Question 1:

1. O(N^2)
2. O(N \* (log(N)^2))

Question 2:

1. O(n ^ (log3 / log2))
2. O(nlogn)
3. O(logn \* log(logn))

Question 3:

The sorting algorithm in the question has the time complexity of O(N^2) due to 2 for loops. However, when using Insertion sort, the time complexity is between O(n), the best case, and O(N^2), the worst case if the pivot is not chosen wisely. In comparison, the Insertion sort would be the better choice in case of time complexity, as the time complexity for Insertion Sort can be much lower than the unchangeable that of the given sorting algorithm.

Question 4:

|  |
| --- |
| #include <stdio.h>  #include <stdbool.h>  #define MAX\_SIZE 100  int findMaxOccurrence(int arr[], int N) {  // Create a hash table to store the count of each number  int count[MAX\_SIZE] = {0};  // Traverse the array and update the count in the hash table  for (int i = 0; i < N; i++) {  count[arr[i]]++;  }  // Find the number with the maximum occurrence  int maxOccurrence = 0;  int maxNumber = 0;  for (int i = 0; i < MAX\_SIZE; i++) {  if (count[i] > maxOccurrence) {  maxOccurrence = count[i];  maxNumber = i;  }  }  return maxNumber;  }  int main() {  int arr[] = {1, 2, 4, 18, 2, 10, 3, 18, 2};  int size = sizeof(arr) / sizeof(arr[0]);  int maxNumber = findMaxOccurrence(arr, size);  printf("The number that appeared the maximum number of times is: %d\n", maxNumber);  return 0;  } |

The time complexity of the program is T(N) = O(max(N, MAX\_SIZE)), which is O(N) if N > MAX\_SIZE and O(MAX\_SIZE) if N < MAX\_SIZE