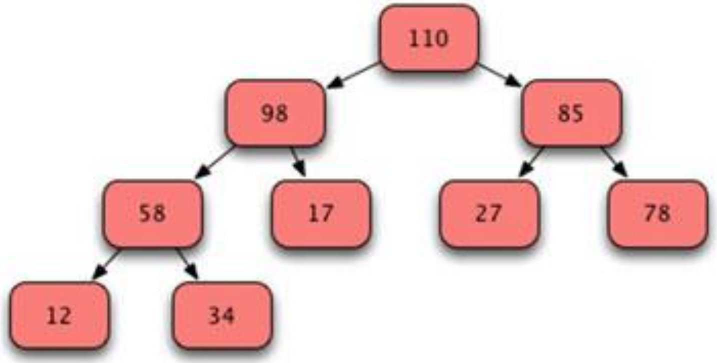
15CSE374
INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS

Sarath tv

Last Lecture

- BST

Binary Tree using Arrays

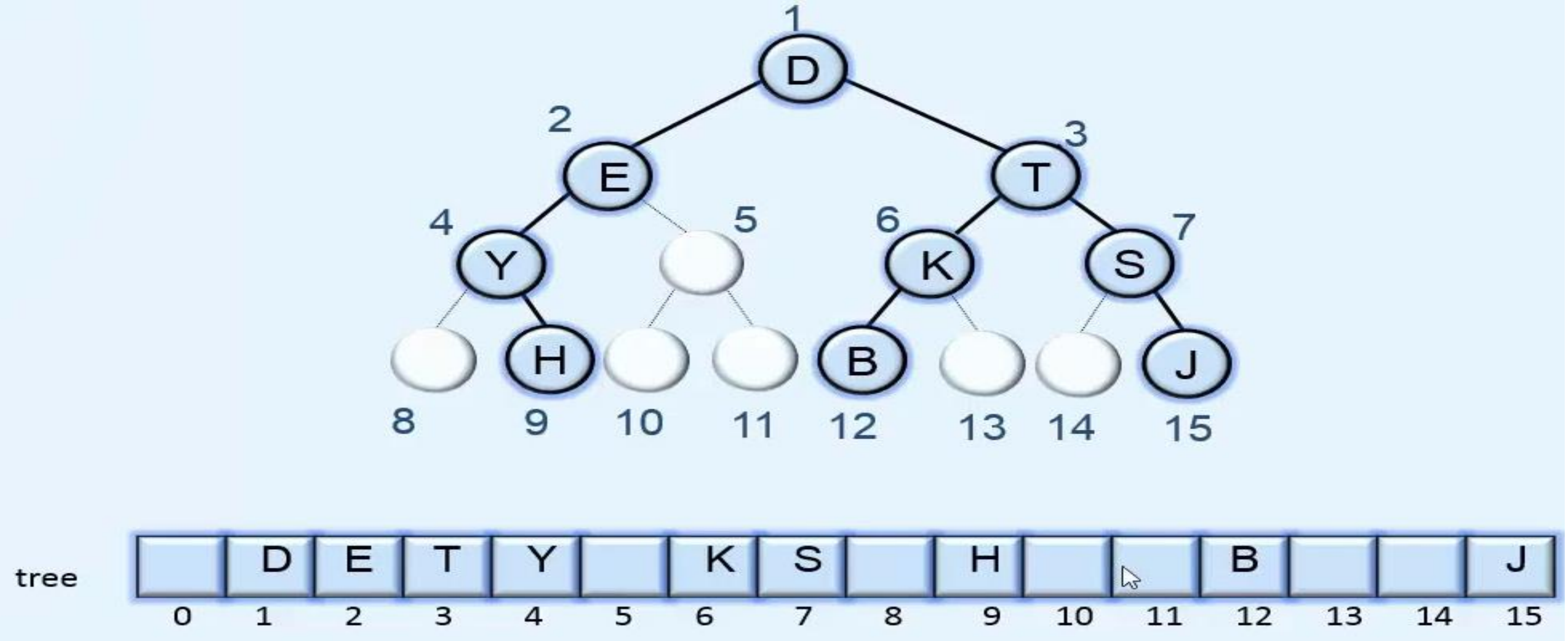


0	1	2	3	4	5	6	7	8	9	...	...	...
110	98	85	58	17	27	78	12	34				

- The children of any element of the array can be calculated from the index of the parent.
  - *leftChildIndex* = 2 ∗ *parentIndex* + 1
  - *rightChildIndex* = 2 ∗ *parentIndex* + 2
- It is also possible to go in the other direction. Given a child's index, we can discover where the parent is located.
  - *parentIndex = (childIndex* − 1*)//*2
- The **//** in the previous formula represents integer division. It means that the result is always an integer.

For indexing from **0** to **n-1**

For indexing from **1** to **n**



tree		D	E	T	Y		K	S		H			B			J
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- The children of any element of the array can be calculated from the index of the parent.
  - $leftChildIndex = 2 * parentIndex$
  - $rightChildIndex = 2 * parentIndex + 1$

Priority Queue

- A priority queue is an abstract data type similar to a regular queue or stack data structure in which **each element additionally has a "priority"** associated with it.
- In a priority queue, an element with high priority is served before an element with low priority. In some implementations, if two elements have the same priority, they are served according to the order in which they were enqueued

- A priority queue is an abstract data type that behaves similarly to the normal queue except that each element has some priority, i.e., the element with the highest priority would come first in a priority queue.
- The priority of the elements in a priority queue will determine the order in which elements are removed from the priority queue
- The priority queue is used in many applications, such as CPU scheduling.

