CS301, Spring 2019 **Homework 2 (March 18, 2019)**

Problem 1 (Order statistics) **Cavit Cakir**

(a) I can sort the numbers using merge sort. It will take $(nlogn)$ worst case time. Take out k largest elements to the array which can be done in (k). Worst case running time is $(nlogn + k) = (nlogn)$ (i $\leq n$).

Recurrence relation for worst case: $T(N) = 2T(N/2) + N$ $a > 1$ and $b > 1$ so we can use master theorem, It is case 2, since Logba is 1 and f(n) = n So it is $(nlogn)$

(b) I can use **Selection Algorithm** which is an algorithm to find the kth smallest number in a list which will be in (n), partition around kth number in (n) time. After that we have to sort a array which is k sized. If we use merge sort which we used in part a it will be in $(klogk)$. All computations will be in $(n + klogk)$.

Recurrence relation for Selection Algorithm; T(n) SELECT(i, n) 1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

2. Recursively SELECT the median x of the n/5 group medians to be the pivot.

3. Partition around the pivot x. Let k = rank(x).

```
4.if $i = k$ then return x
  else if $i < k$
    then recursively SELECT the $ith$ smallest element in the lower part
    else recursively SELECT the $(ik)th$ smallest element in the upper part
```

$1 = (n)$

$2 = T(n/5)$

$3 = (n)$

$4 = T(3n/4)$

if we add them up we will get $T(n) = T(3n/4) + T(n/5) + (n)$ which is (n)

I would prefer **method b** according to asymptotic runtime complexity. Only in worst case they will both run in same asymptotic complexity but in all other cases method b will be better. It will take at most $(n + klogk)$ where k $\leq n.$

Problem 2 (Linear-time sorting) **Cavit Cakir**

(a) I modified the line when we allocate an array for counting as count = [0] * (27) because when we are using this count array for integers it was base (characters + 1) so i created it as (26+1) (I assume alphabet as uppercase english letters) Also we were adding 0s at beginning but now we are dealing with strings so we can not add 0's. I modified it as adding ' ' (blanks) to end.

(b)

1 - EGE , SELIN , YASIN , VEYSEL	index = 5
2 - EGE , VEYSEL , SELIN , YASIN	index = 4
3 - EGE , SELIN , YASIN , VEYSEL	index = 3
4 - EGE , SELIN , YASIN , VEYSEL	index = 2
5 - YASIN , SELIN , VEYSEL , EGE	index = 1
6 - EGE , SELIN , VEYSEL , YASIN	index = 0

(c) When we were dealing with integers, our running time was O(nk) where n input size, k is longest digit number but now we are sorting strings. So our algorithms will do same computations for strings, then asymptotic runtime complexity will be O(nl) where l is lenght of the longest string.