Indian Institute of Information Technology Kottayam

Binary Heaps

By

Arun Cyril Jose

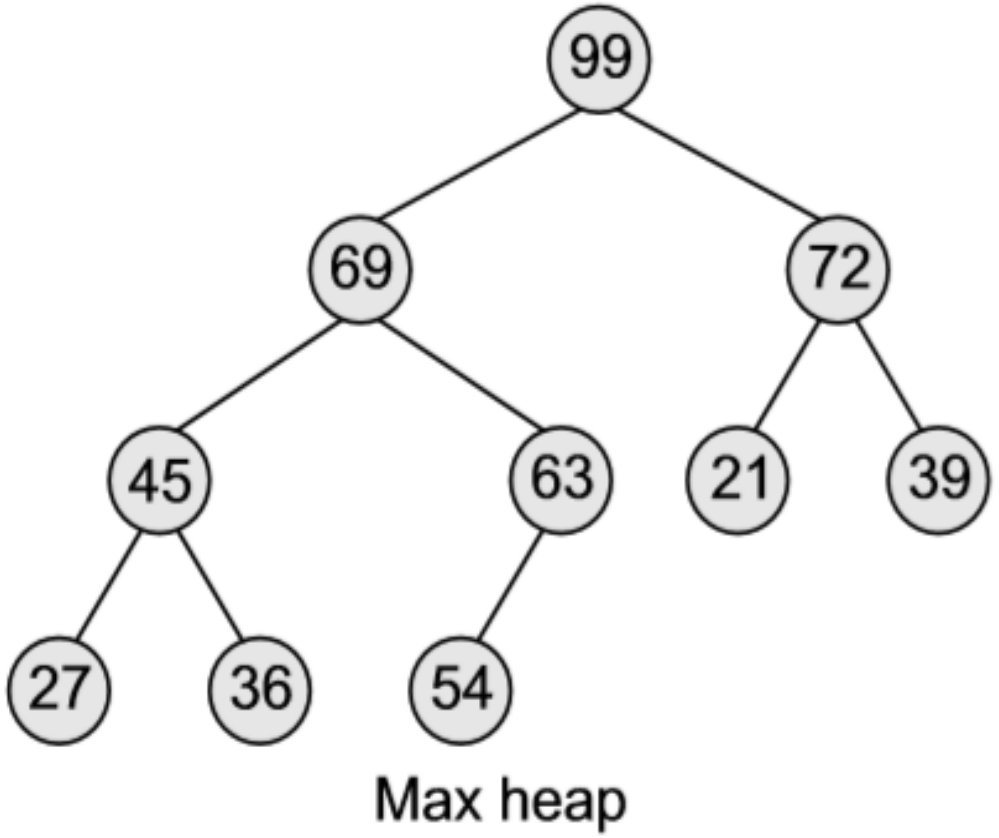
Binary Heaps

- A binary heap is a complete binary tree in which every node satisfies the following property : *If B is a child of A, then key(A) ≥ key(B)*

- i.e. elements at every node will be either greater than or equal to the element at its left and right child.

- Root node has the highest key value in the heap.
- **Max-Heap**

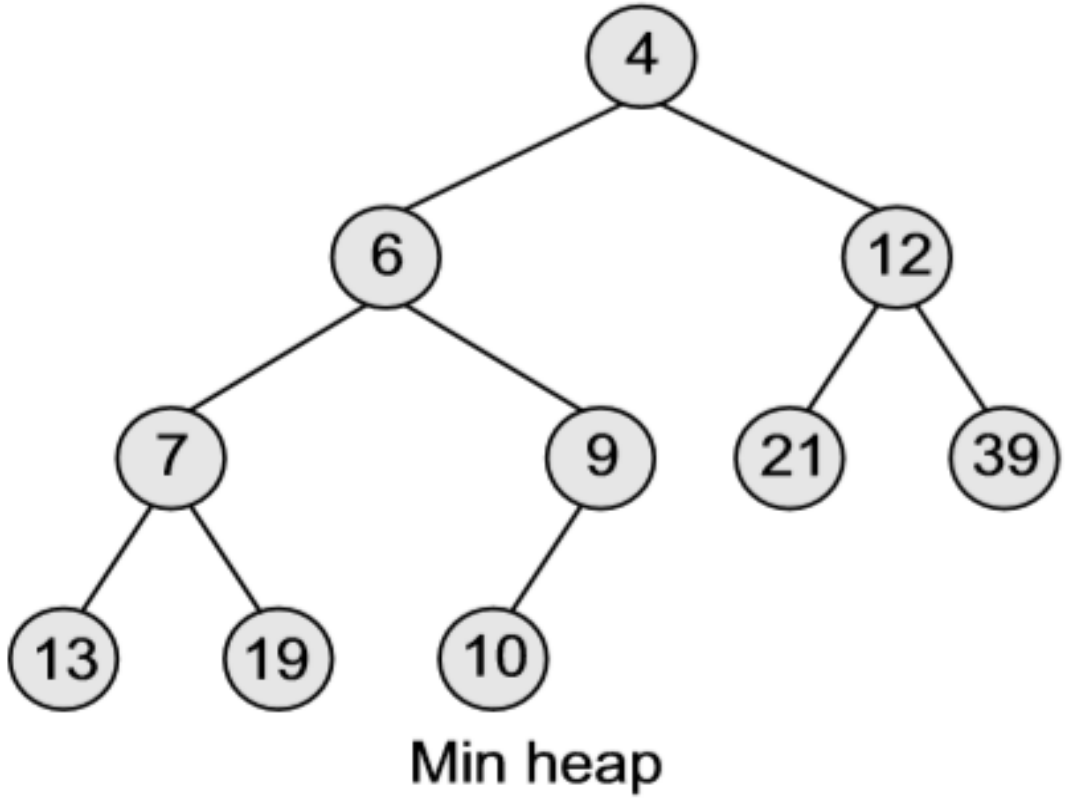
Binary Heaps



Max heap

Binary Heaps

- Elements at every node will be either less than or equal to the element at its left and right child.
- Root will have the lowest key value.
- **Min-Heap**.

