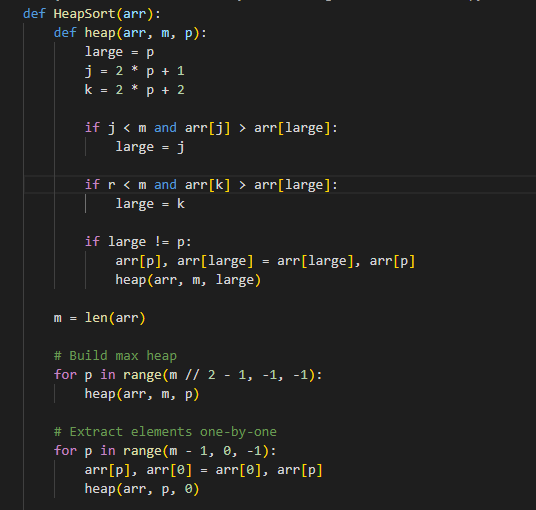
***Aryan Sharma***

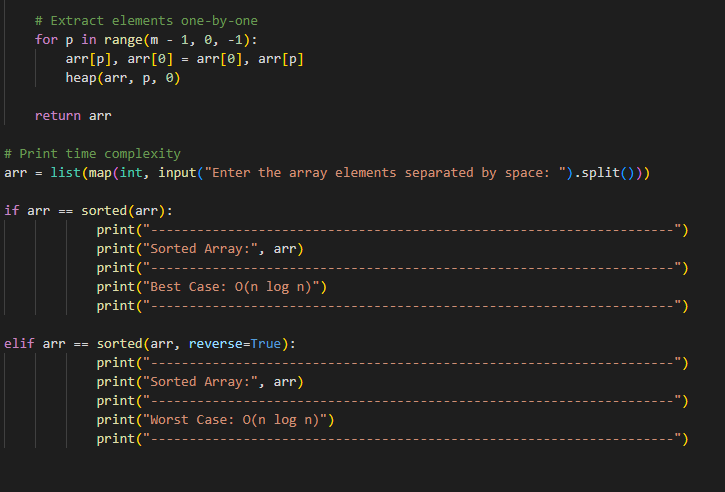
***BT21GCS161***

***Assignment – 07***

***Batch – B3***

***Code***

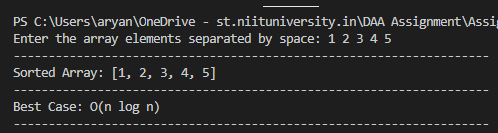
******

******

***Output –***

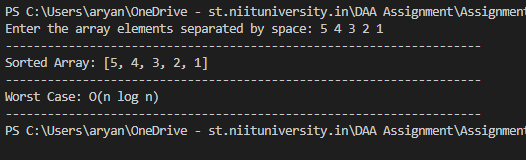
***Best Case***

***In the event that the input array has already been sorted, the best-case scenario***

******

***Worst Case***

***The input array is sorted in reverse order, which is the worst-case scenario***

******

***Heap sort –***

*The average and worst-case time complexity of Heap sort is O (n\*log n). Due to its in-place sorting, it has an O (1) space complexity. Heap sort is unstable, meaning that it does not preserve the relative order of equal elements. Although it may not be the most efficient sorting algorithm for large datasets, Heap sort is frequently used in situations where memory is constrained or when the number of elements to be sorted is small****.***

***Merge Sort –***

*In both the average and worst-case scenarios, Merge sort has a time complexity of O (n\*log n) and a space complexity of O(n) due to the additional memory required for merging subarrays. As Merge sort preserves the relative order of equal elements, it is stable. Merge sort is a commonly used sorting algorithm when memory is not an issue and is suitable for sorting large datasets.*

***Quick Sort –***

*Quick sort has a time complexity of O (n\*log n) in the average case and O(n^2) in the worst case, in addition to a space complexity of O(log n) due to the recursive stack space. Quick sort is not stable because it does not preserve the relative order of equal elements. It is often used when memory is limited due to its good average-case performance. However, quick sort may perform poorly in the worst-case scenario if the pivot element is not chosen wisely*

***Generally, merge sort is a good sorting algorithm that performs well in terms of stability, time and space complexity, and performance. In some circumstances where memory is restricted, more specialised algorithms like heap sort and rapid sort may be useful, but they come with their own trade-offs in terms of stability and worst-case speed.***

***Conclusion:***

*The choice of the best sorting algorithm depends on specific requirements. When a time complexity of O(n log n) is needed, Heap Sort and Quick Sort are the best choices. If stability is required and memory is not a concern, then Merge Sort is the best choice*