BITS Pilani
Pilani Campus

Data Structures & Algorithms Design- SS ZG519
Lecture - 10

Dr. Padma Murali

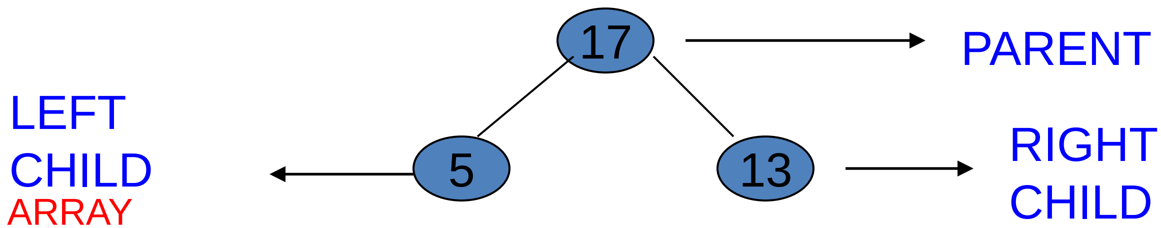
Lecture 10 Topics

- Heap Sort
-

HEAPSORT

Heap

- The heap data structure is an array object which can be viewed as a nearly complete binary tree.
- the two attributes of heap are length[A] and heap-size[A]



Heap as an ARRAY

- given an index i of a node,

  Parent(i)
    return $\lfloor(i/2)\rfloor$
  Left(i)
    return 2i
  Right(i)
    return 2i + 1

HEAPSORT

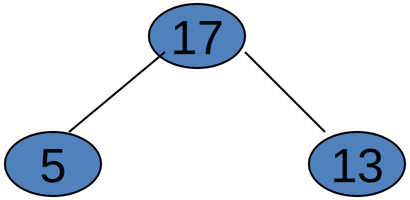
Max-heap

- max-heap property is A[parent(i)] >= A[i]

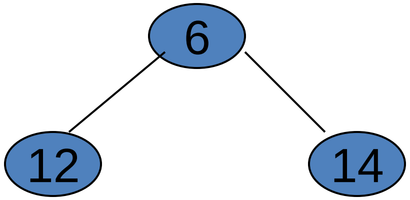
Min-heap

- min-heap property is A[parent(i)] <= A[i]

max-heap



min-heap



HEAPSORT

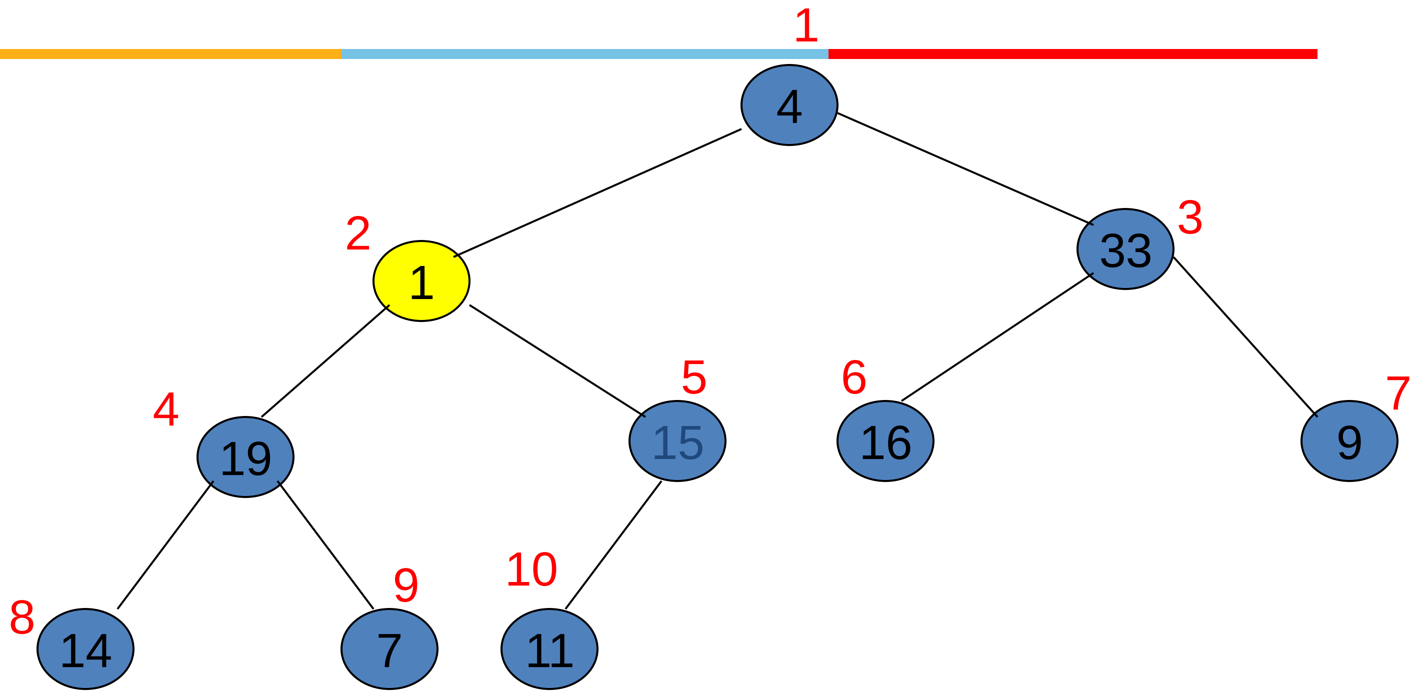
Basic Procedures used in Heapsort algorithm

1. MAX-HEAPIFY – maintains the heap property.
2. BUILD-MAX-HEAP – produces a max-heap from an unordered input array.
3. HEAPSORT – sorts an array in place

HEAPSORT

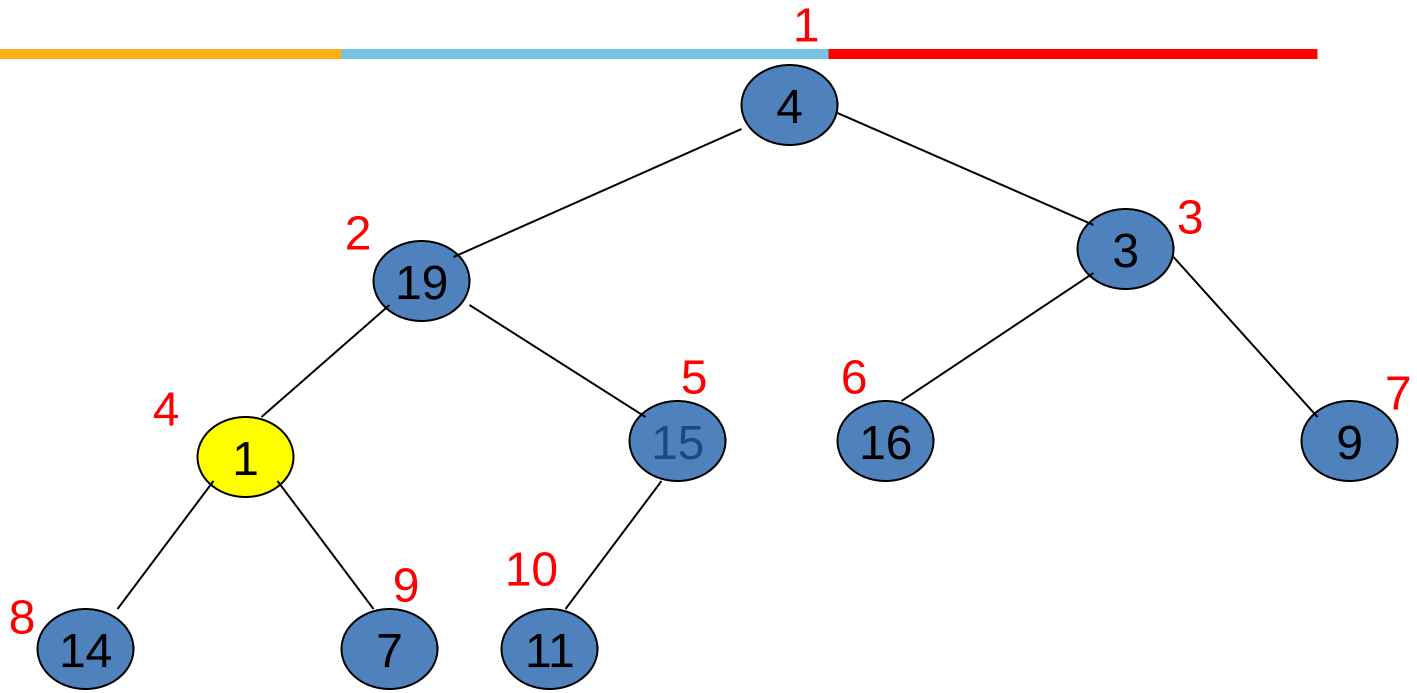
```
MAX-HEAPIFY (A,i)
        l  ←  left(i)
        r  ←  right(i)
    if l <= heap-size[A] and A[l] > A[i]
          then largest ← l
          else largest ← i
    if r <= heap-size[A] and A[r] > A[largest]
          then largest ← r
    if largest ≠ i
          then exchange A[i] <-> A[largest]
                    MAX-HEAPIFY (A,largest)
```

MAX-HEAPIFY



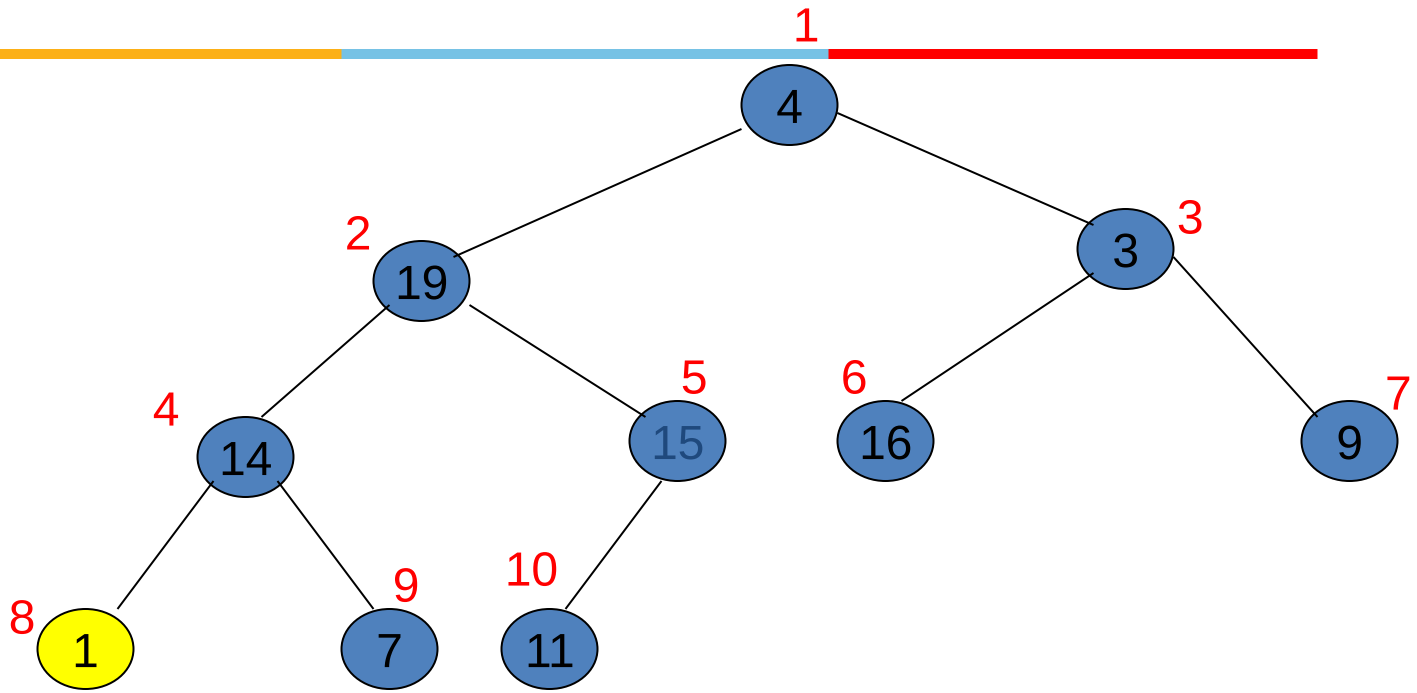
MAX-HEAPIFY(A,2)