- The priority queue supports only comparable elements, which means that the elements are either arranged in an ascending or descending order.

- For example, suppose we have some values like 1, 3, 4, 8, 14, 22 inserted in a priority queue with an ordering imposed on the values is from least to the greatest. Therefore, the 1 number would be having the highest priority while 22 will be having the lowest priority.

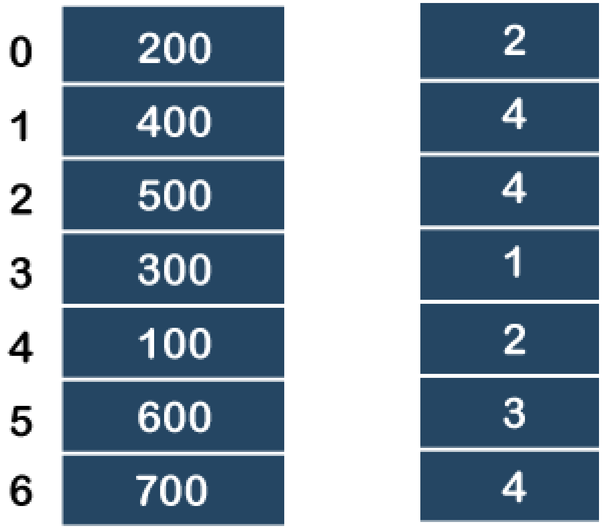
- **Characteristics of a Priority queue**
- A priority queue is an extension of a queue that contains the following characteristics:
  - Every element in a priority queue has some priority associated with it.
  - An element with the higher priority will be deleted before the deletion of the lesser priority.
  - If two elements in a priority queue have the same priority, they will be arranged using the FIFO principle.

The importance of priority queues over regular queues.

- Assume that, in a store, customers queue in a line where service is rendered only at the front of the queue.
- Each customer will spend some time in the queue before getting served. If the units of time spent by four customers in the queue are 4, 30, 2, and 1 respectively, then the average time spent in the queue becomes (4 + 34 + 36 + 37)/4, which is 27.75.
- However, if we associate the priority condition with the data stored in the queue, then we can give more priority to the customer that spends the least time.
- In this situation, the customers will be served in the order of time spent by the customers, that is, in the order of 1, 2, 4, then 30.
- Thus, the average waiting time would be (1 + 3 + 7 + 37)/4, which now equals 12—a better average waiting time.
- Clearly, there is merit to serving the customers by the least time spent. This method of selecting the next item by priority, or some other criterion, is the basis for creating priority queues.

