**Offline 2**

**Problem 1:**

**Machine Configuration:**

Processor: Intel® Core™ i3-7100 CPU @2.40 GHz

RAM: 12 GB

Operating System: Windows 10 64 bit

**Data and Complexity Analysis:**

**Selection Sort:**

Selection sort is a simple sorting algorithm which finds the minimum element each time it iterates through the array and puts it at the beginning. While sorting, the array is divided into two parts, the sorted subarray and the unsorted one. In each iteration the minimum element is put in the sorted subarray from the unsorted one.

T(n)= c1(n-1) +c2(n-2) +………………………+cn

= c\*(n-1) \*n/2

**Time complexity:** O(n2) for average, worst and best case

**Space Complexity:** O(1)

As there are two nested loops, time complexity will be O(n2). In best case scenario, the array is already sorted so there will be no change in the value of minimum in the inner loop. Since that takes a constant time, time complexity for best case will also be O(n2). For worst case scenario, the array will be in reversed order and the value of minimum will be changed in each iteration in inner loop but it also takes a constant time and the time complexity will be the same.

|  |  |  |  |
| --- | --- | --- | --- |
| **Array Size** | **Time for Best Case(microseconds)** | **Time for Worst Case(microseconds)** | **Time for Average Case(microseconds)** |
|  |  |  |  |
| 10 | 299.9 | 699.2 | 596.1 |
| 100 | 16404.3 | 22722.6 | 20579.8 |
| 200 | 70404.3 | 82794.8 | 60931.4 |
| 500 | 459308 | 531847 | 460927 |
| 1000 | 1.70658+E6 | 1.83586+E6 | 1.7594+E6 |
| 2000 | 6.99159+E6 | 7.39324+E6 | 7.08629+E6 |
| 5000 | 4.40668+E7 | 4.83568+E7 | 4.4507+E7 |
| 10000 | 1.75391+E8 | 1.9925+E8 | 1.76175+E8 |

**Insertion Sort:**

Insertion sort is a sorting algorithm that builds the final sorted array one element at a time.

T(n)=c1\*1+c2\*2+c3\*3+…………………+cn\*(n-1)

=c\*(n-1)\*n/2

**Time Complexity:** O(n2) for average and worst case, O(n) for best case

**Space Complexity:** O(1)

In best case scenario, Insertion sort performs no swap and exits the inner loop after checking once. So in best case, the time complexity is O(n). While in worst case scenario, swapping happens in each iteration in inner loop costing the time complexity to be O(n2)

|  |  |  |  |
| --- | --- | --- | --- |
| **Array Size** | **Time for Best Case(microseconds)** | **Time for Worst Case(microseconds)** | **Time for Average Case(microseconds)** |
|  |  |  |  |
| 10 | 198.1 | 499.5 | 402.9 |
| 100 | 398.9 | 19819.8 | 15228.5 |
| 200 | 598.9 | 93827.4 | 50052.3 |
| 500 | 1562.5 | 555128 | 271174 |
| 1000 | 2160.8 | 2.197880+E6 | 1.122850+E6 |
| 2000 | 11404.6 | 8.81389+E6 | 4.35723+E6 |
| 5000 | 23913 | 5.68299+E7 | 2.84548+E7 |
| 10000 | 55915 | 2.22601+E8 | 1.12191+E8 |