# Practice #13

## Sorting
## (Insertion, Quick, Merge, and Heap Sort)

Yunmin Go

School of CSEE

HANDONG GLOBAL UNIVERSITY

# Practice #13 TO-DO List

| To-Do | Submission | Notes |
|---|---|---|
| Sort Class (Insertion, Quick, Merge, Heap Sort) | Screenshot and source code (All files including Sort.cpp) | p.19, 22, 33, 35, 40, 41, Chapter 7 |

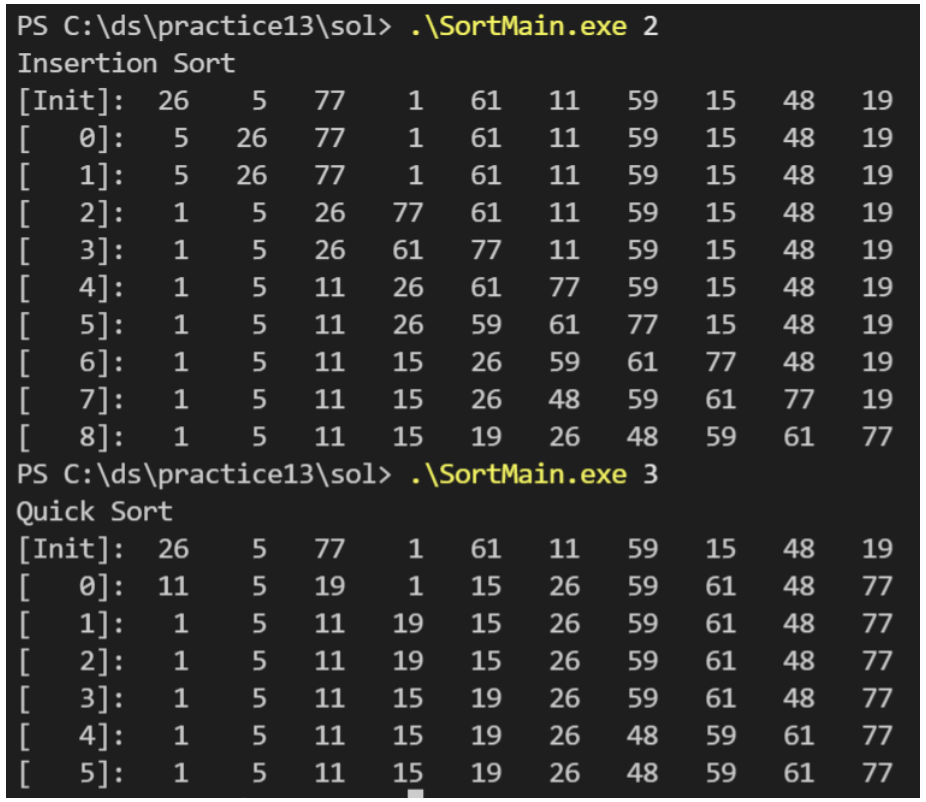
- Upload your screenshot and source codes on LMS by 11pm on 5/26 (Wed).
  - All your screenshots should be merged in one pdf file, screenshot.pdf.
  - Your pdf and all source codes should be compressed into zip file.
- File name: practice13_Your Student ID_Name.zip (only zip, not pdf, docx, c, etc)
  - ex) practice13_20400022_고윤민.zip

## Sorting

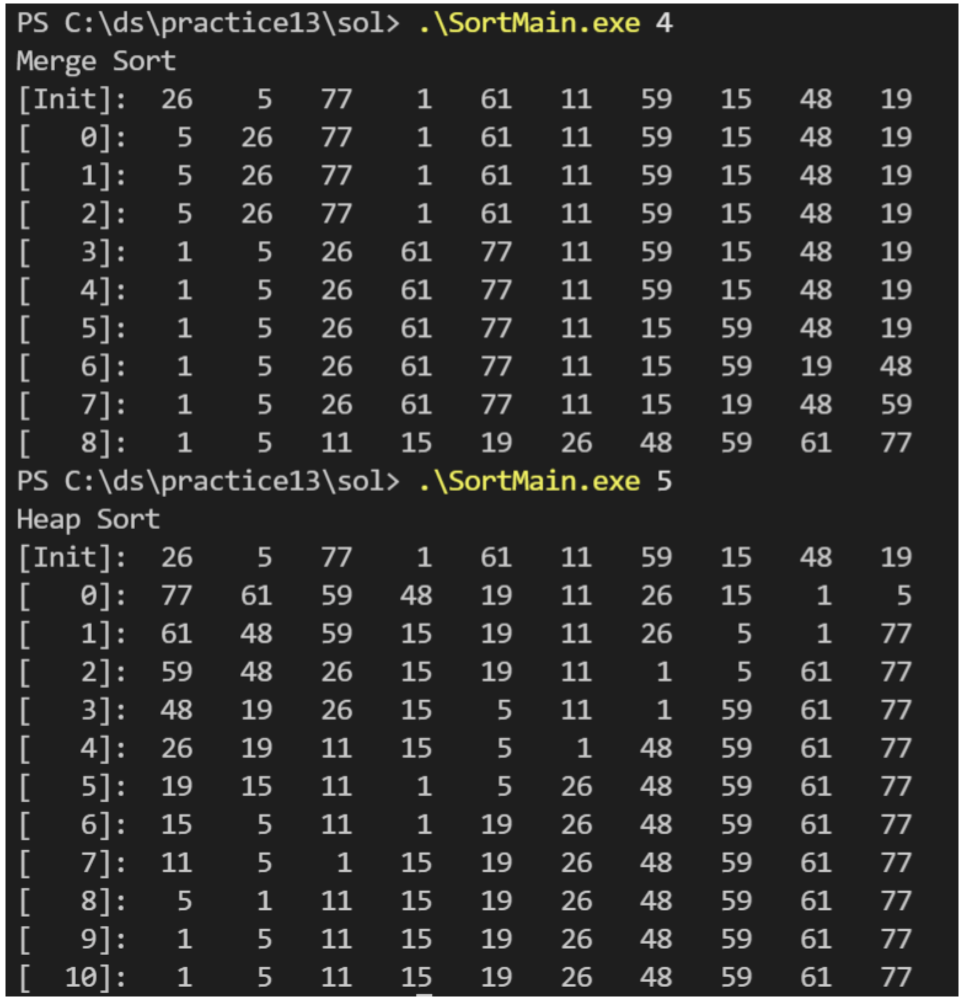
- Implement a Sort class and Dijkstra’s algorithm
  - Complete a Sort.cpp (SortMain.cpp: no need to change)
  - Refer to p.19, 22, 33, 35, 40, 41, Chapter 7
  - You can select a sorting algorithm using command arguments
    - Usage: SortMain.exe <Sorting Alg=0~5>
    - Sorting Alg: 0=Selection, 1:Bubble, 2:Insertion, 3:Quick, 4:Merge, 5:Heap
  - Implement following member functions.
    - InsertionSort(): Insertion sort
    - QuickSort(int, int): Quick sort
    - MergeSort(Element [], int, int), merge(Element [], int, int, in): Merge sort
    - HeapSort() and adjust(int, int): Heap sort

