INFORMATIKA UMN

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

IF232
ALGORITHMS
&
DATA STRUCTURES

10
HEAPS

DENNIS GUNAWAN

REVIEW

Efficient Binary Trees:

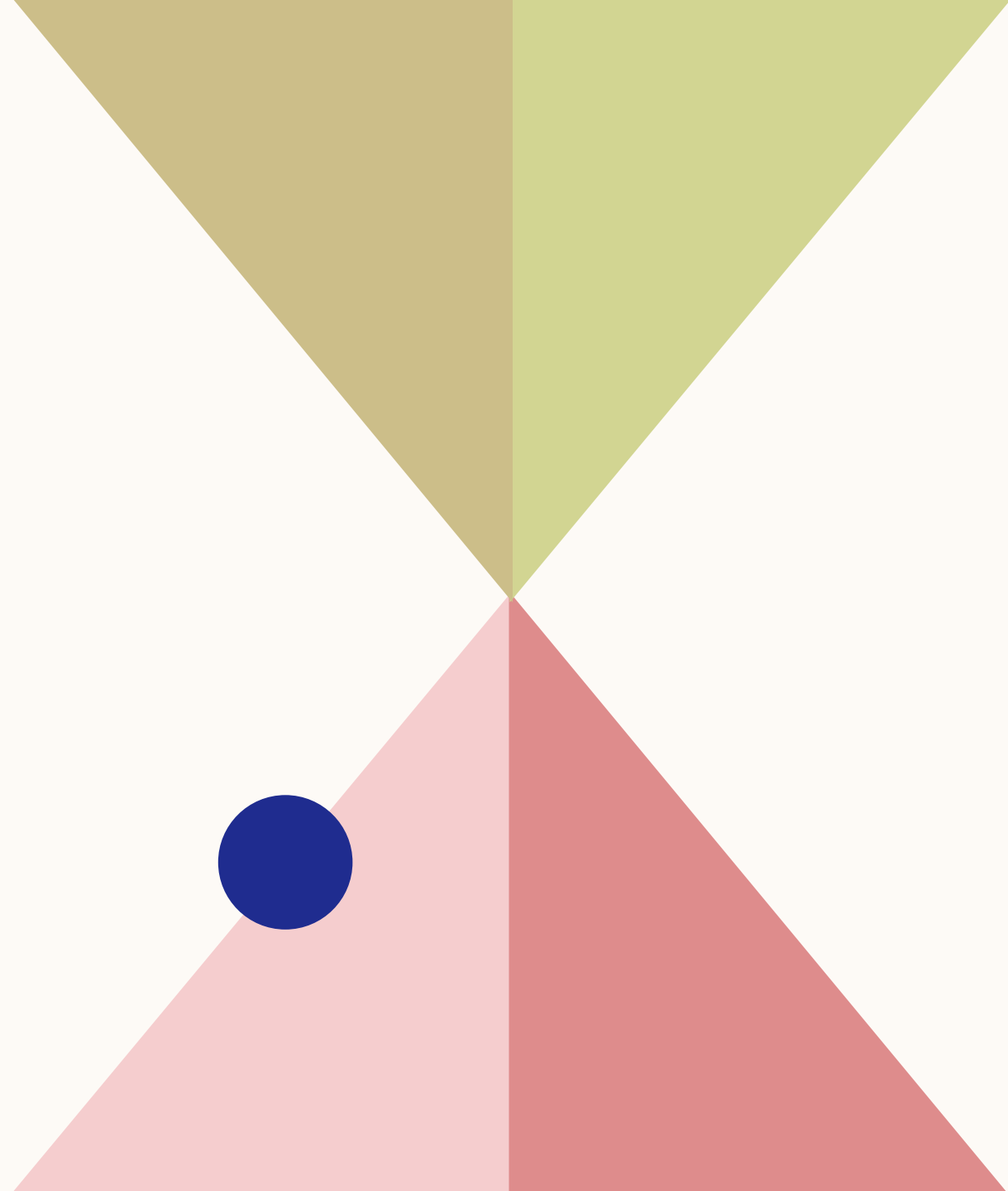
Binary Search Trees

AVL Trees

OUTLINE

Binary Heaps

Applications of Heaps



BINARY HEAPS

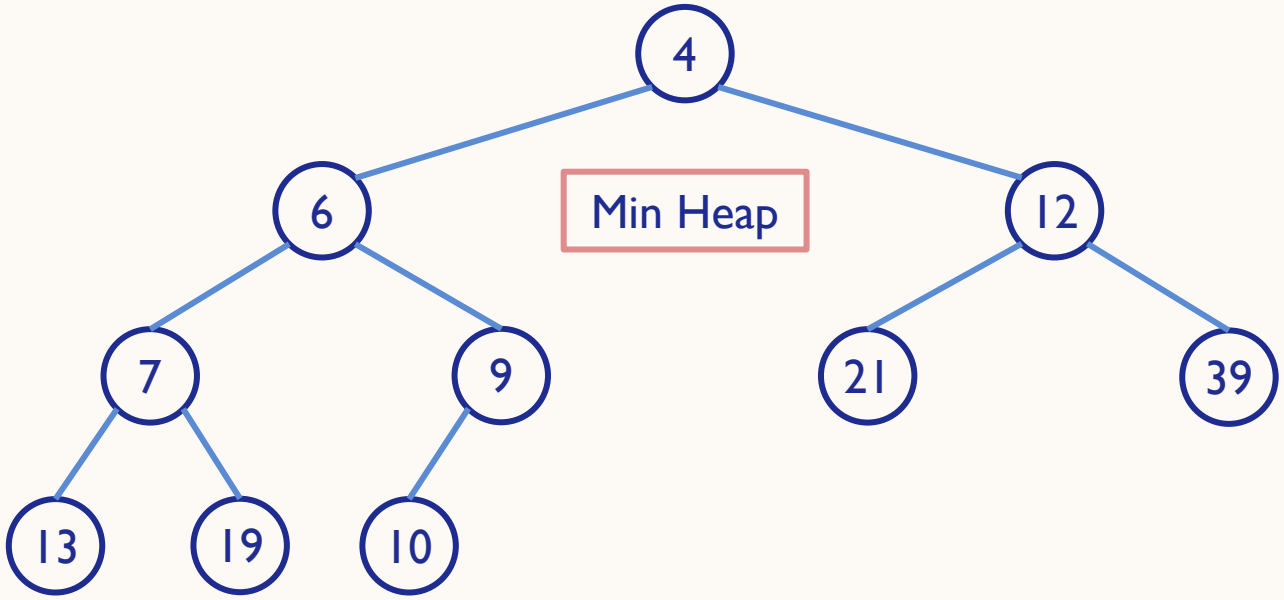
- A heap is a specialized tree-based data structure
- A binary heap is a **complete binary tree** in which every node satisfies the **heap property**

