Assignment 3 (DAA)

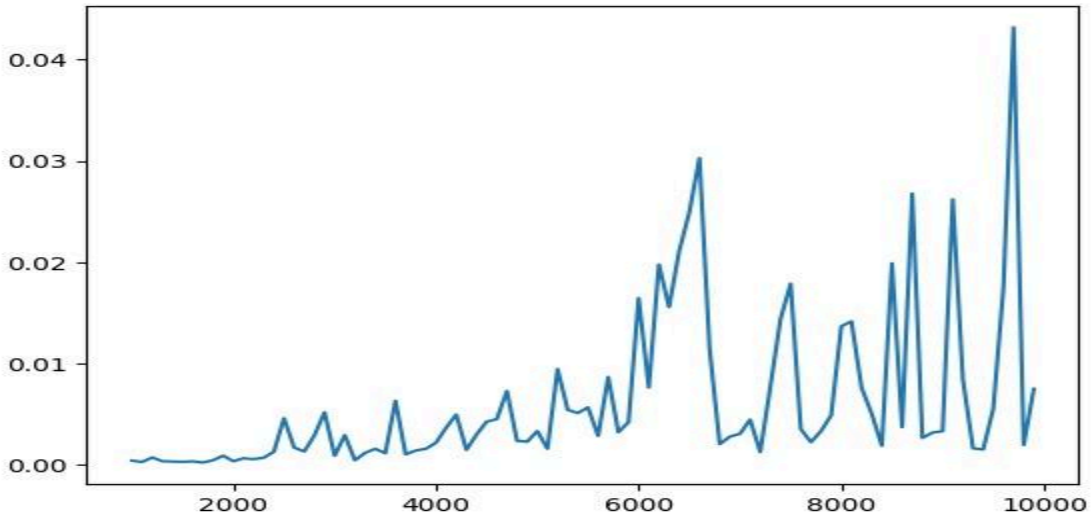
Name: Vatsal Bhuva
Roll No. : IIT2022004
Section: A

Using Merge Sort to search for the kth smallest element:

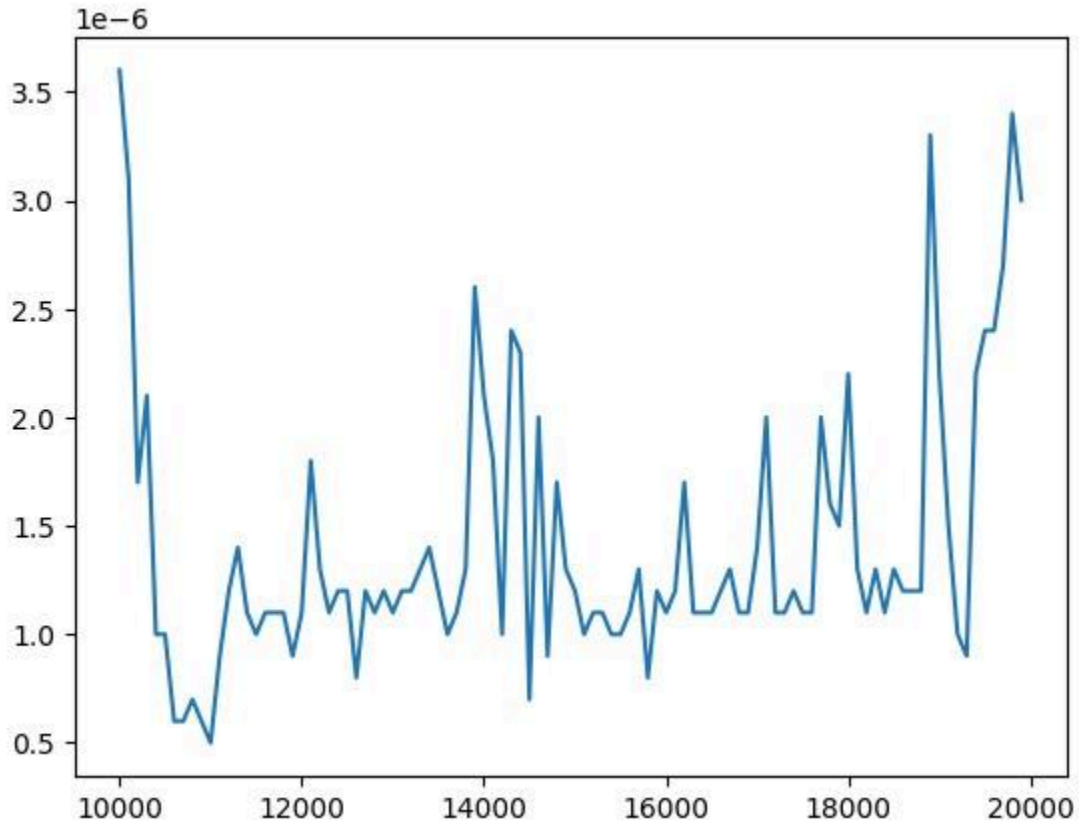
```c
#include <stdio.h>
#include <stdlib.h>

void merge(int arr[], int left[], int leftSize, int right[], int rightSize,int z) {
    int i = 0, j = 0, k = 0;

    if(leftSize+rightSize==5){
        int a=0,b=0;
        for (int i = 0; i < z; ++i)
        {
            if(left[a]<right[b]){
                arr[a+b]=left[a];
                a++;
            }
            if(right[b]<left[a]){
                arr[a+b]=right[b];
                b++;
            }

        }
        return;
    }

    while (i < leftSize && j < rightSize) {
        if (left[i] <= right[j]) {
            arr[k++] = left[i++];
        } else {
            arr[k++] = right[j++];
        }
```

```c
    merge_sort(left, mid,k);
    merge_sort(right, n - mid,k);

    merge(arr, left, mid, right, n - mid,k);

    free(left);
    free(right);
}

int find_kth_smallest_merge_sort(int arr[], int n, int k) {
    merge_sort(arr, n,k);
    return arr[k - 1];
}

int main() {
    int arr[] = {12, 3, 5, 7, 19};
    int n = sizeof(arr) / sizeof(arr[0]);
    int k = 3;

    printf("The %d-th smallest element is: %d\n", k, find_kth_smallest_merge_sort(arr, n, k)

    return 0;
}
```

