Programming Assignment 3

109550127 宋哲頤

1.

Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves.

2.

Time Complexity :O(n*logn)

Reason:

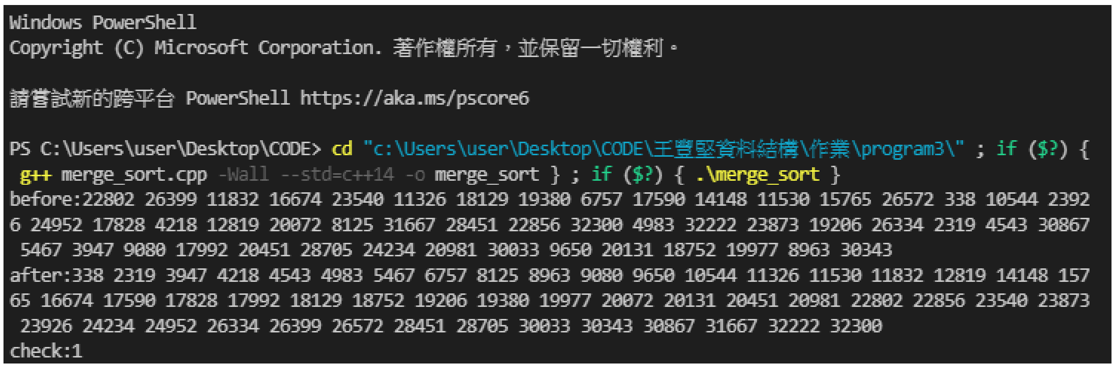
As we have already learned in Binary Search that whenever we divide a number into half in every step, it can be represented using a logarithmic function, which is log n and the number of steps can be represented by log n + 1(at most)

Also, we perform a single step operation to find out the middle of any subarray, i.e. O (1).

And to merge the subarrays, made by dividing the original array of n elements, a running time of O(n) will be required.

Hence the total time for MergeSort function will become n(log n + 1), which gives us a time complexity of O(n*log n).

3.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。

請嘗試新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\user\Desktop\CODE> cd "c:\Users\user\Desktop\CODE\王豐堅資料結構\作業\program3\" ; if ($?) {
 g++ merge_sort.cpp -Wall --std=c++14 -o merge_sort } ; if ($?) { .\merge_sort }
before:22802 26399 11832 16674 23540 11326 18129 19380 6757 17590 14148 11530 15765 26572 338 10544 2392
6 24952 17828 4218 12819 20072 8125 31667 28451 22856 32300 4983 32222 23873 19206 26334 2319 4543 30867
 5467 3947 9080 17992 20451 28705 24234 20981 30033 9650 20131 18752 19977 8963 30343
after:338 2319 3947 4218 4543 4983 5467 6757 8125 8963 9080 9650 10544 11326 11530 11832 12819 14148 157
65 16674 17590 17828 17992 18129 18752 19206 19380 19977 20072 20131 20451 20981 22802 22856 23540 23873
 23926 24234 24952 26334 26399 26572 28451 28705 30033 30343 30867 31667 32222 32300
check:1
```

4.

```
MergeSort(arr[], l,   r)
If r > l
    (1) Find the middle point to divide the array into two halves:
            middle m = (l+r)/2
    (2) Call mergeSort for first half:
            Call mergeSort(arr, l, m)
    (3) Call mergeSort for second half:
            Call mergeSort(arr, m+1, r)
    (4) Merge the two halves sorted in step 2 and 3:
            Call merge(arr, l, m, r)
```

Consider the following example of an unsorted array, which we are going to sort with the help of the Merge Sort algorithm.

A= (36,25,40,2,7,80,15)

Step1: The merge sort algorithm iteratively divides an array into equal halves until we achieve an atomic value. In case if there are an odd number of elements in an array, then one of the halves will have more elements than the other half.

Step2: After dividing an array into two subarrays, we will notice that it did not hamper the order of elements as they were in the original array. After now, we will further divide these two arrays into other halves.

Step3: Again, we will divide these arrays until we achieve an atomic value, i.e., a value that cannot be further divided.

Step4: Next, we will merge them back in the same way as they were broken down.

Step5: For each list, we will first compare the element and then combine them to form a new sorted list.

Step6: In the next iteration, we will compare the lists of two data values and merge them back into a list of found data values, all placed in a sorted manner.

Hence the array is sorted.