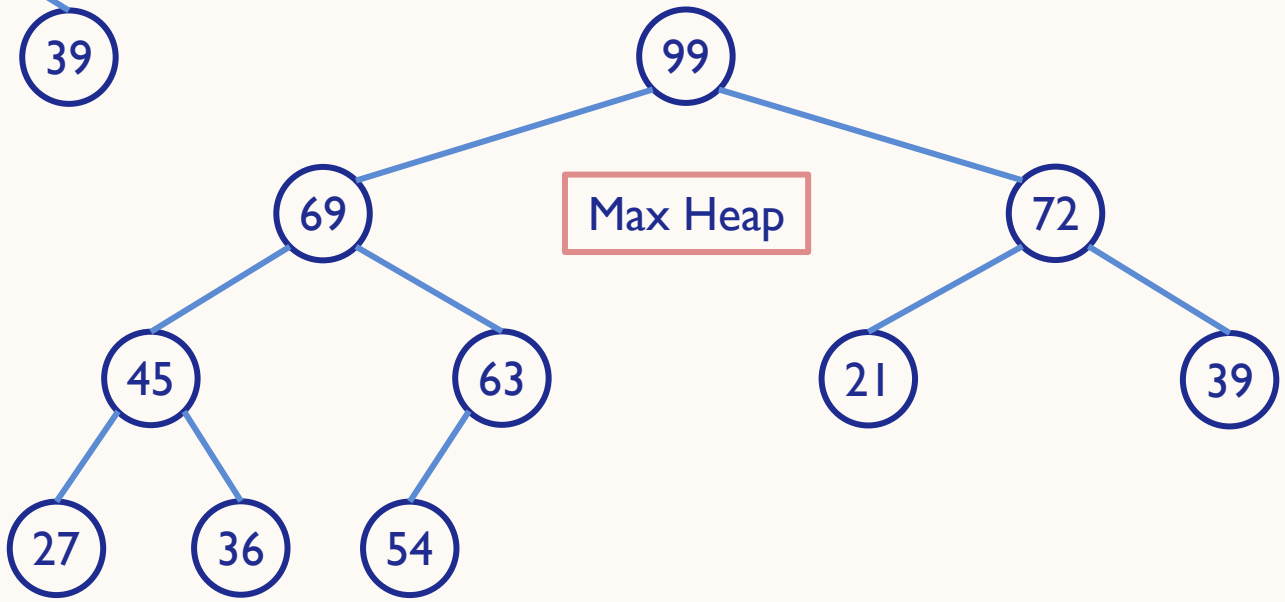
Max Heap	Min Heap
`If B is a child of A, then key(A) ≥ key(B)`	`If B is a child of A, then key(A) ≤ key(B)`
Elements at every node will be either greater than or equal to the element at its left and right child	Elements at every node will be either less than or equal to the element at its left and right child

BINARY HEAPS

- A binary heap is a useful data structure in which elements can be added randomly but only the element with the highest value is removed in case of max heap and lowest value in case of min heap
- A binary tree is an efficient data structure, but a binary heap is more space efficient and simpler

BINARY HEAPS





INSERTING A NEW ELEMENT IN A BINARY HEAP

- Inserting a new value into the heap is done in the following two steps:

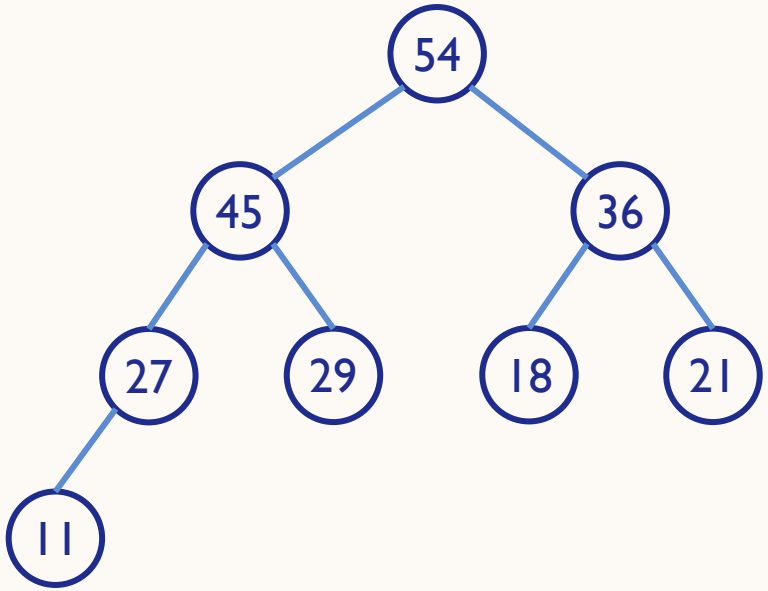
1. Add the new value at the bottom of H in such a way that H is still a complete binary tree but not necessarily a heap

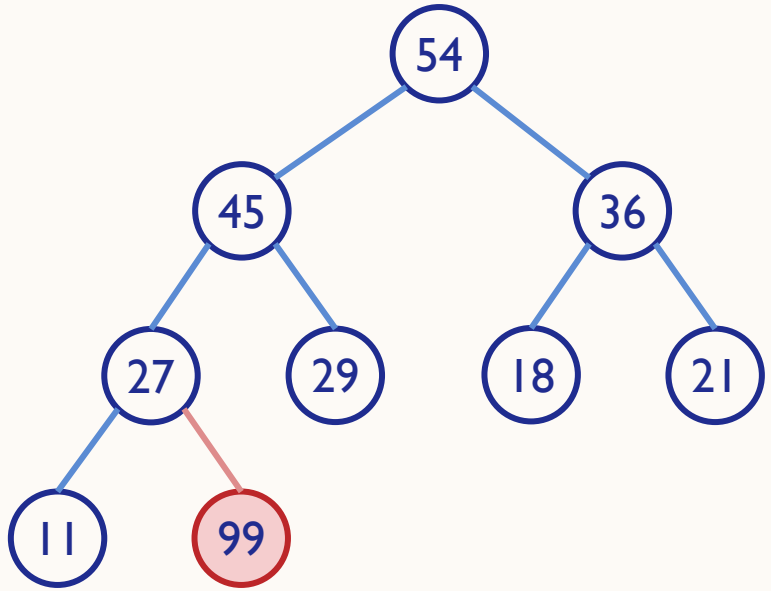
2. Let the new value rise to its appropriate place in H so that H now becomes a heap as well

INSERTING A NEW ELEMENT IN A BINARY HEAP

Consider the max heap given below and insert 99 in it.

