CompSci 260P Project #1

1. Pranav Udupa Shankaranarayana, UCINet ID: pudupash, Student ID no.: 65085145
2. Abhijeet Suresh Bhute, UCINet ID: abhute, Student ID no.: 81004607

1. Analysis for theoretical Worst Case and Average Case

The algorithm used has two parts.

1. Insert first K elements
2. Add next N-K elements

Let's look at them individually.

1. Insert first K elements

```
  // Insert first k items into Best in a reverse sorted fashion.
for (item = 1; item <= K; item++) {
    int insert_index = bin_ins(0, item - 1, item, Best);
    insert_val_and_shift_right(insert_index, item, Best, K);
}
```

Here, we iterate through first K elements, and insert them at the right position in our partially-sorted array. Since we're using binary search to find an index to insert at, the average and maximum number of comparisons is log(K). Since we're iterating through K elements, the number of comparisons would be:

WORST CASE = AVERAGE CASE = $\boldsymbol{K * \log(K)}$

2. Add next N-K elements

```
// Iterate through all the items and insert to Best if
// the item is bigger than the smallest Best element.
for (item = K + 1; item <= n; item++) {
    if (COMPARE(Best[K - 1], item) == 2) {
        int insert_index = bin_ins(0, K - 1, item, Best);
        insert_val_and_shift_right(insert_index, item, Best, K);
    }
}
```

Here, we initially check if an element is bigger than the smallest best element. Since there are N-K elements, we would have N-K such checks for both average case and worst case. In addition, if we need to insert, we shall do a binary search for an index to insert at, which will cost $\log(K)$ per insertion for both worst and average case. Hence the number of comparisons required to execute this part would be:

$$(N-K) + c*(N-K)*\log(K)$$

Where c is the probability where the check is True. In the best case, the check is never true. In the worst case, the check is always true (c=1). i.e. The input array is in sorted fashion.

WORST CASE = $(\boldsymbol{N}-\boldsymbol{K})+(\boldsymbol{N}-\boldsymbol{K}) * \mathbf{log}(\boldsymbol{K})$

In average case, it is hard to determine a value for c. As the algorithm progresses, the probability of finding an item bigger than the smallest in Best array reduces since the Best array will now be populated with relatively bigger elements than in the beginning. Let's assume $c(\text{probability constant}) = 0.5$ to arrive at an average case.

AVERAGE CASE = $(\boldsymbol{N}-\boldsymbol{K})+\mathbf{0.5} * (\boldsymbol{N}-\boldsymbol{K}) * \mathbf{log}(\boldsymbol{K})$

Hence, the combined number of comparisons for a given N, K will be as follows:

WORST CASE

$$\boldsymbol{K} * \boldsymbol{log}(\boldsymbol{K})+(\boldsymbol{N}-\boldsymbol{K})+(\boldsymbol{N}-\boldsymbol{K}) * \mathbf{log}(\mathbf{K})$$

AVERAGE CASE

$$\boldsymbol{K} * \boldsymbol{log}(\boldsymbol{K})+(\boldsymbol{N}-\boldsymbol{K})+\mathbf{0.5} * (\boldsymbol{N}-\boldsymbol{K}) * \mathbf{log}(\boldsymbol{K})$$

2. Analysis for theoretical expected Worst Case

After the initial insertion of K elements in Best array, in the worst case, all the remaining N-K elements will always be greater than the smallest(last) element in Best array. This is only possible if the input Numbers array is sorted in ascending manner. However, the probability of the input array being sorted is very low. We can thus formulate the number of comparisons in theoretical worst case as

$$\boldsymbol{K} * \boldsymbol{log}(\boldsymbol{K})+(\boldsymbol{N}-\boldsymbol{K})+\boldsymbol{c} * (\boldsymbol{N}-\boldsymbol{K}) * \mathbf{log}(\mathbf{K})$$

Here the probability constant *c* will be smaller than that in theoretical worst case but greater than that in average case. ie) 0.5 < *c* < 1.

