Assignment 6 – Sorting Algorithms

TA: Henry(**asd56565656@gmail.com**)

Deadline: **Dec. 31, 11:59pm**

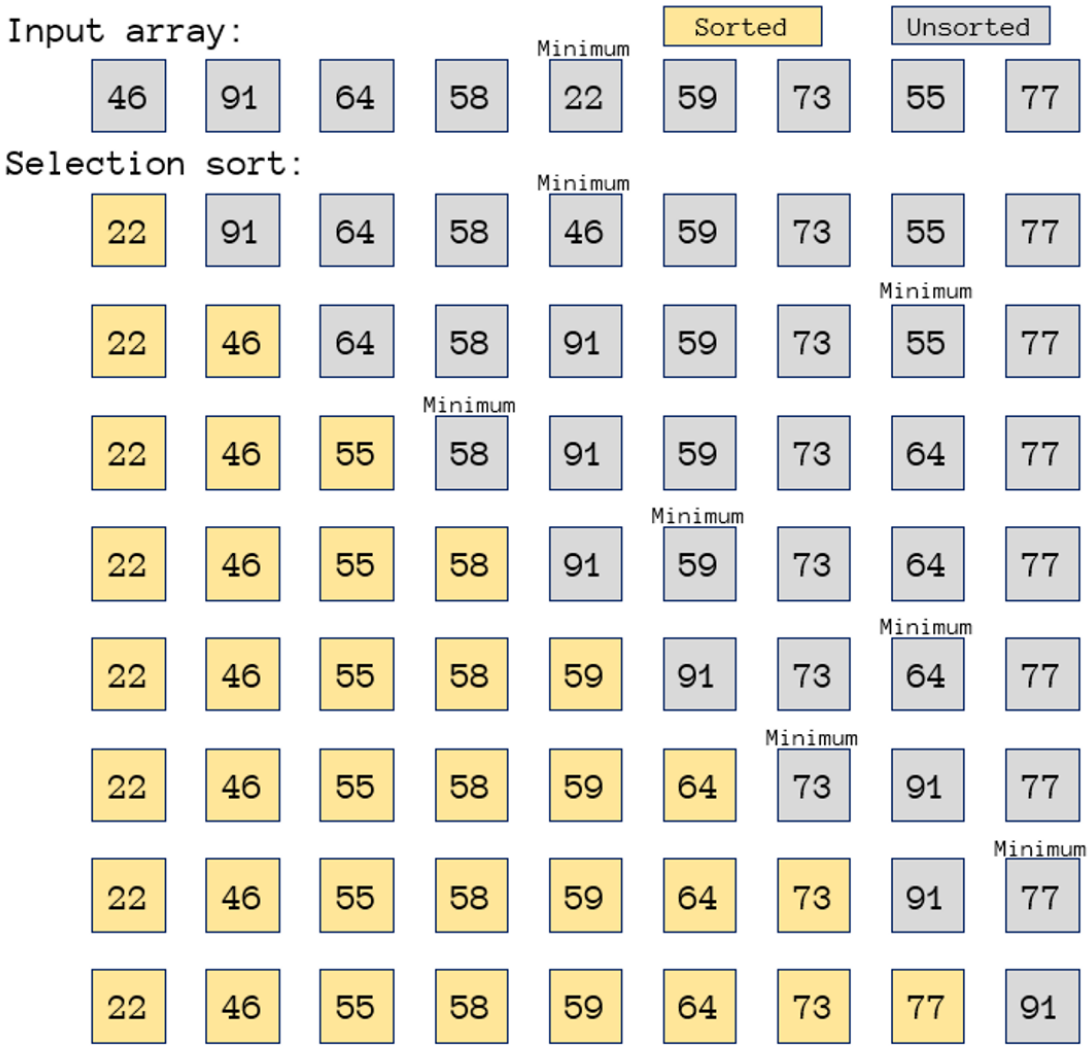
1. Implementation of sorting algorithms

Sorting is a common and important problem in programming. To understand the different sorting algorithms, you are asked to implement four sorting algorithms for this assignment.

The four sorting algorithms separately is **Selection Sort**, **Insertion Sort**, **Merge Sort** and **Bucket Sort**. The description of each sorting algorithm is shown below.

1.1 Selection Sort 25%

Please write a program that can read data from a standard input (stdin). In this program, you are asked to read multiple input data from a given file. The selection sort algorithm searches the whole array for the smallest element. First, the data is divided into two parts, sorted and unsorted. Suppose you want to put n data from small to large, and the n data at the beginning is "unsorted". At this time, the minimum value of n "unsorted" data is taken out and placed at the end of the "sorted" data, that is, it is exchanged with the first unsorted number, and the array result is printed, and so on. Until there is no "unsorted" data, the procedure stops. The format of the output is as follows.