# Sorting

- Expected results



```
PS C:\ds\practice13\sol> .\SortMain.exe 2
Insertion Sort
[Init]:  26    5   77    1   61   11   59   15   48   19
[   0]:   5   26   77    1   61   11   59   15   48   19
[   1]:   5   26   77    1   61   11   59   15   48   19
[   2]:   1    5   26   77   61   11   59   15   48   19
[   3]:   1    5   26   61   77   11   59   15   48   19
[   4]:   1    5   11   26   61   77   59   15   48   19
[   5]:   1    5   11   26   59   61   77   15   48   19
[   6]:   1    5   11   15   26   59   61   77   48   19
[   7]:   1    5   11   15   26   48   59   61   77   19
[   8]:   1    5   11   15   19   26   48   59   61   77
PS C:\ds\practice13\sol> .\SortMain.exe 3
Quick Sort
[Init]:  26    5   77    1   61   11   59   15   48   19
[   0]:  11    5   19    1   15   26   59   61   48   77
[   1]:   1    5   11   19   15   26   59   61   48   77
[   2]:   1    5   11   19   15   26   59   61   48   77
[   3]:   1    5   11   15   19   26   59   61   48   77
[   4]:   1    5   11   15   19   26   48   59   61   77
[   5]:   1    5   11   15   19   26   48   59   61   77
```



```
PS C:\ds\practice13\sol> .\SortMain.exe 4
Merge Sort
[Init]:  26    5   77    1   61   11   59   15   48   19
[   0]:   5   26   77    1   61   11   59   15   48   19
[   1]:   5   26   77    1   61   11   59   15   48   19
[   2]:   5   26   77    1   61   11   59   15   48   19
[   3]:   1    5   26   61   77   11   59   15   48   19
[   4]:   1    5   26   61   77   11   59   15   48   19
[   5]:   1    5   26   61   77   11   15   59   48   19
[   6]:   1    5   26   61   77   11   15   59   19   48
[   7]:   1    5   26   61   77   11   15   19   48   59
[   8]:   1    5   11   15   19   26   48   59   61   77
PS C:\ds\practice13\sol> .\SortMain.exe 5
Heap Sort
[Init]:  26    5   77    1   61   11   59   15   48   19
[   0]:  77   61   59   48   19   11   26   15    1    5
[   1]:  61   48   59   15   19   11   26    5    1   77
[   2]:  59   48   26   15   19   11    1    5   61   77
[   3]:  48   19   26   15    5   11    1   59   61   77
[   4]:  26   19   11   15    5    1   48   59   61   77
[   5]:  19   15   11    1    5   26   48   59   61   77
[   6]:  15    5   11    1   19   26   48   59   61   77
[   7]:  11    5    1   15   19   26   48   59   61   77
[   8]:   5    1   11   15   19   26   48   59   61   77
[   9]:   1    5   11   15   19   26   48   59   61   77
[  10]:   1    5   11   15   19   26   48   59   61   77
```

HANDONG GLOBAL UNIVERSITY

## Heap Sort

- Array index begins from 0

```
void Sort::HeapSort()
{
    for (i = (num-1)/2; i >= 0; i--)
    {
        adjust(i, num);
    }
    Print(cur++);

    for (i = num - 1; i >= 0; i--)
    {
        swap(&list[0], &list[i]);
        adjust(0, i);
        Print(cur++);
    }
}
```

```
void Sort::adjust(int root, int n)
{
    int child, rootkey;
    Element temp = list[root];
    rootkey = list[root].key;
    child = 2 * root + 1;
    while (child <= n-1)
    {
        if ((child < n-1) && (list[child].key < list[child+1].key))
            child++;
        if (rootkey > list[child].key)
        {
            break;
        }
        else
        {
            list[(child-1)/2] = list[child];
            child = child * 2 + 1;
        }
    }
    list[(child-1)/2] = temp;
}
```