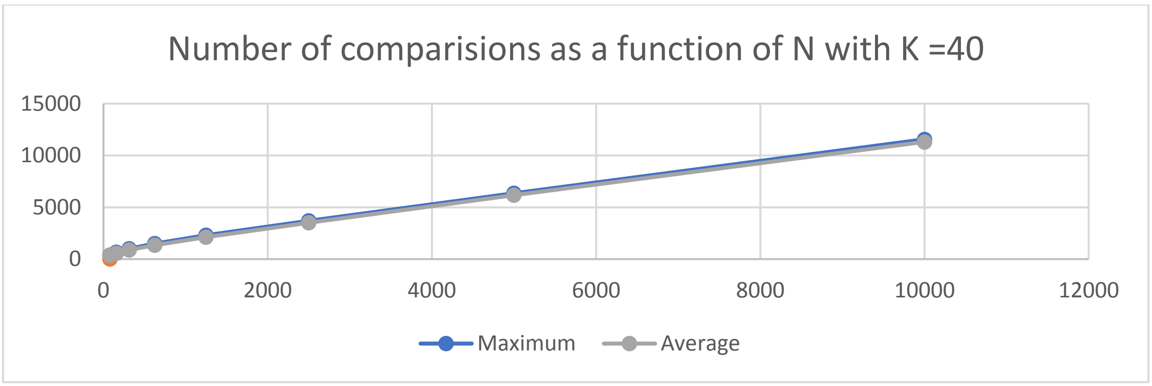
3. Analysis for observed Worst Case and Average Case

WORST CASE: $\boldsymbol{N}+\mathbf{0.5} * (\boldsymbol{N}-\boldsymbol{K}) * \mathbf{log}(\boldsymbol{K})+\boldsymbol{K} * \mathbf{log}\,(\mathbf{K})-\boldsymbol{K}$

AVERAGE CASE: $\boldsymbol{N}+\mathbf{0.25} * (\boldsymbol{N}-\boldsymbol{K}) * \mathbf{log}(\boldsymbol{K})+\boldsymbol{K} * \mathbf{log}\,(\mathbf{K})-\boldsymbol{K}$

Let's use a constant K = 40 and change the value of N, the observed values are as follows.

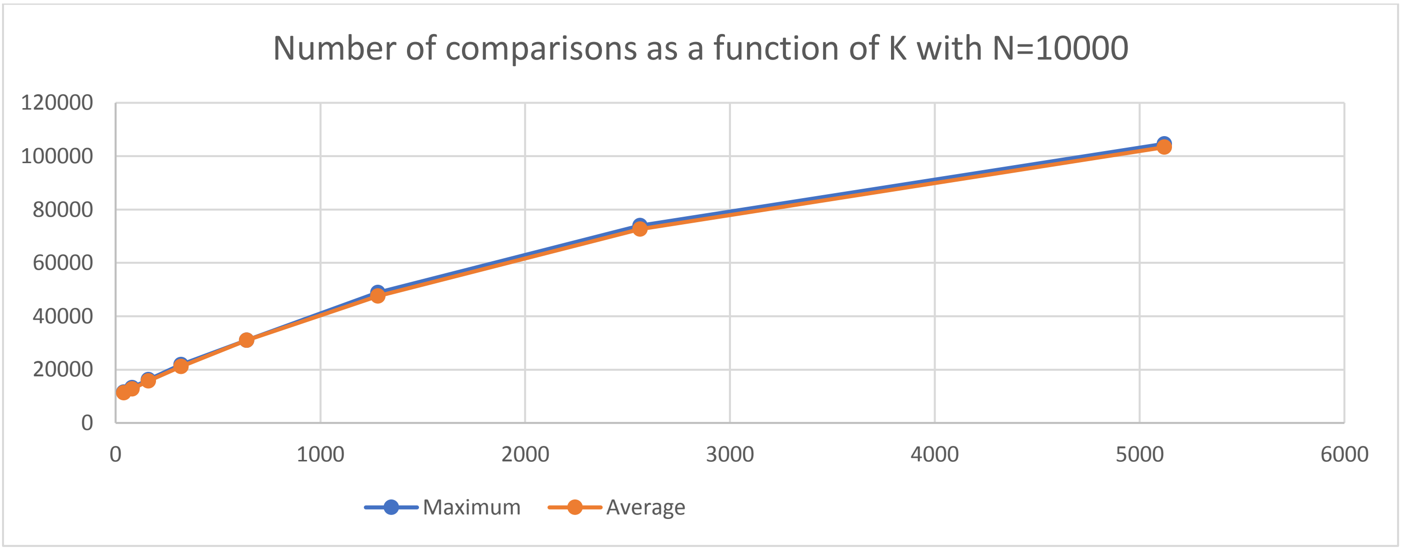
N	Maximum	Average
78	395	342.78
156	664	571.44
312	997	876.37
625	1484	1336.51
1250	2311	2112.21
2500	3687	3512.57
5000	6351	6161
10000	11553	11309



