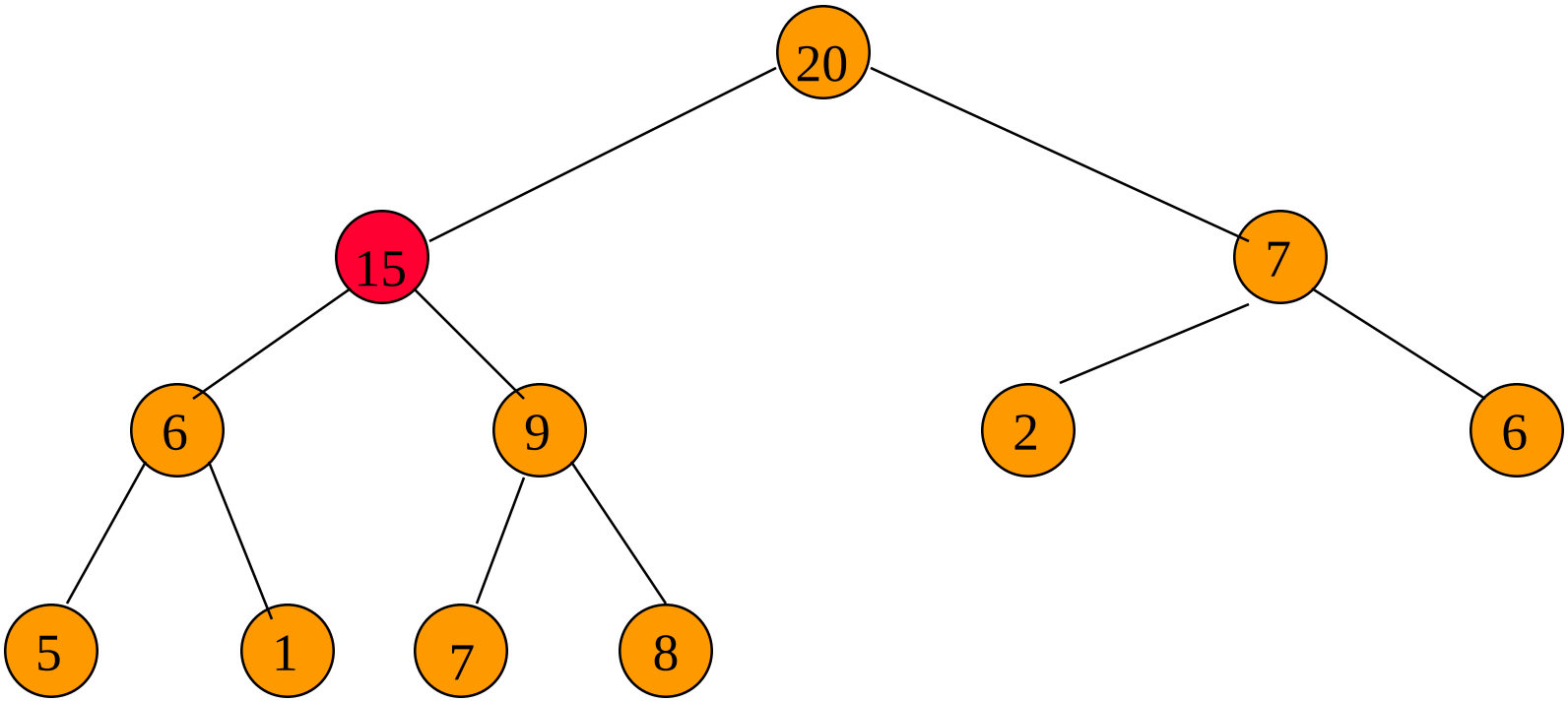
New element is 15.

Putting An Element Into A Max Heap



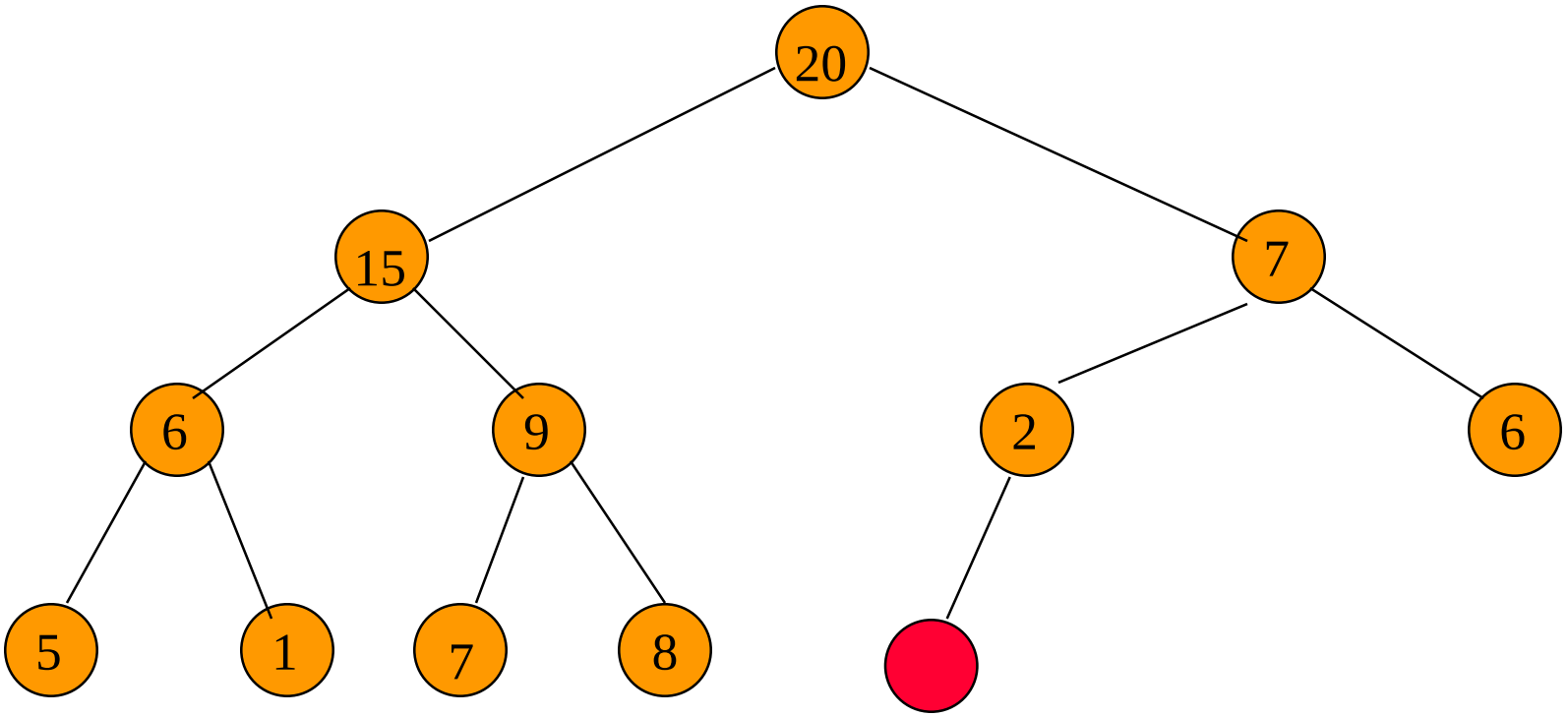
New element is 15.

Putting An Element Into A Max Heap



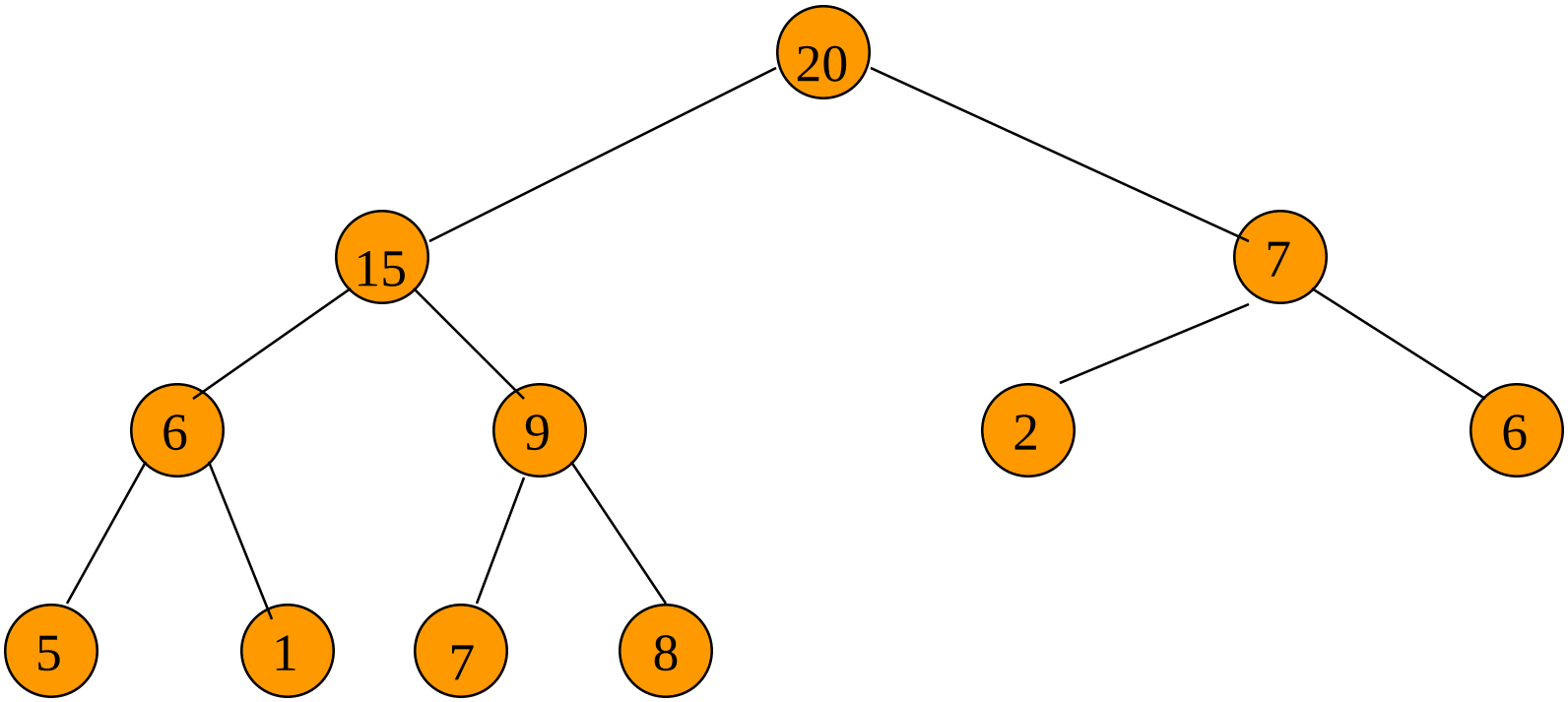
New element is 15.

Complexity Of Put



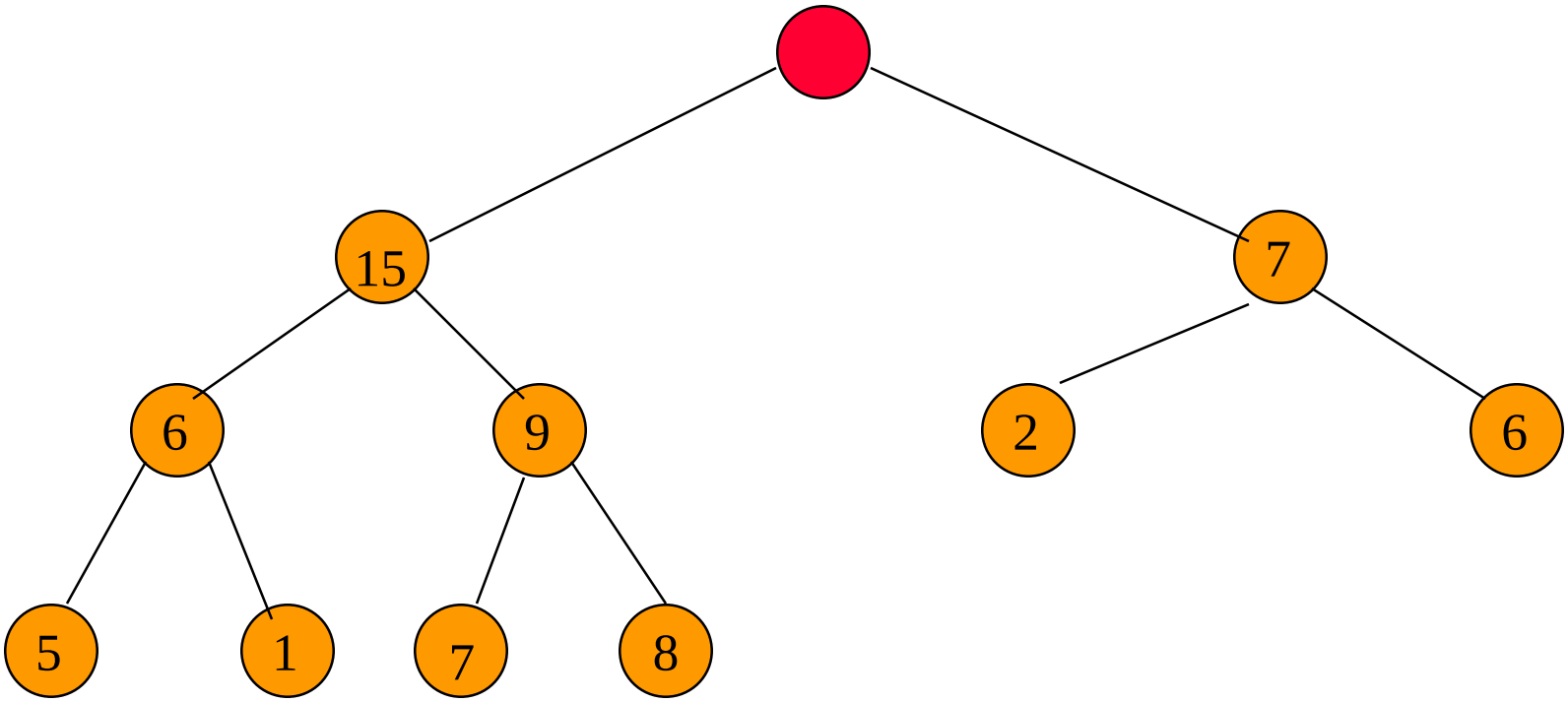
Complexity is O(log n), where n is heap size.