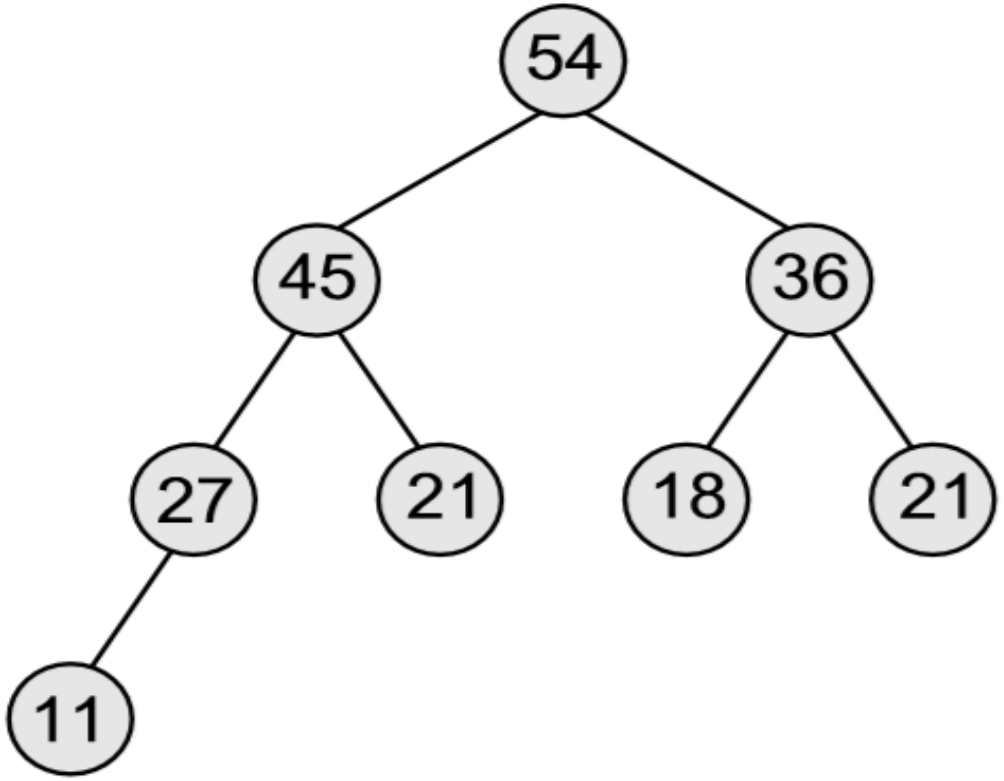


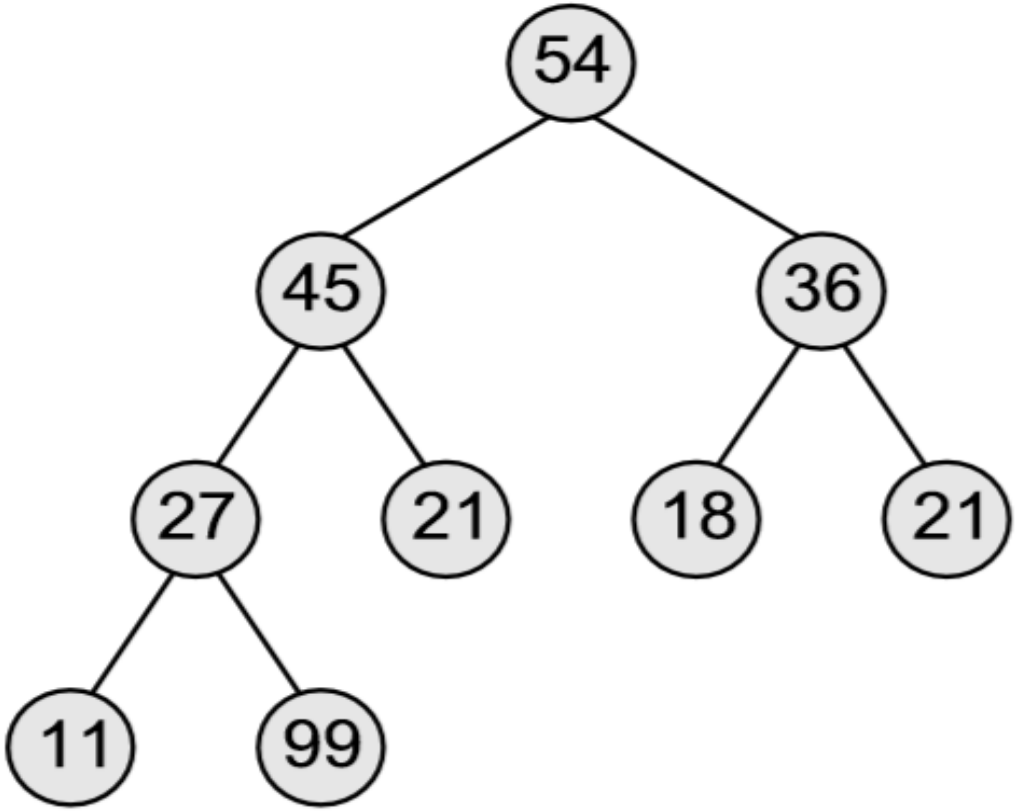
Inserting a New Element in a Binary Heap