Step 1: Add the new value at the bottom of H in such a way that H is still a complete binary tree but not necessarily a heap

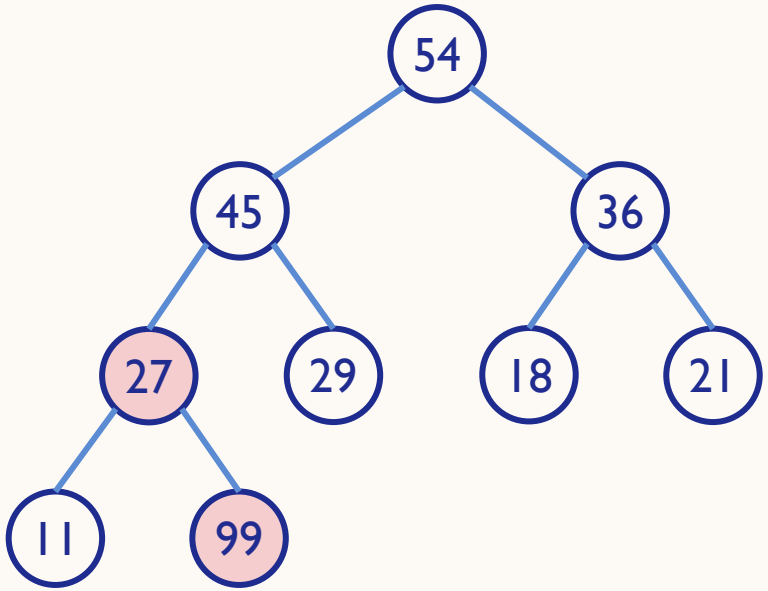


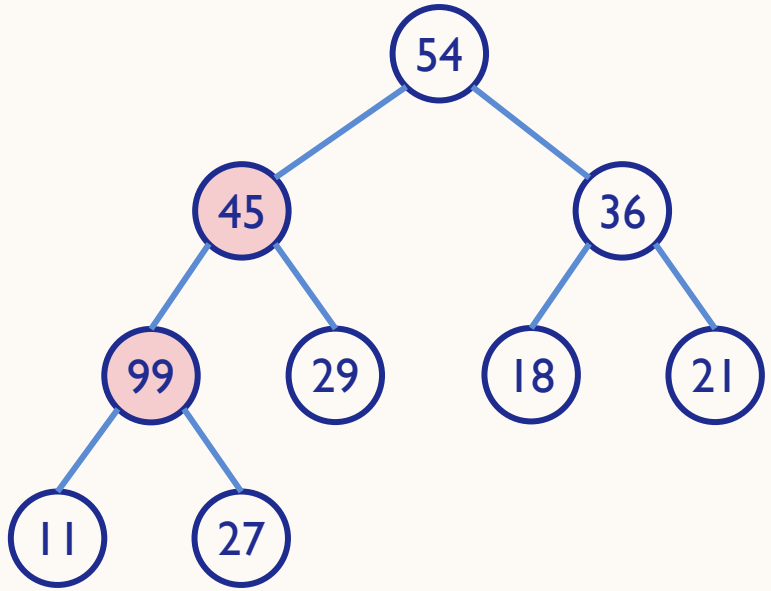


INSERTING A NEW ELEMENT IN A BINARY HEAP

Consider the max heap given below and insert 99 in it.

Step 2: Let the new value rise to its appropriate place in H so that H now becomes a heap as well