Graph-1:



Graph-2:



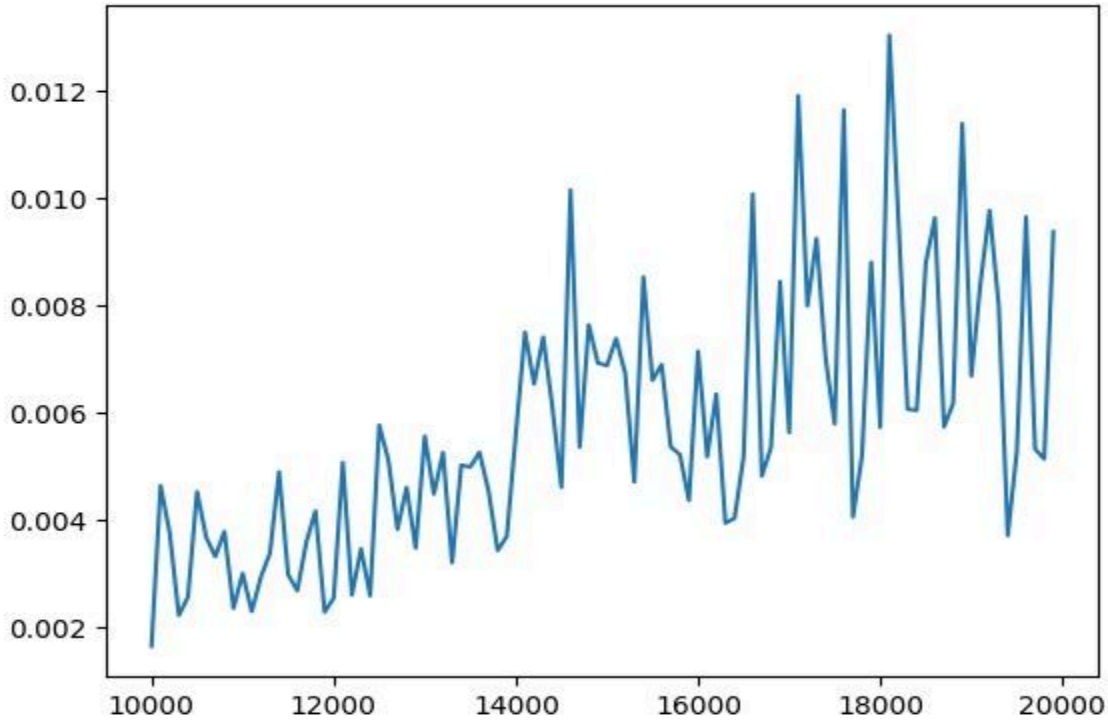
Using Quick Sort for the kth smallest element:

The logic is relatively simple, we implement normal quick sort and break when the kth index gets sorted (i.e., the partition index = k)

```cpp
#include <bits/stdc++.h>
using namespace std;

int merge(vector<int> &arr, int l, int r)
{
    int pivot = arr[r];
    int i = l - 1;
    for (int j = l; j < r; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[r]);
    return i + 1;
}

void quickSort(vector<int> &arr, int l, int r, int k)
{
    if (l < r)
    {
        int pi = merge(arr, l, r);
        if (pi == k)
        {
            cout << k << "th smallest element: " << arr[pi] << endl;
            return;
        }
        quickSort(arr, l, pi - 1, k);
        quickSort(arr, pi + 1, r, k);
    }
}
```

```cpp
int main()
{
    int n;
    cout << "Enter size of array: ";
    cin >> n;
    vector<int> arr;
    for (int i = 0; i < n; i++)
    {
        int temp;
        cin >> temp;
        arr.push_back(temp);
    }
    cout << "Enter k (kth smallest element): ";
    int k;
    cin >> k;
    quickSort(arr, 0, n, k);

    return 0;
}
```

Graph-1:



Graph-2: