Sorting Algorithms

- A sorting algorithm is *comparison-based* if the only operation we can perform on keys is to compare two keys.
- A sorting algorithm is *in place* if only a constant number of elements of the input array are ever stored outside the array.

Running Time of Comparison-Based Sorting Algorithms

	worst-case	average-case	best-case	in place
Insertion Sort				
Merge Sort				
Quick Sort				
Heap Sort				

Heaps

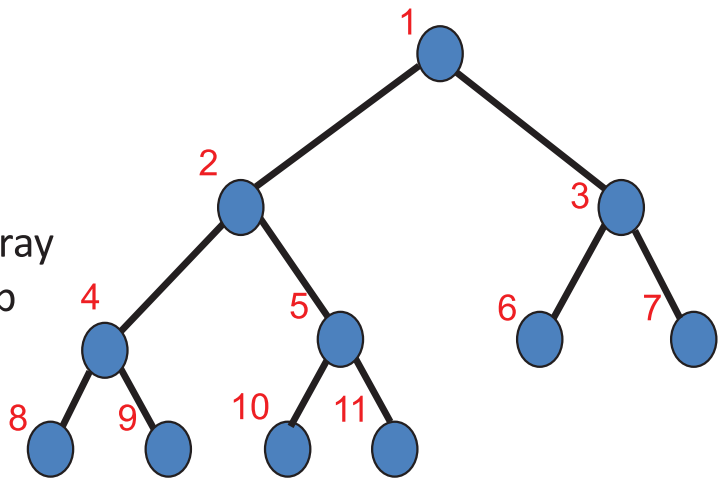
- The ***(binary) heap*** data structure is an array object that can be viewed as a nearly complete binary tree.

Binary Heap

- A heap data structure created using a **binary tree**.
- It can be seen as a binary tree with two additional constraints:
  - The ***shape property***: the tree is an *almost complete binary tree*; that is, all levels of the tree, except possibly the last one (deepest) are fully filled.
  - The ***heap property***: each node is greater than or equal to each of its children according to some comparison predicate which is fixed for the entire data structure.

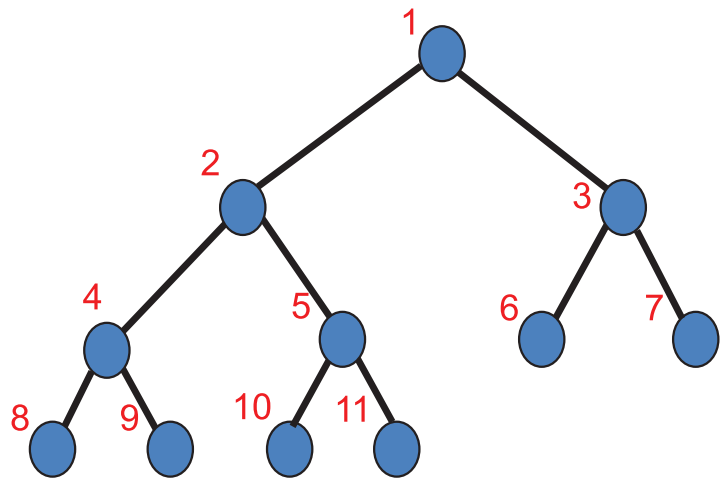
Binary tree: an array implementation

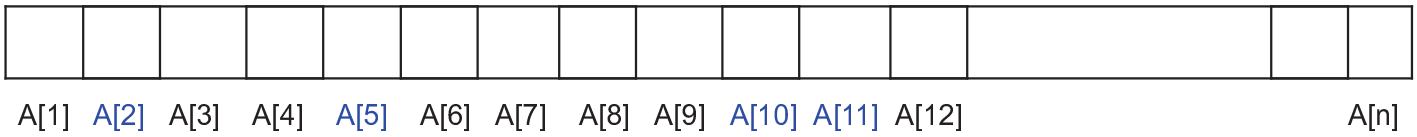
- root is A[1]
- for element A[i]
  - left child is in position A[2i]
  - right child is in position A[2i + 1]
  - parent is in A[$\lfloor i/2 \rfloor$]
- Example: i = 5
- heap as an array implementation
  - store heap as a logical binary tree in array
  - *heapsize* is number of elements in heap
  - *length* is number of elements in array