Listing 1.1: Selection Sort

Input：

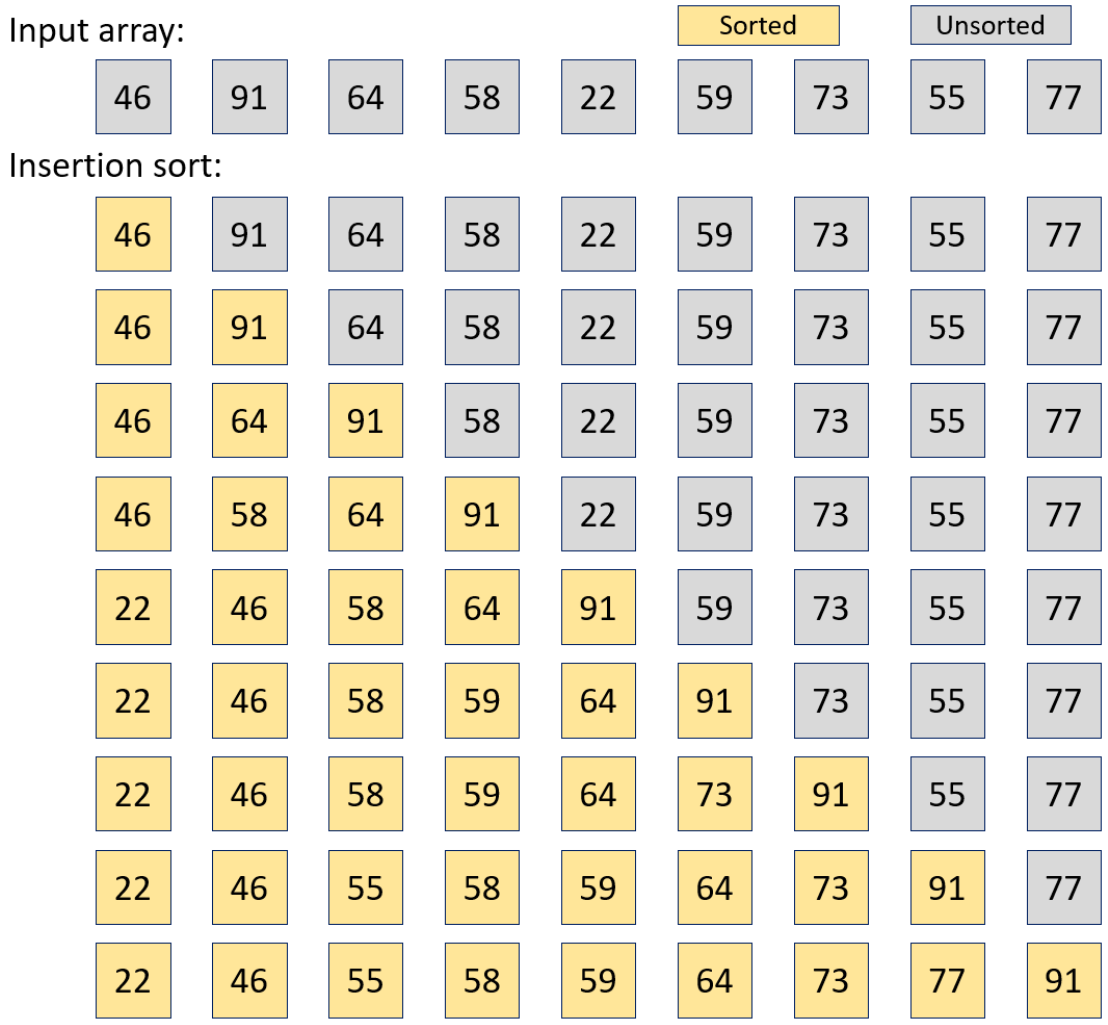
46, 91, 64, 58, 22, 59, 73, 55, 77

Output：

22, 91, 64, 58, 46, 59, 73, 55, 77
22, 46, 64, 58, 91, 59, 73, 55, 77
22, 46, 55, 58, 91, 59, 73, 64, 77
22, 46, 55, 58, 91, 59, 73, 64, 77
22, 46, 55, 58, 59, 91, 73, 64, 77
22, 46, 55, 58, 59, 64, 73, 91, 77
22, 46, 55, 58, 59, 64, 73, 91, 77
22, 46, 55, 58, 59, 64, 73, 77, 91

1.2 Insertion Sort 25%

Please write a program that can read data from a standard input (stdin). In this program, you are asked to read multiple input data from a given file. Insertion sort divides the data into sorted and unsorted parts. The procedure starts from taking an element from the top of the unsorted array and check the sorted array from the back to the front until the value is not larger than the element and insert this value. Repeat the above procedure until all the unsorted numbers are all processed. The format of the output is as follows.