MAX-HEAPIFY



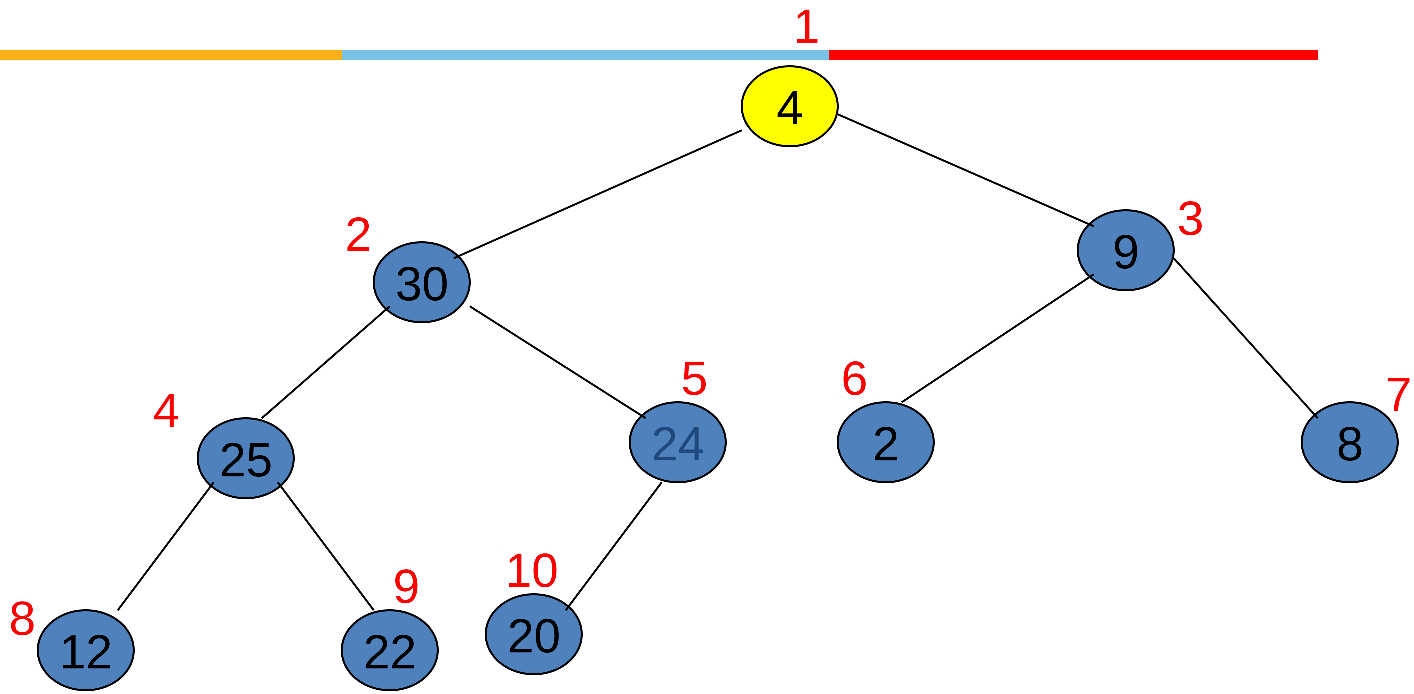
MAX-HEAPIFY(A,4)

MAX-HEAPIFY



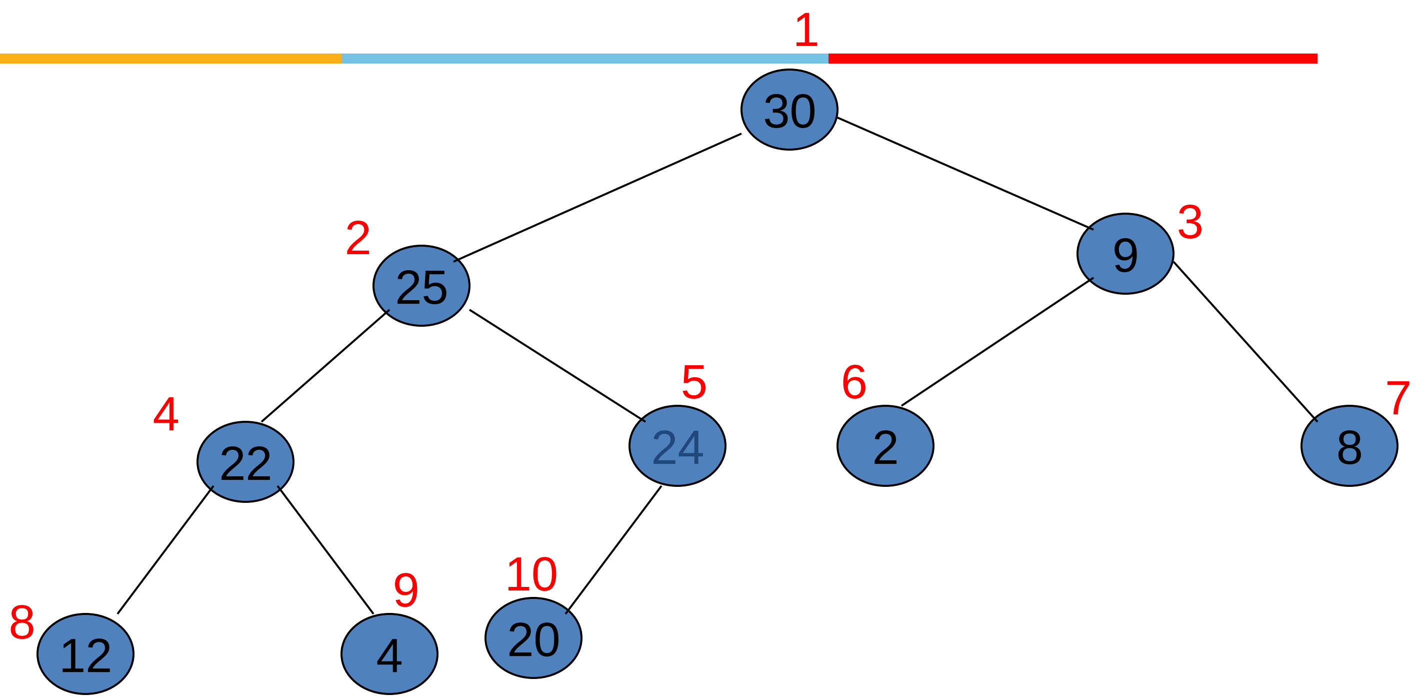
MAX-HEAPIFY(A,8)

MAX-HEAPIFY



MAX-HEAPIFY(A,1)

MAX-HEAPIFY



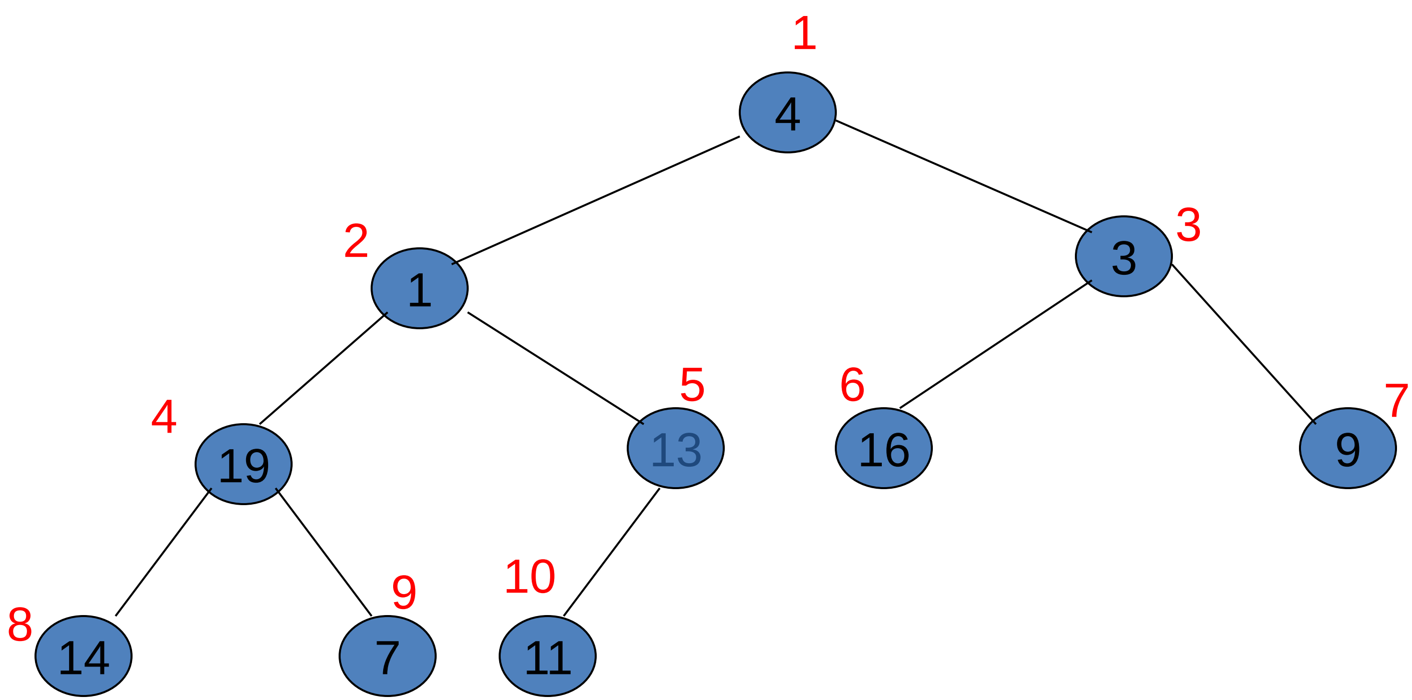
MAX-HEAPIFY(A,1)

Building a HEAP

```
BUILD-MAX-HEAP (A)
    Heap-size[A] ← length[A]
    for i ← length[A]/2 downto 1
        do MAX-HEAPIFY (A,i)
```