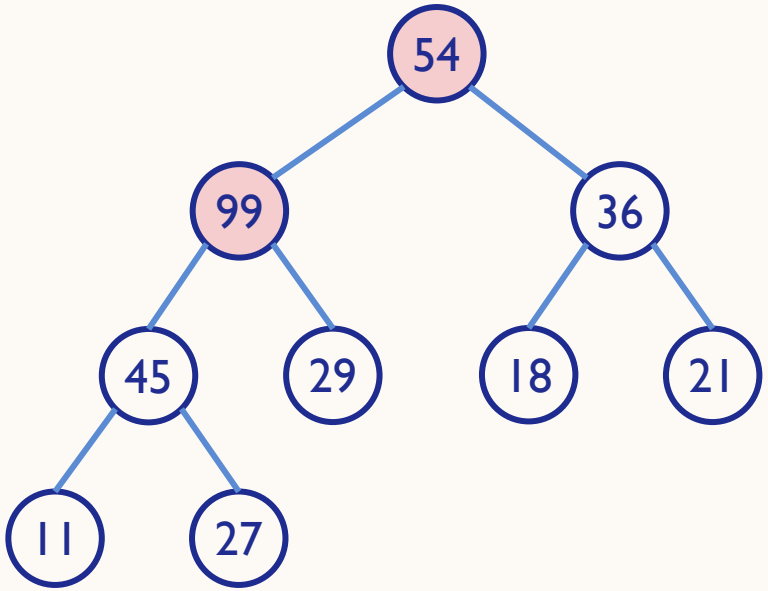


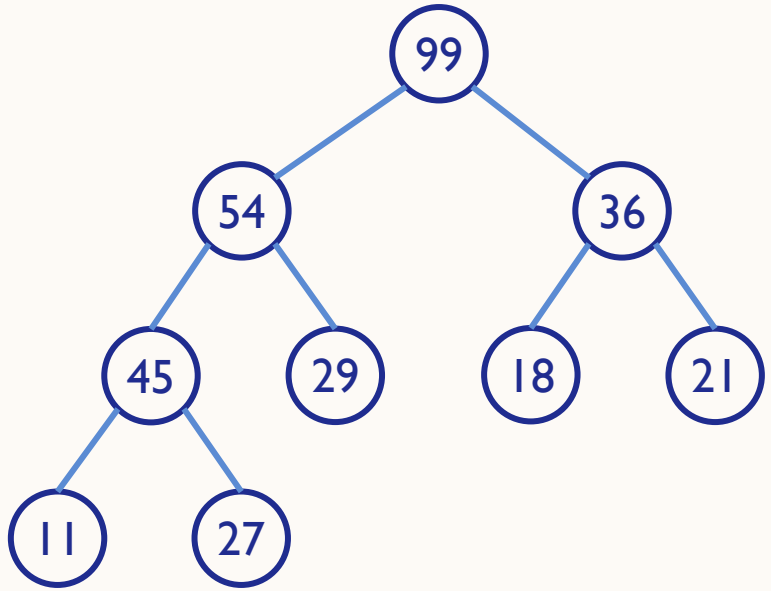


INSERTING A NEW ELEMENT IN A BINARY HEAP

Consider the max heap given below and insert 99 in it.

Step 2: Let the new value rise to its appropriate place in H so that H now becomes a heap as well

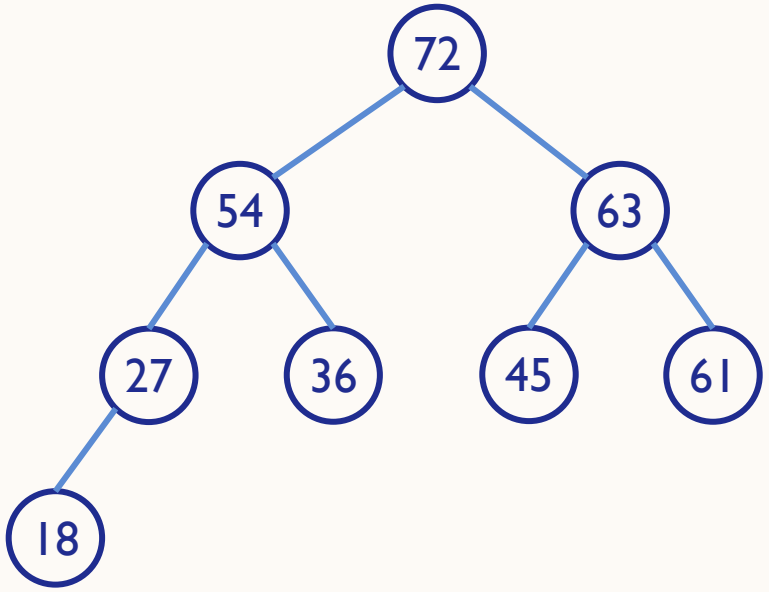




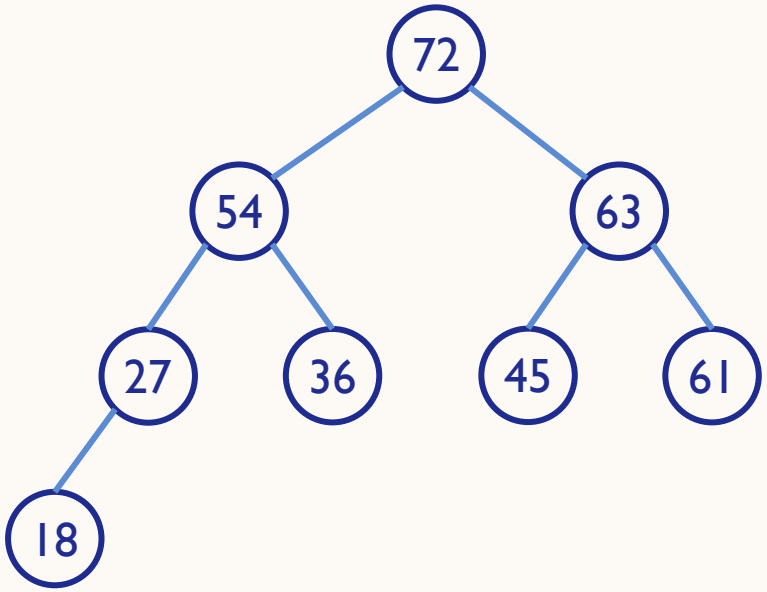
INSERTING A NEW ELEMENT IN A BINARY HEAP

Build a max heap H from the given set of numbers:

45, 36, 54, 27, 63, 72, 61, 18



MEMORY REPRESENTATION OF THE HEAP



- Since a heap is defined as a complete binary tree, all its elements can be stored sequentially in an array
- If an element is at position **`i`** in the array
  - Left child: **`2i`**
  - Right child: **`2i+1`**
  - Parent: **`i/2`**

	72	54	63	27	36	45	61	18	
0	1	2	3	4	5	6	7	8	9

INSERTING A NEW ELEMENT IN A BINARY HEAP

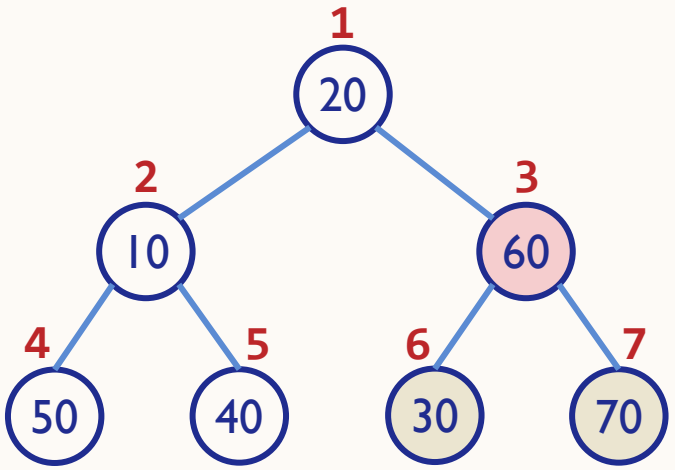
- Algorithm to insert an element in a max heap