Binary tree:
an array implementation

- root is A[1]
- for element A[i]
  - left child is in position A[2i]
  - right child is in position A[2i + 1]
  - parent is in A[⌊i/2⌋]
- Example: i = 5
- Parent(i) return (⌊i/2⌋)
- LeftChild(i) return (2i)
- RightChild(i) return(2i+1)





Max-heap property

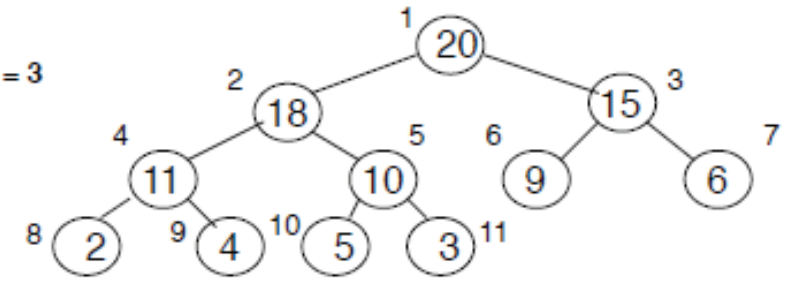
In the array representation of a max-heap, the root of the tree is in A[*1*], and given the index *i* of a node,

Parent(i)	LeftChild(i)	RightChild(i)
return (⌊i/2⌋)	return 2i	return (2i + 1)

Max-heap property: A[Parent(i)] ≥ A[i]

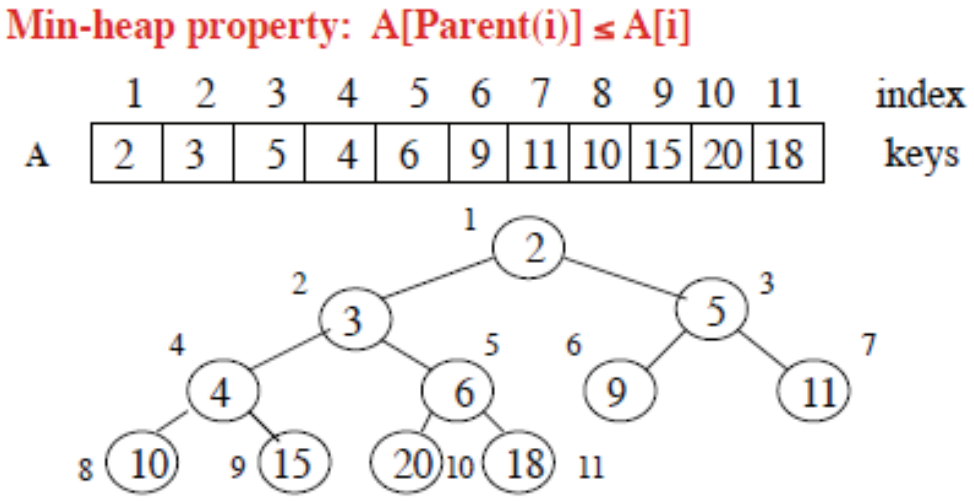
	1	2	3	4	5	6	7	8	9	10	11	
A	20	18	15	11	10	9	6	2	4	5	3	keys
												index

n = 11
height = 3



Min-heap property

Min-heaps are commonly used for priority queues in event-driven simulators.



We will use Max-heap for heapsort

Heap Sort

***Input*:** An *n*-element array A (unsorted).

***Output*:** An *n*-element array A in sorted order, smallest to largest.

HeapSort(A)

1. **Build-Max-Heap(A**) /* put all elements in heap */
2. **for** i $\leftarrow$ length(A) **downto** 2
3. **do** swap A[1] $\leftrightarrow$ A[i]/*puts max in *i*-th array position*/
4. heap-size[A] $\leftarrow$ heap-size[A] - 1
5. **Max-Heapify(A,1)** /* restore heap property */