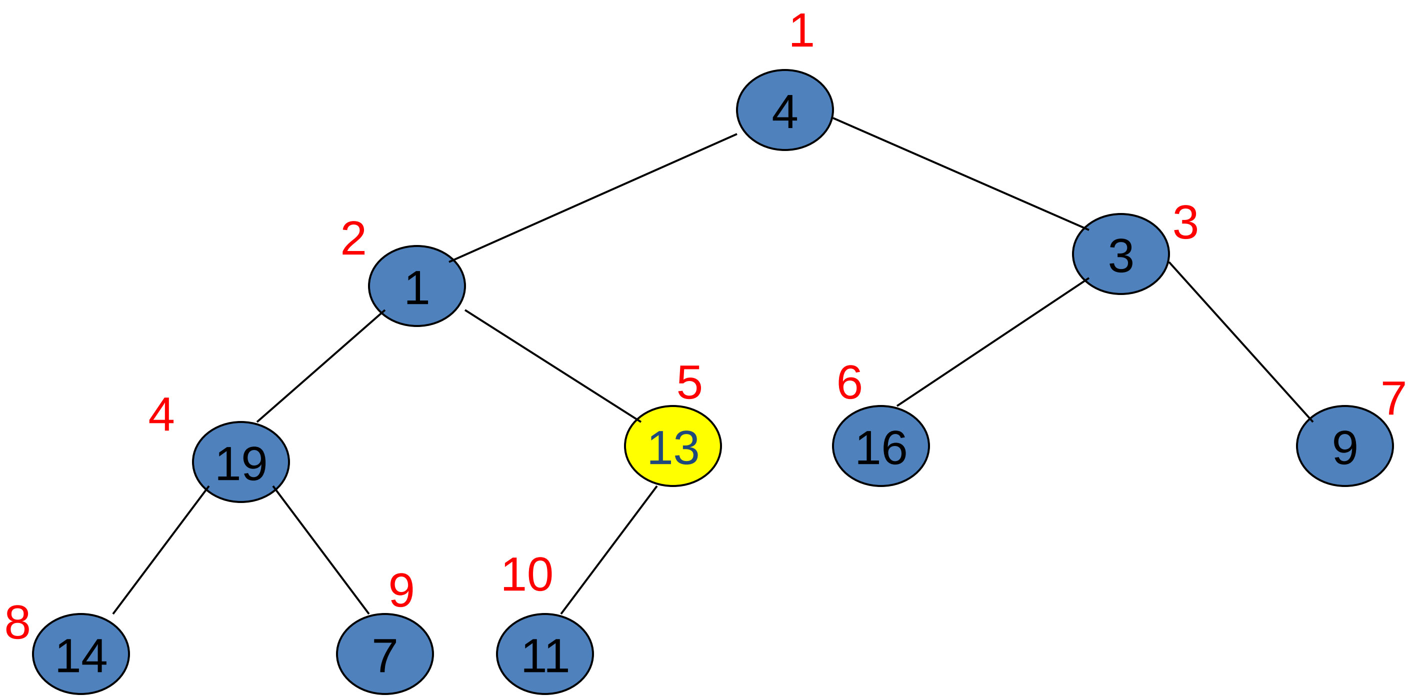
Building a HEAP

4	1	3	19	13	16	9	14	7	11



Building a HEAP

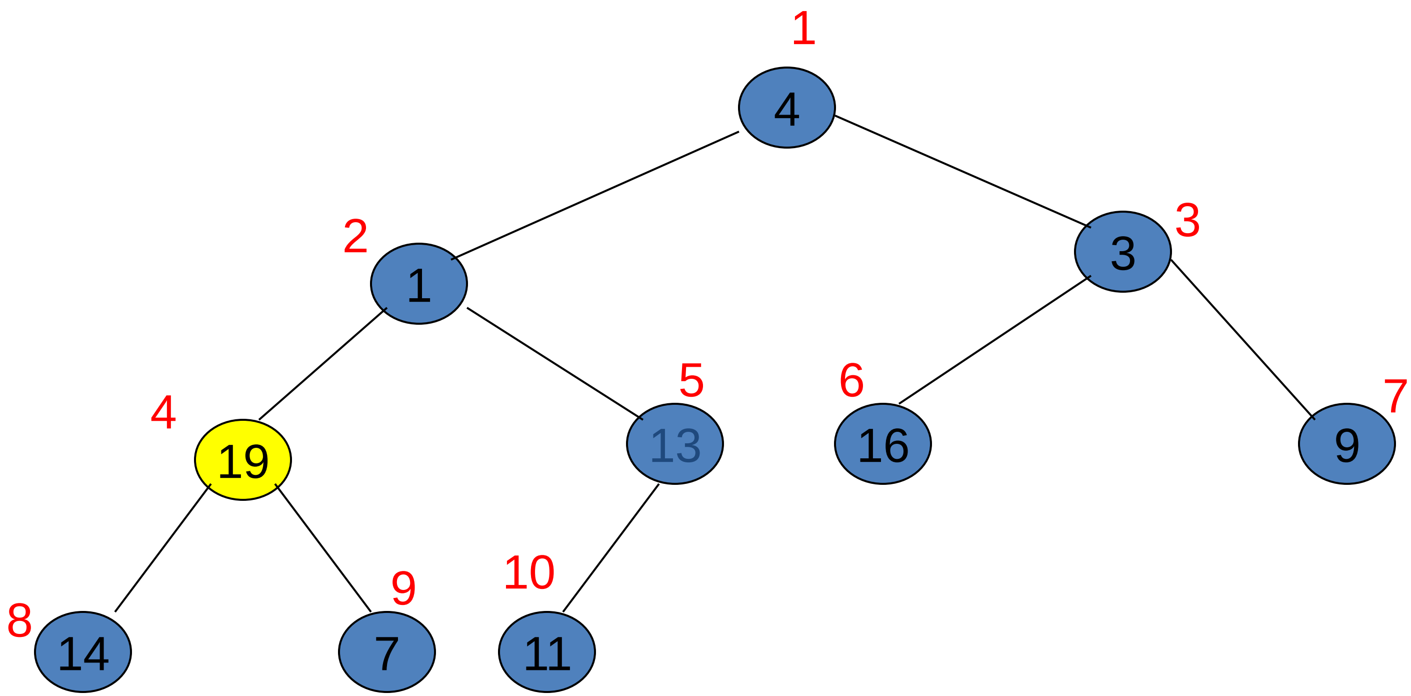
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,5)

Building a HEAP

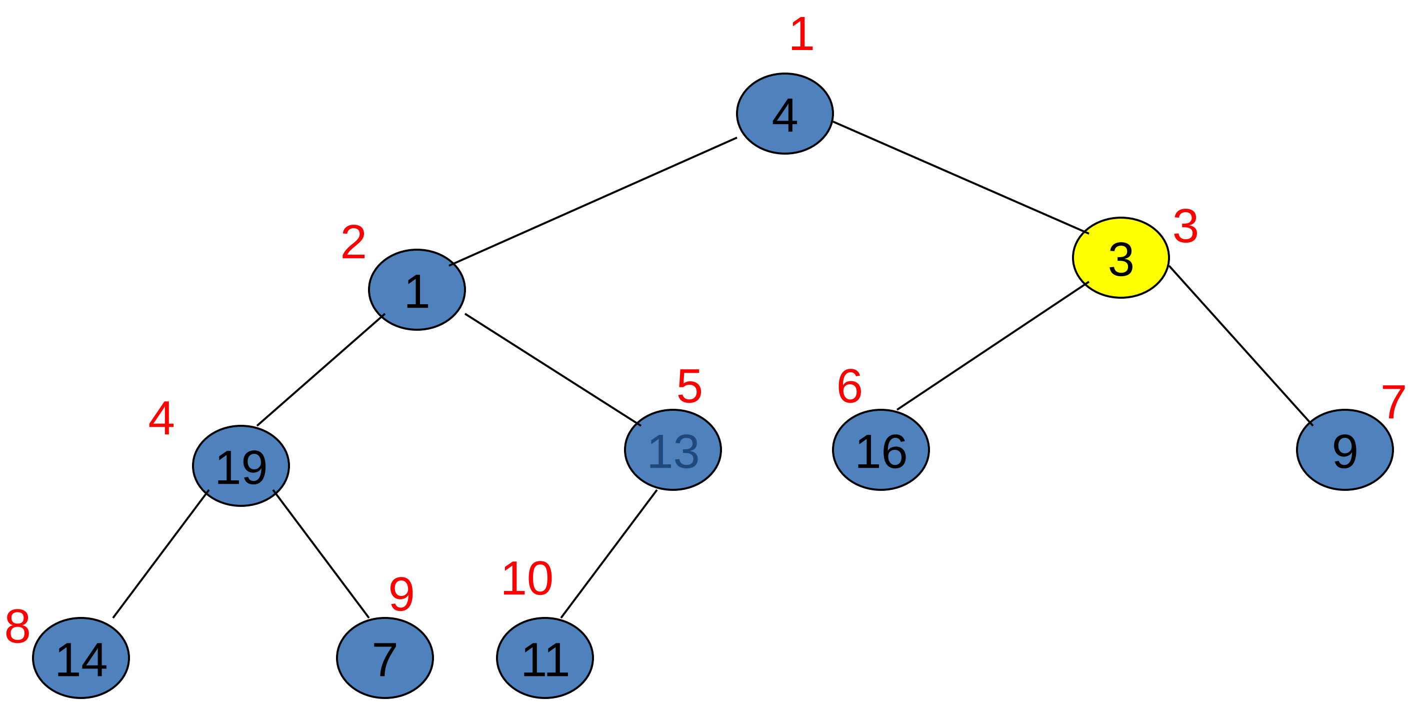
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,4)

Building a HEAP

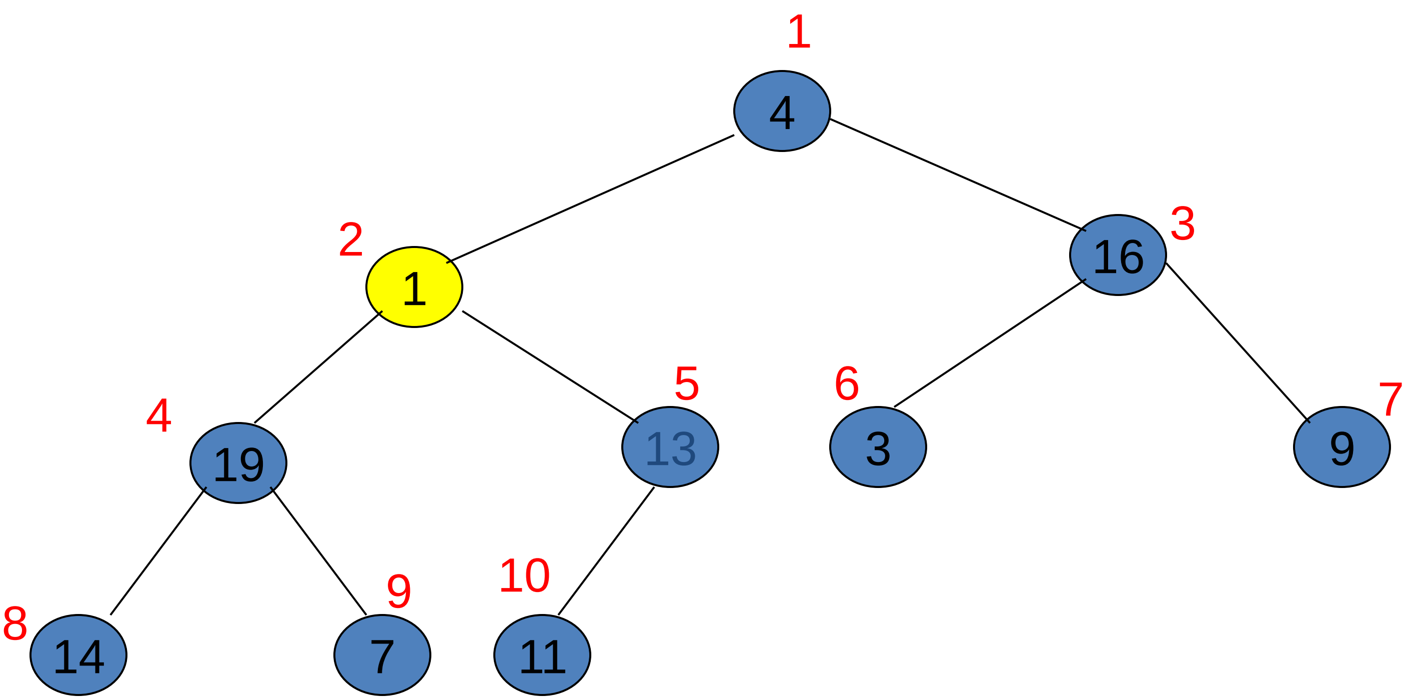
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,3)