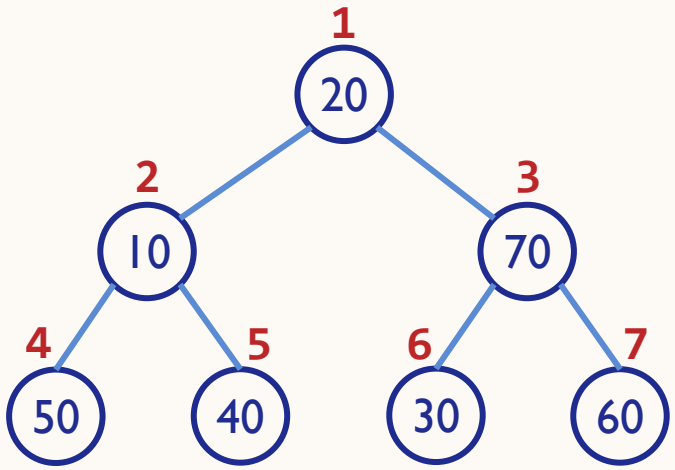
```
Step 1: [Add the new value and set its POS]
        SET N = N + 1, POS = N
Step 2: SET HEAP[N] = VAL
Step 3: [Find appropriate location of VAL]
        Repeat Steps 4 and 5 while POS > 1
Step 4: SET PAR = POS / 2
Step 5: IF HEAP[POS] <= HEAP[PAR]
           Go to Step 6
        ELSE
           SWAP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

HEAPIFY

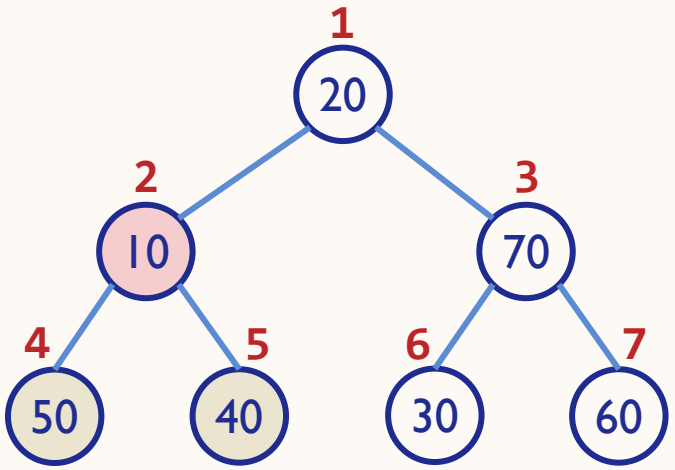
- Heapify is the process of creating a heap data structure from a complete binary tree
- Start from a non-leaf node whose index is given by $n/2$

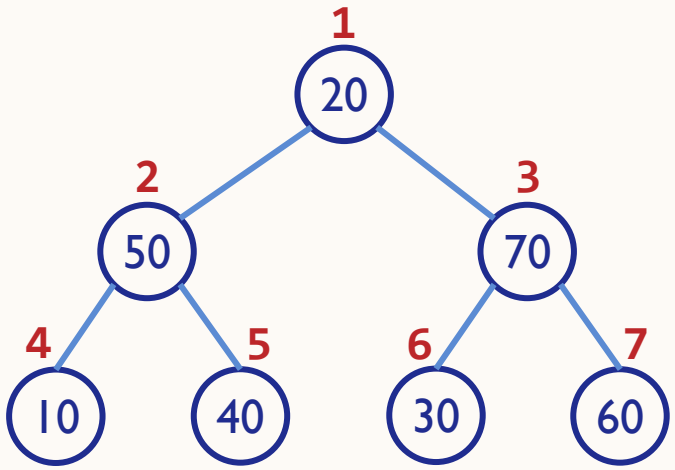
HEAPIFY (MAX HEAP)



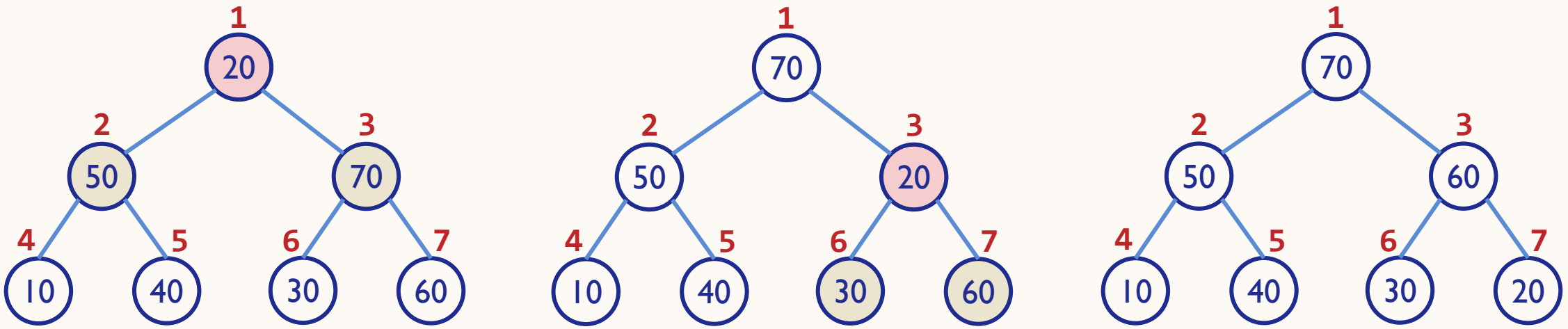


HEAPIFY (MAX HEAP)





HEAPIFY (MAX HEAP)



HEAPIFY (MAX HEAP)

```
void maxheapify(int arr[], int i, int n){
    int child, temp, left, right;

    temp = arr[i];
    while(i <= n / 2){
        left = 2 * i;
        right = 2 * i + 1;

        if(left < n && (arr[right] > arr[left])) child = right;
        else child = left;

        if(arr[child] > temp) arr[i] = arr[child];
        else break;

        i = child;
    }
    arr[i] = temp;
}
```

```
void maxheap(int arr[], int n){
    int i;

    for(i = n / 2; i > 0; i--)
        maxheapify(arr, i, n);
}
```

HEAPIFY (MAX HEAP)

```
void maxheapify(int arr[], int i, int n){
    int largest = i;
    int left = 2 * i;
    int right = 2 * i + 1;

    if(left <= n && arr[left] > arr[largest])
        largest = left;

    if(right <= n && arr[right] > arr[largest])
        largest = right;

    if(largest != i){
        swap(&arr[i], &arr[largest]);
        maxheapify(arr, largest, n);
    }
}
```

```
void maxheap(int arr[], int n){
    int i;

    for(i = n / 2; i > 0; i--)
        maxheapify(arr, i, n);
}
```

HEAPIFY (MIN HEAP)