Consider a max heap H with n elements,

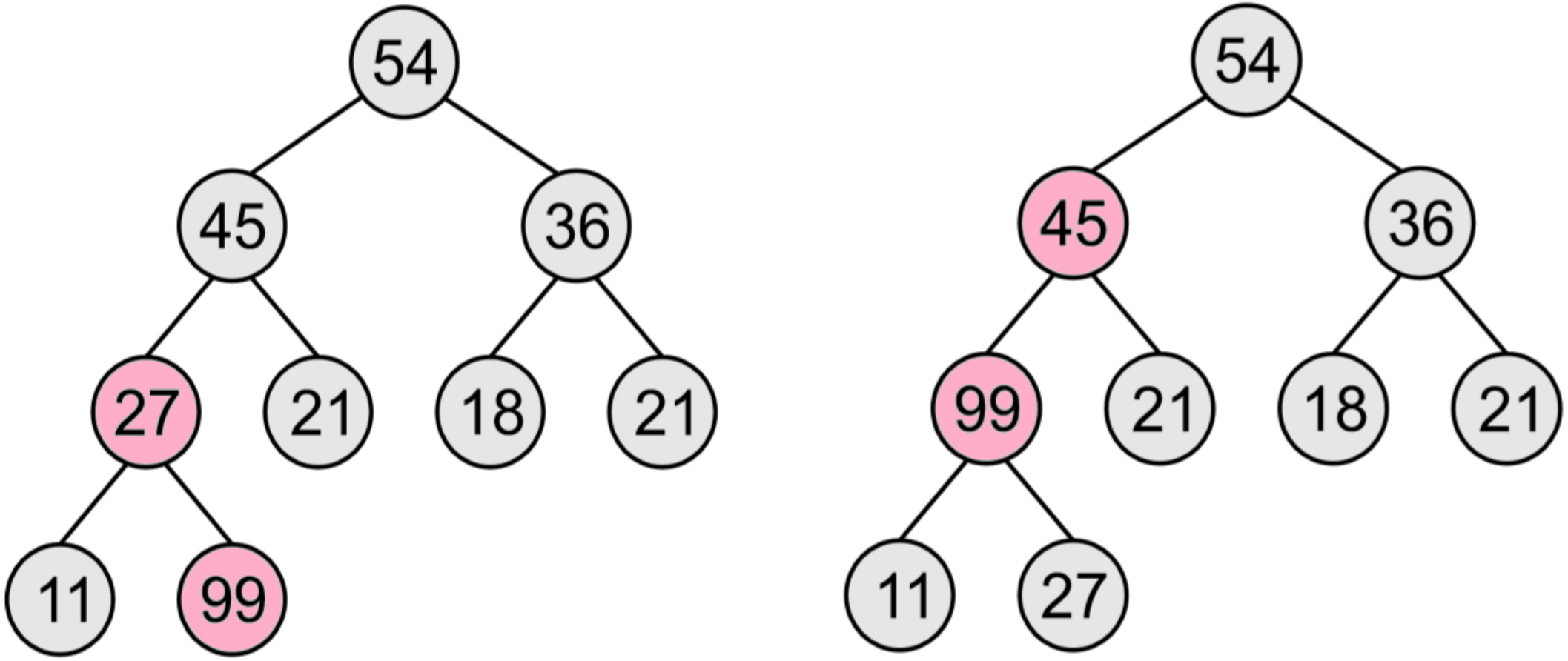
- Add the new value at the bottom of H so that H is still a complete binary tree but not necessarily a heap.
- Let the new value rise to its appropriate place in H so that H now becomes a heap.
- Compare the new value with its parent to check if they are in the correct order. If they are, then the procedure halts, else the new value and its parent's value are swapped and the process is repeated.