Building a HEAP

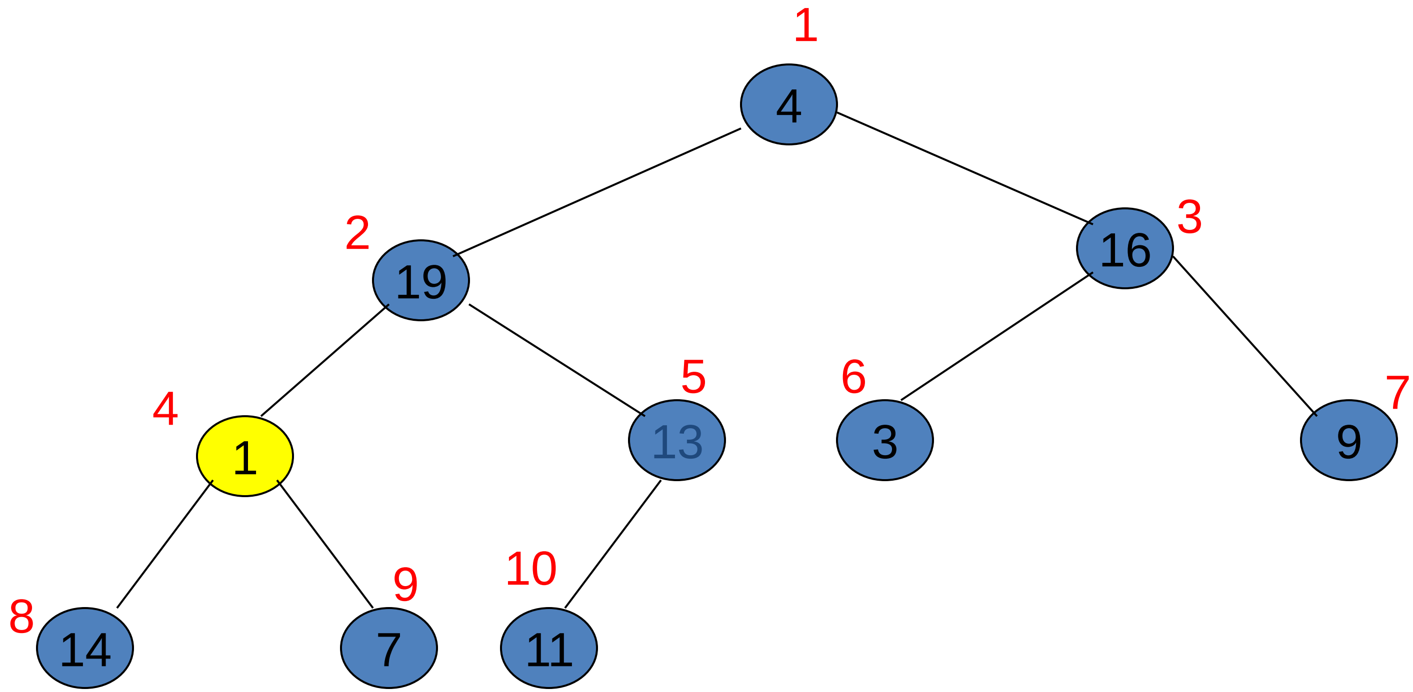
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,2)

Building a HEAP

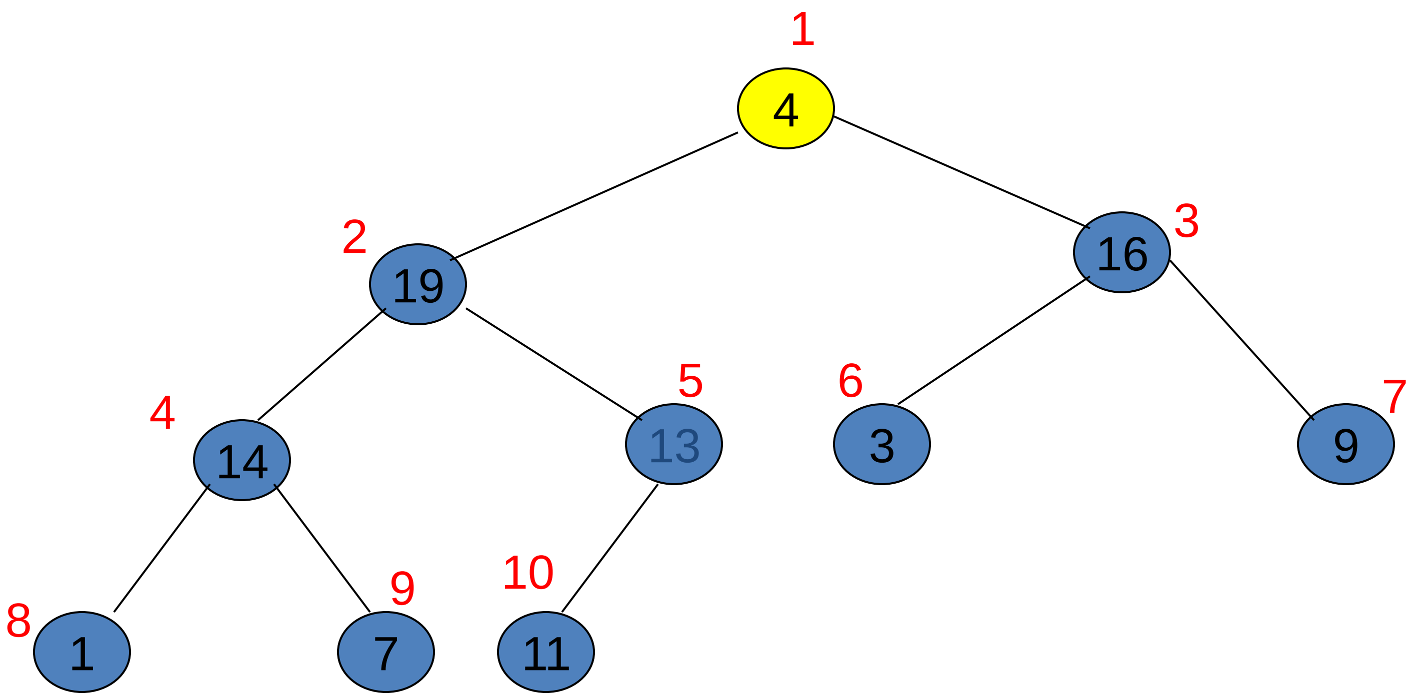
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,4)

Building a HEAP

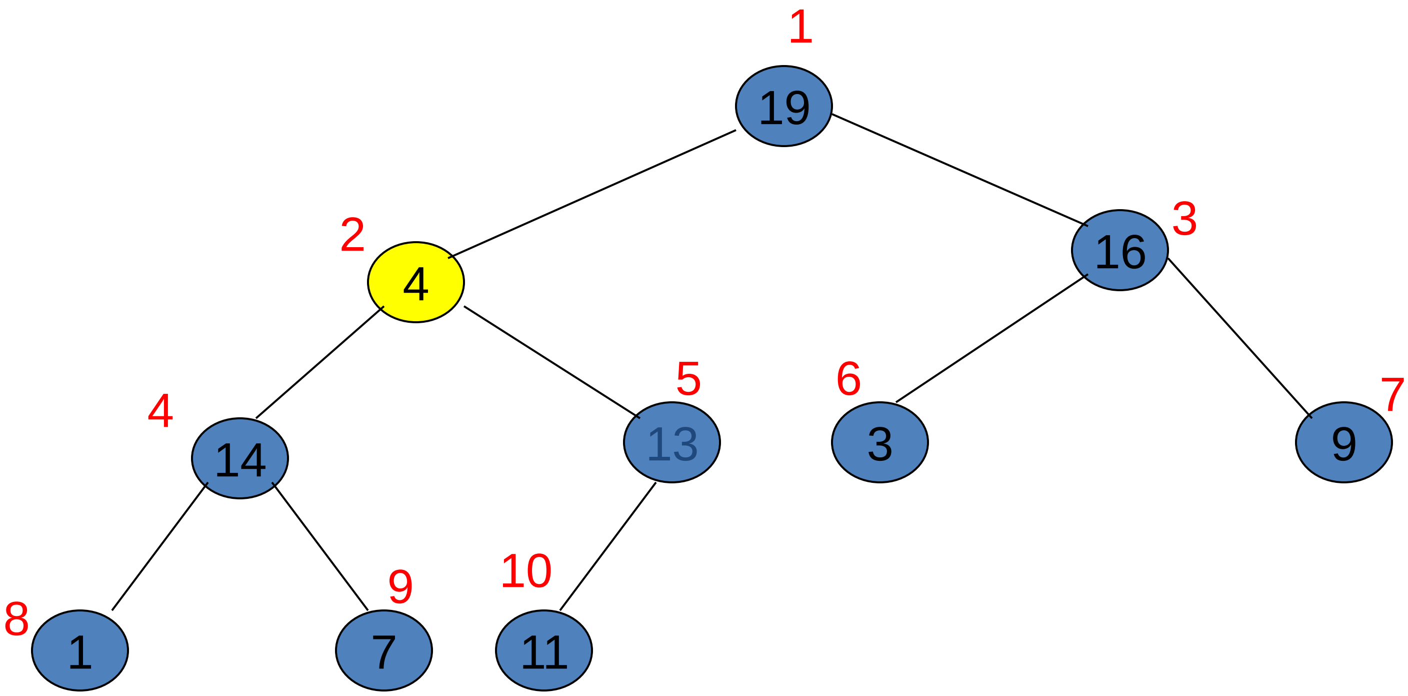
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,1)