```
void minheapify(int arr[], int i, int n){
    int lowest = i;
    int left = 2 * i;
    int right = 2 * i + 1;

    if(left <= n && arr[left] < arr[lowest])
        lowest = left;

    if(right <= n && arr[right] < arr[lowest])
        lowest = right;

    if(lowest != i){
        swap(&arr[i], &arr[lowest]);
        minheapify(arr, lowest, n);
    }
}
```

```
void minheap(int arr[], int n){
    int i;

    for(i = n / 2; i > 0; i--)
        minheapify(arr, i, n);
}
```

DELETING AN ELEMENT FROM A BINARY HEAP

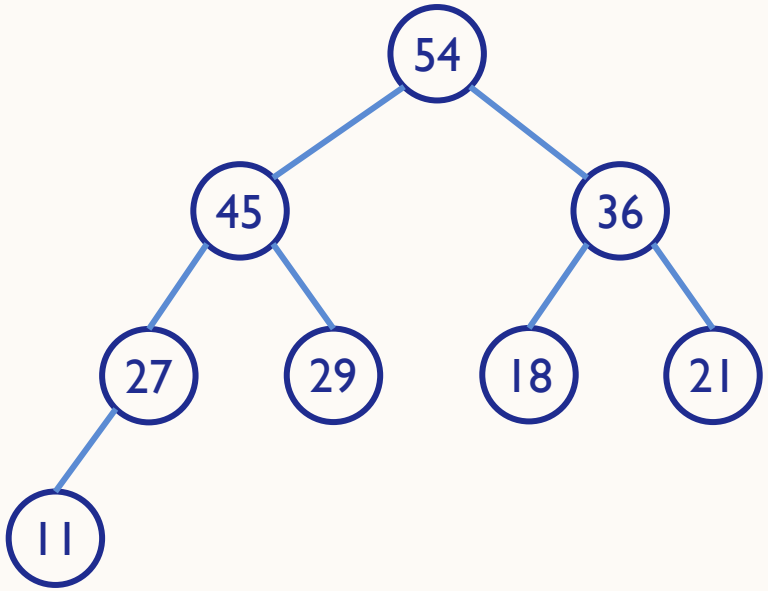
- An element is always deleted from the root of the heap
- Deleting an element from the heap is done in the following three steps:

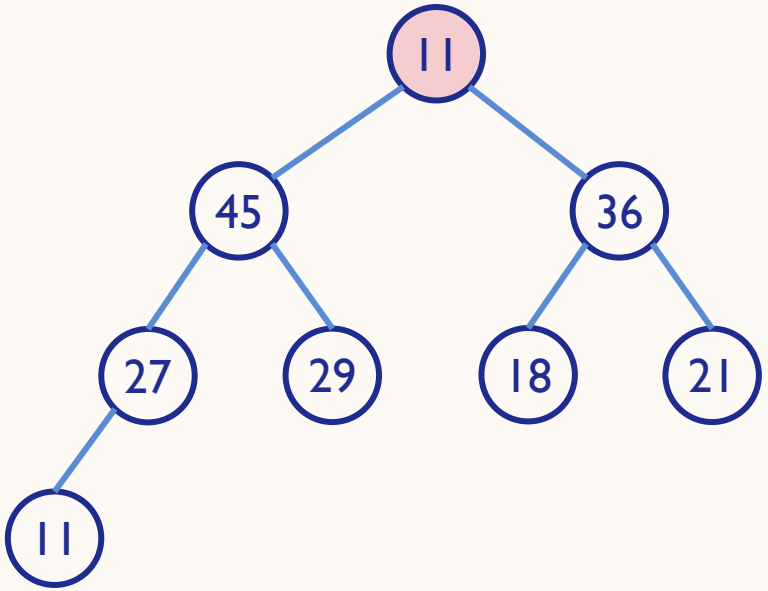
1. Replace the root node's value with the last node's value so that H is still a complete binary tree but not necessarily a heap
2. Delete the last node
3. Sink down the new root node's value so that H satisfies the heap property

DELETING AN ELEMENT FROM A BINARY HEAP

Consider the max heap H shown below and delete the root node's value.

Step 1: 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

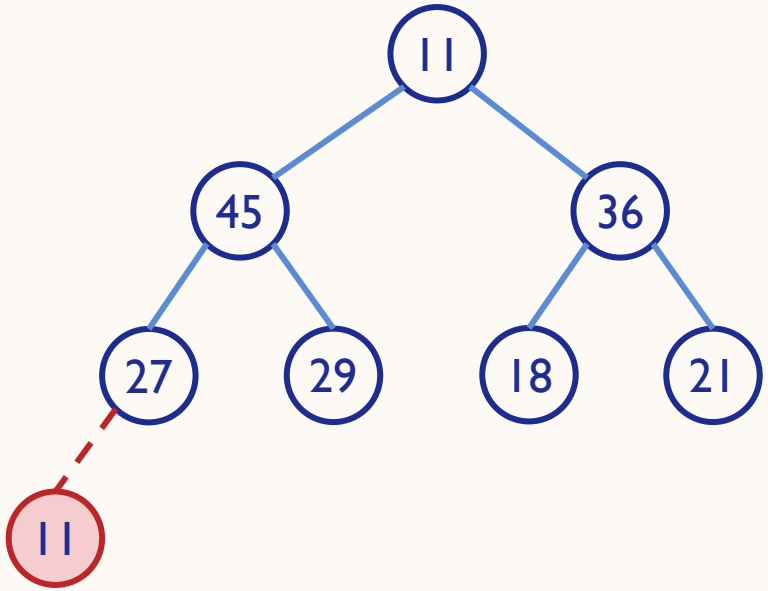


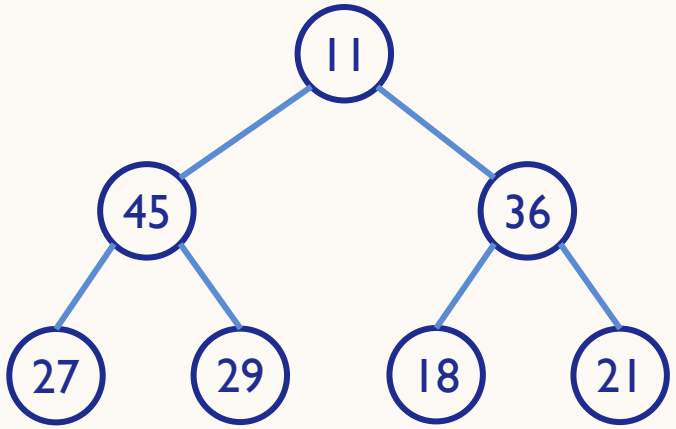


DELETING AN ELEMENT FROM A BINARY HEAP

Consider the max heap H shown below and delete the root node's value.

Step 2: Delete the last node

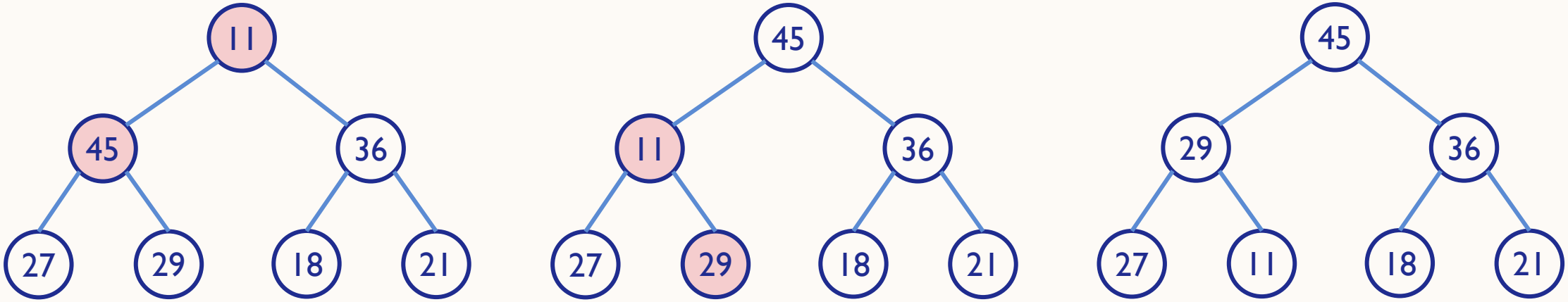




DELETING AN ELEMENT FROM A BINARY HEAP

Consider the max heap H shown below and delete the root node's value.

Step 3: Sink down the new root node's value so that H satisfies the heap property



DELETING AN ELEMENT FROM A BINARY HEAP

- Algorithm to delete the root element from a max 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
```

