**Areeba M Farooq**

**CS162008**

**CS5A**

**Design Analysis and Algorithms**

**Assignment 03**

**Q1:**

**Q3:**

**Part (B):**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo name | Stable | Inplace | Worst Case Scenario | *Best Case Scenario* | Worst Case time complexity | Best Case time complexity | Average Case time complexity | Worst Case space complexity | Best Case space complexity |
| Selection sort | Not stable | Inplace | In increasing order the array must be sorted. | The numbers in an array already in order | O(n^2) | O(n^2) | O(n^2) | O(1) |  |
| Bubble sort | Yes stable | Inplace | The array is in complete opposite order | In ascending order the array is already sorted. | O(n^2) | O(n) | O(n^2) | O(1) |  |
| Modified bubble sort | Yes stable | Inplace | The array is in complete opposite order | In ascending order the array is already sorted. | O(n^2) | O(n) | O(n^2) | O(1) |  |
| Insertion sort | Yes stable | Inplace | The array is sorted in complete reverse order | The array is already sorted. | O(n^2) | O(n) | O(n^2) | O(1) |  |
| Bin sort | Yes stable | Not Inplace | When all the numbers are in a single bucket | When all the numbers are not in a single bucket | O(n^2) | O(n+k ) | O(n+k) | O(n) |  |
| Tree sort | Yes stable | Not Inplace | The balanced trees results in worst case | When the number we are looking for is in the middle because we’ll get the output in a single step. | O(n^2) | O(n log (n)) | O(n log (n)) | O(n ) |  |
| Quick sort | Not stable | Inplace | The worst case is when the array is in complete reverse order and when the array is sorted or when all numbers are same | The best case is that the pivot is on right or in the middle expect for last. | O(n^2) | O(n log (n)) | O(n log (n)) | O(n ) |  |
| Merge sort | Yes stable | Not Inplace | The worst case is When the maximum number of comparisions occur. | The best case is when the n/2 comparisions are made in an array. | O(n log (n)) | O(n log (n)) | O(n log (n)) | O(n) |  |
| Inplace merge sort | Yes stable | Inplace | The worst case is When the maximum number of comparisions occur. | The best case is when the n/2 comparisions are made in an array. | O(n log (n)) | O(n log (n)) | O(n log (n)) | O(n ) |  |
| Heap sort | Not stable | Inplace | The worst case is when the multiple of (log (n)) swaps are made | The best case is when all the numbers are same | O(n log (n)) | O(n log (n)) | O(n log (n)) | O(1) |  |
| Counting sort | Yes stable | Not Inplace | The worst case is when all the numbers are equal | The best case is when the numbers are sorted | O(n+k) | O(n+k) | O(n+k) | O(k) |  |
| Radix Sort | Yes stable | Not Inplace | The worst case is when there is small radix but large key area | The best case is when the capacity of the word is awaited to be reduced than log (n) | O(wn) | O(wn) | O(wn) | O(n+r) |  |
| Shell sort | Not stable | Inplace | The worst case is that the sorting should be in squaric time. | The best case is when the numbers are already sorted | O(n (log n)) ^ 2) | O(n (log n)) | O(n (log n) ^ 2) | O(1) |  |