Removing The Max Element



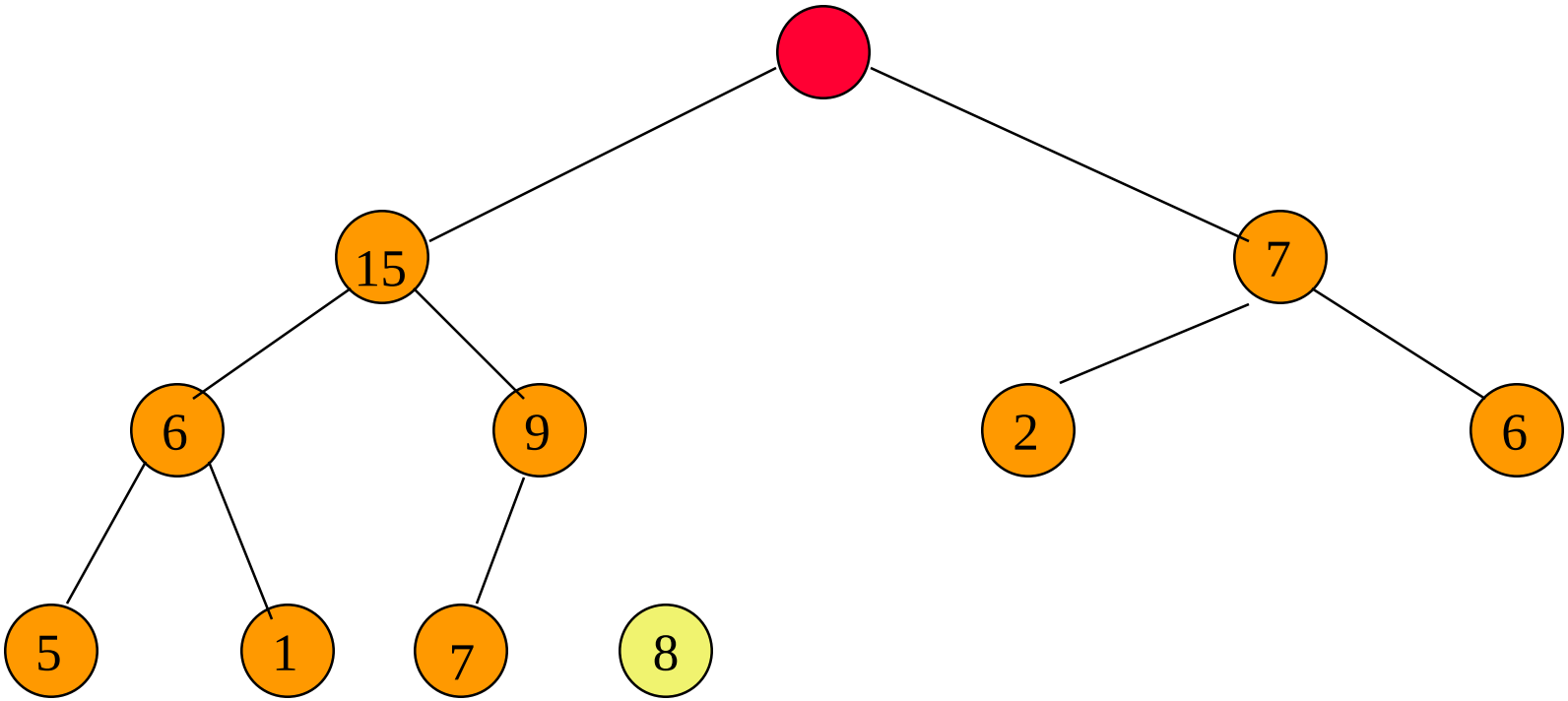
Max element is in the root.

Removing The Max Element



After max element is removed.

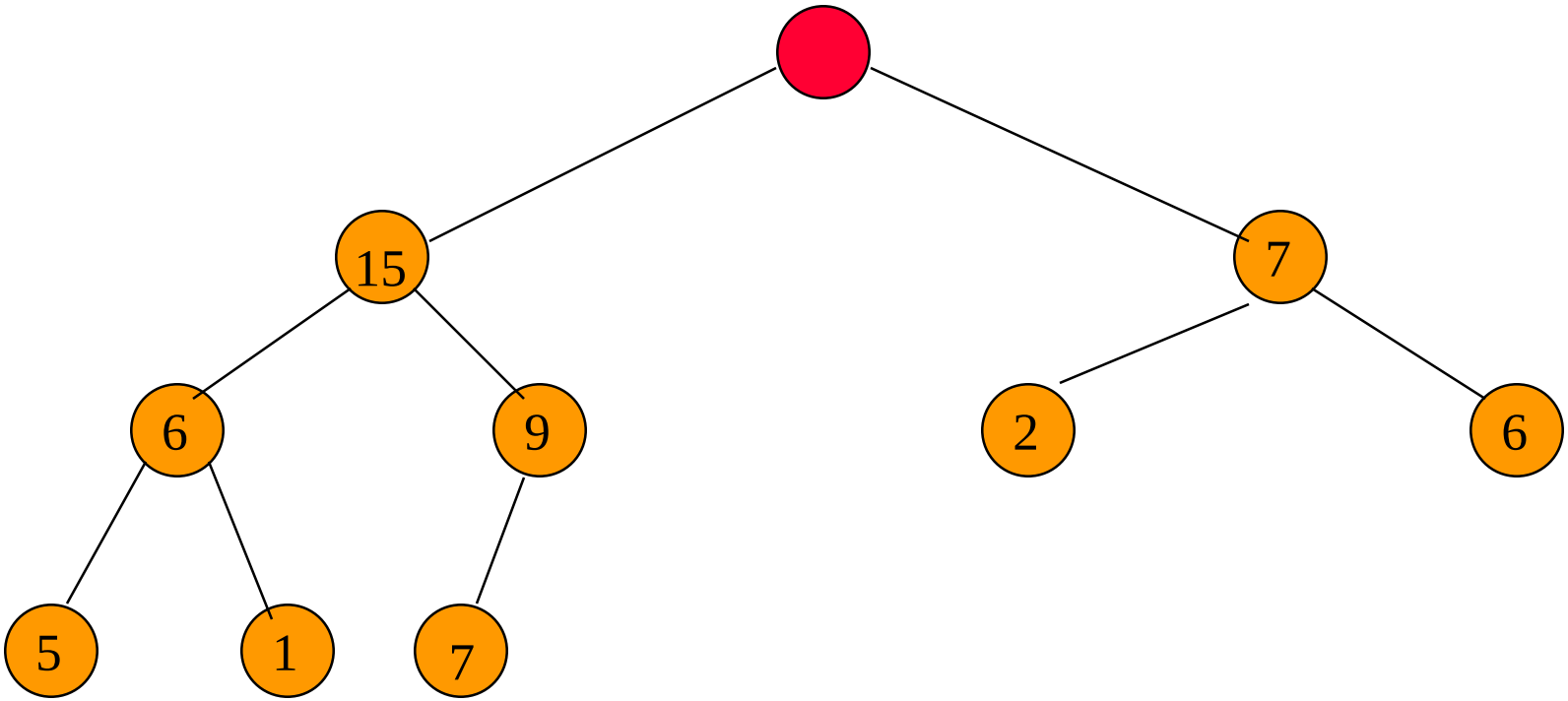
Removing The Max Element



Heap with 10 nodes.

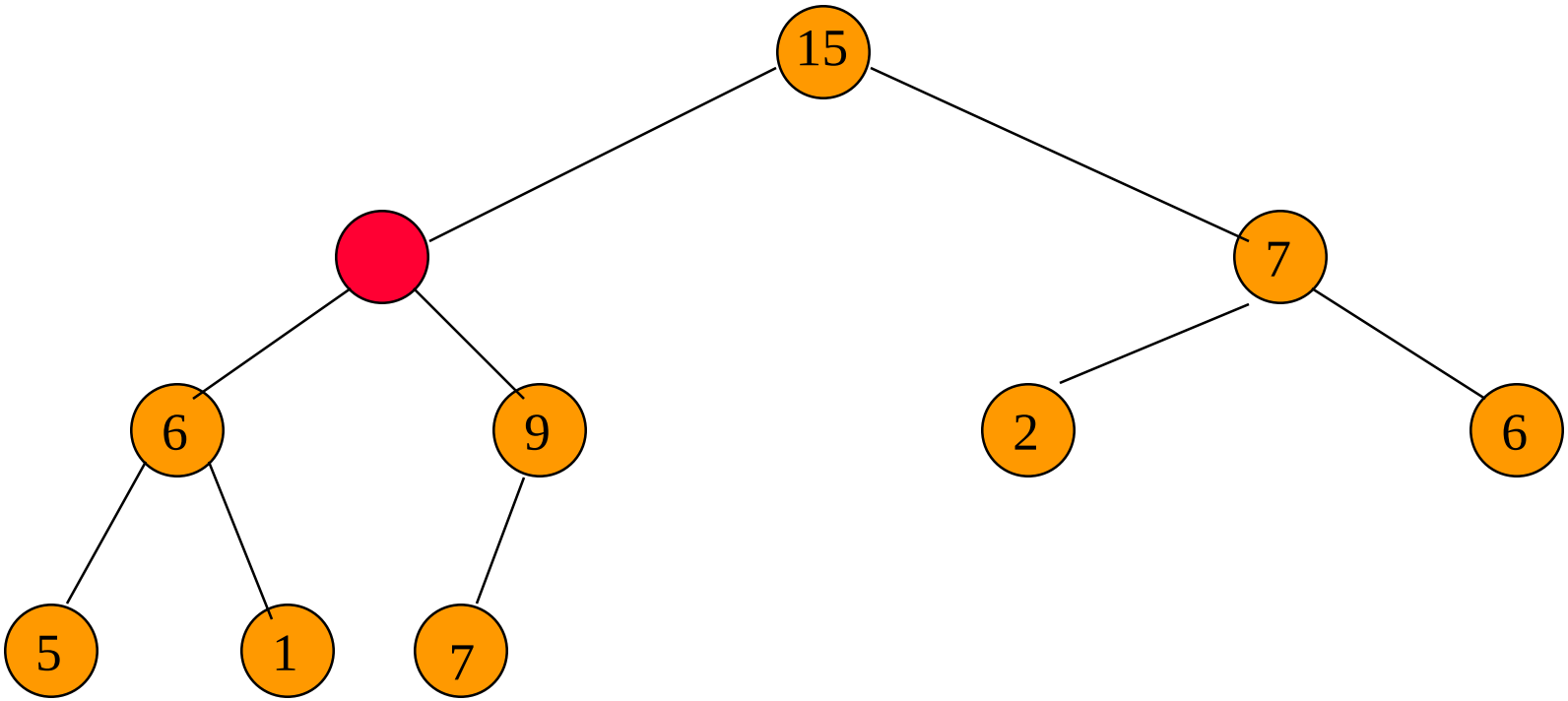
Reinsert 8 into the heap.

Removing The Max Element



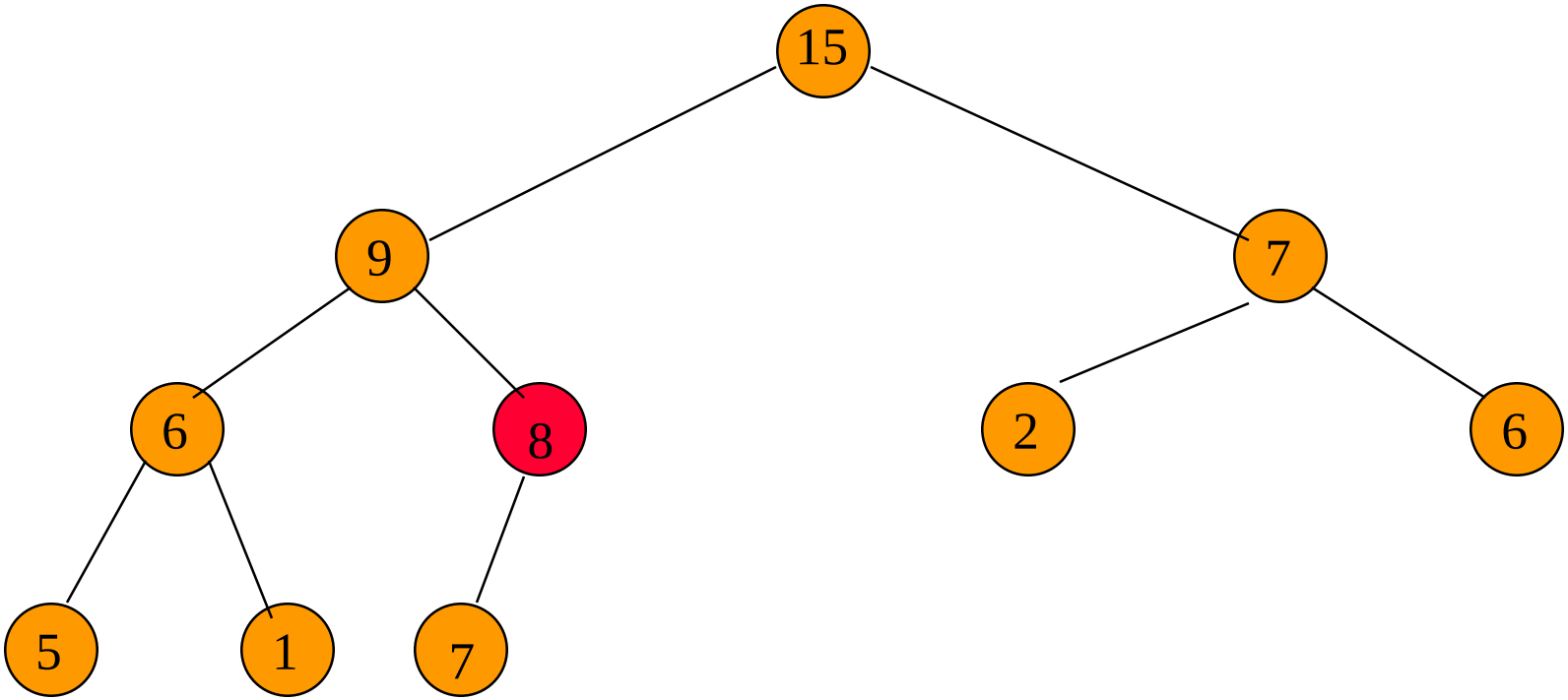
Reinsert 8 into the heap.

Removing The Max Element



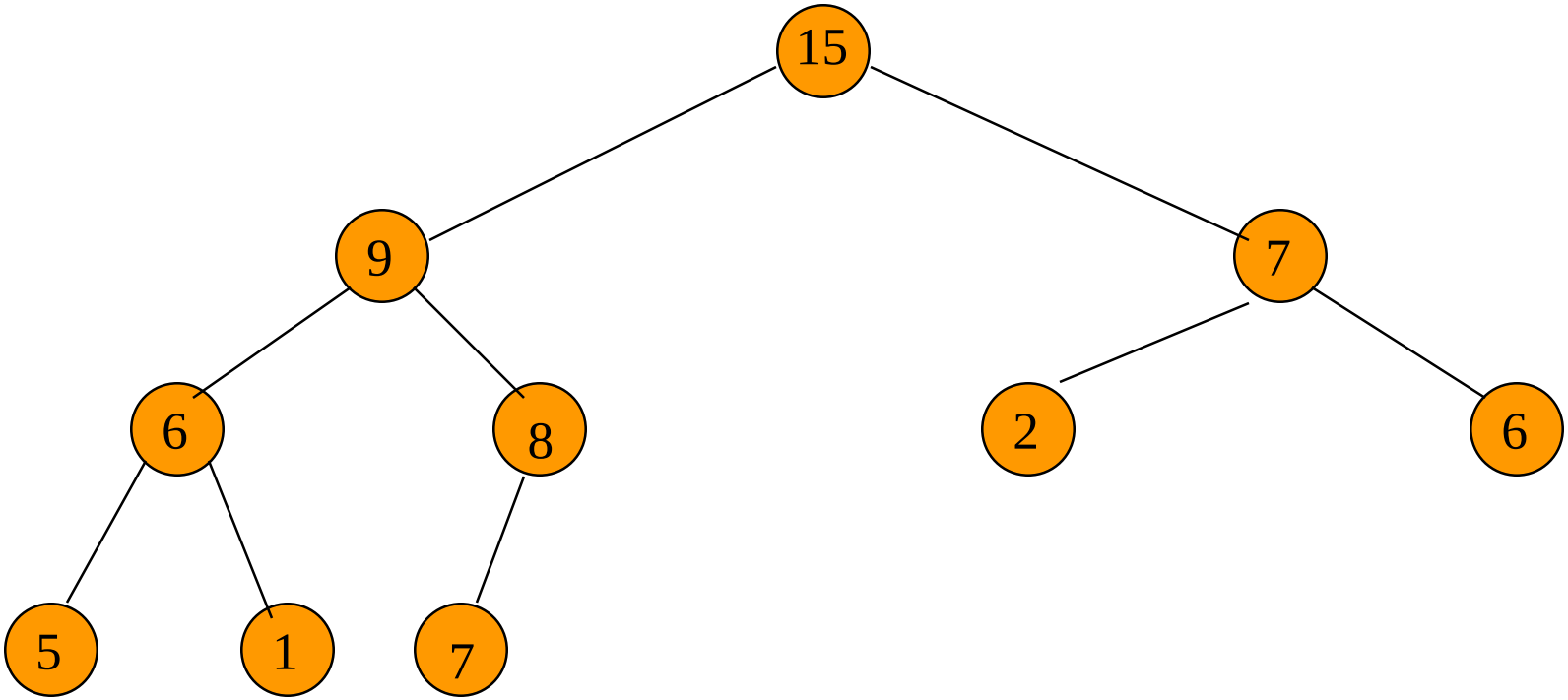
Reinsert 8 into the heap.

Removing The Max Element



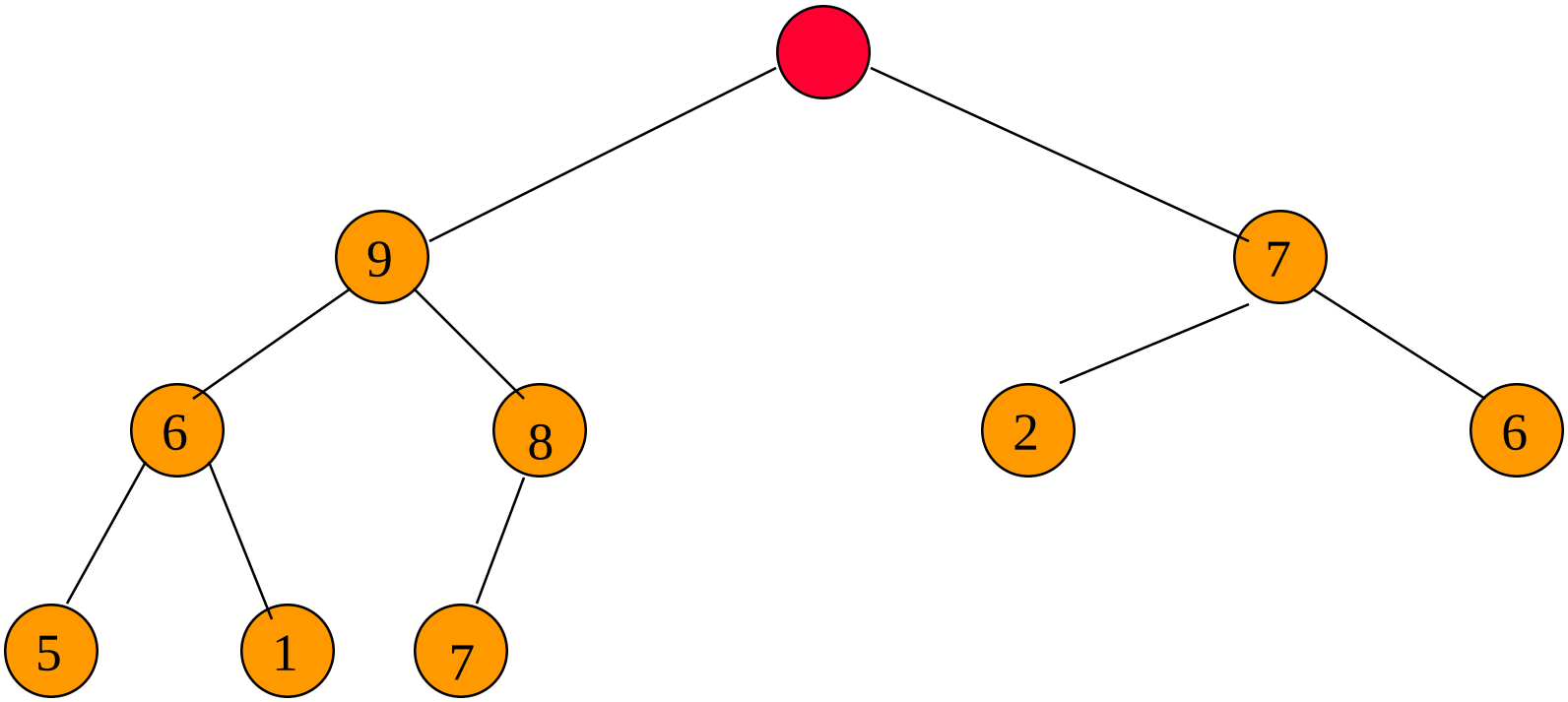
Reinsert 8 into the heap.

Removing The Max Element



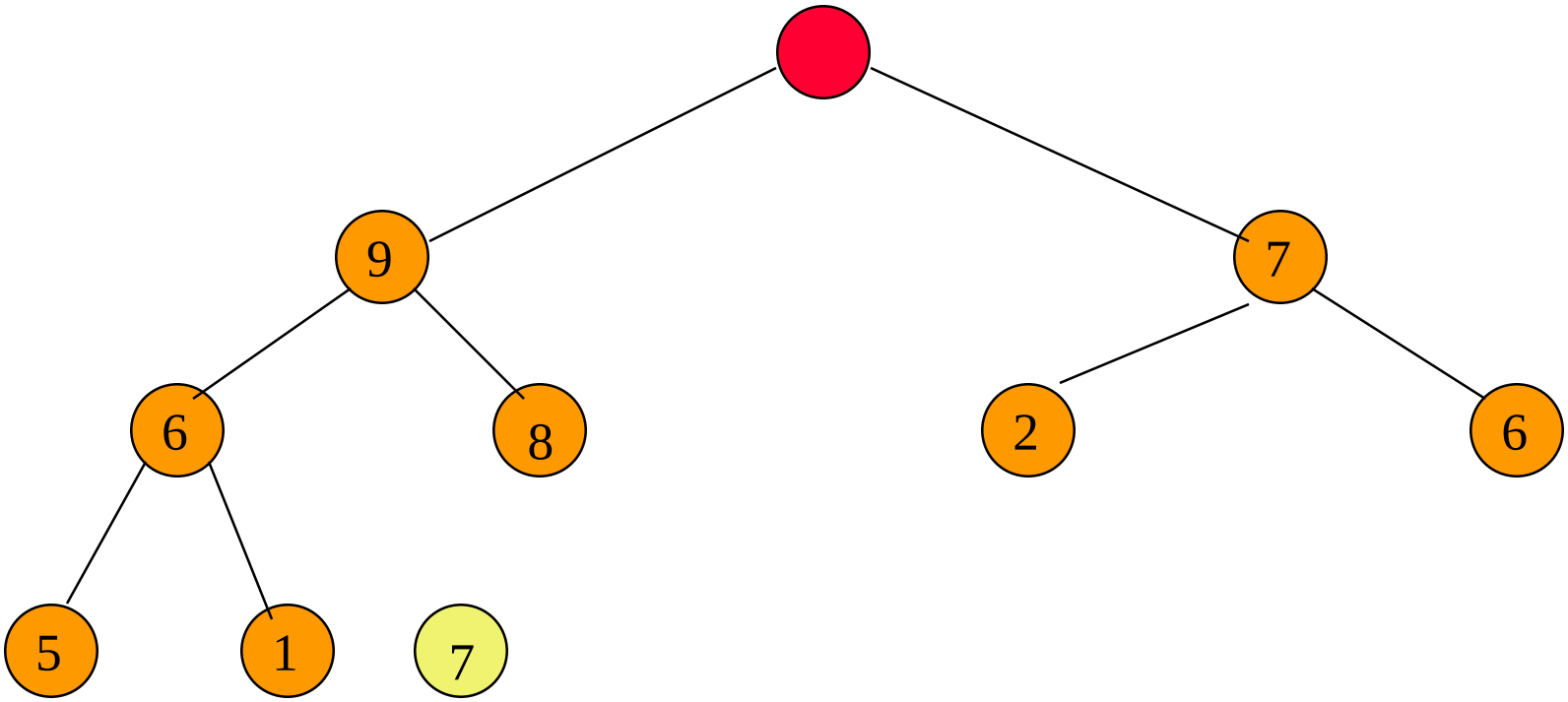
Max element is 15.

Removing The Max Element



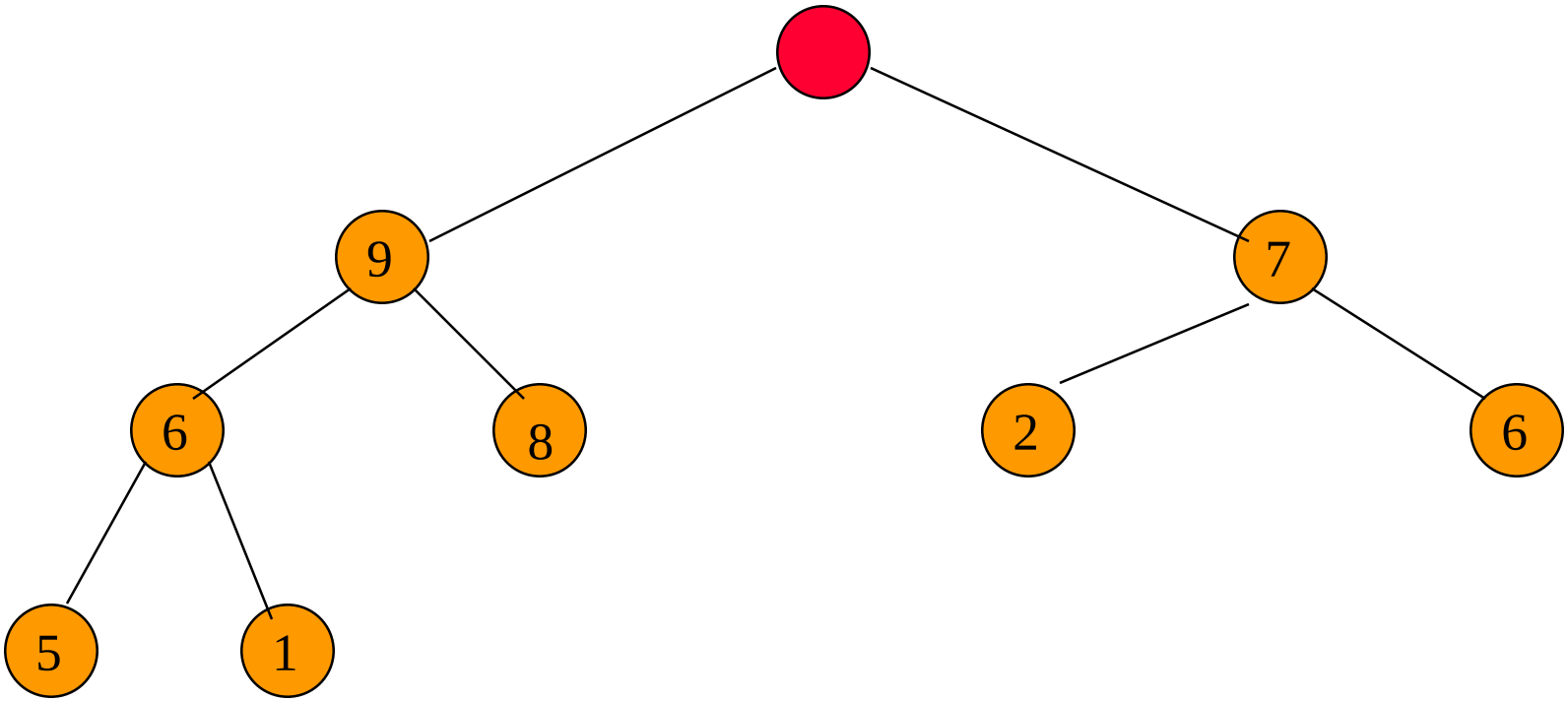
After max element is removed.

Removing The Max Element



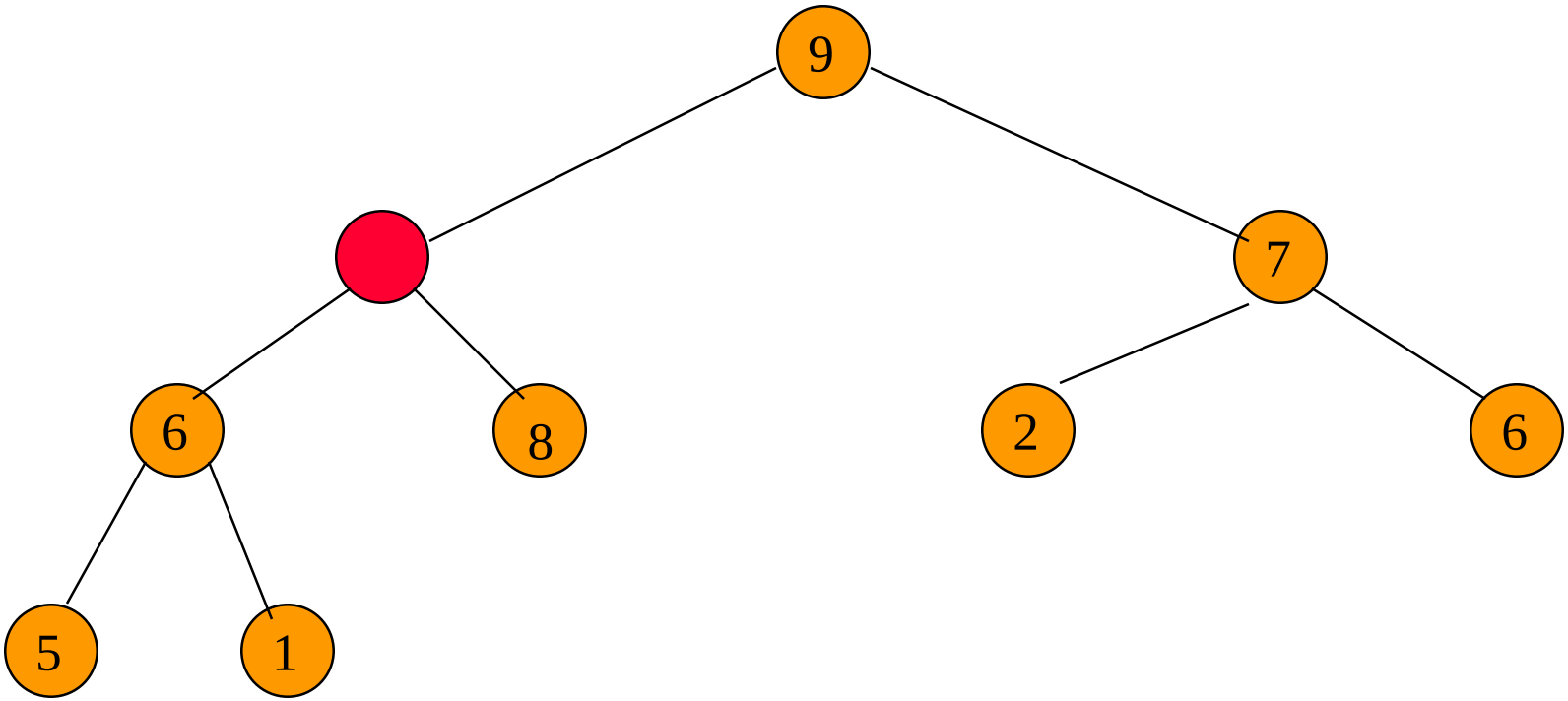
Heap with 9 nodes.