APPLICATIONS OF HEAPS

- Priority queues
- Heap sort
- Selection algorithms
- Graph algorithms

PRACTICE

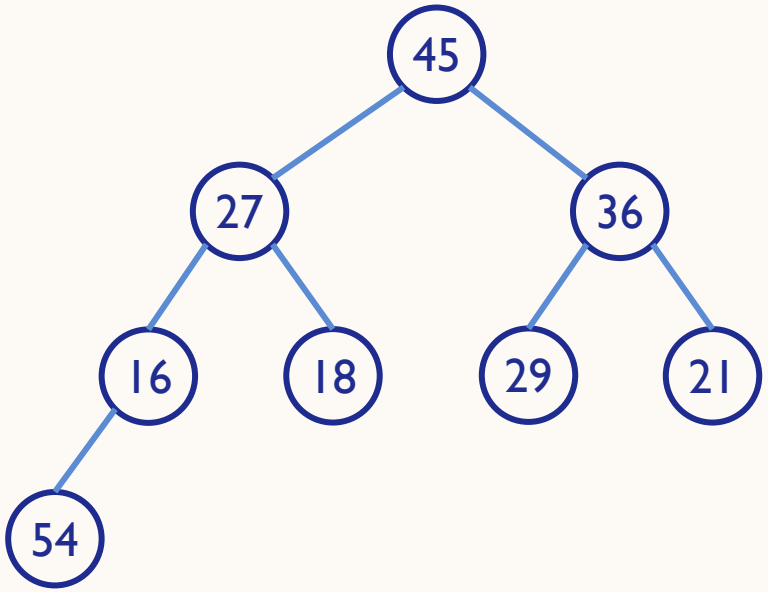
EXERCISES

1. Form a binary max-heap and min-heap from the following sequence of data:

   `50, 40, 35, 25, 20, 27, 33`

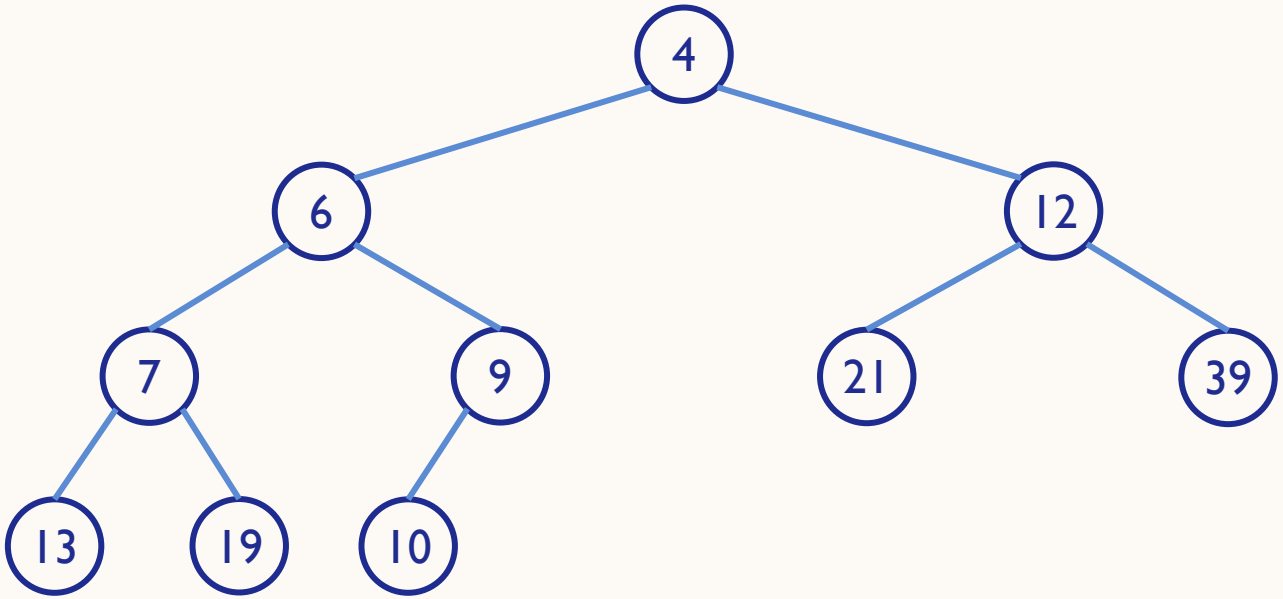
EXERCISES

2. Heapify the following structure to make it a max-heap and min-heap.



EXERCISES

3. Show the array implementation of the following heap.



EXERCISES

4. Given the following array structure, draw the heap.

45	27	36	18	16	21	23	10

Also, find out

a. the parent of nodes 10, 21, and 23

b. index of left and right child of node 23

EXERCISES

5. Which of the following sequences represents a binary heap?

   a. `40, 33, 35, 22, 12, 16, 5, 7`

   b. `44, 37, 20, 22, 16, 32, 12`

   c. `15, 15, 15, 15, 15, 15`

EXERCISES

6. A heap sequence is given as:

   `52, 32, 42, 22, 12, 27, 37, 12, 7`

   a. Which element will be deleted when the deletion algorithm is called thrice?

   b. Show the resulting heap when values 35, 24, and 10 are added to the heap above.

