

Assignment 7 Technical Report

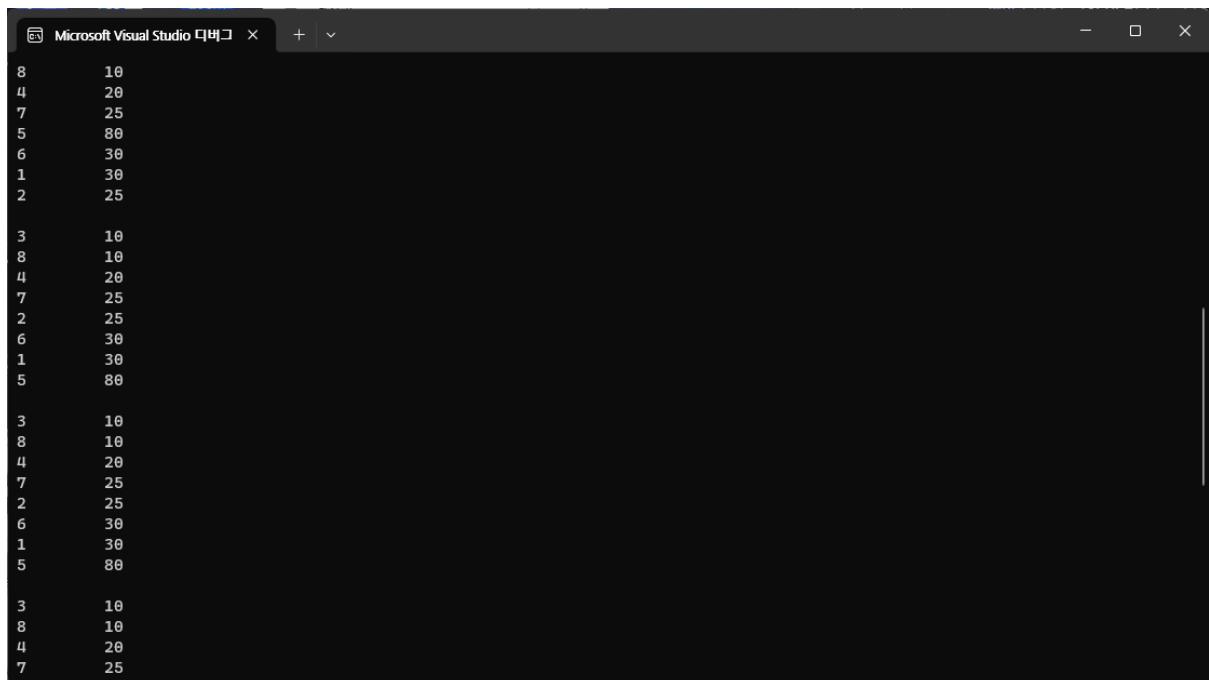
2171039 Chaewon Lee

Code 1

This code sorts the given array with standard selection sort and stable selection sort, and prints the result in the console to see whether the sort is done well.

First, the given structure named data has two fields : id and score. In main function, the code sets the value of data.

Then, in selection_sort and selection_sort_stable function, both sorts the given array with selection sort, but it is slightly different. Since in selection_sort function, it does not check whether the element in the front has the bigger id or not. To be more in detail, we can see this example.



```
Microsoft Visual Studio 디버그 콘솔
8      10
4      20
7      25
5      80
6      30
1      30
2      25
3      10
8      10
4      20
7      25
2      25
6      30
1      30
5      80
3      10
8      10
4      20
7      25
2      25
6      30
1      30
5      80
3      10
8      10
4      20
7      25
```

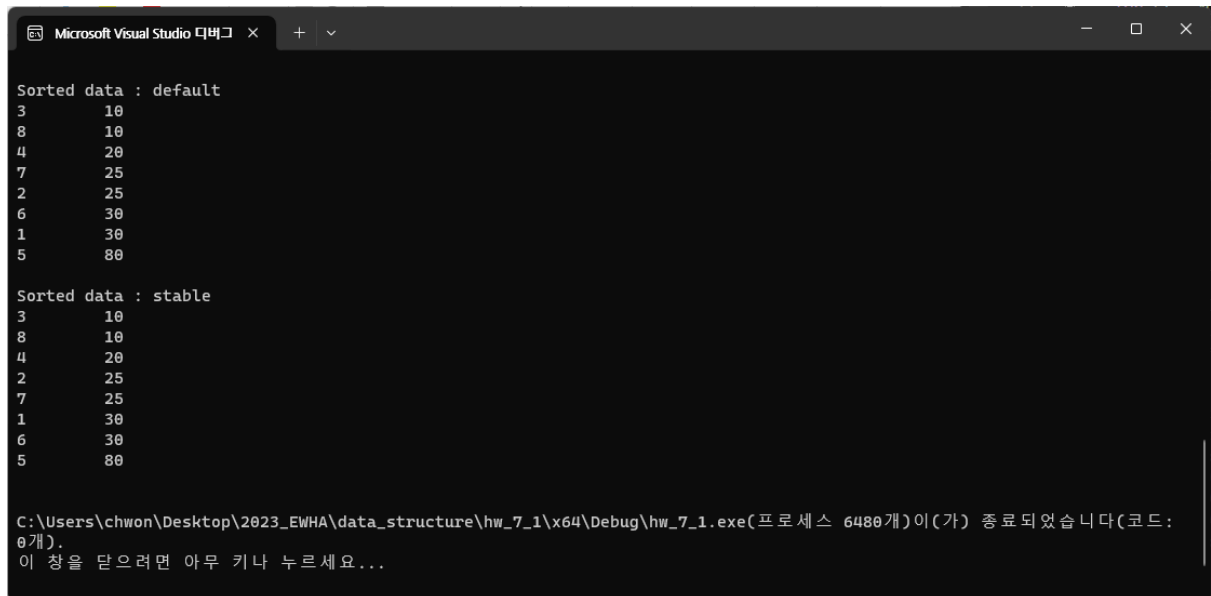
The element 7 and 2 both have same data, 25. But when 2 is moved to the front by sorting, it does not check the element in the front since it suppose that every element in the front is sorted well. However, because 2 came to the front later than 7, because of swapping occurred with element 8, it makes 2 to be placed behind the 7.

Therefore, to fix this problem, we should first check the element in the front has the same data, and if it does, we should check its id, and if the moving is needed, we operate swap function.

The function selection_sort_stable is the modified version of selection_sort function applying the changes I said before. After placing the least element in the array, it checks whether the index of that element is bigger than 0 (since if it is 0, there is no

element in the front, therefore it cause an error), and if the data of front element and that element is same, and finally, the id of front element is bigger or not. If it is bigger, we should swap two elements to set the result stable.

- Result



```
Sorted data : default
3      10
8      10
4      20
7      25
2      25
6      30
1      30
5      80

Sorted data : stable
3      10
8      10
4      20
2      25
7      25
1      30
6      30
5      80

C:\Users\chwon\Desktop\2023_EWHA\data_structure\hw_7_1\x64\Debug\hw_7_1.exe(프로세스 6480개)이(가) 종료되었습니다(코드:
0개).
이 창을 닫으려면 아무 키나 누르세요...
```

You can see the result of both sorting, and with stable sort function, the stably sorted result is given.

Code 3

This code generates random 24 bit integers and sorts them with radix sort. The radix sort used in this code is implemented with counting sort, not queue.

First, there is randomNum function, which generates random 24 bits integers. This first generates 4 elements of 6 bits integers, and then merges them in one number by shifting each numbers. It is same with this : to construct a 3-digit-decimal, first picking three numbers : ex) 5, 3, 4, and combining them by multiplying the value of each digit : ex) $5*100 + 3*10 + 4*1$.

I generated number in this way, since the rand function only can generate number smaller than 32767.

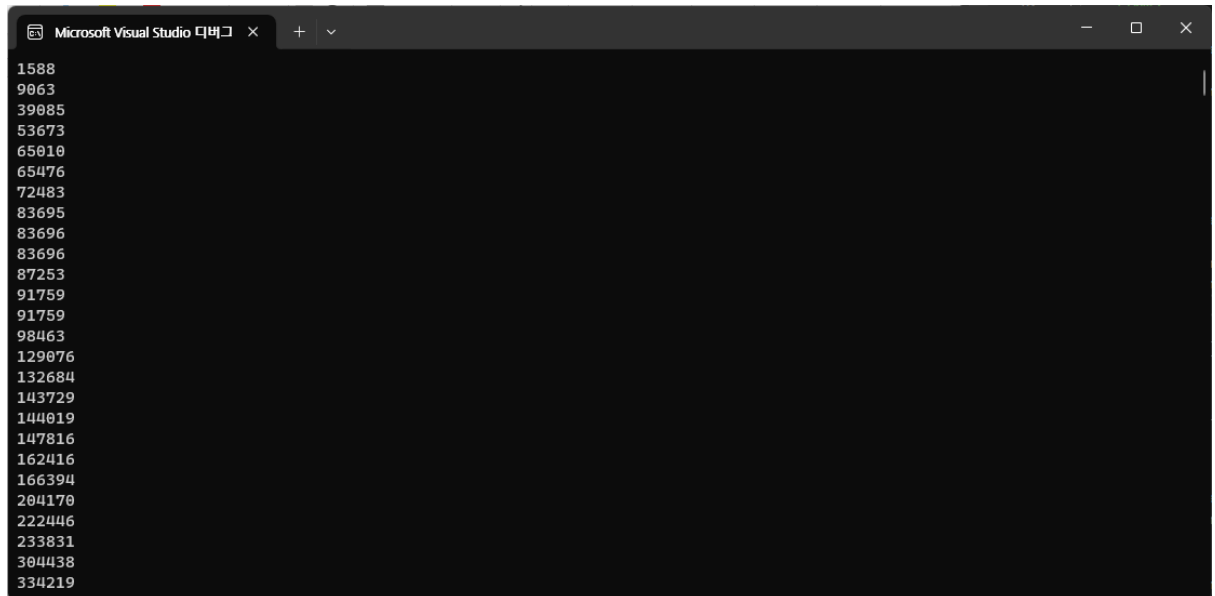
After generating all 1000 numbers with randomNum function, the code sorts the number from lsb to msb. The code checks 0~5 digit, 6~11 digit, 12~17 digit and 18~23 digit in order. To get the number of that digit, the code uses bit operator. First it shifts right until the smallest bit we want is placed in the rightmost side of the number. Then to erase all numbers in front of that smallest 6 bits, the code operates & operation with 63, which is 0b111111. It masks all digits to 0 without the part we want to get. Then we can get the number of that digit we want to find.

Then, the code uses the method of counting sort. It first counts how many times that

digit number comes out. Then, from the reverse order, the function checks the digit of the number, and places that number in the position where is 1 smaller than the count. And it decreases the count, since one element is used. By repeating this operation, we can get the result that is sorted by the digit number.

Since there are 4 digits, the code does this operation 4 times, and prints the result to see the function is working well.

Result



```
Microsoft Visual Studio 디버그 x + -
```

```
1588  
9063  
39085  
53673  
65010  
65476  
72483  
83695  
83696  
83696  
87253  
91759  
91759  
98463  
129076  
132684  
143729  
144019  
147816  
162416  
166394  
204170  
222446  
233831  
304438  
334219
```

You can see all numbers are sorted well.