Building a HEAP

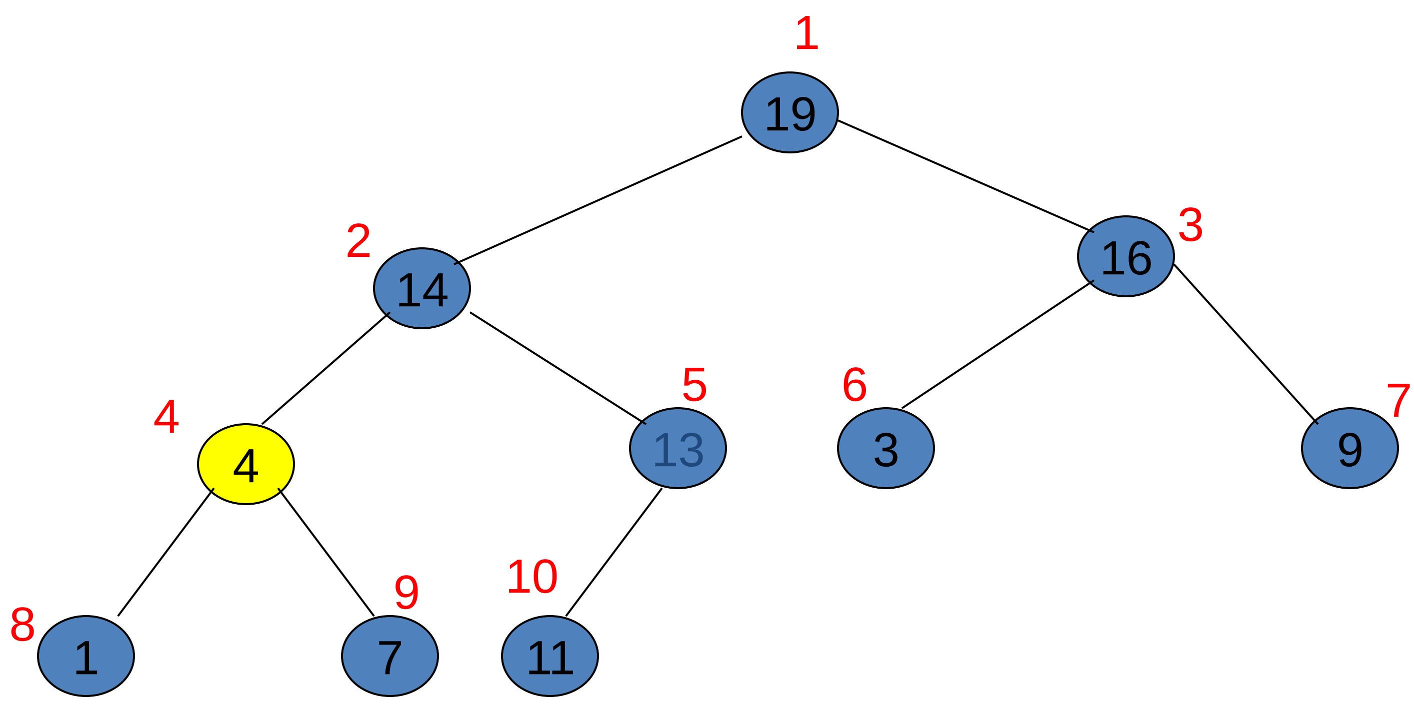
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,2)

Building a HEAP

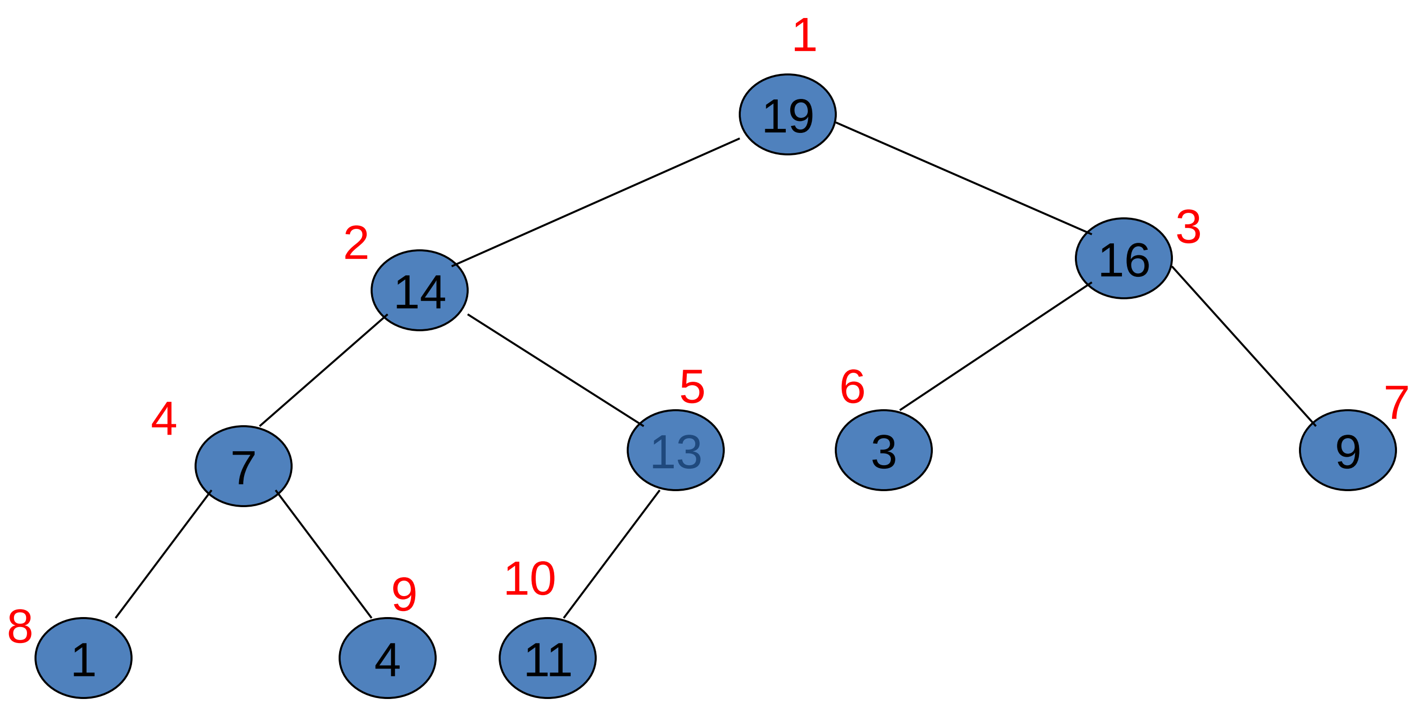
4	1	3	19	13	16	9	14	7	11



MAX-HEAPIFY(A,4)

Building a HEAP

4	1	3	19	13	16	9	14	7	11



DONE

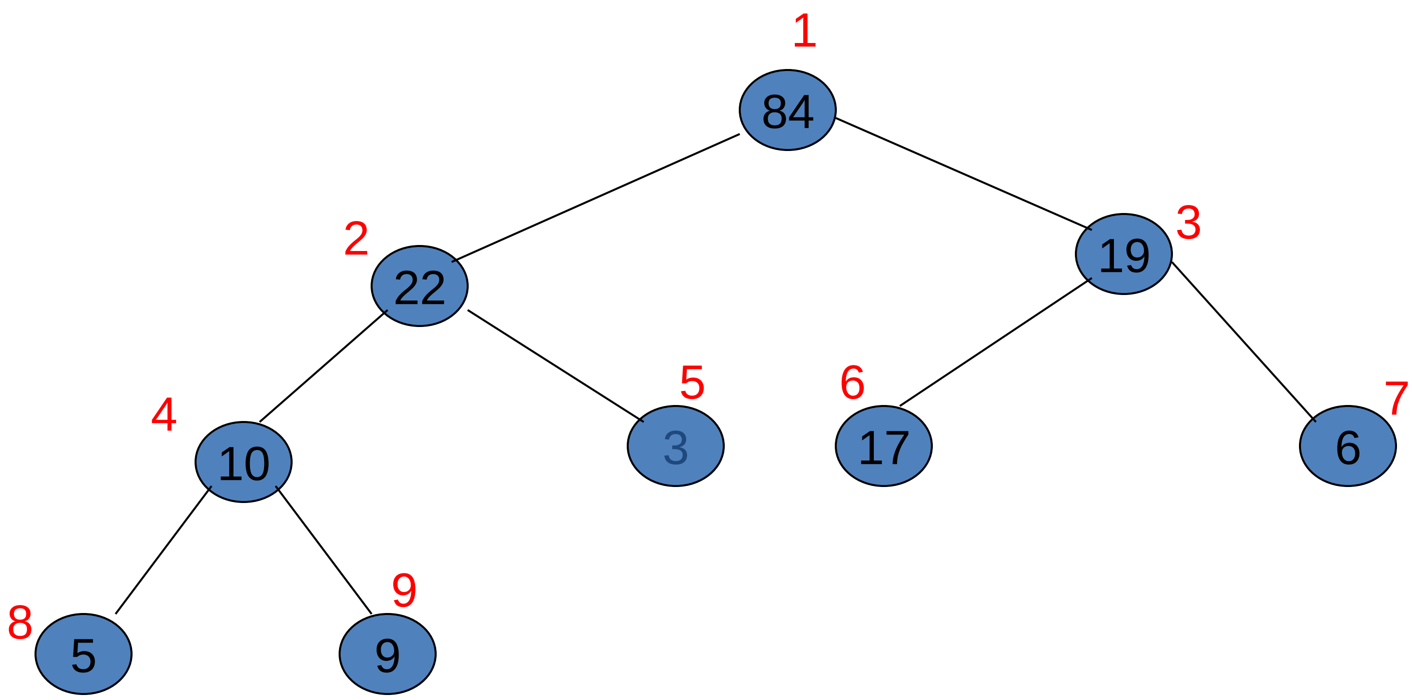
Building a HEAP

5	3	17	10	84	19	6	22	9

BUILD a heap on the above array

Building a HEAP

5	3	17	10	84	19	6	22	9

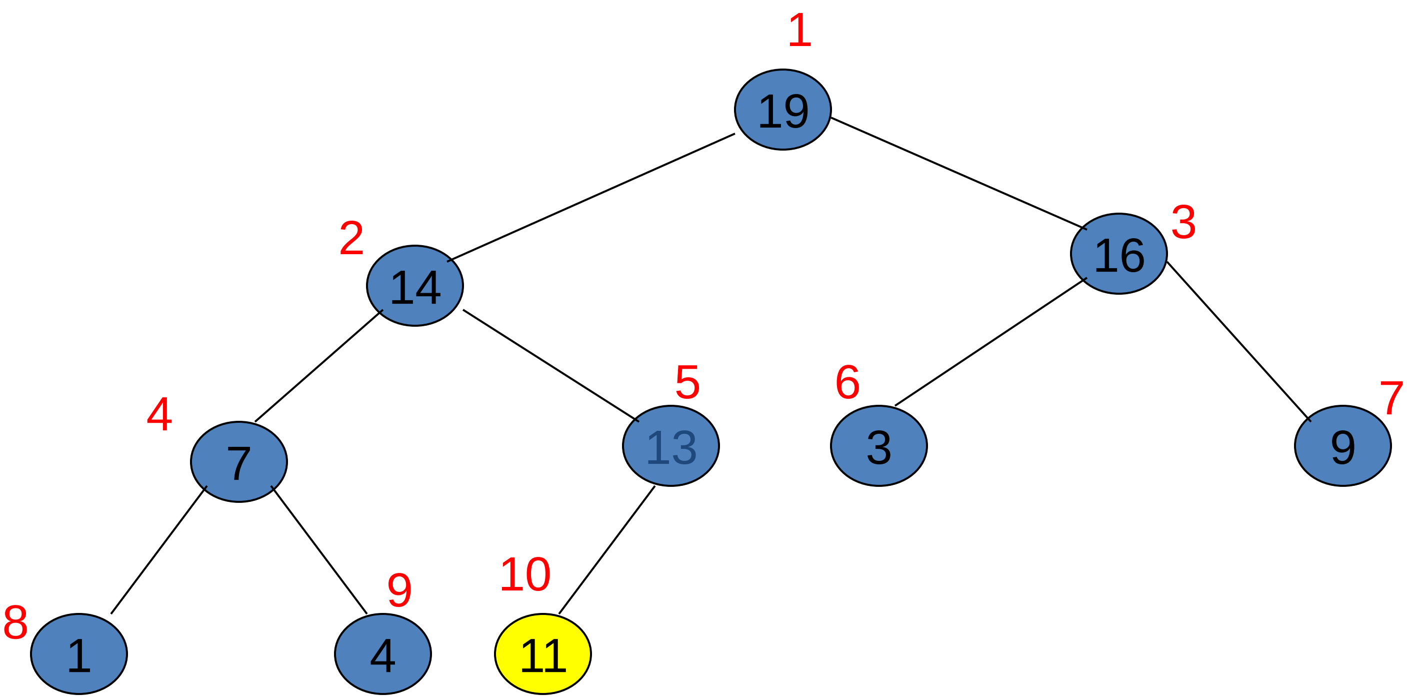


HeapSort algorithm

```
HEAPSORT(A)
    BUILD-MAX-HEAP(A)
    for i ← length[A] downto 2
        do exchange A[1] <-> A[i]
            heap-size [A] ← heap-size[A] – 1
            MAX-HEAPIFY(A,1)
```