Example

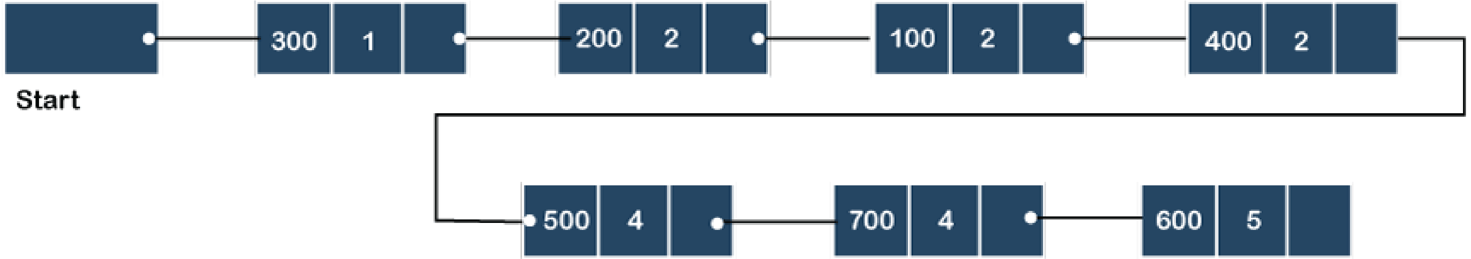
- Representation of priority queue using list



	DATA	PRIORITY
0	200	2
1	400	4
2	500	4
3	300	1
4	100	2
5	600	3
6	700	4

Let's create the priority queue step by step.

0	200	2
1	400	4
2	500	4
3	300	1
4	100	2
5	600	3
6	700	4



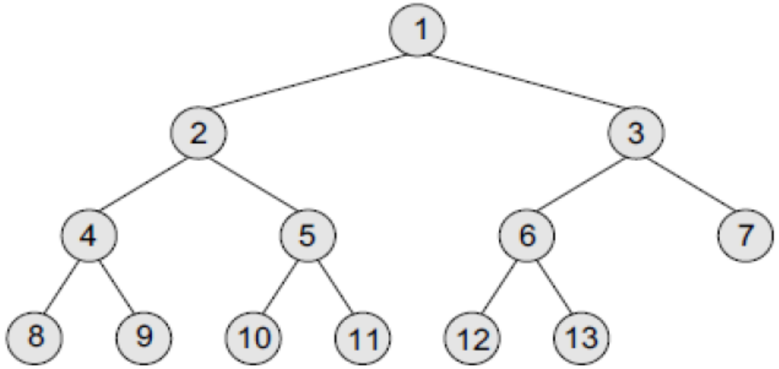
- **In the case of priority queue, lower priority number is considered the higher priority, i.e.,** lower priority number = higher priority.
- **Step 1:** In the list, lower priority number is 1, whose data value is 300, so it will be inserted in the list as shown in the below diagram:
- **Step 2:** After inserting 300, priority number 2 is having a higher priority, and data values associated with this priority are 200 and 100. So, this data will be inserted based on the FIFO principle; therefore 200 will be added first and then 100.
- **Step 3:** After inserting the elements of priority 2, the next higher priority number is 4 and data elements associated with 4 priority numbers are 400, 500, 700. In this case, elements would be inserted based on the FIFO principle; therefore, 400 will be added first, then 500, and then 700.
- **Step 4:** After inserting the elements of priority 4, the next higher priority number is 5, and the value associated with priority 5 is 600 so it will be inserted at the end of the queue.

Implementation of Priority Queue

- Priority queues are mostly implemented using heaps.
- A heap is a data structure that satisfies a **heap property.**
- A heap property states that
  - there must be a **certain relationship between a parent node and its child nodes.**
  - This property must **apply throughout the entire heap.**