Removing The Max Element



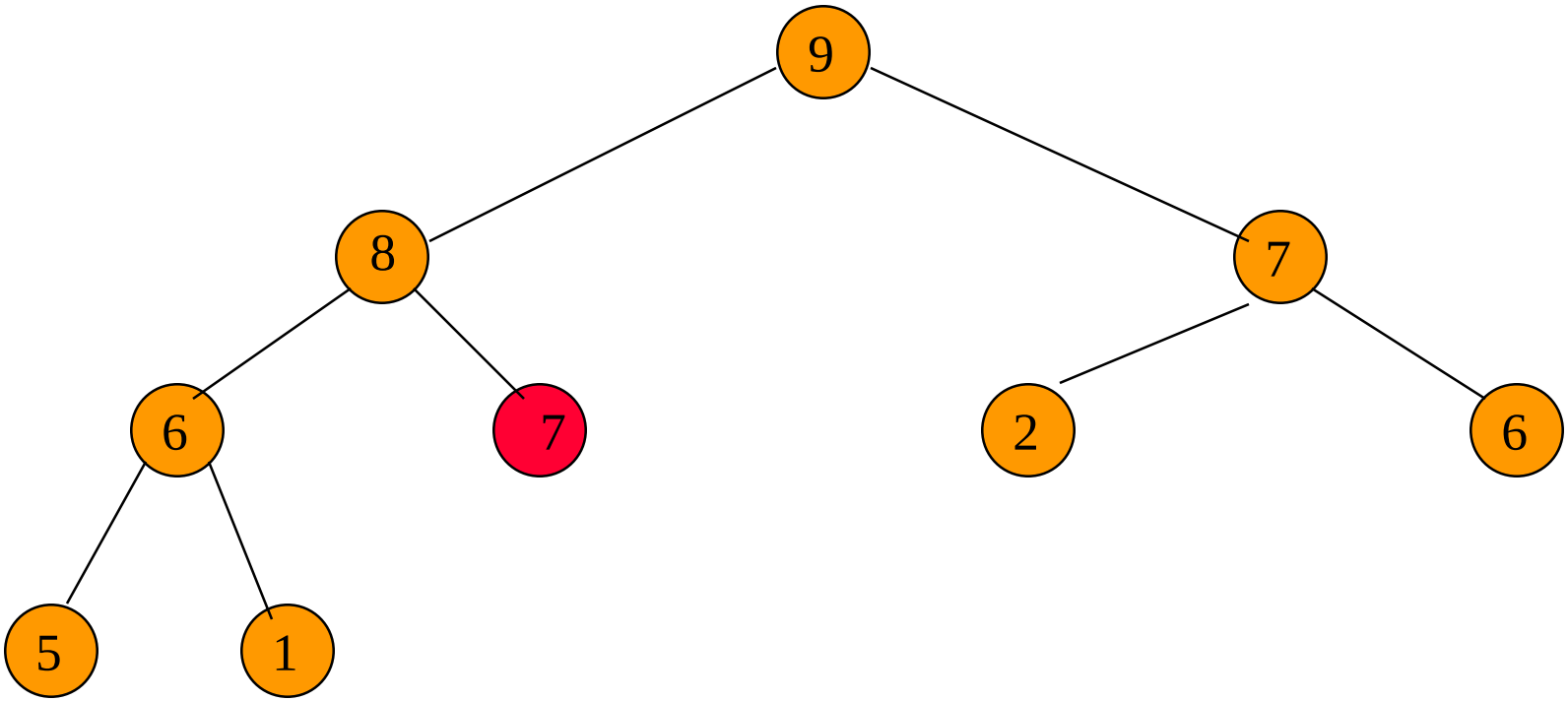
Reinsert 7.

Removing The Max Element



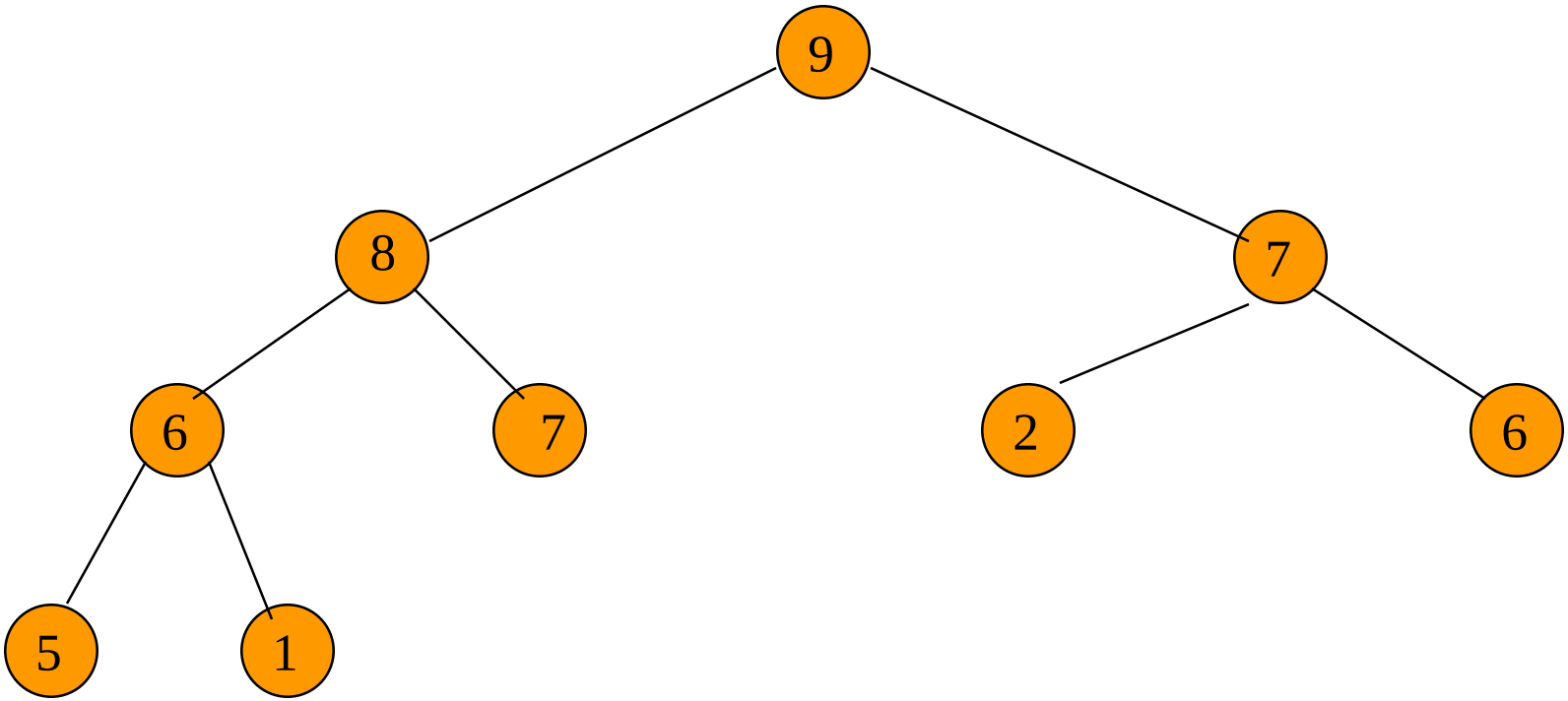
Reinsert 7.

Removing The Max Element



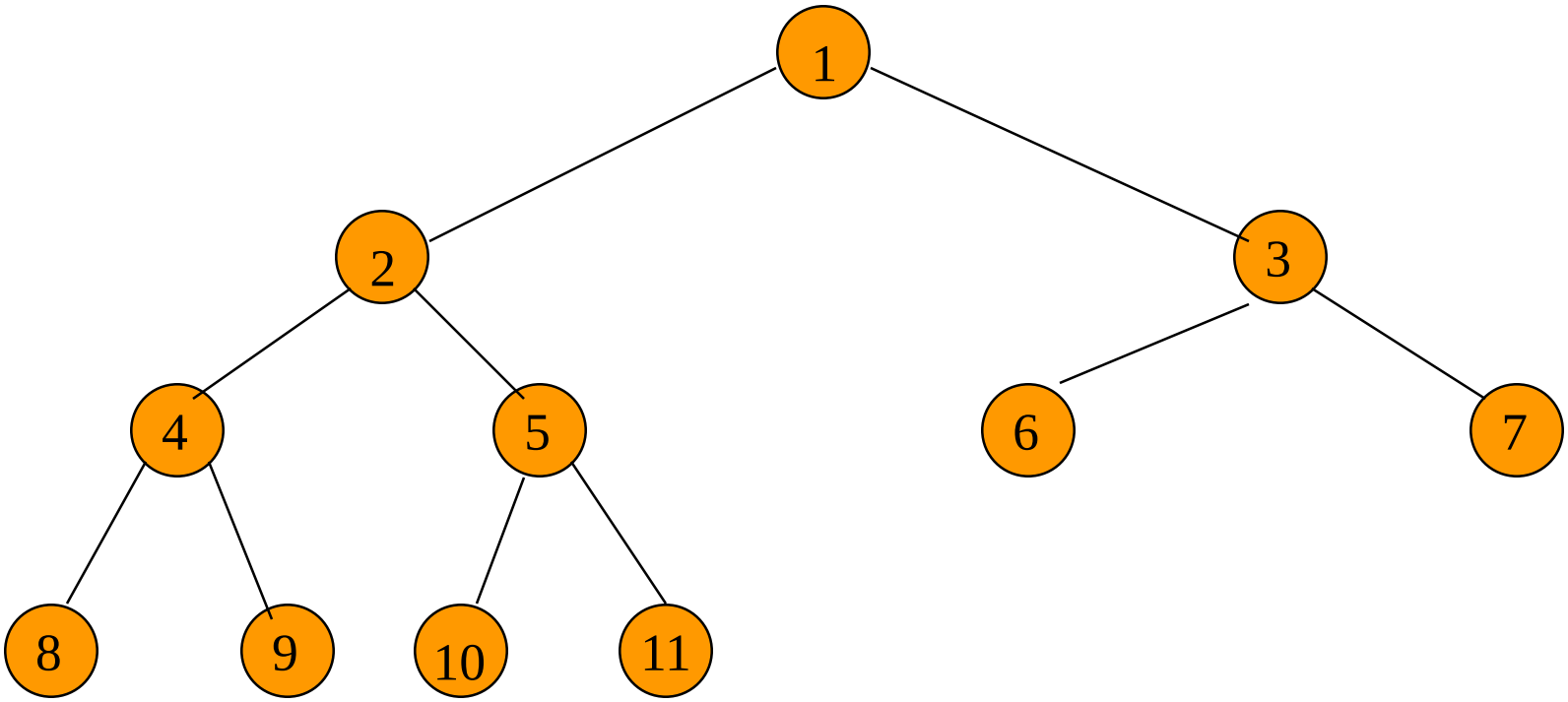
Reinsert 7.

Complexity Of Remove Max Element



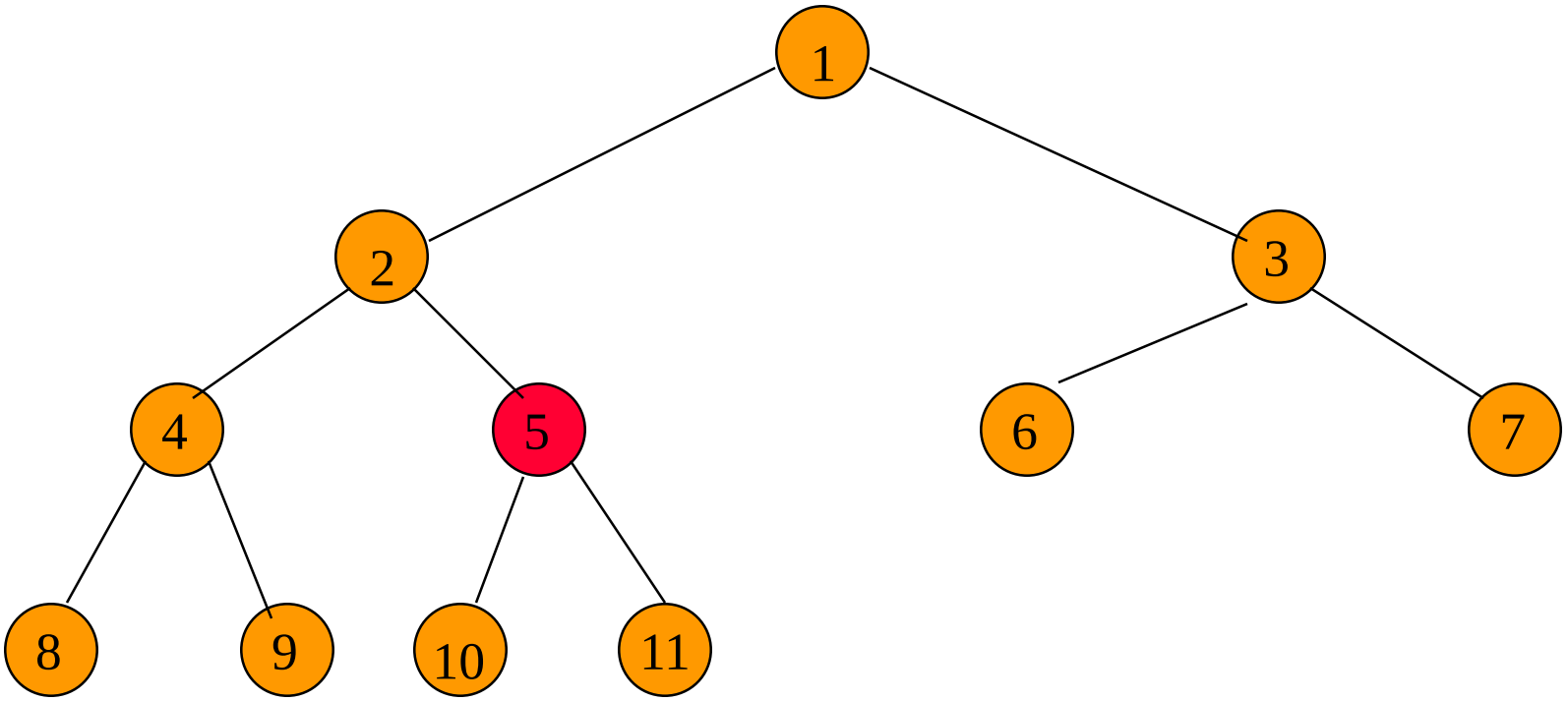
Complexity is O(log n).

Initializing A Max Heap



input array = [-, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

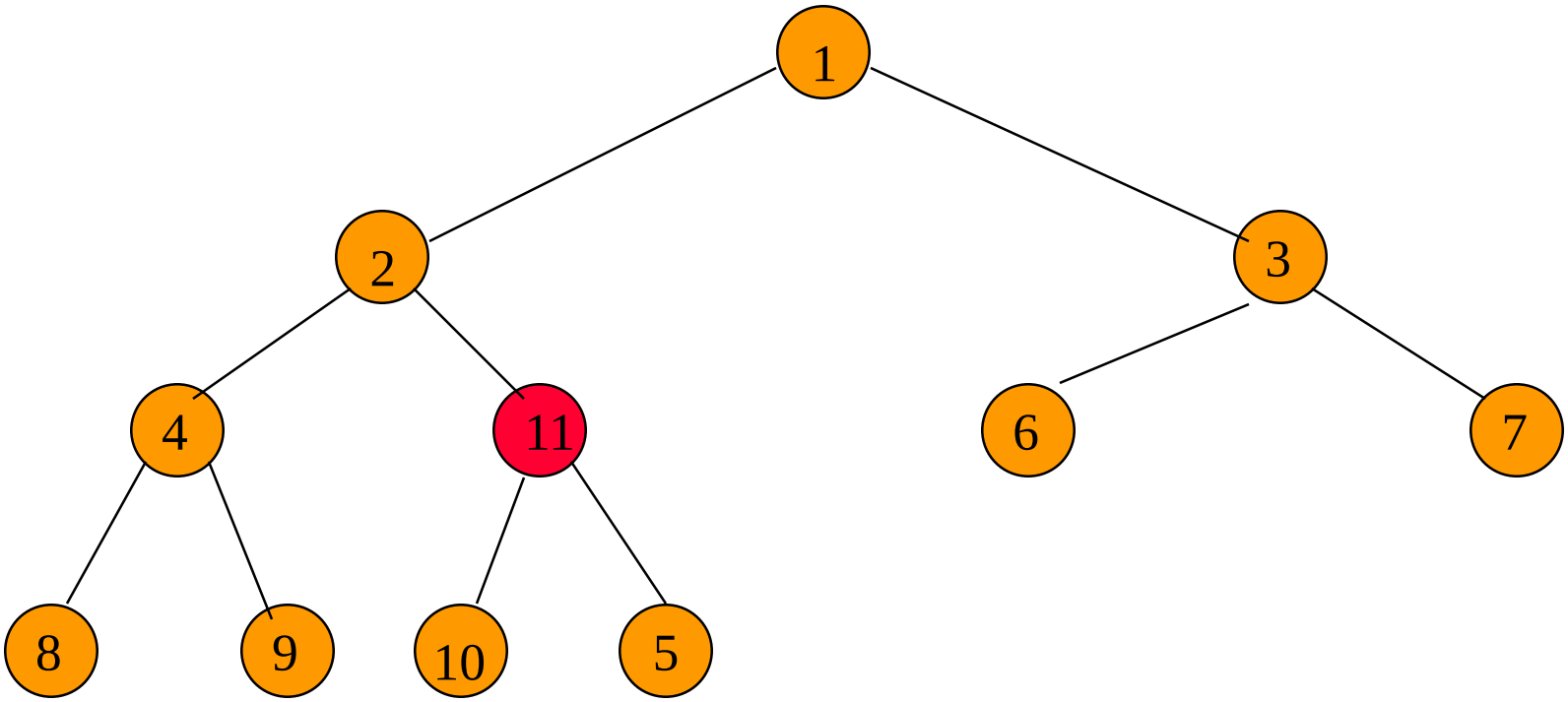
Initializing A Max Heap



Start at rightmost array position that has a child.