HeapSort algorithm

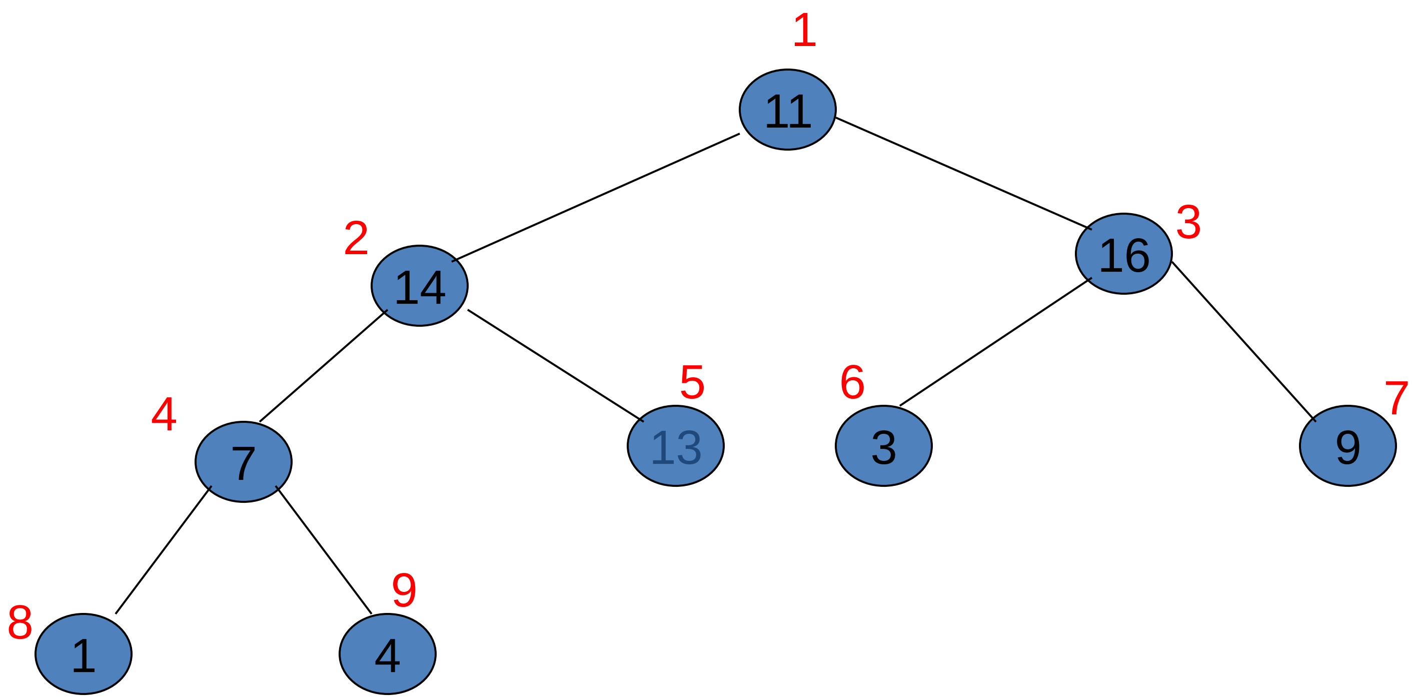
19	14	16	7	13	3	9	1	4	11



A

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

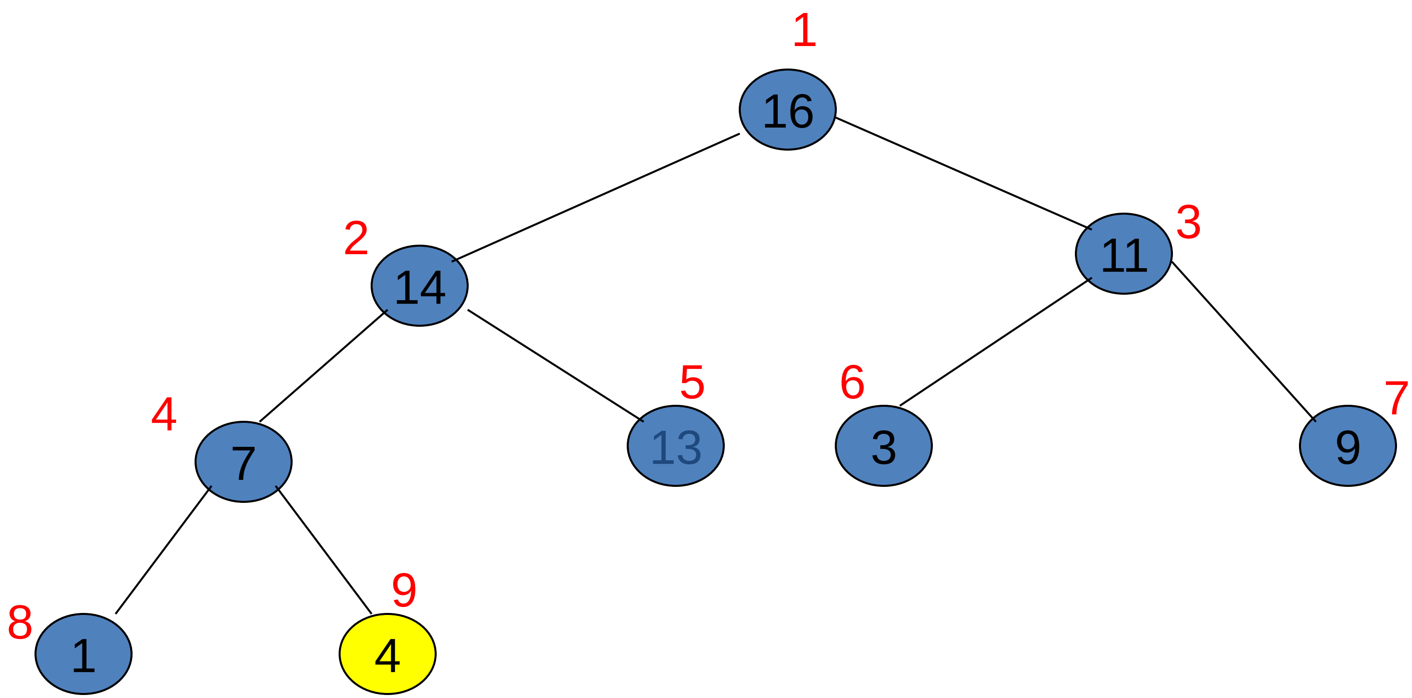


A

									19

HeapSort algorithm

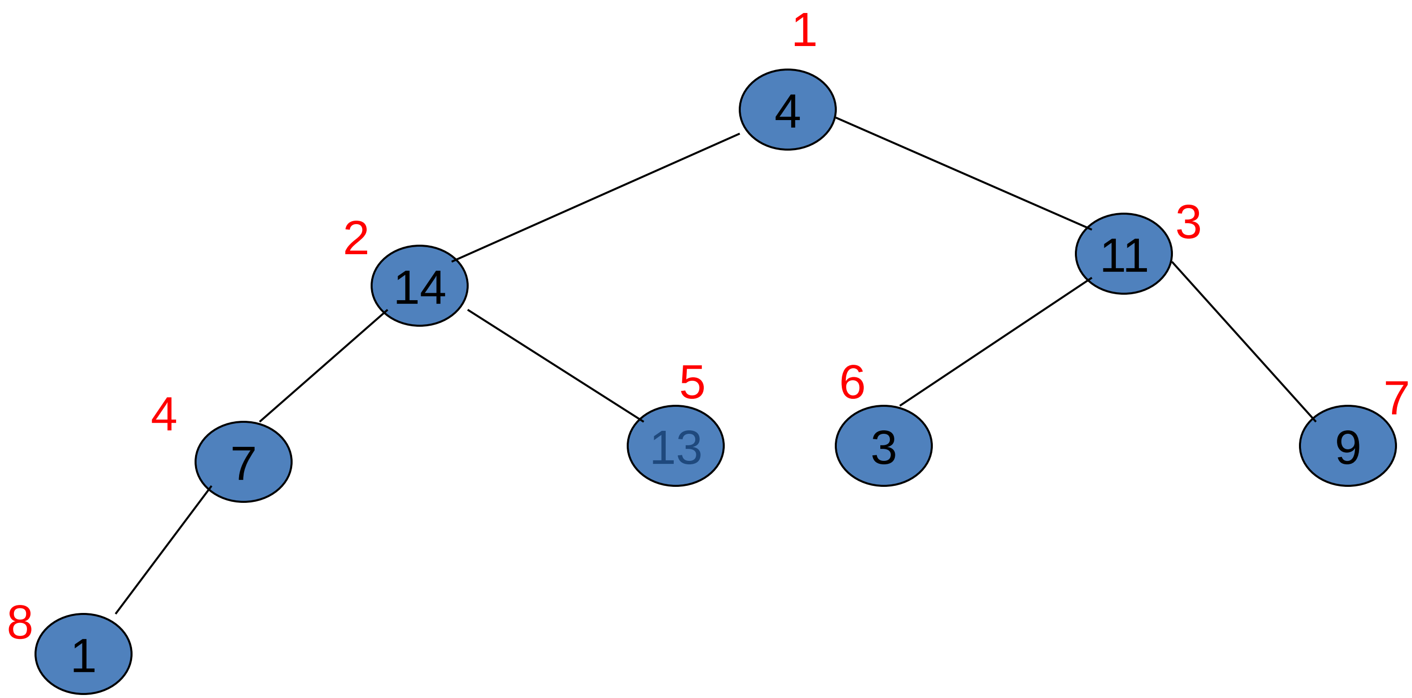
19	14	16	7	13	3	9	1	4	11



A										
										19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

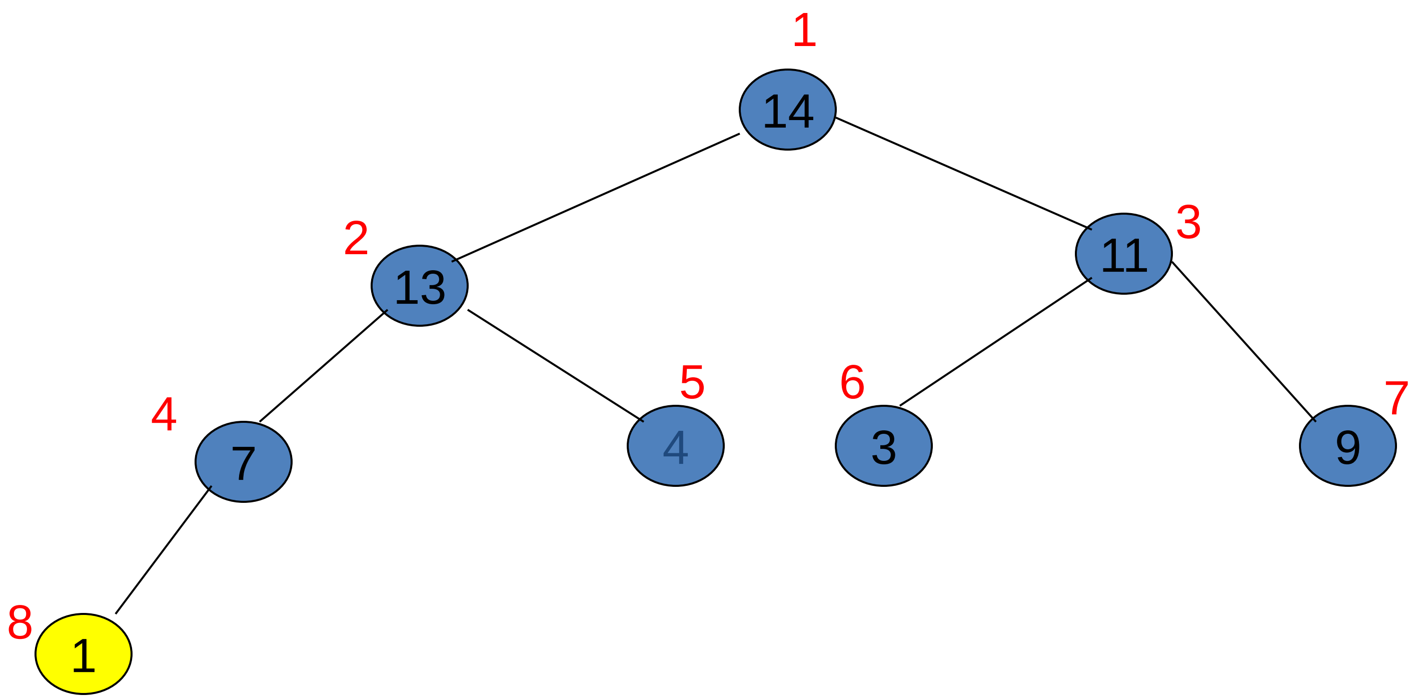


A

								16	19

HeapSort algorithm

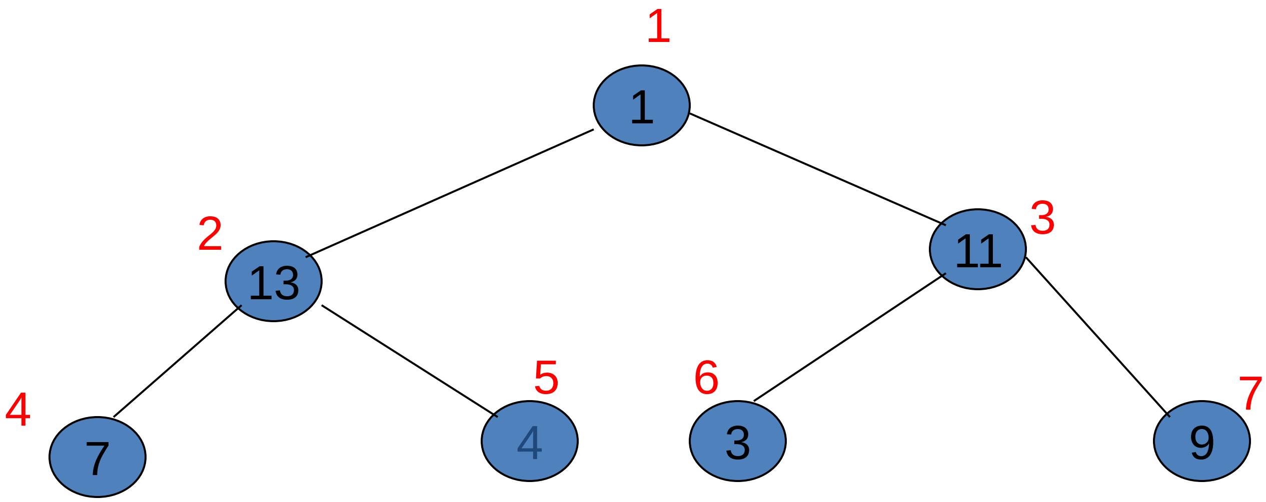
19	14	16	7	13	3	9	1	4	11



A

								16	19

19	14	16	7	13	3	9	1	4	11

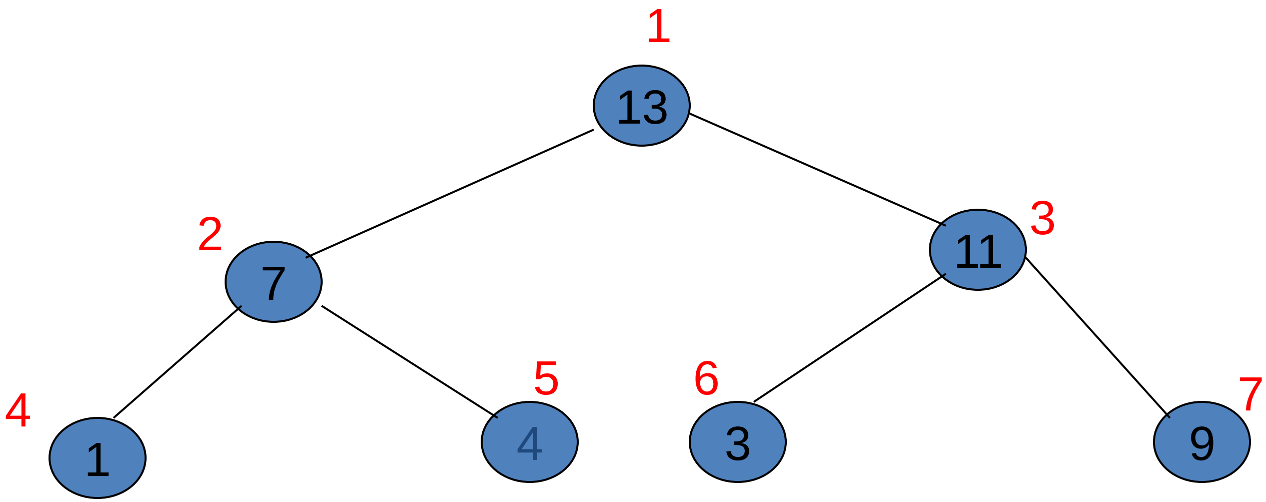


A

							14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

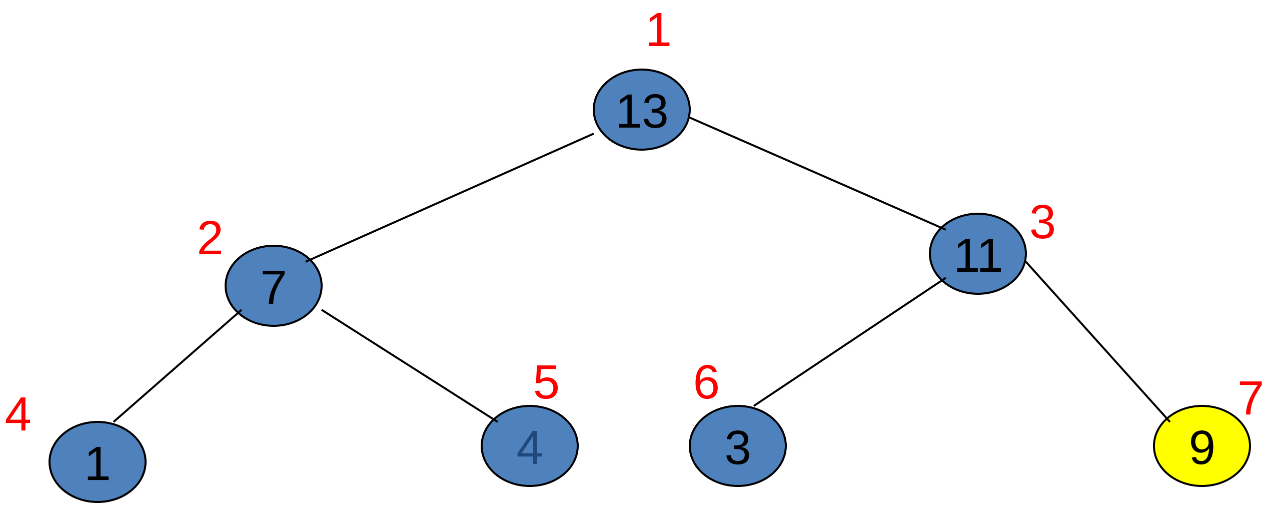


A

							14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

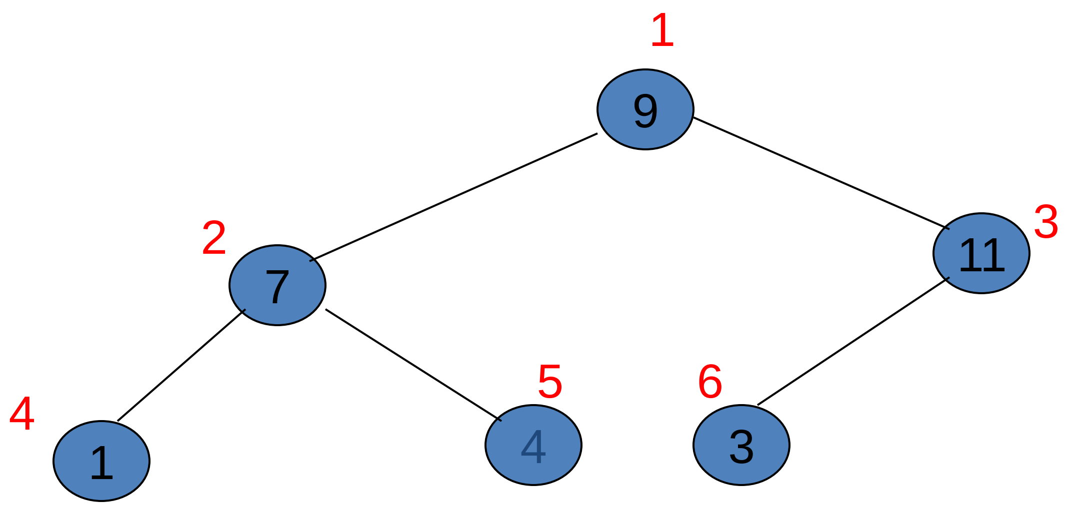


A

							14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

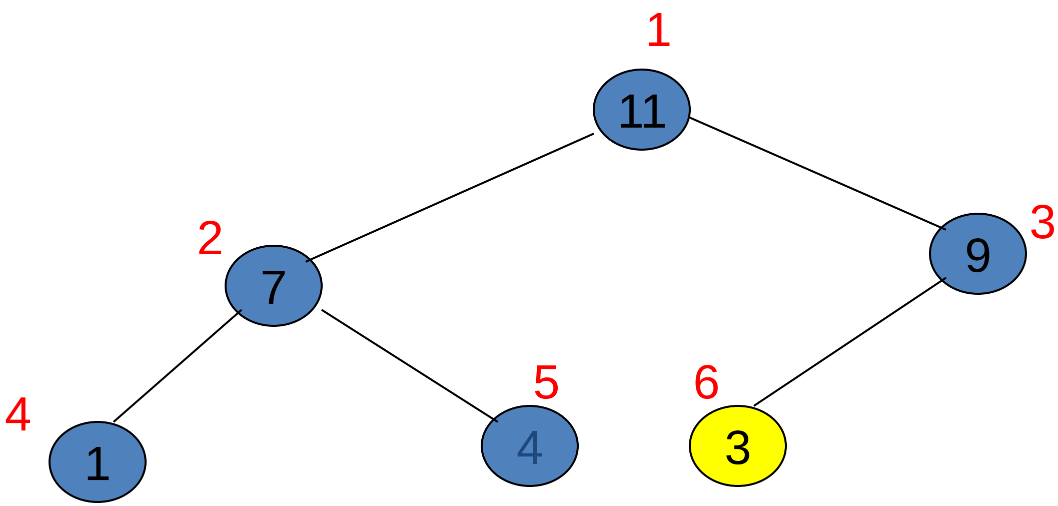


A