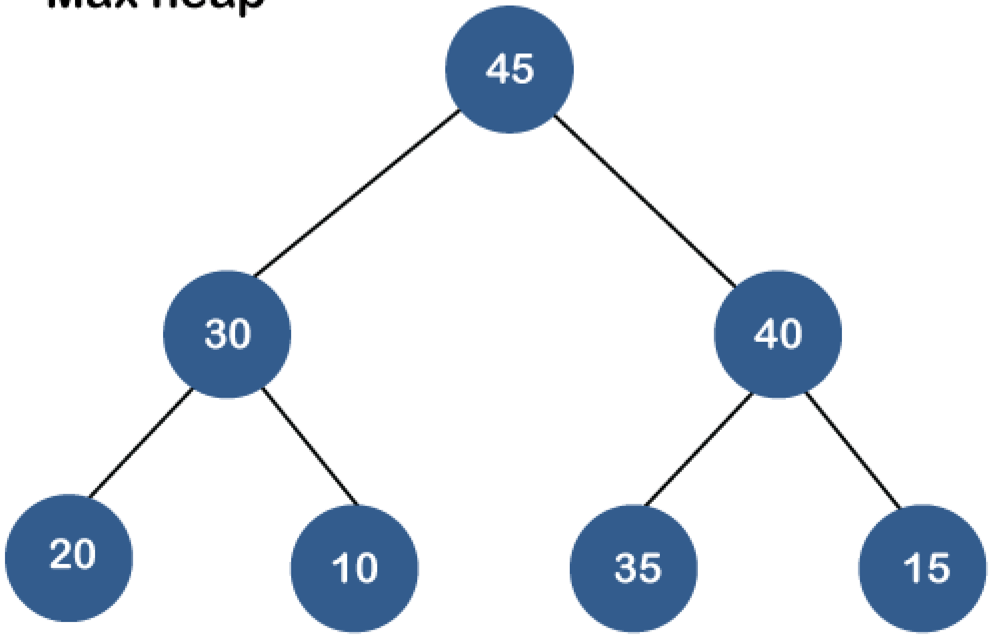
HEAP

complete binary tree

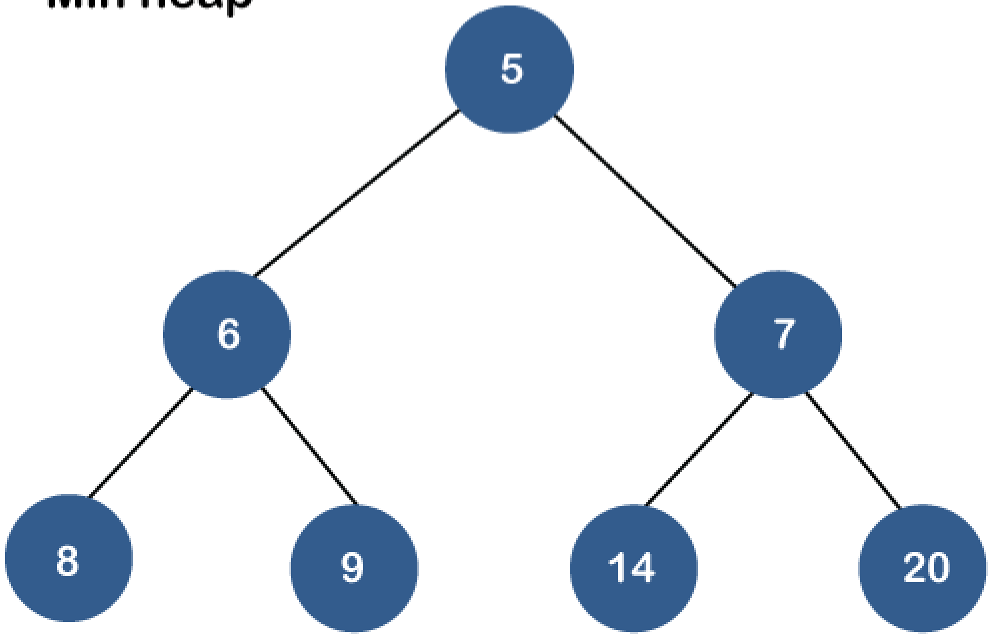


- A heap is a tree-based data structure that forms a complete binary tree.
- Heap property
  - Min heap, the relationship between parent and children is that the value at the parent must always be less than or equal to its children. As a consequence of this, the **lowest element in the heap must be the root node.**
  - Max heap, on the other hand, the parent is greater than or equal to its child or its children. It follows from this that the **largest value makes up the root node**