Inserting a New Element in a Binary Heap

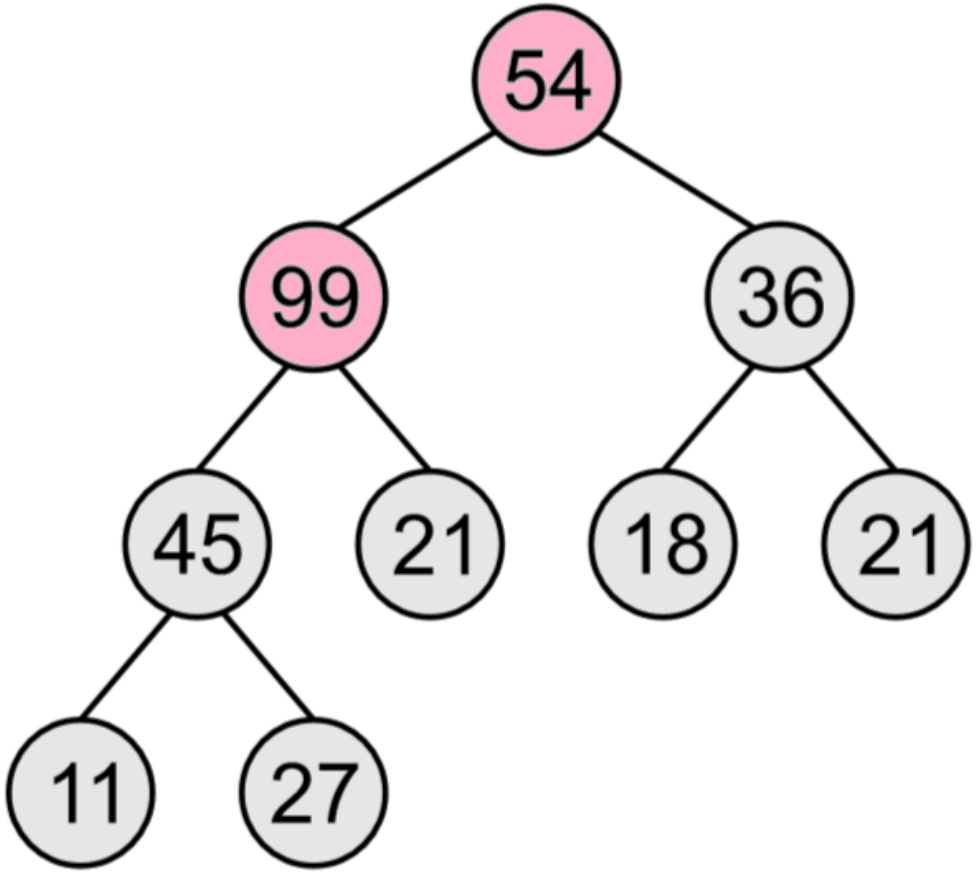


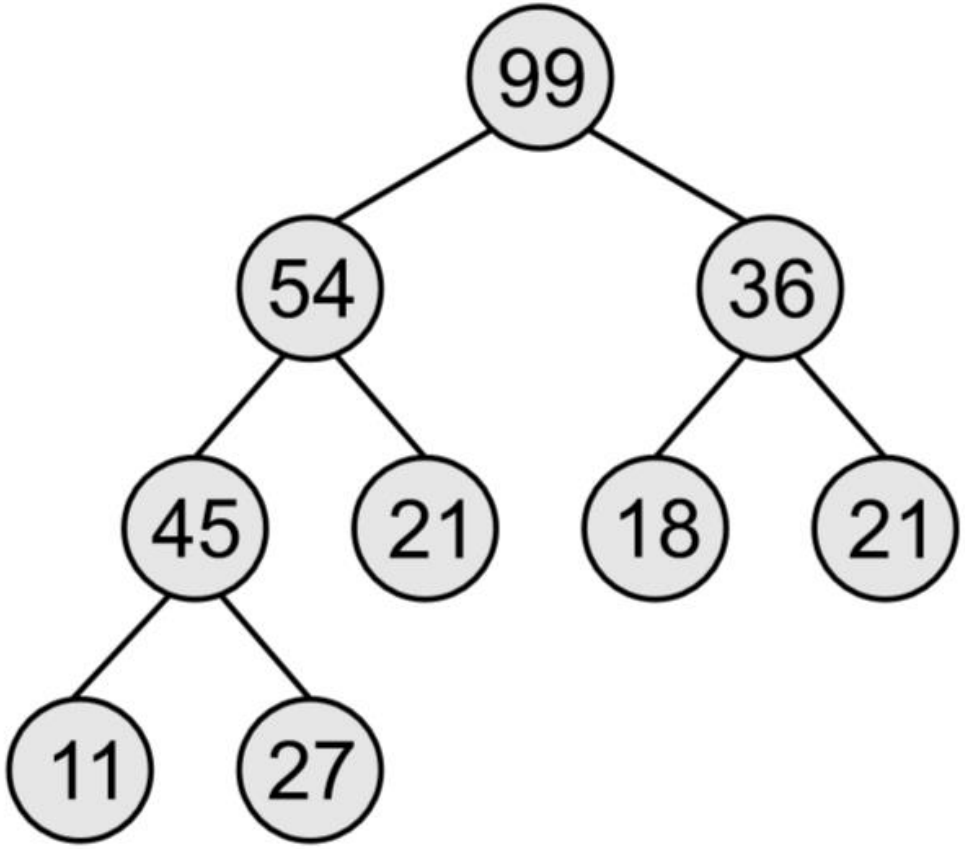


Inserting a New Element in a Binary Heap



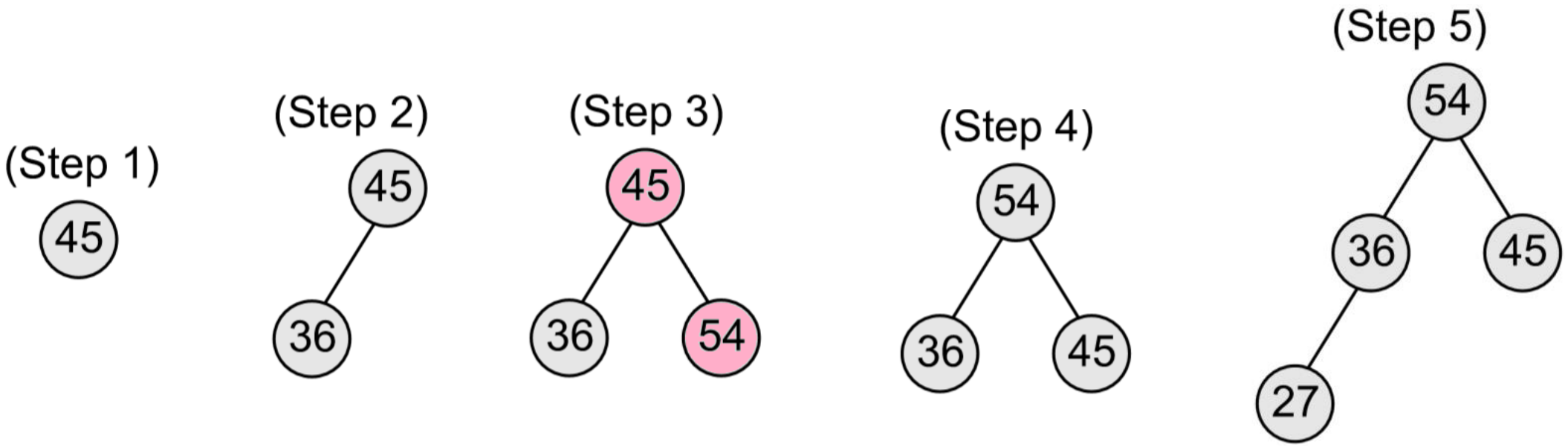
Inserting a New Element in a Binary Heap





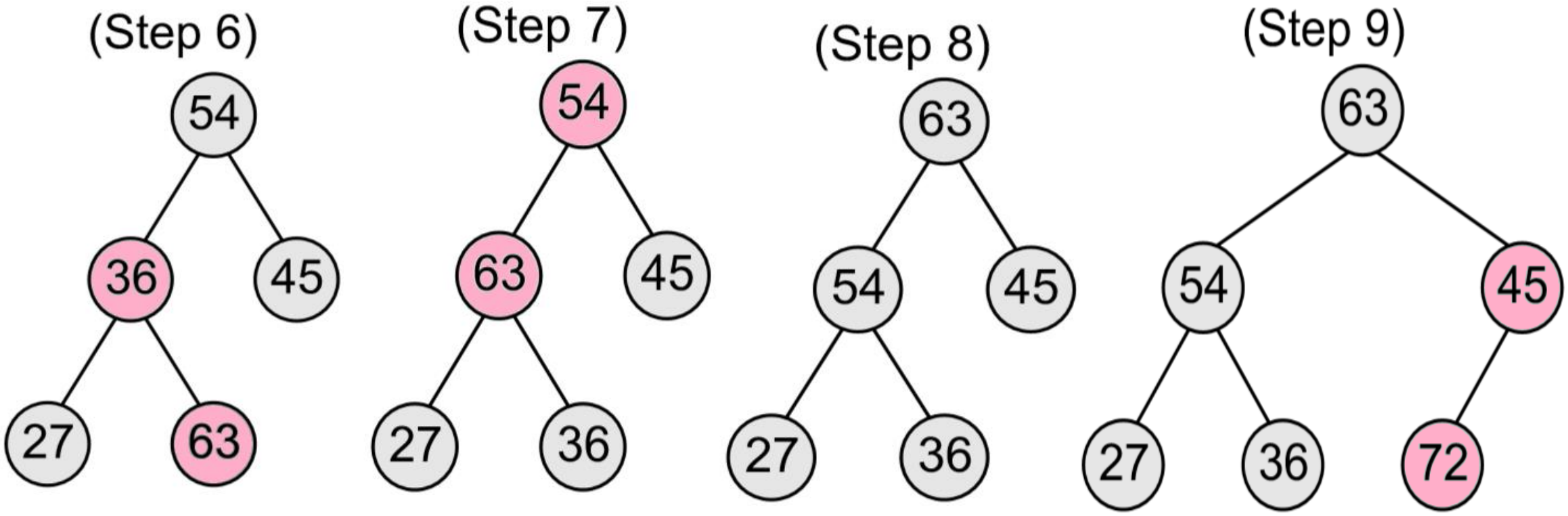
Building a Max Heap

- Building a max heap from these numbers: 45, 36, 54, 27, 63, 72, 61, 18.



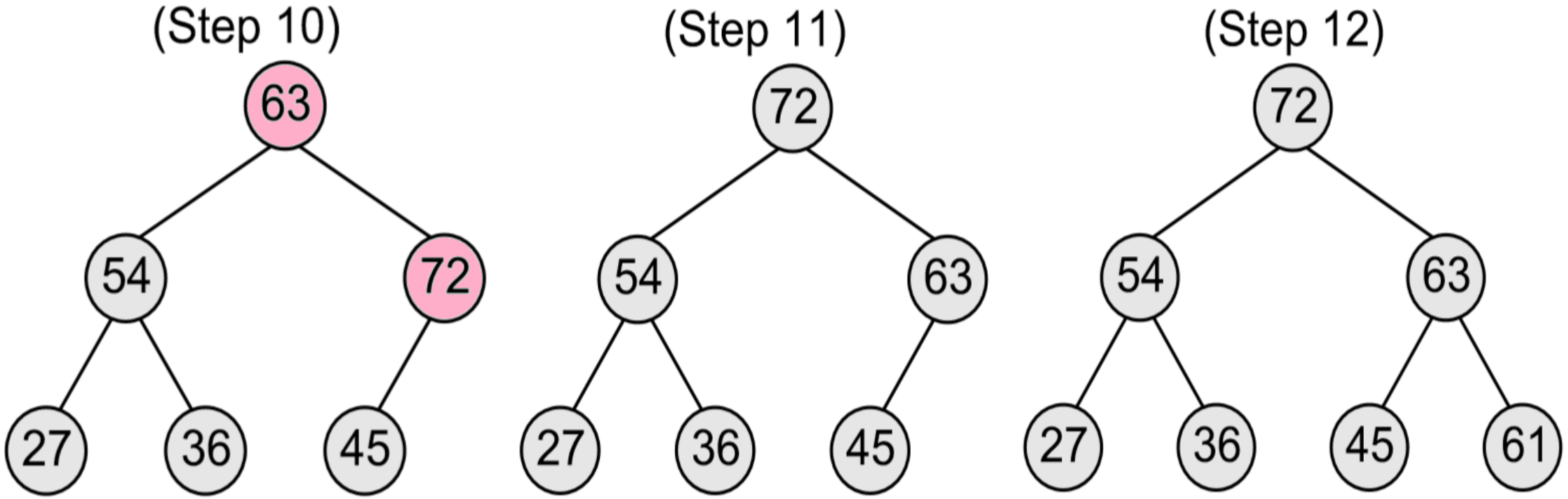
Building a Max Heap

- Building a max heap from these numbers: 45, 36, 54, 27, 63, 72, 61, 18.



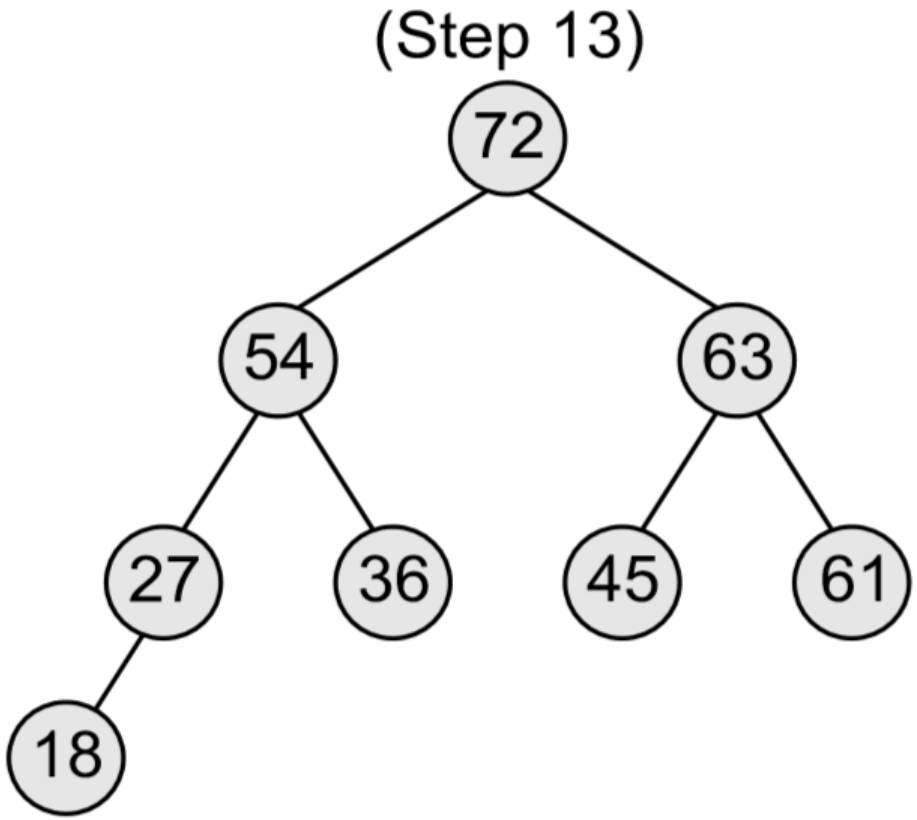
Building a Max Heap

- Building a max heap from these numbers: 45, 36, 54, 27, 63, 72, 61, 18.

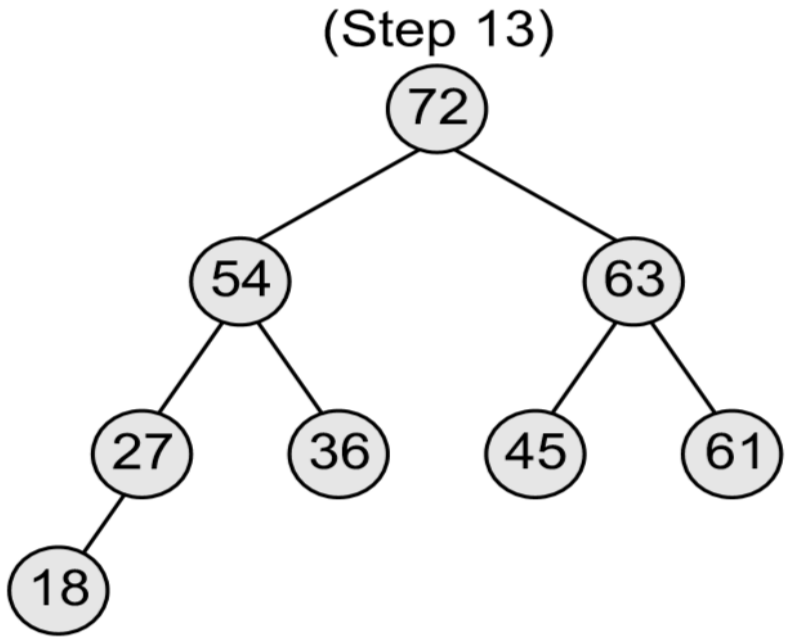


Building a Max Heap

- Building a max heap from these numbers: 45, 36, 54, 27, 63, 72, 61, 18.