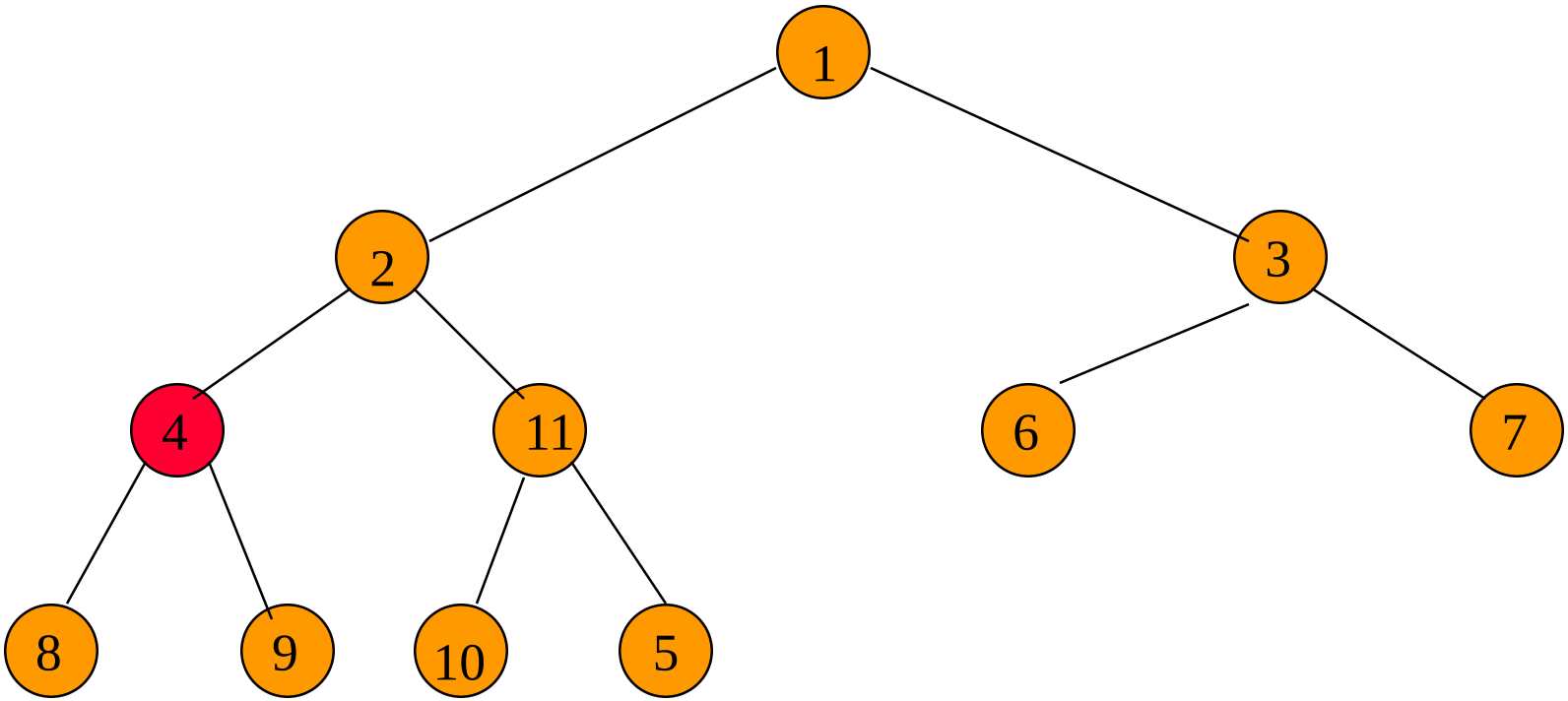
Index is n/2.

Initializing A Max Heap

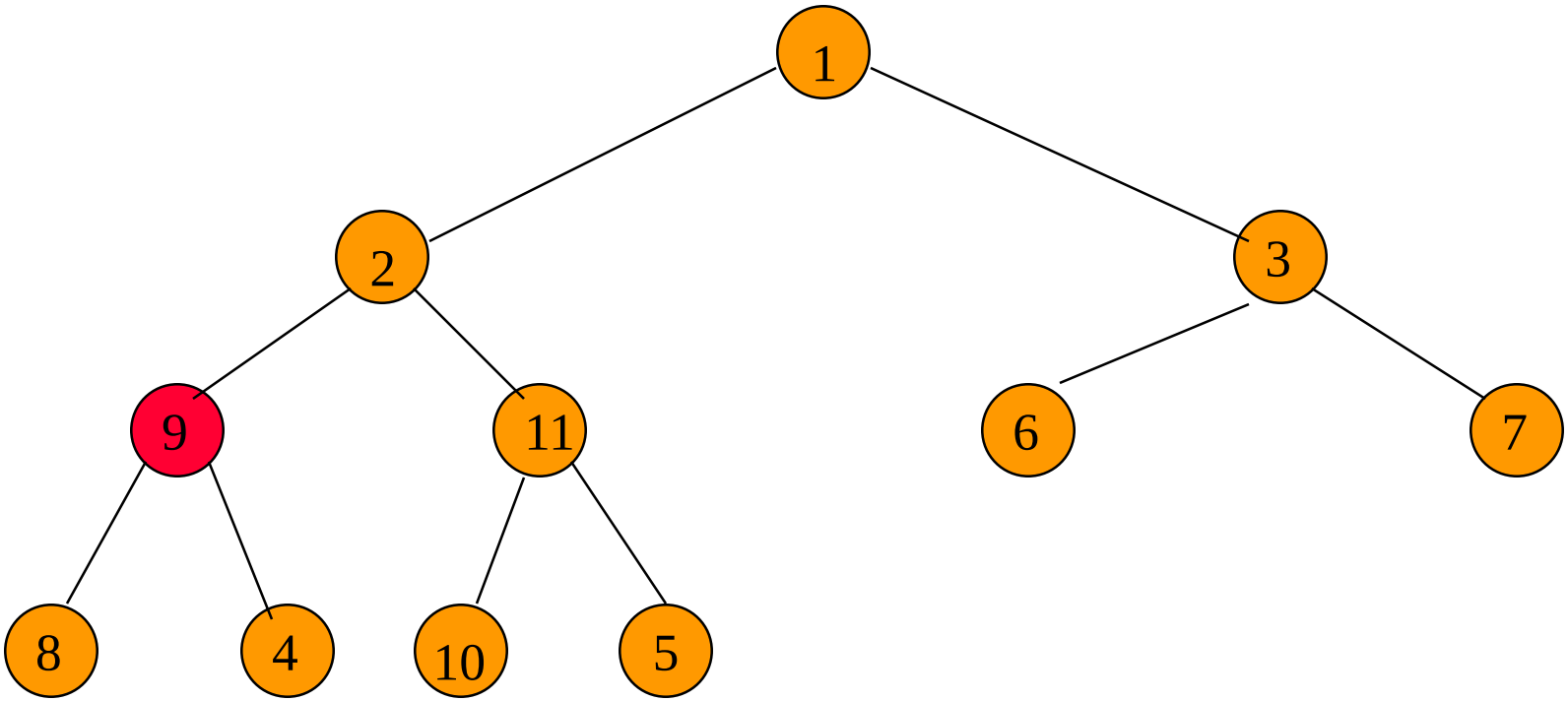


Move to next lower array position.

