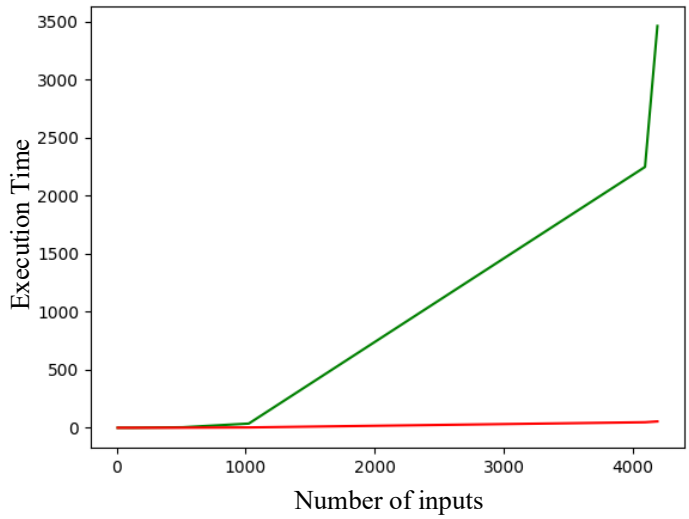
Question 1:




| Line | Legend |
|---|---|
| | $O(n^3)$: Naïve Implementation |
| | $O(n^2 \log n)$: Sophisticated Implementation |

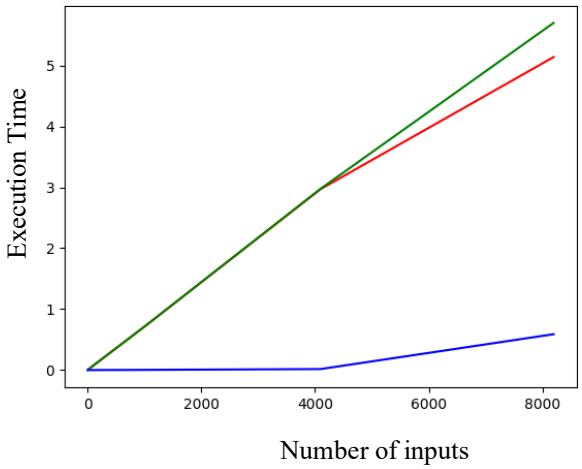
NOTE: The above plot has been generated in Python using 'Matplotlib' library

**Observation**: Due to improper scale selection (happens by default in the library I have used for plotting python), the graph is somewhat skewed. Due to thus, the actual shape of the plot can't be seen.

**Analysis:** Here, somewhere up to 1000 number of inputs, both the algorithms almost take about the same time. Thus, for data sets containing about 1000 inputs, both the algorithms perform about the same.
Once the number of inputs is increased beyond 1000, a drastic change in computation time is seen. The 'Sophisticated' algorithm takes drastically less time as compared to 'Naïve' algorithm.
It can also be seen that once the number of inputs exceeds beyond 4000, the computation time of 'Naïve' implementation increases manifold, whereas that for 'Sophisticated' algorithm still remains very low.

While the code was getting executed, the initial .txt files i.e. 8int.txt", "32int.txt", "128int.txt", "512int.txt", "1024int.txt" were done with very quickly; whereas "4096int.txt" took a comparatively greater time. However, files "4192int.txt", "8192int.txt" took considerable amount of time (in 'Naïve' implementation).

Question 2:




| Line | Legend |
|---|---|
| | Quick Find |
| | Quick Union |
| | Quick Union with Weight Balancing |

NOTE: The above plot has been generated in Python using 'Matplotlib' library

**Observation**: Due to improper scale selection (happens by default in the library I have used for plotting python), the graph is somewhat skewed. Due to thus, the actual shape of the plot can't be seen.

**Analysis:** In Quick Find algorithm, the execution time is about linear with the number of inputs. The execution time is directly proportional to the number of inputs. The execution time for files having a smaller number of inputs e.g. "128pair.txt" takes lesser amount of time as compared to "8192pair.txt". Thus, the complexity is O(N)

In Quick Union algorithm, the execution time is also about linear with the number of inputs. All the above-mentioned analysis is also valid here. Thus, complexity is O(N).
However, comparing between 'Quick Find' and 'Quick Union', at number of inputs greater than 4000, the 'Quick Union' algorithm takes a smaller execution time.

In 'Quick Union with Weight Balancing' algorithm, the time taken in execution is exceptionally lower as compared to the other two union-find algorithms. Even for large inputs, about the order of 8000, the execution time is fairly low.
For data input size as lower than 4000, the execution time is nearly negligible. Even for input size greater than 4000, it takes fairly less amount of time. The complexity of the same is $O(\log_2 N)$.

Question 3:

Using the recursive equation for the case:
$C_N = C_{N-1} + N$ for where $C_1 = 1$
$= C_{N-2} + (N-1) + N$
$= C_{N-3} + (N-2) + (N-1) + N$
.
.
.
$= 1 + 2 + \ldots + (N-1) + N$
$= N(N+1)/2$
$= (N^2 + N)/2$
Now, this can be written as $O(N^2)$
According to the provided definition,
Thus, $(N^2 + N)/2 < kN^2$
⇨ $(2k-1)N^2-N>0$
⇨ $N[(2k-1)N-1]>0$
⇨ $K>(N+1)/2N$
Thus, $k_{min} = (N+1)/2N$
So, the required 'c' for the question is $(N+1)/2N$

Also, if we do the graph analysis, for question 1, the value of 'c' will be about 900
Similarly, for question 2, the value of 'c' will be about 4000.

(For the graph analysis, I compared the graphs. The point from where the behavior of the graph changes is the required point, according to me.)