						13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

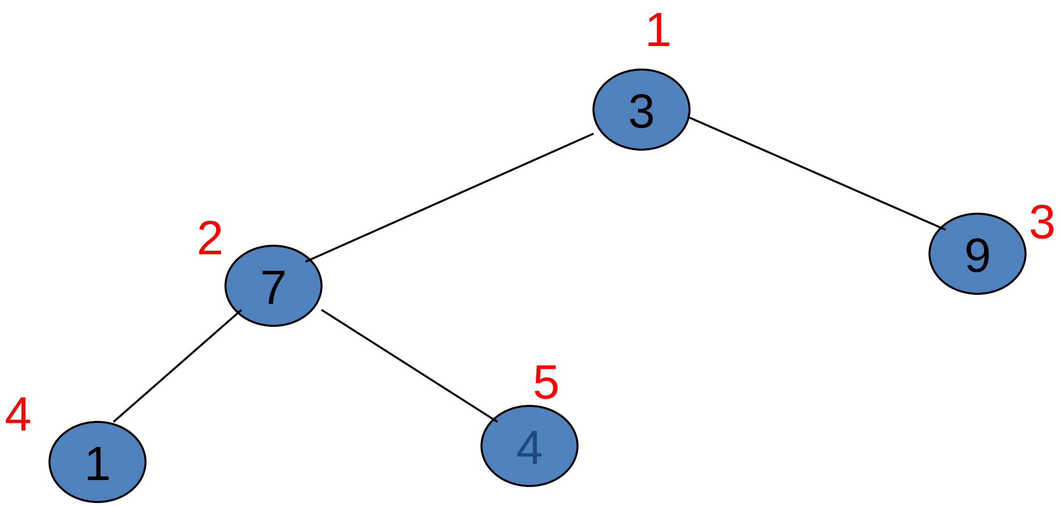


A

						13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

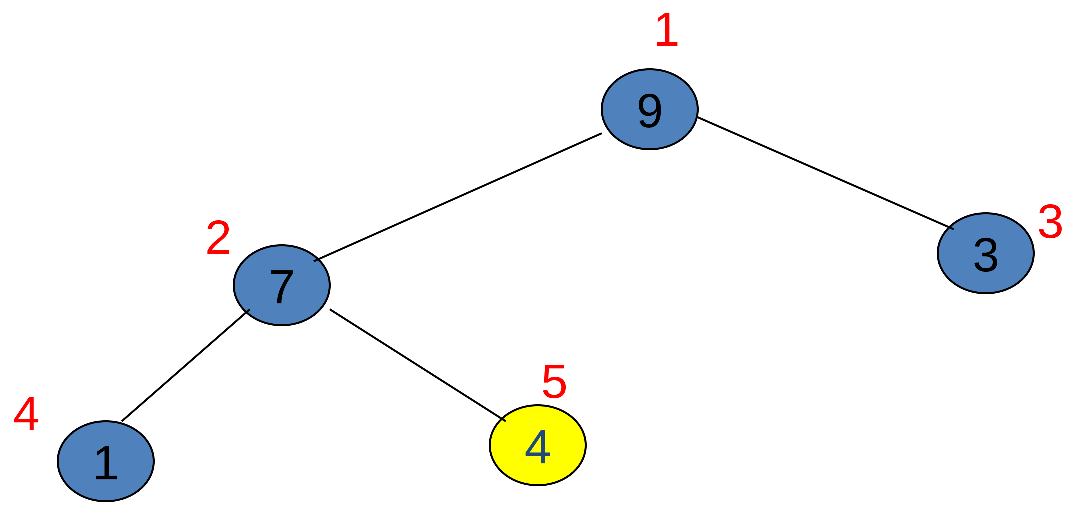


A

					11	13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

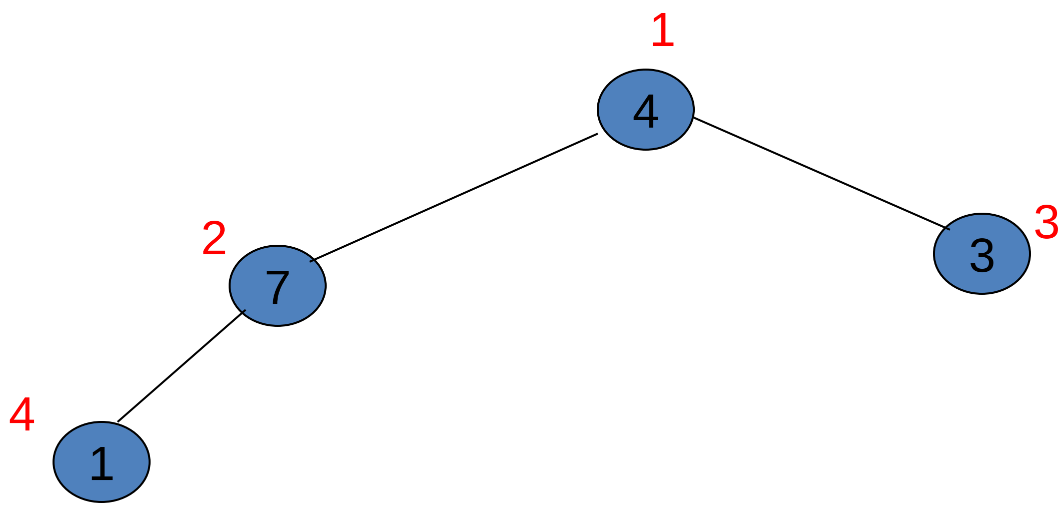


A

					11	13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

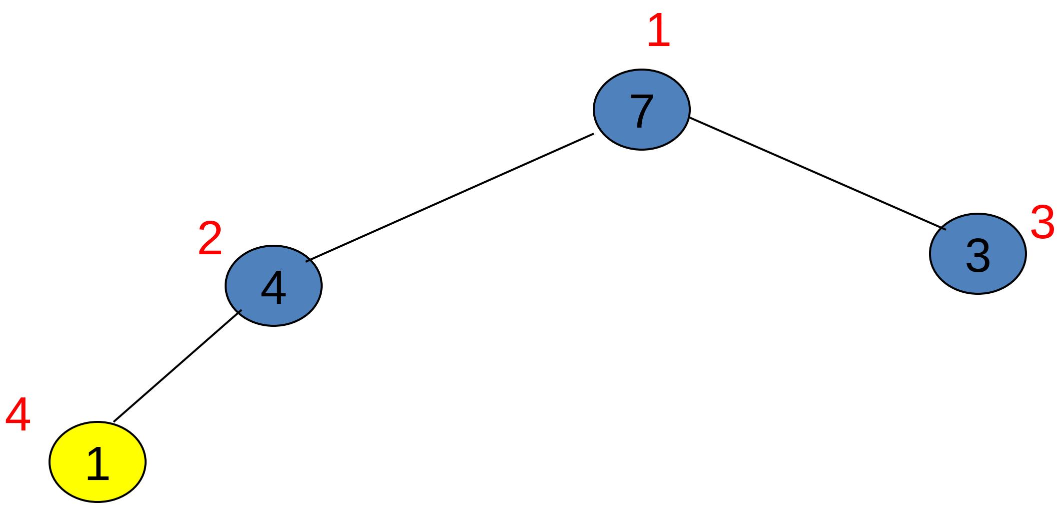


A

				09	11	13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

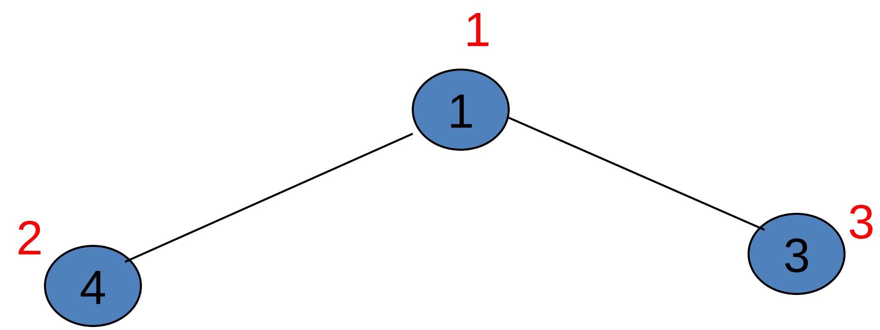


A

				09	11	13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

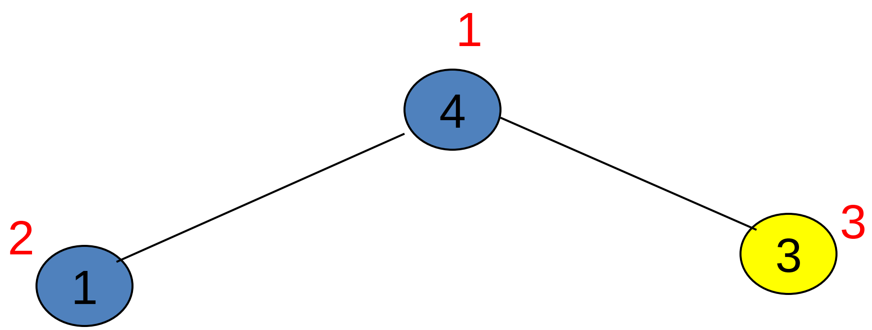


A

			07	09	11	13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11

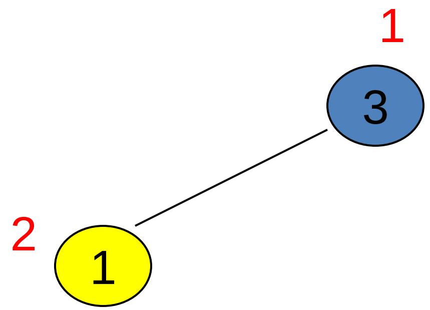


A

			07	09	11	13	14	16	19

HeapSort algorithm

19	14	16	7	13	3	9	1	4	11



A

		04	07	09	11	13	14	16	19

19	14	16	7	13	3	9	1	4	11

A	01	03	04	07	09	11	13	14	16	19

