Samip Shah
samipjanish.shah@mavs.uta.edu
1001709873
CSE5311

(1) Using Figure 8.2 as a model, illustrate the operation of COUNTING-SORT on the array A = <6; 0; 2; 0; 1; 3; 4; 6; 1; 3; 2>.

We have C = <2, 4, 6, 8, 9, 9, 11>
Running iterations on B

B						2					
B						2		3			
B				1		2		3			
B				1		2		3			6
B				1		2		3	4		6
B				1		2	3	3	4		6
B			1	1		2	3	3	4		6
B		0	1	1		2	3	3	4		6
B		0	1	1	2	2	3	3	4		6
B	0	0	1	1	2	2	3	3	4		6
B	0	0	1	1	2	2	3	3	4	6	6

(2) Suppose that we were to rewrite the for-loop header in line 10 of the COUNTINGSORT
As 10 for j D 1 to A.length. Show that the algorithm still works properly. Is the modified algorithm stable?

For example, there are two elements a1 and a2 where a1=a2 and a1 is before a2 in the input array A. In the original algorithm, if a2 is put into the output array B at position i, i.e., put a2 into B[i], then a1 will be put before that i.e. into B [i – 1]. But in the modified algorithm, a2 is in B [i – 1] and a1 is in B[i]. They are reversed in the output. For the reversing is ok because a1 = a2. So, the modified algorithm works properly.
From the above example, we know that the modified algorithm will not be stable. It will work properly, but they are not stable. Equal elements will appear in reverse order in the sorted array

(3) Show how to sort n integers in the range 0 to n^3 - 1 in O(n) time.

Using Radix Sort with base n. Now each digit will be in log n^3 and it will require 3 epochs. Further we only need to run 3 epochs and so for n possible values total time will be O(n).

(3) Show that the second smallest of n elements can be found with n + [lg n] - 2 comparisons in the worst case.

The second smallest can be found in (n-1) comparisons where at least one out of two elements move to next comparison. After log n rounds only the smallest elements remains. So, for second smallest element we have to do log n -1 comparisons
For Example,

1 2 5 6 7 9	So for 6 elements we get 3 elements as second smallest.
1 5 7	No of comparisons is (6-1) + (3-1) = 7
1	So for 6 elements smallest element is 1

Reference: https://sites.math.rutgers.edu
https://walkccc.github.io/CLRS/
https://github.com/gzc/CLRS

7 No of calculations is (6-1) + (1-1) = 6

(4) In the algorithm SELECT, the input elements are divided into groups of 5. Will the algorithm work in linear time if they are divided into groups of 7? Argue that SELECT does not run in linear time if groups of 3 are used.

Yes, it will work in linear time as median of median is less than less than 4 elements from the group. So it is around 4n/14 elements. So, we can say that it is
T(n) =< T(n/7) =< T(10n/14) =< O(n). Thus, it will work in linear time.
Now if the group of 3 is used, we have
T(n) = T([n/3]) + T(4n/6) + O(n) >= T(n/3) + T(2n/3) + O(n). so we have >= cn lg n. Thus it will grow faster than linear.

(5) Exercise 9.3-8 on Page 223 [20 points]

```
if n = =1 then
        return min( X[1], Y [1] )
end if
if X[n/2] < Y [n/2] then
        return Median (X[n/2 + 1...n], Y [1...n/2], n/2)
else
        if X[n/2] >= Y [n/2] then
                return Median(X[1...n/2], Y [n/2 + 1...n], n/2)
        end if
end if
```

Reference: https://sites.math.rutgers.edu
https://walkccc.github.io/CLRS/
https://github.com/gzc/CLRS