Listing 1.2: Insertion Sort

```
Input：
46, 91, 64, 58, 22, 59, 73, 55, 77
Output：
46, 91, 64, 58, 22, 59, 73, 55, 77
46, 91, 64, 58, 22, 59, 73, 55, 77
46, 64, 91, 58, 22, 59, 73, 55, 77
46, 58, 64, 91, 22, 59, 73, 55, 77
22, 46, 58, 64, 91, 59, 73, 55, 77
22, 46, 58, 59, 64, 91, 73, 55, 77
22, 46, 58, 59, 64, 73, 91, 55, 77
22, 46, 55, 58, 59, 64, 73, 91, 77
22, 46, 55, 58, 59, 64, 73, 77, 91
```

1.3 Merge Sort 25%

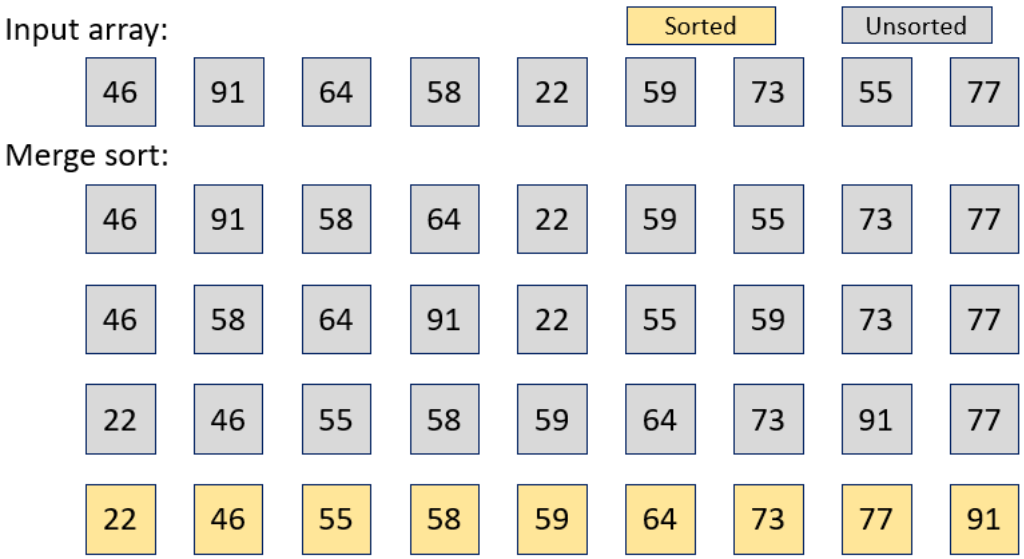
Please write a program that can read data from a standard input (stdin). In this program, you are asked to read multiple input data from a given file. Merge Sort is a divide and conquer algorithm which divides input array in two halves, calls itself for the two halves, and then merges the two sorted halves. The algorithm works as follows.

Divide：

Divide the unsorted list into n sublists until each sublist only contains one element (a list of one element is considered sorted).

Conquer：

Repeatedly merge sublists to produce new sorted sublists until there is only one sublist remaining. This will be the sorted list.