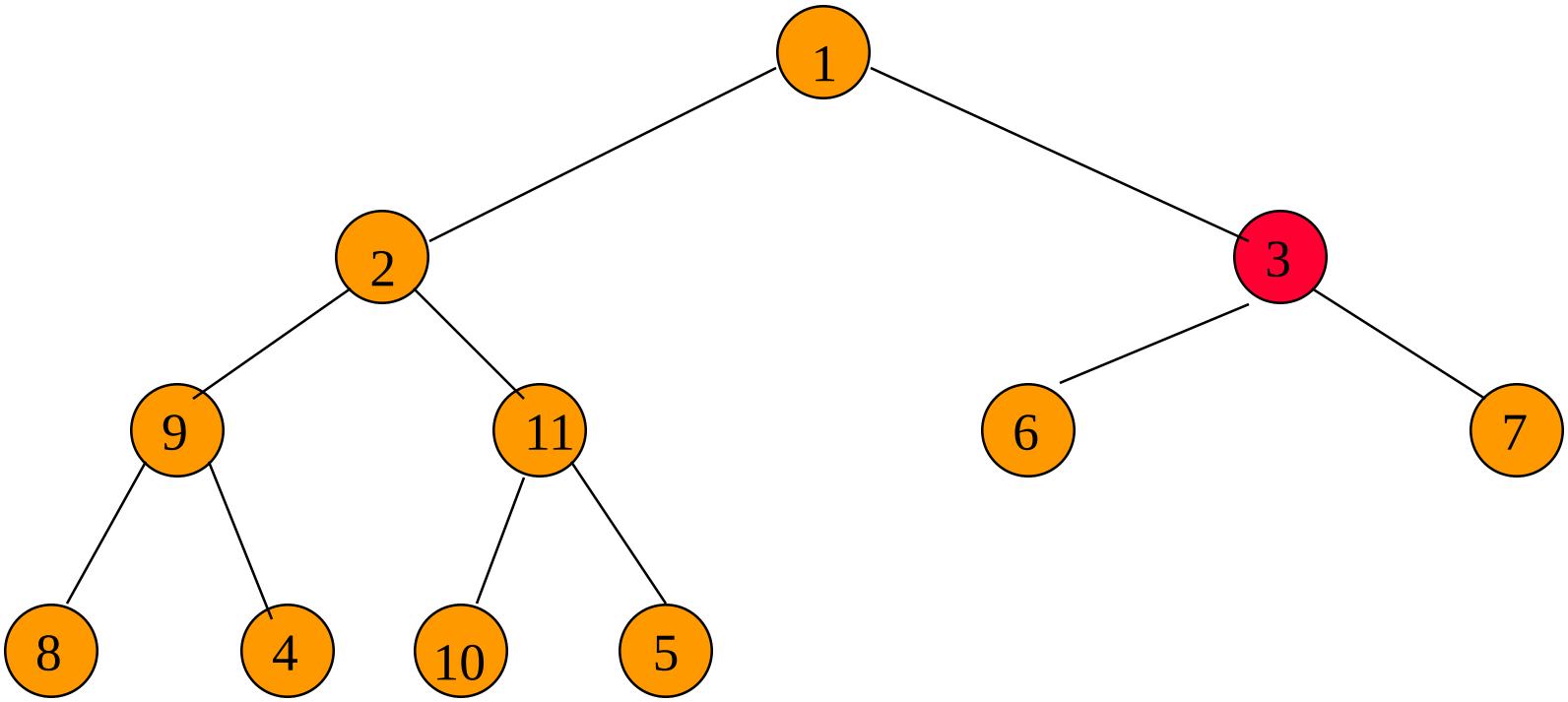
Initializing A Max Heap



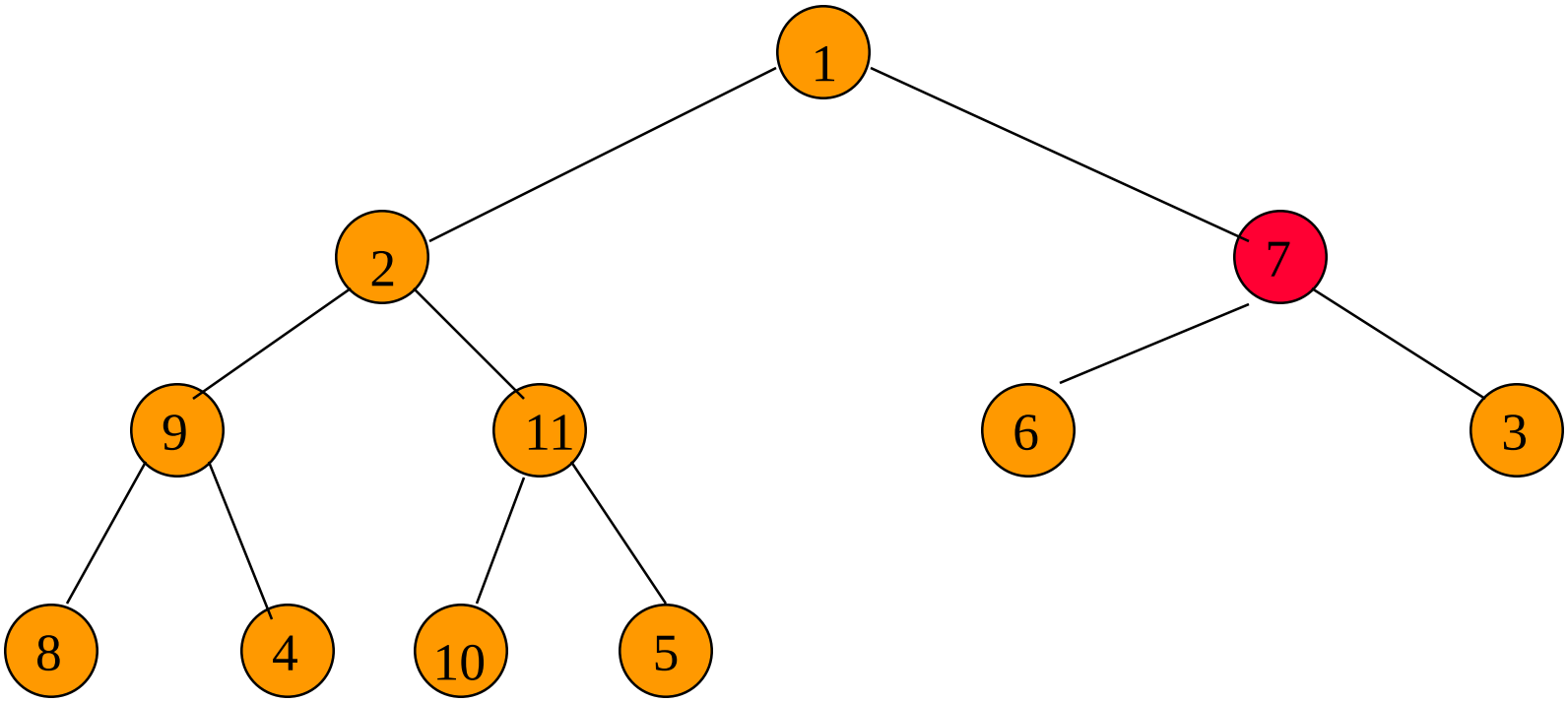
Initializing A Max Heap



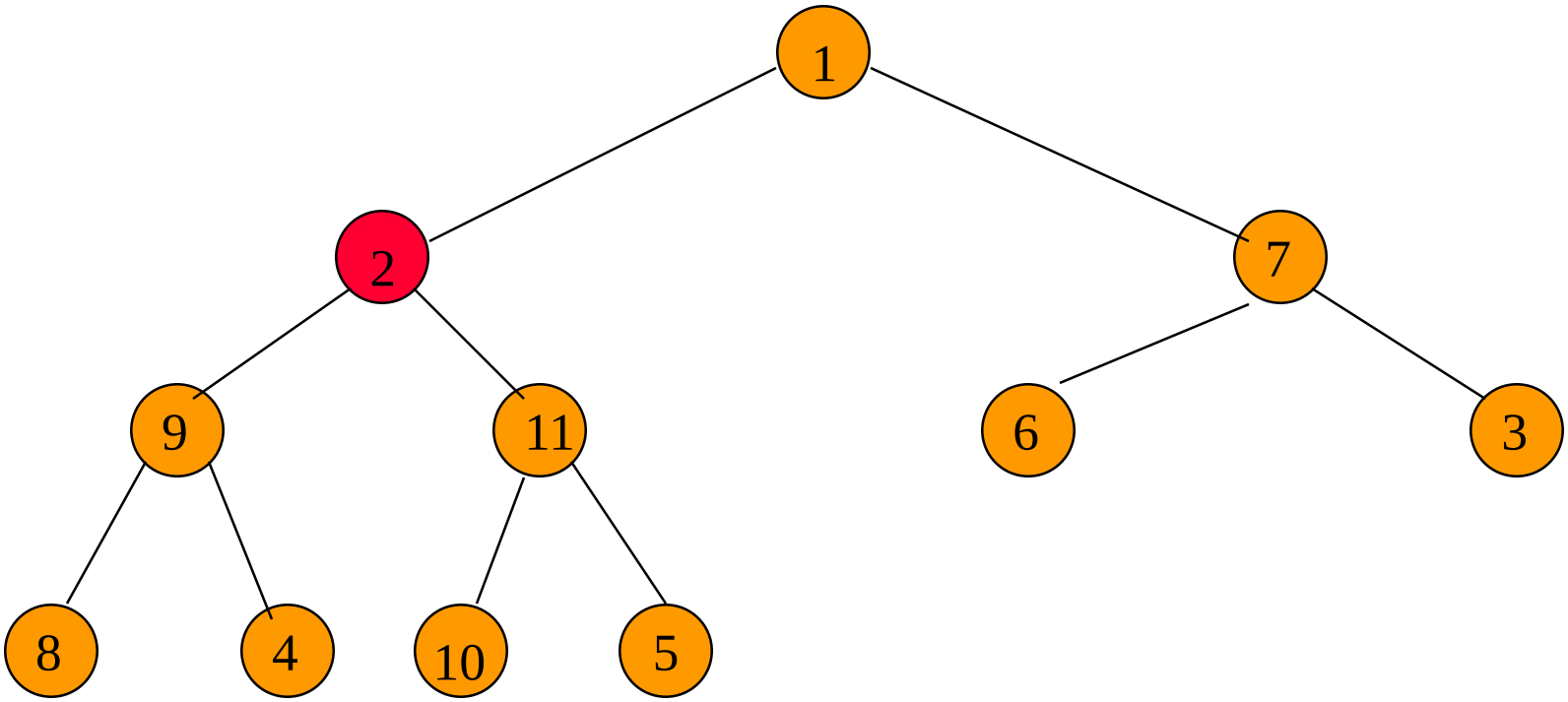
Initializing A Max Heap



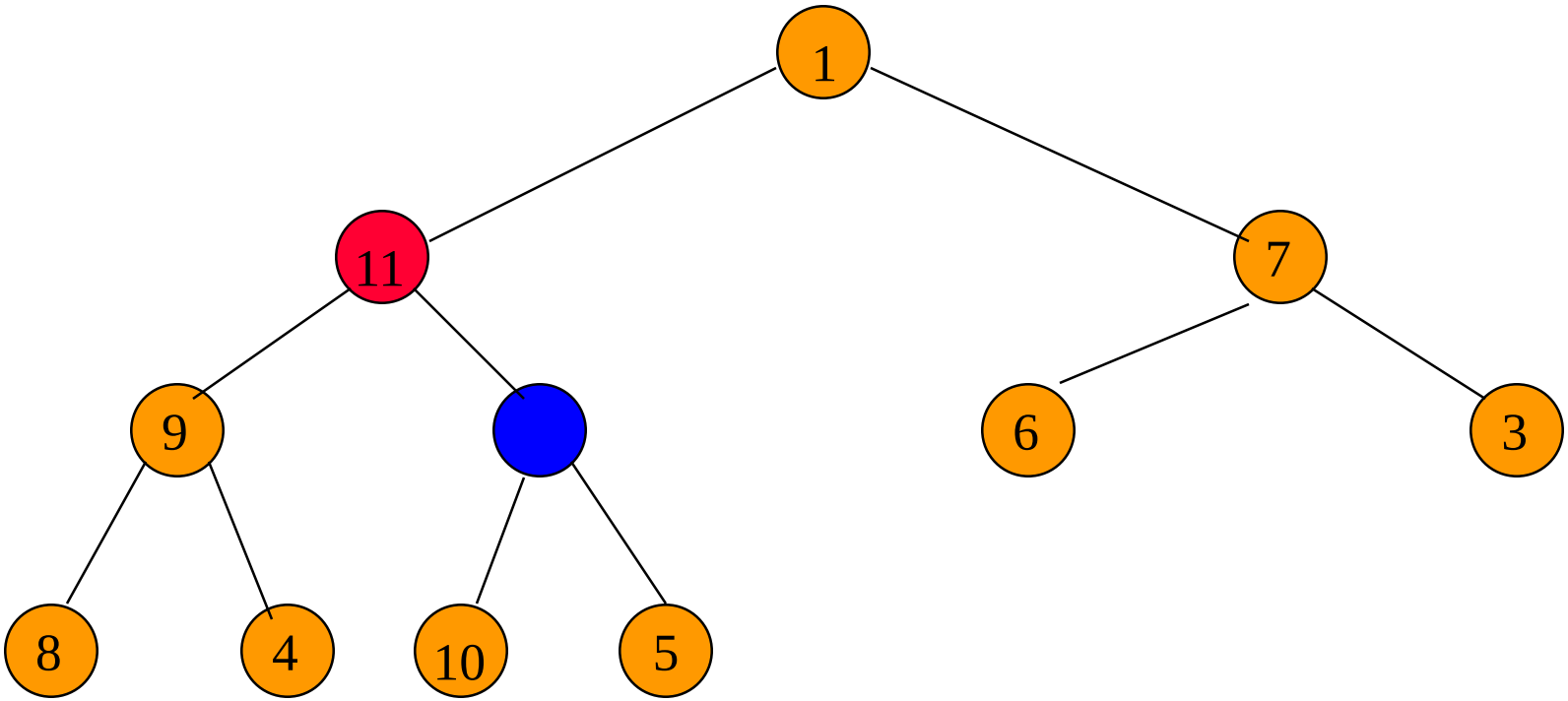
Initializing A Max Heap



Initializing A Max Heap

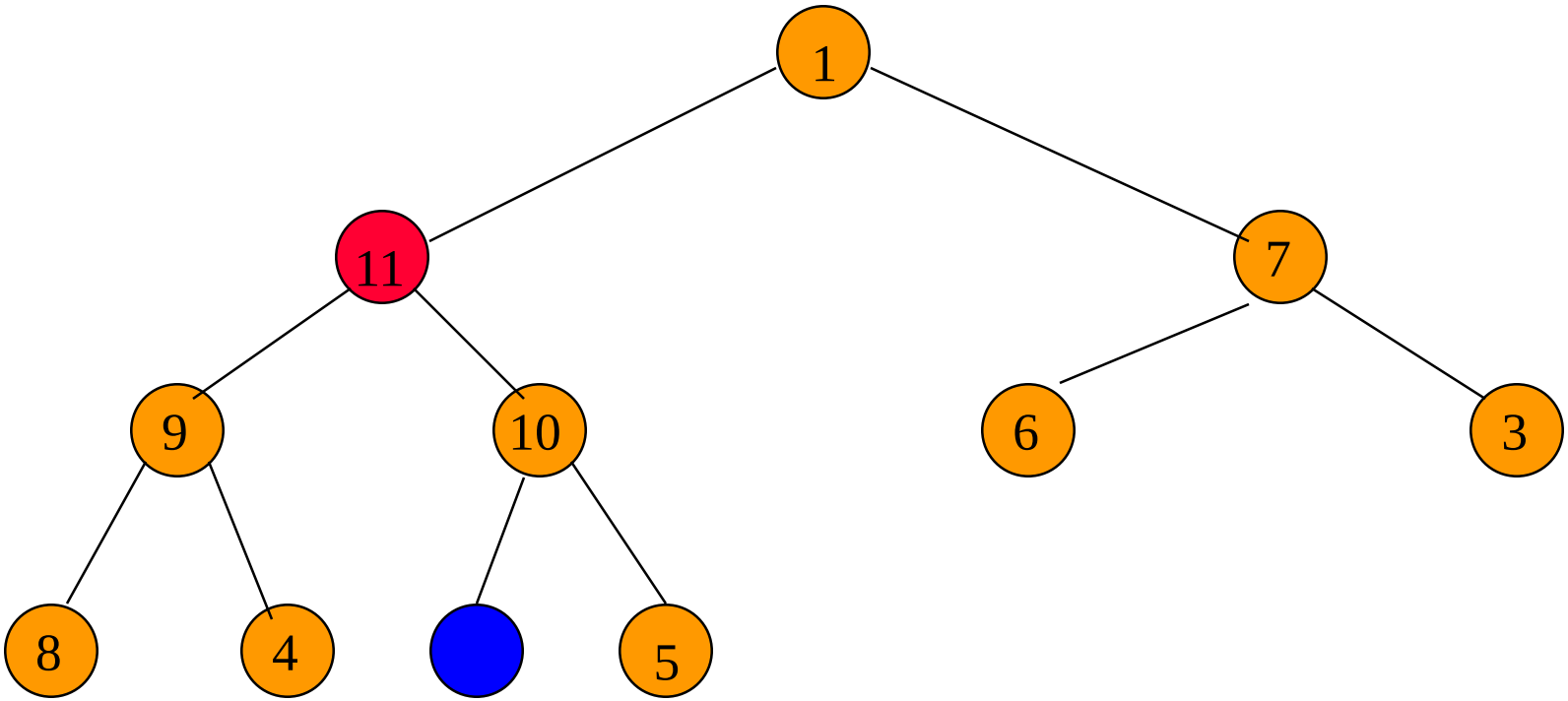


Initializing A Max Heap



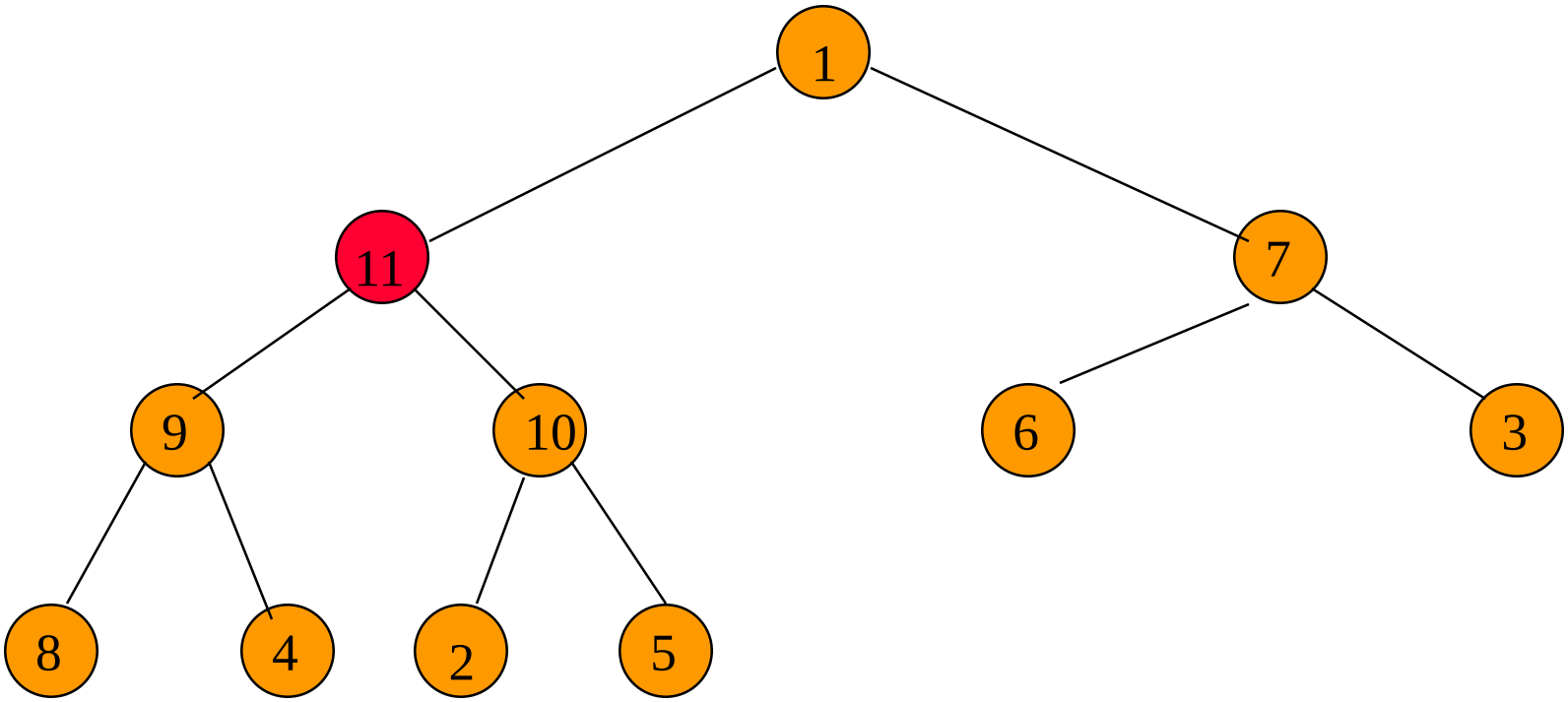
Find a home for 2.

Initializing A Max Heap



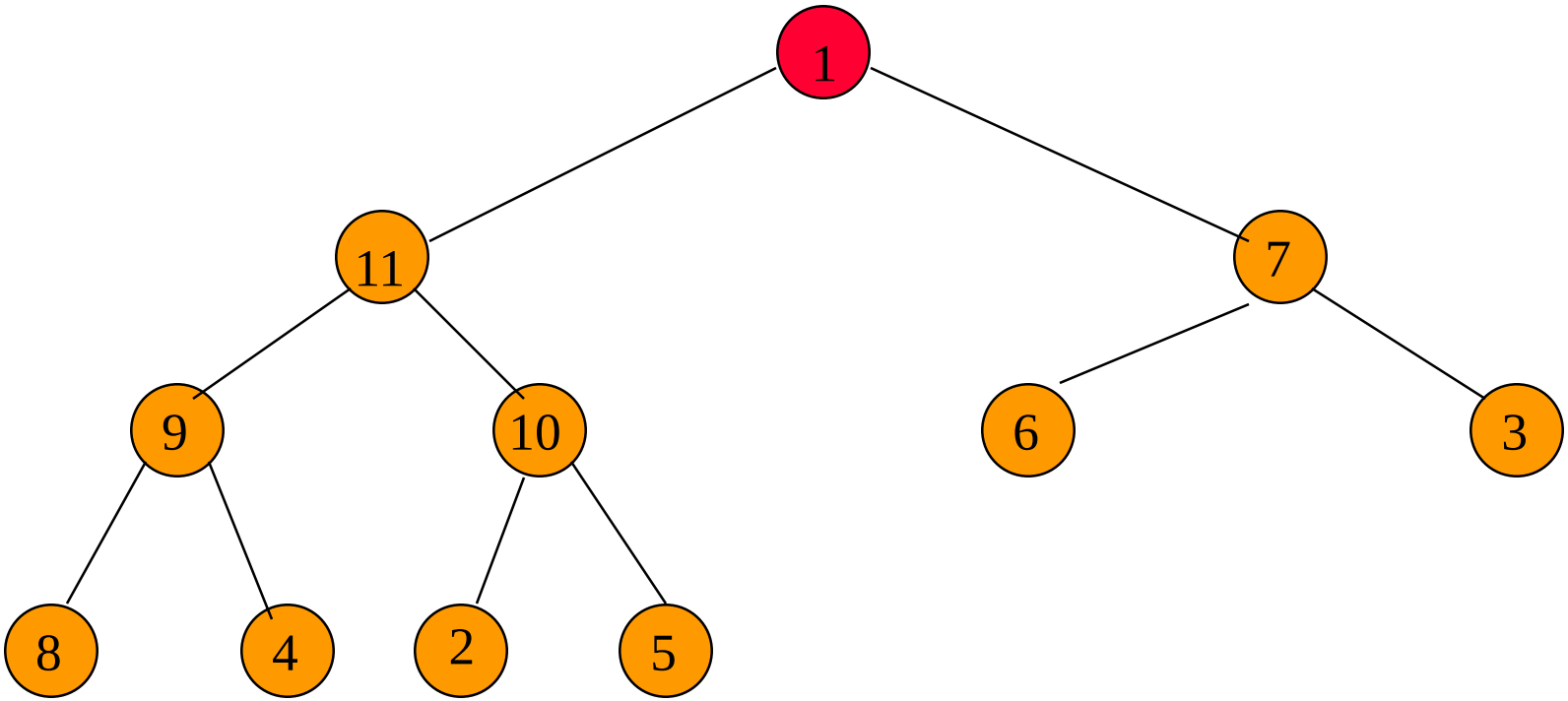
Find a home for 2.

Initializing A Max Heap



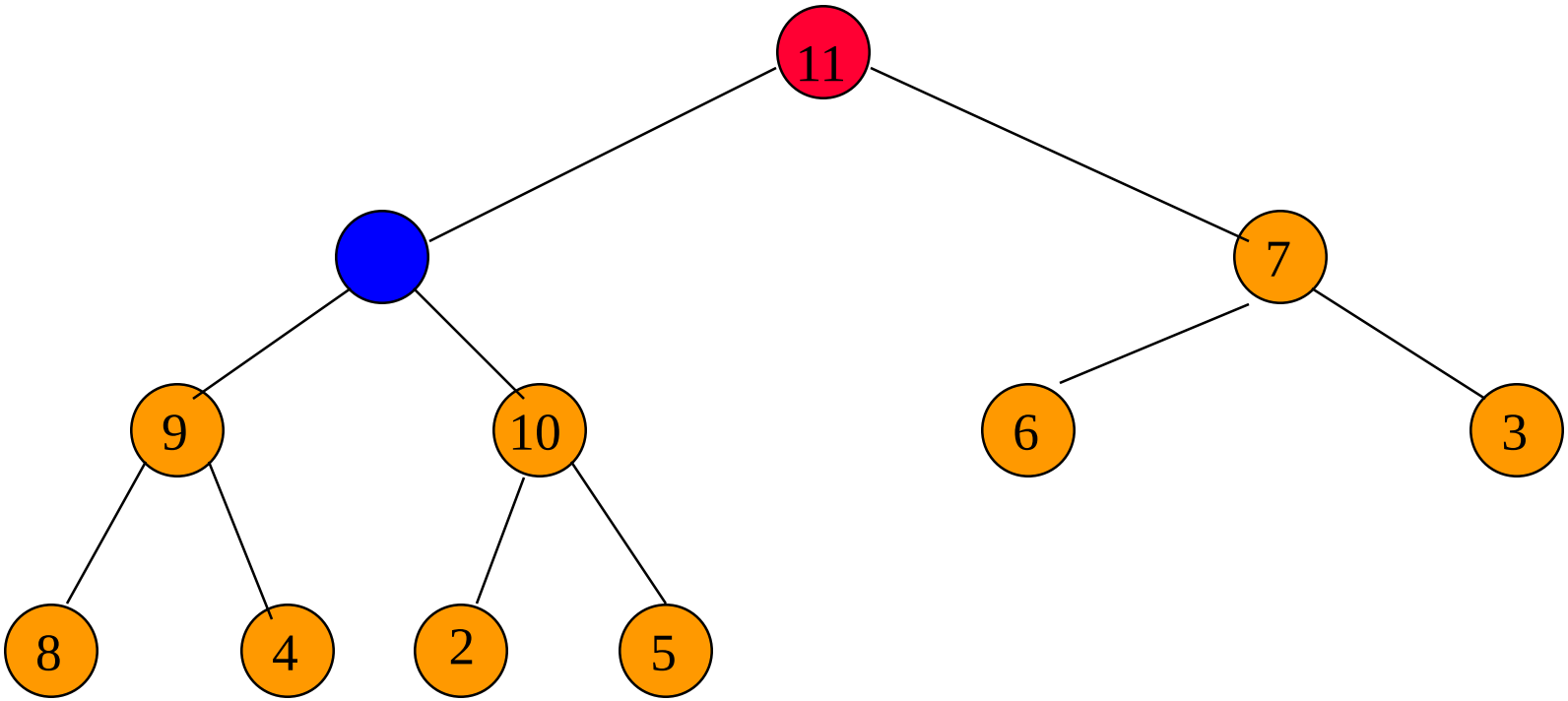
Done, move to next lower array position.

Initializing A Max Heap



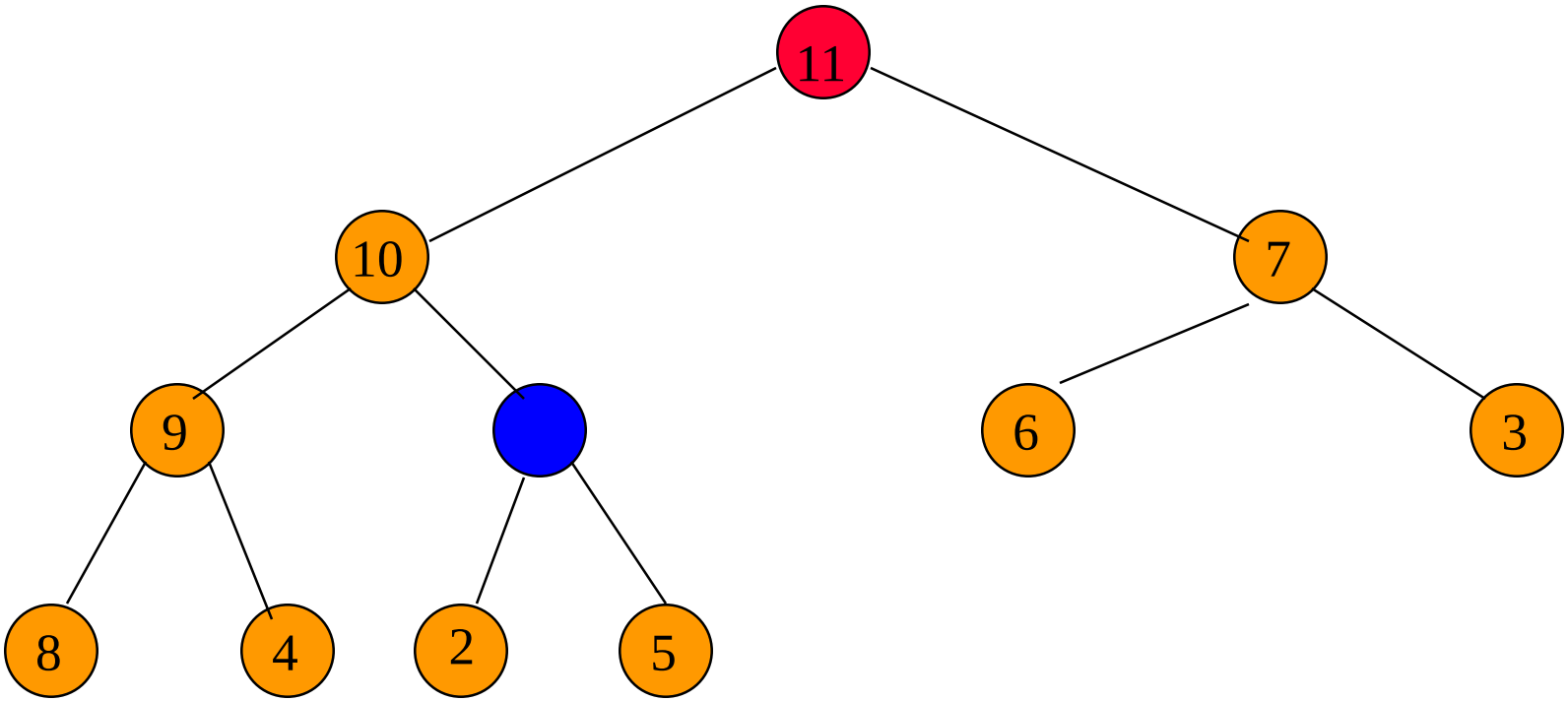
Find home for 1.

Initializing A Max Heap



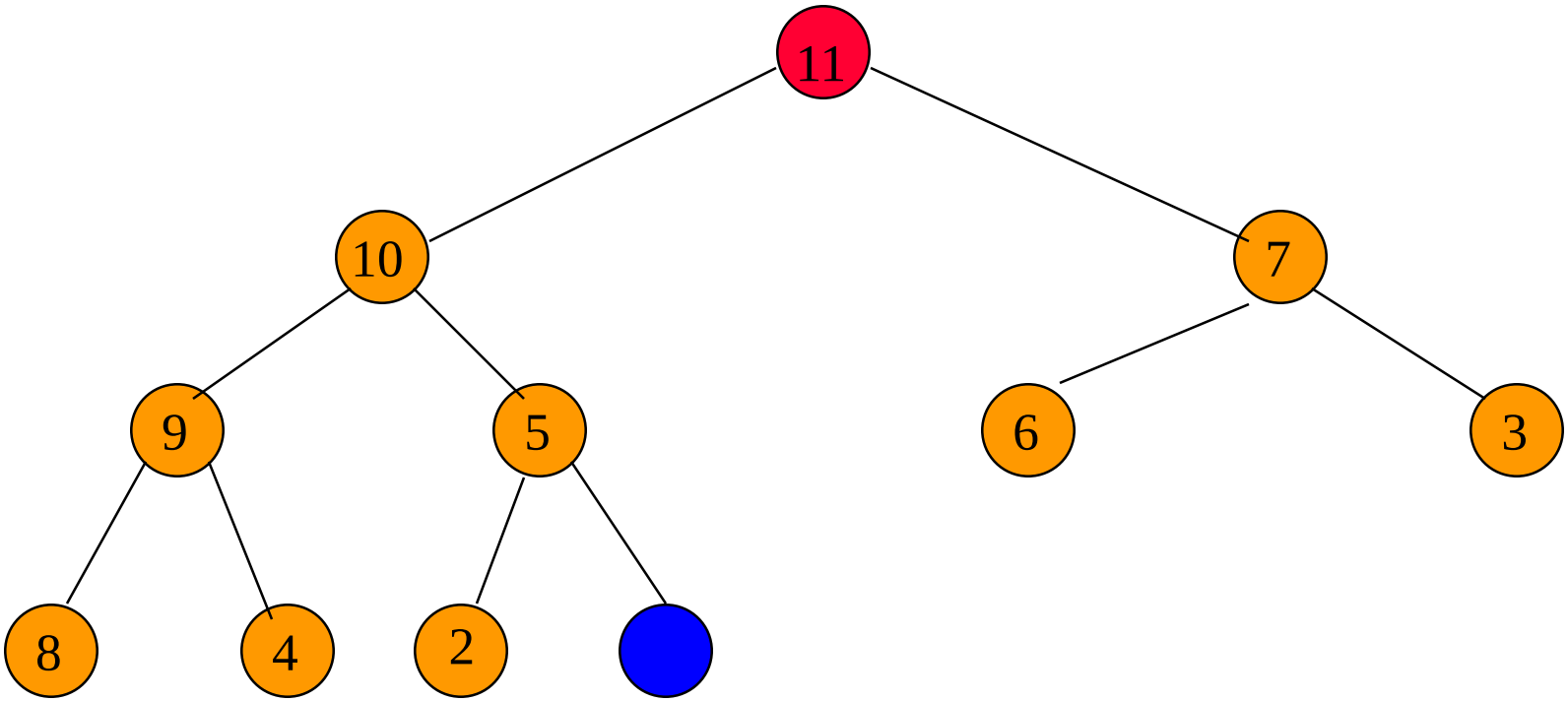
Find home for 1.

Initializing A Max Heap



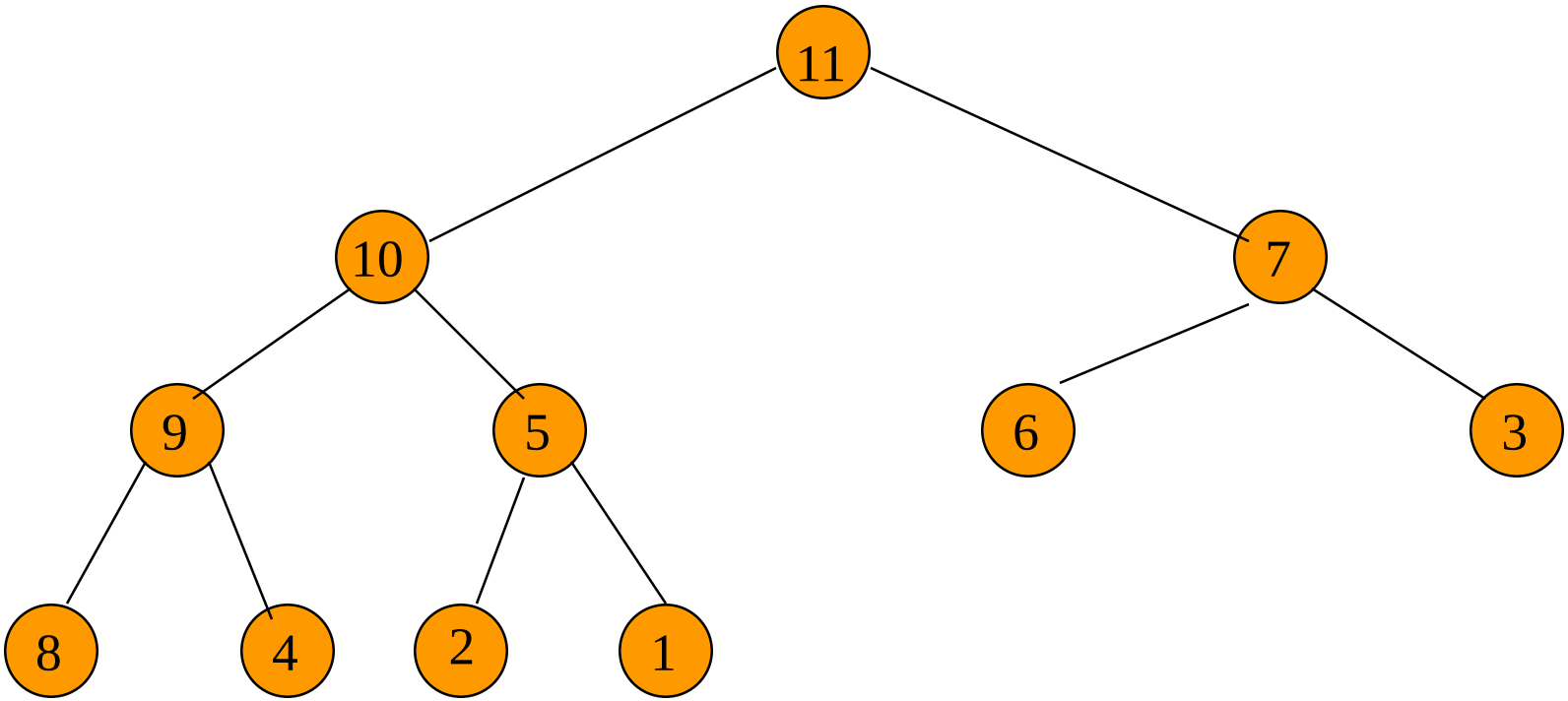
Find home for 1.

