

CS 222

ASSIGNMENT 1: SORTING

Name – Gautam Kumar Mahar

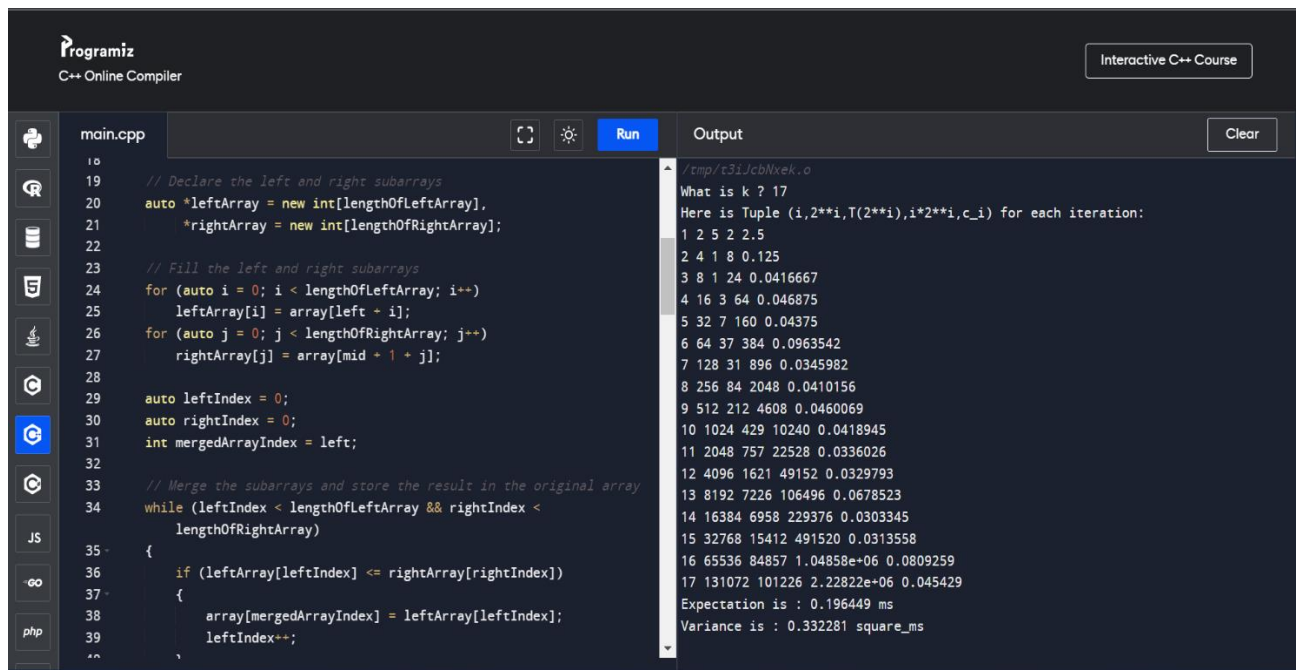
Roll No. – 2103114

Branch – Computer Science Engineering

QUESTION 1 – MERGE SORT

- Because $T(n)$ has a relatively low value, there are large variations in the proportion $c_i = T(n) / n \log(n)$ for the first few examples. The values of c_i then continue to fall until the computational capacity of our computer or laptop is met, at which point the significance of c_i 's increases exponentially.

Here is the Output –



The screenshot shows the Programiz C++ Online Compiler interface. The code in `main.cpp` implements a Merge Sort algorithm. The output window displays the results of the program execution, including a prompt for a value `k`, a tuple of values for each iteration, and the final expectation and variance values.

```
18
19 // Declare the left and right subarrays
20 auto *leftArray = new int[lengthOfLeftArray];
21 *rightArray = new int[lengthOfRightArray];
22
23 // Fill the left and right subarrays
24 for (auto i = 0; i < lengthOfLeftArray; i++)
25     leftArray[i] = array[left + i];
26 for (auto j = 0; j < lengthOfRightArray; j++)
27     rightArray[j] = array[mid + 1 + j];
28
29 auto leftIndex = 0;
30 auto rightIndex = 0;
31 int mergedArrayIndex = left;
32
33 // Merge the subarrays and store the result in the original array
34 while (leftIndex < lengthOfLeftArray && rightIndex <
    lengthOfRightArray)
35 {
36     if (leftArray[leftIndex] <= rightArray[rightIndex])
37     {
38         array[mergedArrayIndex] = leftArray[leftIndex];
39         leftIndex++;
40     }
```

Output:

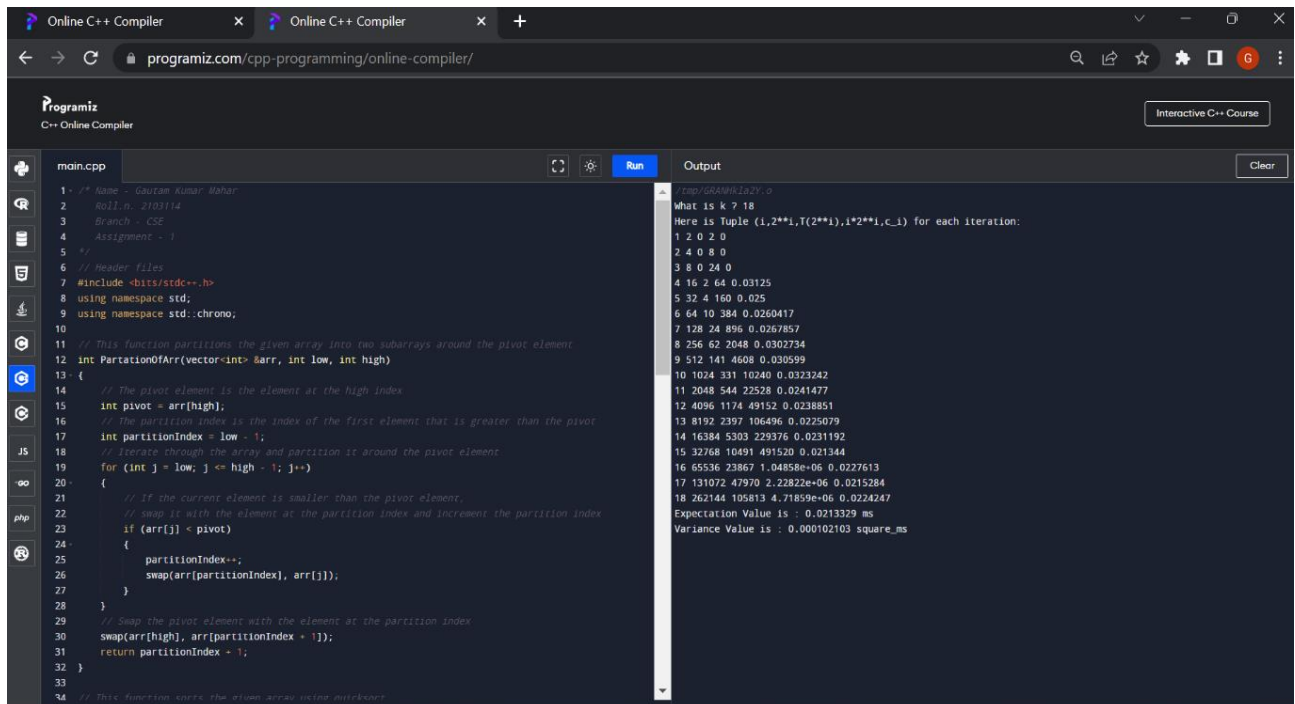
```
/tmp/c31jcbNxeK.o
What is k ? 17
Here is Tuple (i,2**i,T(2**i),i*2**i,c_i) for each iteration:
1 2 5 2 2.5
2 4 1 8 0.125
3 8 1 24 0.0416667
4 16 3 64 0.046875
5 32 7 160 0.04375
6 64 37 384 0.0963542
7 128 31 896 0.0345982
8 256 84 2048 0.0410156
9 512 212 4608 0.0460069
10 1024 429 10240 0.0418945
11 2048 757 22528 0.0336026
12 4096 1621 49152 0.0329793
13 8192 7226 106496 0.0678523
14 16384 6958 229376 0.0303345
15 32768 15412 491520 0.0313558
16 65536 84857 1.04858e+06 0.0809259
17 131072 101226 2.22822e+06 0.045429
Expectation is : 0.196449 ms
Variance is : 0.332281 square_ms
```

QUESTION 2 – QUICK SORT

$T(n)$: Time of execution

- Since the rapid sort's execution time $T(n)$ is so brief in this situation, the fraction $c_i = T(n) / n \log(n)$ is initially quite small. The value of the C_i typically ranges between 1 and 2. Additionally, it varies from compiler to compiler.

Here is the Output –



```
main.cpp
1 /* Name - Gautam Kumar Mahar
2 Roll no. 2103114
3 Branch - CSE
4 Assignment - 1
5 */
6 // Header files
7 #include <bits/stdc++.h>
8 using namespace std;
9 using namespace std::chrono;
10
11 // This function partitions the given array into two subarrays around the pivot element
12 int PartitionOfArr(vector<int> &arr, int low, int high)
13 {
14     // The pivot element is the element at the high index
15     int pivot = arr[high];
16     // The partition index is the index of the first element that is greater than the pivot
17     int partitionIndex = low - 1;
18     // Iterate through the array and partition it around the pivot element
19     for (int j = low; j <= high - 1; j++)
20     {
21         // If the current element is smaller than the pivot element,
22         // swap it with the element at the partition index and increment the partition index
23         if (arr[j] < pivot)
24         {
25             partitionIndex++;
26             swap(arr[partitionIndex], arr[j]);
27         }
28     }
29     // Swap the pivot element with the element at the partition index
30     swap(arr[high], arr[partitionIndex + 1]);
31     return partitionIndex + 1;
32 }
33
34 // This function sorts the given array using quicksort.
```

```
Output
/tmp/GRAHMLa2V.o
what is k ? 18
Here is Tuple (1,2**1,T(2**1),1*2**1,C_i) for each iteration:
1 2 0 2 0
2 4 0 8 0
3 8 0 24 0
4 16 2 64 0.03125
5 32 4 160 0.025
6 64 10 384 0.0260417
7 128 24 896 0.0267857
8 256 62 2048 0.0302734
9 512 141 4608 0.030599
10 1024 331 10240 0.0323242
11 2048 544 22528 0.0241477
12 4096 1174 49152 0.0238851
13 8192 2397 106496 0.025079
14 16384 5303 229376 0.0231192
15 32768 10491 491520 0.021344
16 65536 23867 1.04858e+06 0.0227613
17 131072 47970 2.22822e+06 0.0215284
18 262144 105813 4.71859e+06 0.0224247
Expectation Value is : 0.0213329 ms
Variance Value is : 0.000102103 square_ms
```

- THANK YOU -