Building a Max Heap



HEAP[1]	HEAP[2]	HEAP[3]	HEAP[4]	HEAP[5]	HEAP[6]	HEAP[7]	HEAP[8]	HEAP[9]	HEAP[10]
72	54	63	27	36	45	61	18		

Building a Max Heap

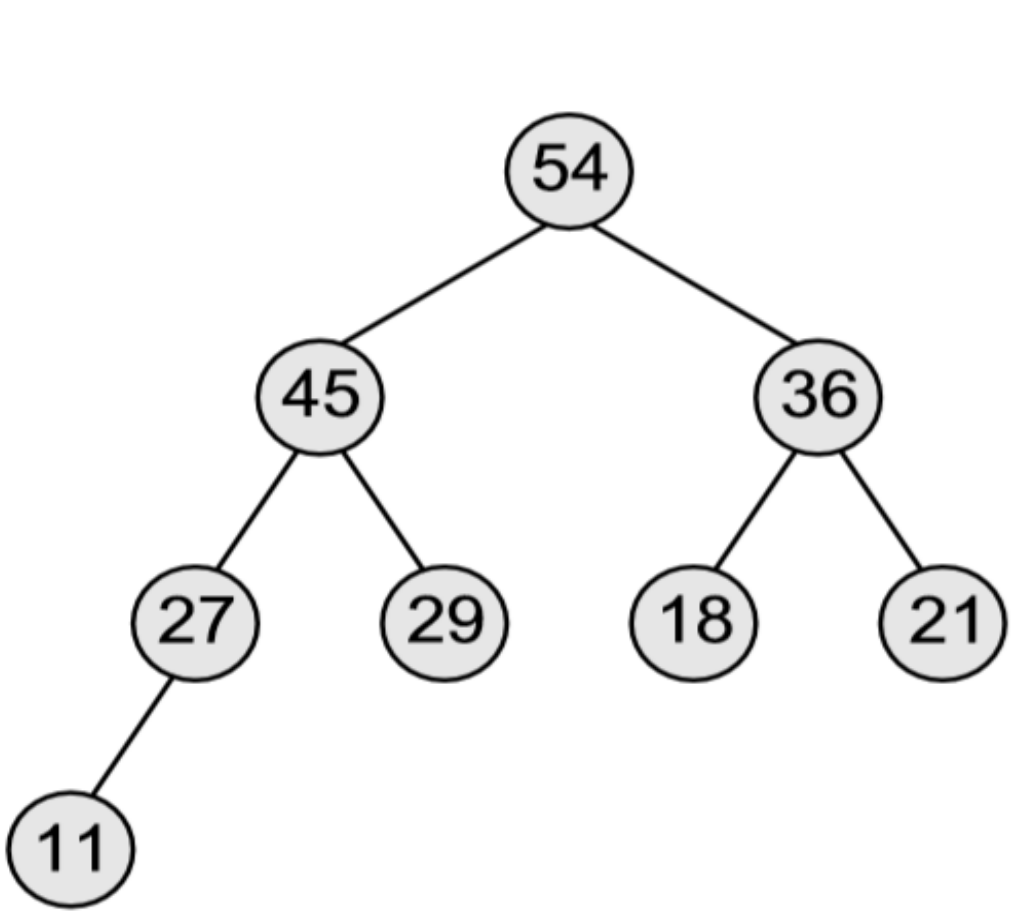
```
Step 1: [Add the new value and set its POS]
        SET N = N + 1, POS = N
Step 2: SET HEAP[N] = VAL
Step 3: [Find appropriate location of VAL]
        Repeat Steps 4 and 5 while POS > 1
Step 4:     SET PAR = POS/2
Step 5:     IF HEAP[POS] <= HEAP[PAR],
            then Goto Step 6.
            ELSE
                  SWAP HEAP[POS], HEAP[PAR]
                  POS = PAR
            [END OF IF]
      [END OF LOOP]
Step 6: RETURN
```

Indian Institute of Information Technology Kottayam

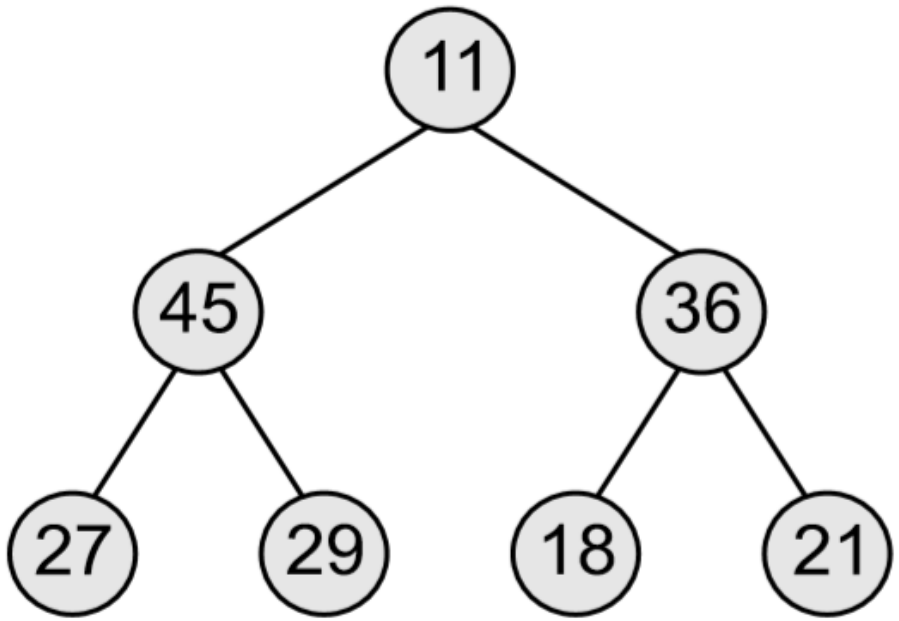
Deleting an Element from a Binary Heap

- Element is always deleted from the root of the heap.
- Consider a max heap H having n elements.
- Replace the root node's value with the last node's value so that H is still a complete binary tree but not necessarily a heap.
- Delete the last node.
- Sink down the new root node's value so that H satisfies the heap property. i.e. interchange the root node's value with its child node's value (whichever is largest among its children for MAX Heap).