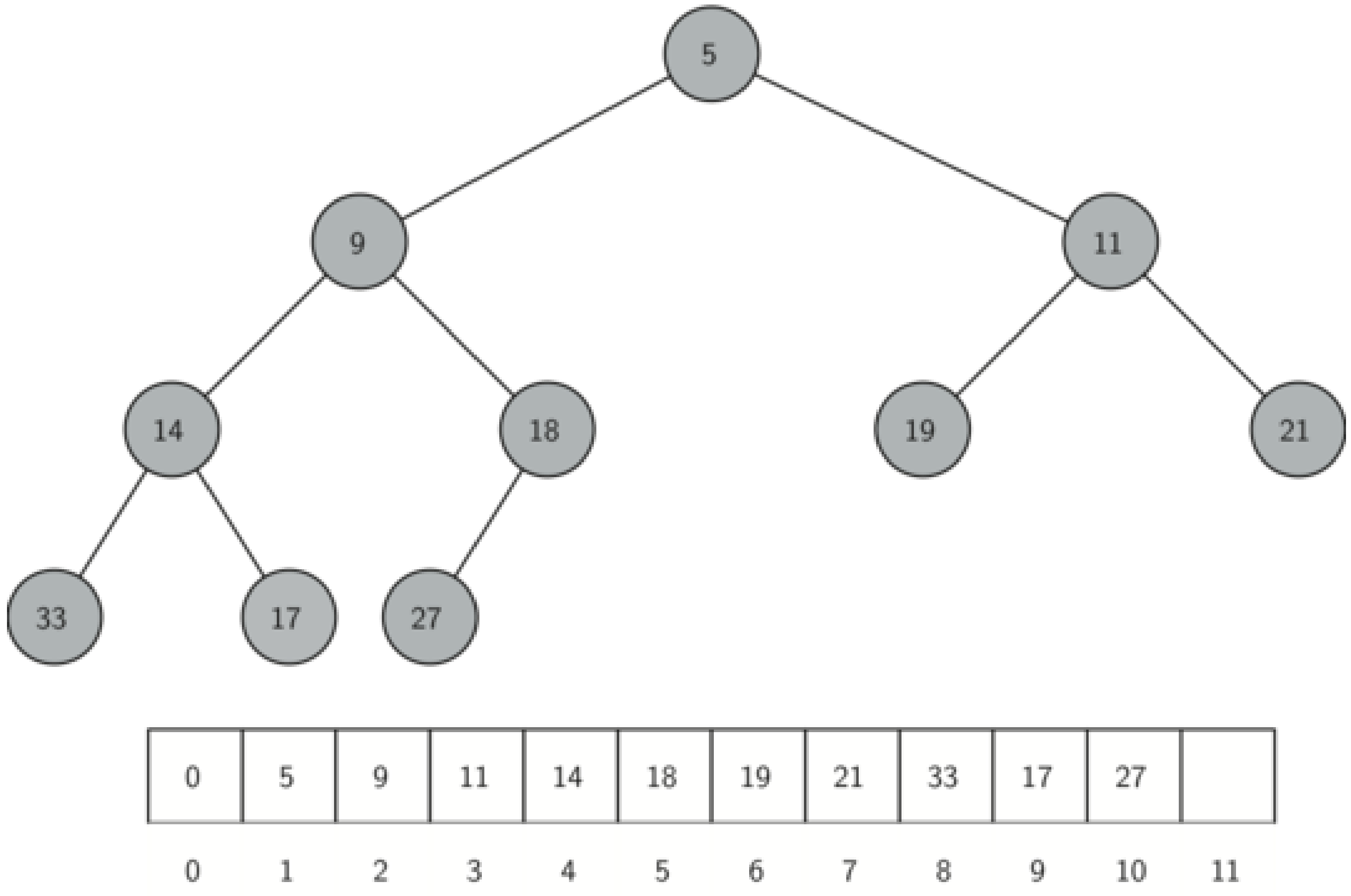
Max heap



Min heap

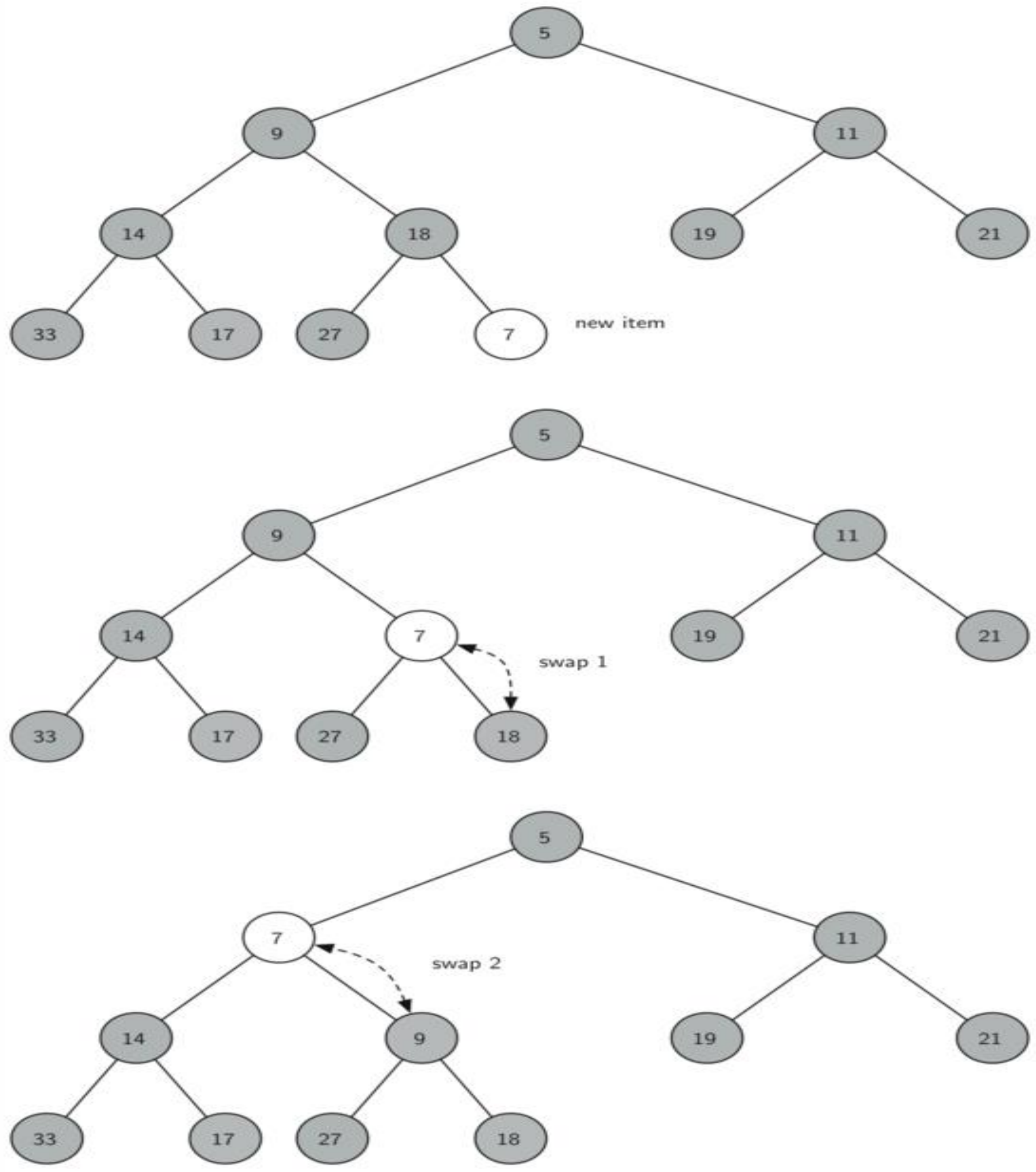


Operations