Listing 1.3: Merge Sort

Input：

46, 91, 64, 58, 22, 59, 73, 55, 77

Output：

46, 91, 58, 64, 22, 59, 55, 73, 77

46, 58, 64, 91, 22, 55, 59, 73, 77

22, 46, 55, 58, 59, 64, 73, 91, 77

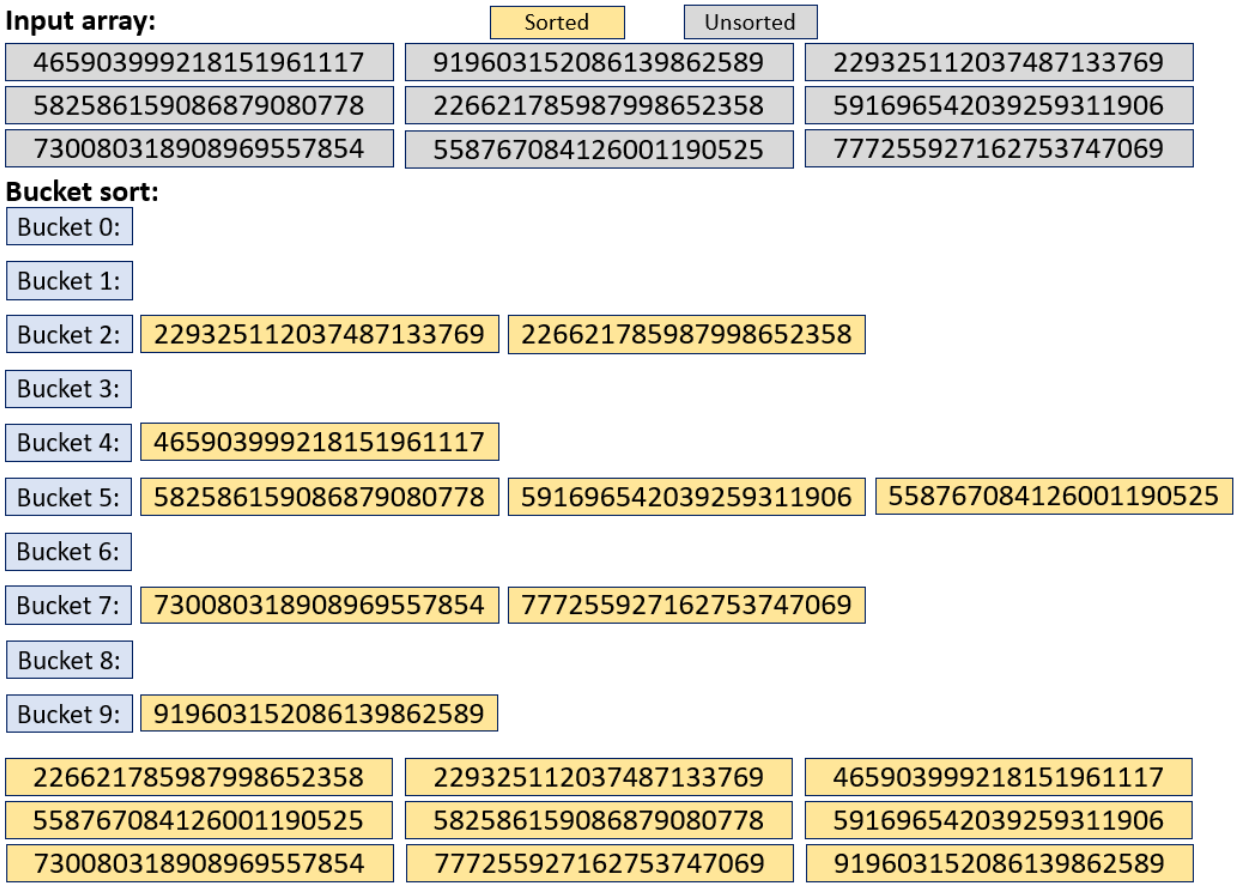
22, 46, 55, 58, 59, 64, 73, 77, 91

1.4 Bucket Sort 20%

Please write a program that can read data from a standard input (stdin). In this program, you are asked to read multiple input data from a given file. Bucket sort is a sorting algorithm that works by distributing the elements of an input array into a number of buckets. Each bucket using a different sorting algorithm to sort the list.

Bucket sort works as follows:

1. Read the **21-digit bignum** data from the standard input (stdin).
2. Set up an array for initially empty "buckets".
3. Go over the original array, and put each number in its bucket by using **MSB**.
4. Use **Insertion Sort** to sort the numbers in each non-empty bucket.
5. Visit the buckets in order and put all elements back into the original array.



Listing 1.4: Bucket Sort

Input：

46590399921815196117, 91960315208613986258, 22932511203748713376, 58258615908687908077, 22662178598799865235, 59169654203925931190, 73008031890896955785, 55876708412600119052, 77725592716275374706

Output：

Bucket 0:

Bucket 1:

Bucket 2: 22932511203748713376, 22662178598799865235

Bucket 3:

Bucket 4: 46590399921815196117

Bucket 5: 58258615908687908077, 59169654203925931190, 55876708412600119052

Bucket 6:

Bucket 7: 73008031890896955785, 77725592716275374706

Bucket 8:

Bucket 9: 91960315208613986258

22662178598799865235, 22932511203748713376, 46590399921815196117, 55876708412600119052, 58258615908687908077, 59169654203925931190, 73008031890896955785, 77725592716275374706, 91960315208613986258

2. Submission

To submit your files electronically, login DomJudge website through the following url :

https://108-1-ds-judge.ate.cs.ccu.edu.tw/

Press the submit button and choose the homework questions you want to submit. After submitting your code, DomJudge will give you a result to tell you whether your code is correct or not. However, during the demo time, your code will be evaluated by different sets of test cases. Please make sure your code can work correctly based on the description above. Additionally, you must compress your code and the README file into a zip file and upload it to Ecourse2.

3. Output, readme, comments and style (5%)

The TA(s) will mark and give points according to the following grading policies:

- 25 % **Selection sort**
- 25 % **Insertion sort**
- 25% **Merge sort**
- 20% **Bucket sort**
- 5% **Readme file, code style, and comments in source code**

Readme file should include your name, class ID, a brief description of the code, and other issues students think that will be helpful for the TAs to understand their homework.