REFERENCES

- Deitel, P. and Harvey Deitel (2022), C How to Program (9th Edition), Pearson Education.
- Thareja, R. (2014), Data Structures Using C (2nd Edition), India: Oxford University Press.

NEXT

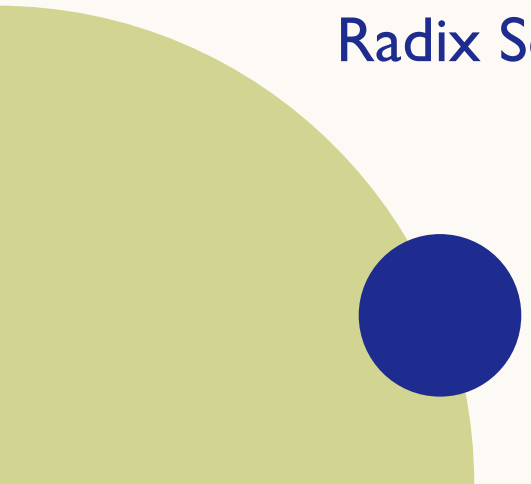
Sorting:

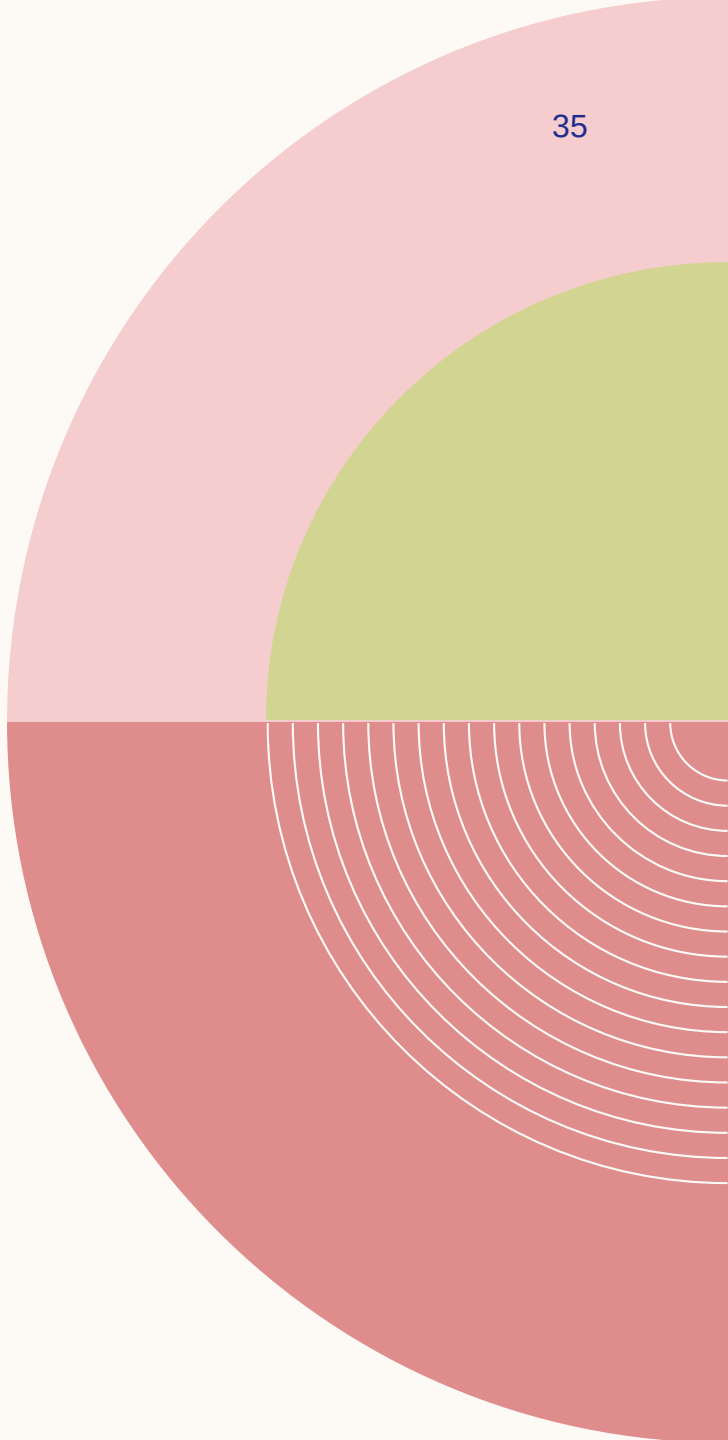
Bubble Sort

Selection Sort

Insertion Sort

Radix Sort





VISION

To become an **outstanding** undergraduate Computer Science program that produces **international-minded** graduates who are **competent** in software engineering and have **entrepreneurial spirit** and **noble character**.

MISSION

1. To conduct studies with the best technology and curriculum, supported by professional lecturer
2. To conduct research in Informatics to promote science and technology
3. To deliver science-and-technology-based society services to implement science and technology

Without hard work,
nothing grows but weeds.

UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

INFORMATIKA UMN

Have patience.
All things are difficult before they become easy.