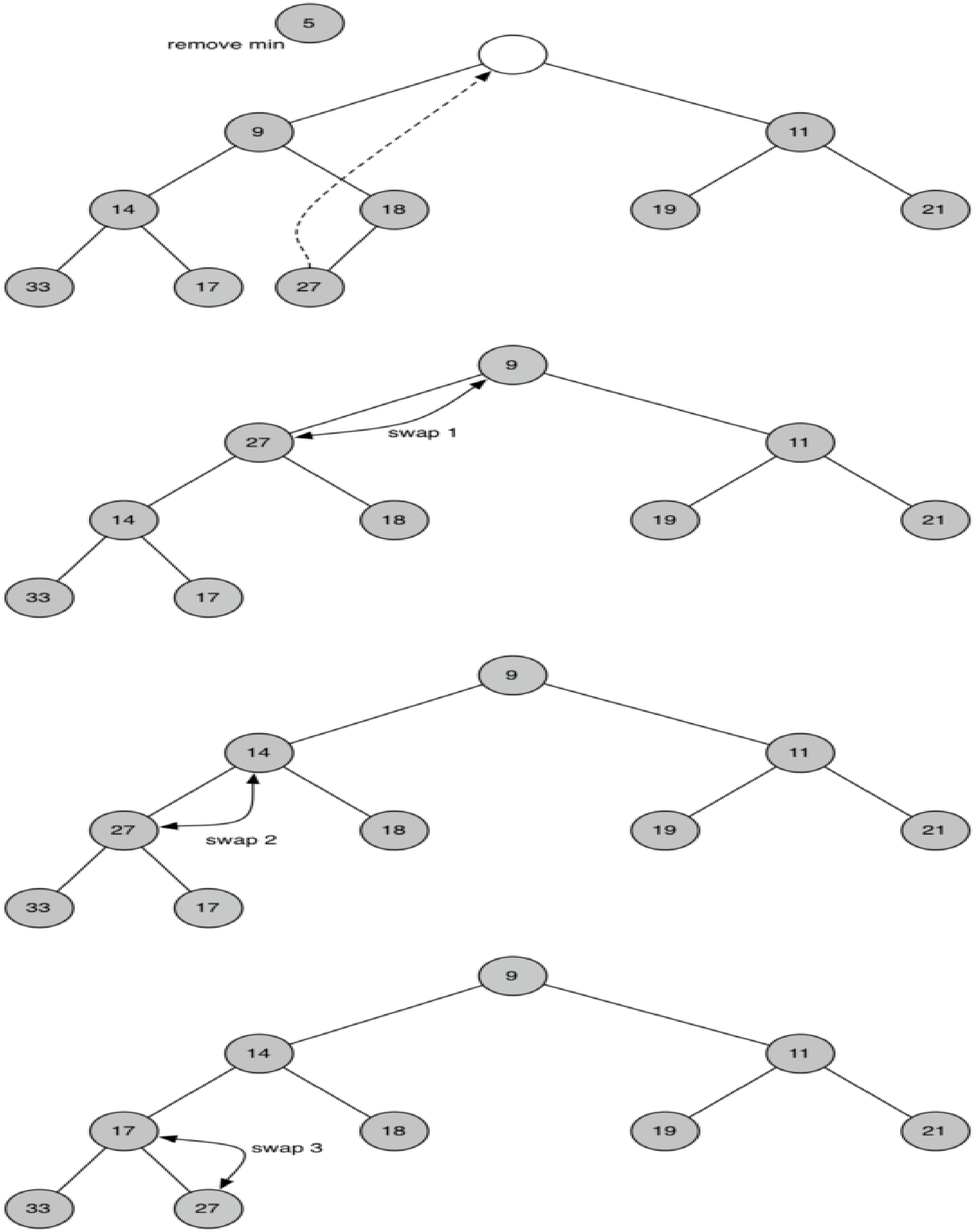


0	5	9	11	14	18	19	21	33	17	27	
0	1	2	3	4	5	6	7	8	9	10	11

Insert new element



Delete Min


5
remove min
9
11
14
18
19
21
33
17
27
9
27
swap 1
11
14
18
19
21
33
17
9
14
11
27
swap 2
18
19
21
33
17
9
14
11
17
18
19
21
swap 3
33
27

THANK YOU!!!!!