Initializing A Max Heap



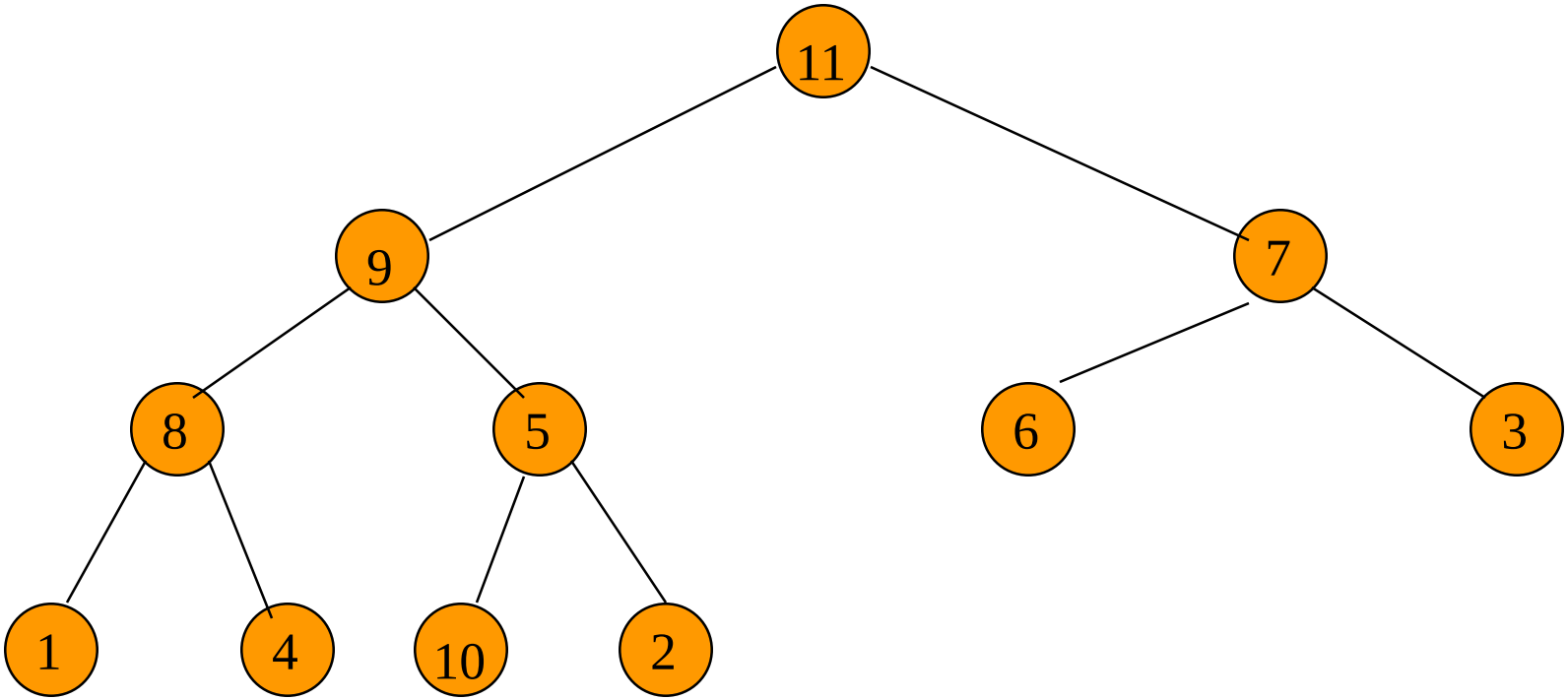
Find home for 1.

Initializing A Max Heap



Done.

Time Complexity



Height of heap = h.

Number of subtrees with root at level j is <= 2^{j-1}.

Time for each subtree is O(h-j+1).

Complexity

Time for level j subtrees is $<= 2^{j-1}(h-j+1) = t(j)$.

Total time is $t(1) + t(2) + \ldots + t(h-1) = O(n)$.

Huffman codes

Fixed length codes

- Want to represent data as a sequence of 0's and 1's

For example: BACADAEAFABBAAAGAH

- A-000 B-001 C-010 D-011 E-100 F-101 G-110 H-111

001 000 010 000 011 000 100 000 101 000 001 001 000 000 000 110 000 111

- This is a fixed length code.
- Can we make the sequence of 0's and 1's shorter ?

Variable length codes

- A 0 B 100 C 1010 D 1011

  E 1100 F 1101 G 1110 H 1111

10001010010110110001101010010000111001111

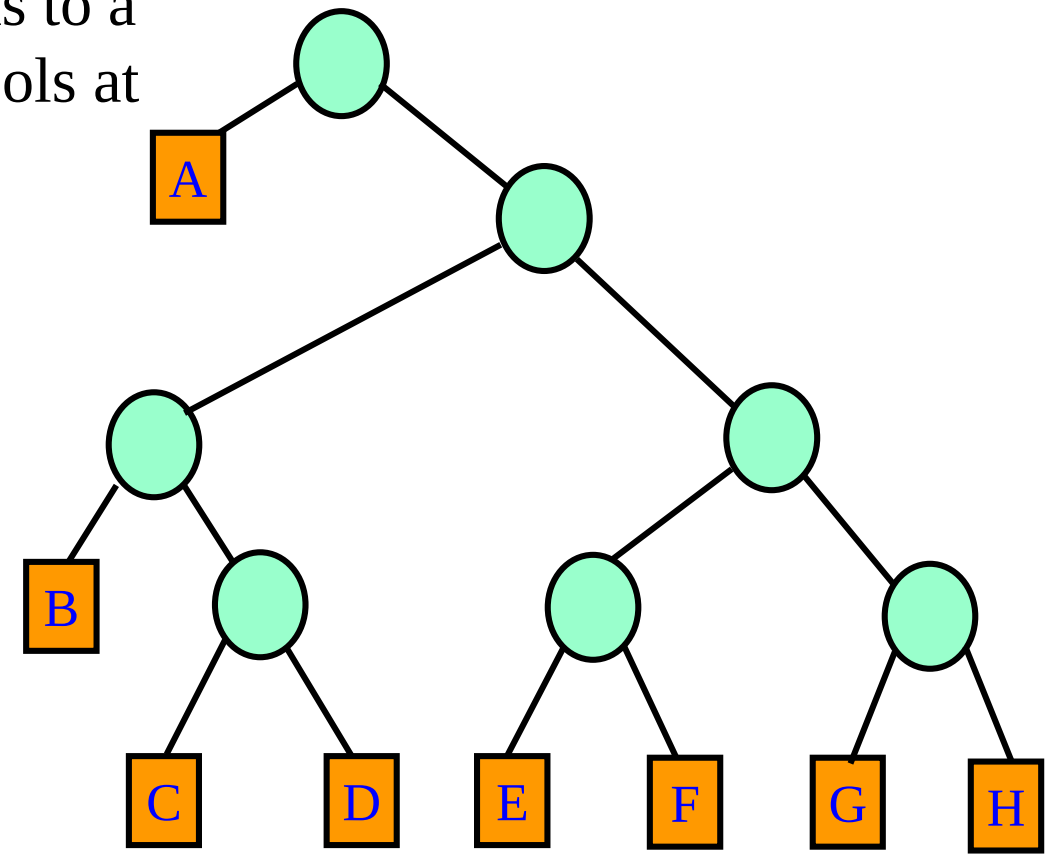
- This is a variable length code.
- How do we decode ?
- Use prefix codes: No codeword is a prefix of the other

Representing prefix codes

A 0 B 100 C 1010 D 1011

E 1100 F 1101 G 1110 H 1111

- A prefix code corresponds to a binary tree with the symbols at the leaves and vice versa.

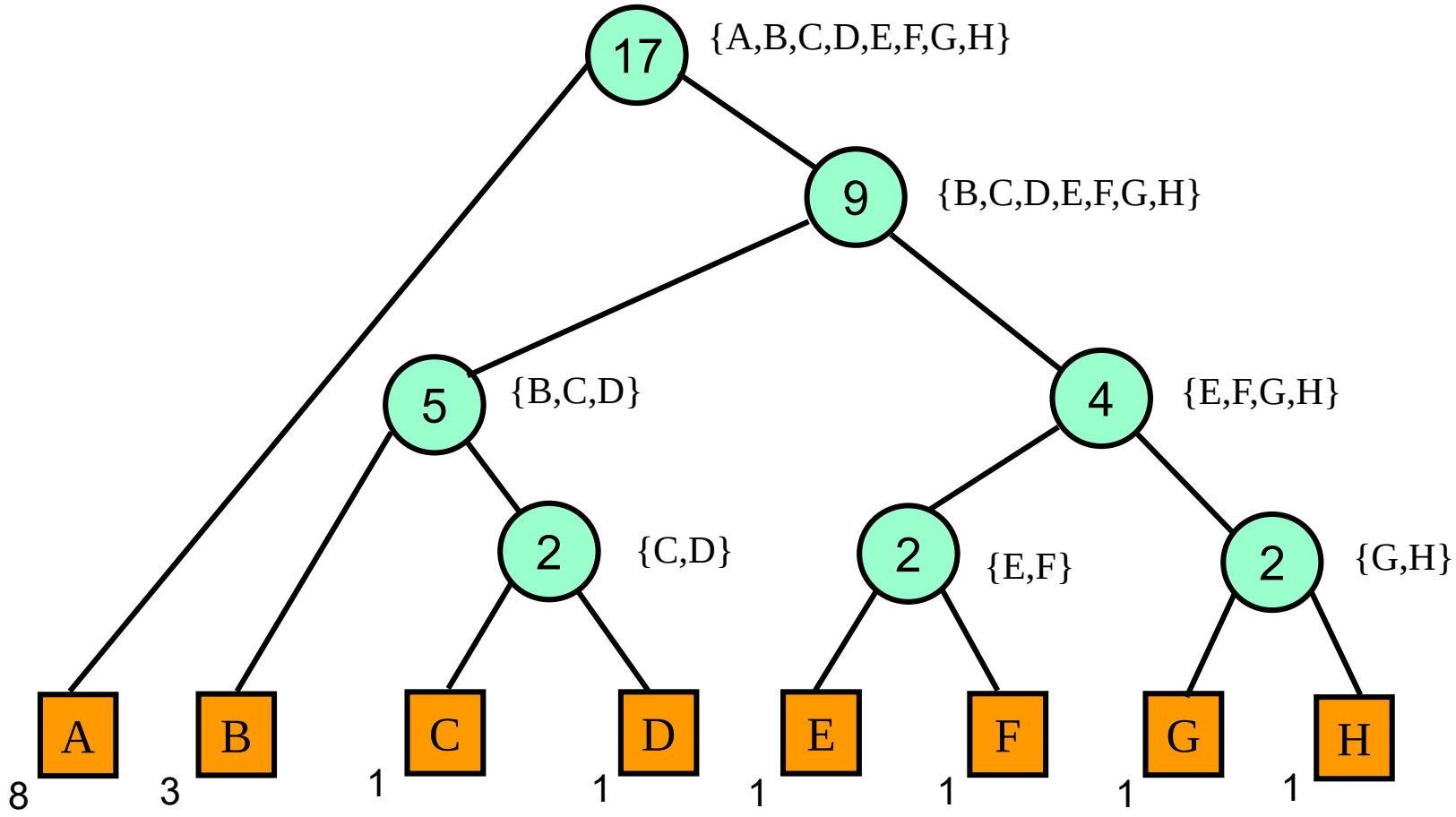


Construction Algorithm

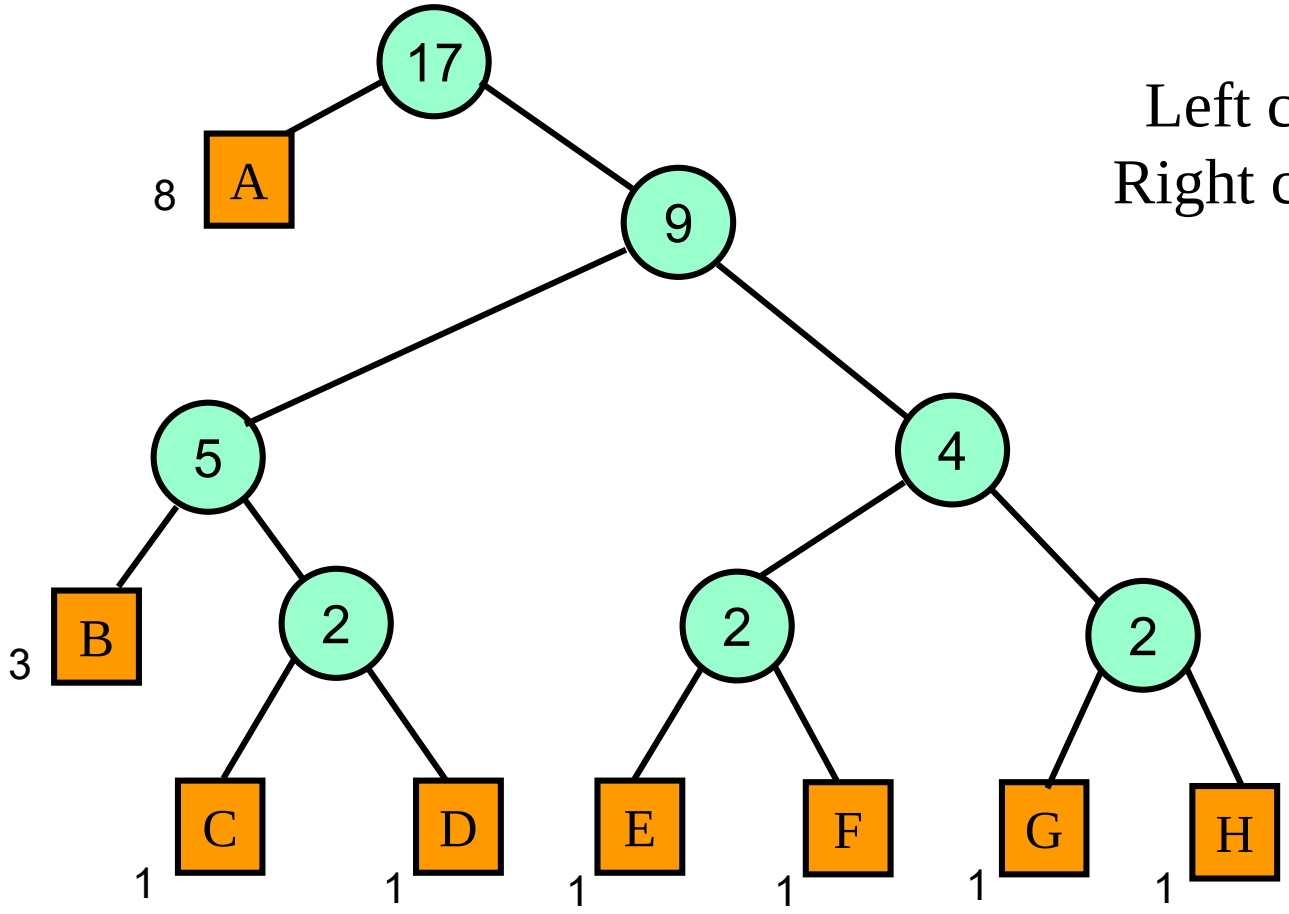
- Compute the character frequencies
- Insert the frequencies into a min heap
- Repeat
  - Delete the two minimum elements, f1 and f2 from the heap
  - insert f1+f2 into the heap

- Complexity: O(n log n).

Construction of Huffman tree



Representation



Left child – 0
Right child – 1

code(D) = 1011; code(F) = 1101