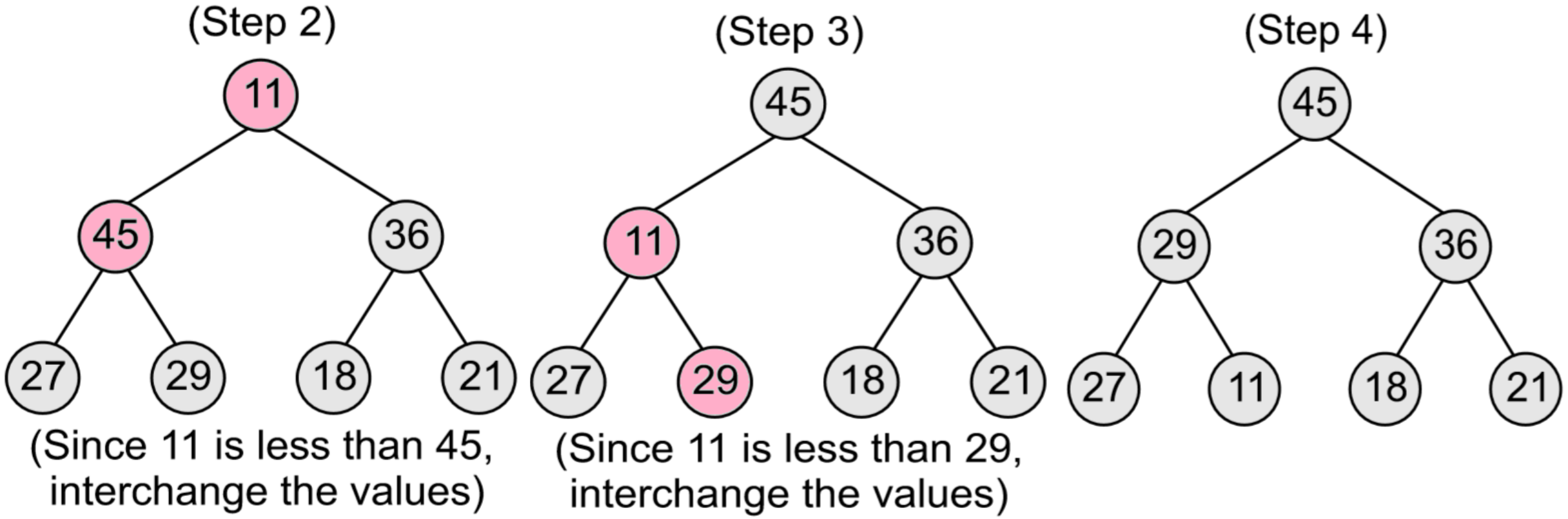
Deleting an Element from a Binary Heap



(Step 1)



Deleting an Element from a Binary Heap



Deleting an Element from a Binary Heap

```
Step 1: [Remove the last node from the heap]
        SET LAST = HEAP[N], SET N = N - 1
Step 2: [Initialization]
        SET PTR = 1, LEFT = 2, RIGHT = 3
Step 3: SET HEAP[PTR] = LAST
Step 4: Repeat Steps 5 to 7 while LEFT <= N
Step 5: IF HEAP[PTR] >= HEAP[LEFT] AND
         HEAP[PTR] >= HEAP[RIGHT]
              Go to Step 8
        [END OF IF]
Step 6: IF HEAP[RIGHT] <= HEAP[LEFT]
              SWAP HEAP[PTR], HEAP[LEFT]
              SET PTR = LEFT
        ELSE
              SWAP HEAP[PTR], HEAP[RIGHT]
              SET PTR = RIGHT
        [END OF IF]
Step 7: SET LEFT = 2 * PTR and RIGHT = LEFT + 1
        [END OF LOOP]
Step 8: RETURN
```

Heap Sort

- We already know, how to build a heap from an array.
- How to insert a new element in an already existing heap, and how to delete an element from H.
- In phase 1, build a heap H using the elements of ARR.
- In phase 2, repeatedly delete the root element of the heap formed in phase 1.
- Which heap do you want to build ??

Heap Sort

```
HEAPSORT(ARR, N)

Step 1: [Build Heap H]
      Repeat for I = 0 to N-1
            CALL Insert_Heap(ARR, N, ARR[I])
      [END OF LOOP]
Step 2: (Repeatedly delete the root element)
      Repeat while N>0
            CALL Delete_Heap(ARR, N, VAL)
            SET N = N + 1
      [END OF LOOP]
Step 3: END
```

Heap Sort

6 5 3 1 8 7 2 4

Heap Sort Complexity

- Running time to sort an array of n elements in worst case is proportional to *O(n log n)*

Searching in Binary Heap