National University of Singapore
School of Continuing and Lifelong Education
TIC1002: Introduction to Computing and Programming II
Semester II, 2020/2021

Tutorial 3
Complexity and Sorting

1. [Not very complex complexity] Give the big-O for following code fragments

a.
```
for (int i = 0; i < n; i++)
   for (int j = 0; j < n; j++)
        <2 operations>
```
$\sum = 2n^2 \Rightarrow O(n^2)$

b.
```
for (int i = 0; i < n; i++)
    for (int j = 0; j < 2; j++)
         <5 operations>
```
$\sum = 10n \Rightarrow O(n)$

c.
```
for (int i = 0; i < n; i++)
      for (int j = n - 1; j >= i; j--)
           <4 operations>
```
$\sum = 1+2+3+\cdots+n = (n+1)\times\frac{n}{2} = O(n^2)$

d.
```
i = 1;
j = 0;
while (j < n ){
    if (i % n == 0){
       j++;
       <1 operation>
    }
    i++;
}
```
$n \times \{1\cdots n-1,\ n\cdots 2n-1,\ 2n\cdots,\ \vdots,\ n\times(n-1)\}$ (i) $\times \{0, 1, 2, \vdots, n-1\}$ (j) $\Rightarrow O(n^2)$

2. [Selection Sort] Trace the working of selection sort on the following array. You can use the given table to show the changes after each outer-loop iteration. Indicate clearly the largest item, the location of the largest item for each iteration.

56	12	34	19	18	79	25	31
56	12	34	19	18	31	25	79
25	12	34	19	18	31	56	79
25	12	31	19	18	34	56	79
25	12	18	19	31	34	56	79
19	12	18	25	31	34	56	79
18	12	19	25	31	34	56	79
12	18	19	25	31	34	56	79

max=31→56→79
max=79→25→56
max=56→31→34
max=34→18→31
max=31→19→25
max=25→18→19
max=19→12→18

3. [Bubble Sort Version 3.0] Let us see how bubble sort can be further improved.
    a. [What's the issue?] Try sorting an array like {2, 3, 4, 5, 1}. How many outer-loop iteration do we need? Identify the issue with the standard bubble sort algorithm.

    array is almost sorted, but have to iterate through anyways.

    b. [Solve the issue] Solve the issue posed by (a). Hint: It is like bubble sort with a twist....

    cocktail sort : In for loop:
    // 1 iteration from left → right , swap
    // 1 iteration from right → left , swap

    {2,3,4,1,5}
    {1,2,3,4,5}

    c. [Analyzing the change] Did we improve the big-O of bubble sort?

    worst case : O(n²)
    best case : O(n)
    (when almost sorted)

4. [Sorting is general] For simplicity, sorting is almost always taught using an integer array. However, it should be clear that the sorting algorithms can be easily generalized. Let us take the **insertion sort** code as a case study in this question.

    a. [What to change?] Identify all necessary changes for the insertion sort code if we need to sort a different type of array (e.g. an array of student records / double values / strings etc). Whenever possible, focus more on the higher level requirement (**"what kind of operation is needed?**") rather than low level details ("**how do I write this in C++?**")

    b. [Actual change] Using your findings in (a), change the ***insertion sort*** to work on an array of **fraction** structure as defined below:

```
struct Fraction {
    int num, den;
};
```

    Note: You should avoid converting the fractions into a floating point values for comparison.

    Note$_2$: Use the provided ***Q4-Template.cpp*** to actually code it out!