RUNNING TIME OF HEAPSORT

RUNNING TIME OF MAX-HEAPIFY

Running time of MAX-HEAPIFY depends on the height h of a heap i.e O(h)

Height of a heap = lg n

T(n) = O(lg n)

RUNNING TIME OF HEAPSORT

```
HEAPSORT(A)
   BUILD-MAX-HEAP(A)
   for i ← length[A] downto 2
      do exchange A[1] <-> A[i]
              heap-size [A] ← heap-size[A] – 1
                 MAX-HEAPIFY(A,1)
```

- Running time of BUILD-MAX-HEAP is $O(n)$
- MAX-HEAPIFY is executed n-1 times with running time $O(\log n)$
- Total Running time of Heap Sort is $O(n) + n-1(O(\log n)) = O(n \log n)$

RUNNING TIME OF HEAPSORT

Running time of BUILD-MAX-HEAP

Total cost of BULID-MAX-HEAP is $\sum_{h=0}^{\lg n} n/2^{h+1} \; O(h) = O\left(n \sum_{h=0}^{\lg n} h/2^{h}\right)$

$$\sum_{h=0}^{\infty} h/2^{h} = \frac{1/2}{(1-1/2)^2} = 2.$$

Running time of BUILD-MAX-HEAP can be bounded as

$$O\left(n \sum_{h=0}^{\lg n} h/2^{h}\right) = O\left(n \sum_{h=0}^{\infty} h/2^{h}\right) = O(n) .$$