It is clear that, the number of comparisons for both worst/average case grows almost linearly to N when K is constant. We can approximately equate that,

$\therefore Number\ of\ comparisons = N + c1(N - c2) + c3$. Since c1, c2 and c3 are not constants, they could be derived from K.
Now, let's keep N as constant, i.e. $N = 10000$. The observed values are:

K	Maximum	Average	Max/N	Average/N
40	11553	11309.59	1.1553	1.130959
80	13167	12783.83	1.3167	1.278383
160	16240	15685.47	1.624	1.568547
320	21909	21145.73	2.1909	2.114573
640	30972	30972.24	3.0972	3.097224
1280	48864	47520.36	4.8864	4.752036
2560	73968	72618.28	7.3968	7.261828
5120	104628	103349.11	10.4628	10.334911



Here, it is clear that as K approaches to N, the number of comparisons increases significantly. For small values of K, its contribution towards the total number of comparisons is small compared to N. From our previous equation, we find that $\boldsymbol{N + c * (N - K) * \log(K) + K * \log(K) - K}$ is a very close match to arrive at the number of comparisons.
Since there is no other input variable, c must be derived from how the input data is distributed.

WORST CASE

In the worst case, c is maximum. Looking at the observations, c ranges from **0.03** to **0.5**. We could estimate that, at the worst possible case, c = 1.

Hence, our worst possible number of comparisons could be $\boldsymbol{N + (N-K) * \log(K) + K * \log(\mathrm{K}) - K}$
However, using empirical evidence as the only source of truth, worst possible number of comparisons would be

$$\boldsymbol{N - K + 0.5 * (N-K) * \log(K) + K * \log(\mathrm{K})}$$

AVERAGE CASE

In the average case in our empirical data, we see that $c \approx 0.25$. Hence, average number of comparisons can be assumed as follows.

$$\boldsymbol{N - K + 0.25 * (N-K) * \log(K) + K * \log(\mathrm{K})}$$

4. Discrepancies between Empirical observation and Theoretical prediction

To summarize, let’s look at Worst case and Average case individually.

WORST CASE

Empirical: $\boldsymbol{K * \log(\mathrm{K}) + (N-K) + 0.5 * (N-K) * \log(K)}$
Theoretical: $\boldsymbol{K * log(K) + (N-K) + (N-K) * \log(K)}$

It is clear that the constants used for these two approaches are different. Theoretically, we will arrive at c=1 when the input array is sorted in increasing order. However, empirically the probability of arriving at a randomized sorted array for N=10000 is extremely low. Hence, the empirical evidence has a lower value for c than theoretically possible. Thus, the reduced number of comparisons in our empirical observation.

AVERAGE CASE

Empirical: $\boldsymbol{K * \log(\mathrm{K}) + (N-K) + 0.25 * (N-K) * \log(K)}$
Theoretical: $\boldsymbol{K * log(K) + (N-K) + 0.5 * (N-K) * \log(\mathrm{K})}$

Theoretically it is hard to calculate the average probability constant c, since c can change based on N, K and the input distribution. We arrived at c=0.5 by taking the average of c=0 (Best case) and c=1 (Worst case). Practically, the probability of finding an element bigger than the smallest Best decreases as we progress in our iterations. It decreases further when K is much smaller than N and the input array is in almost descending order. Hence, empirically c is much lesser. Hence the reduced number of comparisons in our empirical observation.