30 Lecture - CS301

Important Subjective

1. **What is a Min-Heap?**
   A Min-Heap is a binary tree-based data structure where the value of the parent node is always less than or equal to the value of its children.

2. **What is the time complexity for inserting a new element into a Min-Heap?**
   The time complexity for inserting a new element into a Min-Heap is O(log n).

3. **How do you insert a new element into a Min-Heap?**
   To insert a new element into a Min-Heap, you first add the element to the end of the heap, and then you compare the element with its parent. If the parent is larger, you swap the element with its parent, and then you continue comparing the element with its new parent until you reach the root node.

4. **What is the process of maintaining the Min-Heap property after insertion?**
   The process of maintaining the Min-Heap property after insertion involves comparing the new element with its parent and swapping it with the parent if the parent is larger. This process is repeated until the parent is smaller than the new element or until the new element becomes the root node.

5. **What happens if you try to insert a larger element into a Min-Heap?**
   If you try to insert a larger element into a Min-Heap, the Min-Heap property will be violated, and you will need to restore the property by swapping the element with its parent and possibly its children.

6. **Can a Min-Heap have duplicate values?**
   Yes, a Min-Heap can have duplicate values.

7. **What is the height of a Min-Heap with n elements?**
   The height of a Min-Heap with n elements is log(n).

8. **How do you find the minimum element in a Min-Heap?**
   The minimum element in a Min-Heap is always the root node.

9. **What is the time complexity for finding the minimum element in a Min-Heap?**
   The time complexity for finding the minimum element in a Min-Heap is O(1).

10. **How do you delete the minimum element in a Min-Heap?**
    To delete the minimum element in a Min-Heap, you first remove the root node and replace it with the last element in the heap. You then compare the new root with its children and swap it with the smaller child if necessary. This process is repeated until the new root is smaller than both of its children or until it becomes a leaf node.