Session

Mohamed Emad Mahrous

July 2024

1st Possible Solution

Require: Finding the number of admissible degrees within a given range.

We can do that easily by applying a brute force algorithm.

```
for Every-Query do
  for Every-Value-In-Query do
    for Every-Input-Range do
      SUM
    end for
  end for
end for
```

But solving this problem this way is inefficient. The time complexity of the previous solution is: Time complexity = $O(n^3)$

There are better ways to solve the same problem in $O(n^2)$ and $O(n)$.

One approach is to optimize the **Brute Force** to achieve $O(n^2)$ instead of $O(n^3)$.

2nd Possible Solution

Require: To apply the algorithm this way we need to do the following:

```
for Every-Input-Range do
  Accumulate in the array
end for
```

Figure 1: A Simple Array of the first Test-case

91	92	93	94	95	96	97	98	99
1	2	2	2	1	1	2	1	1

Next, we need to change every value in the array to 1 if it is greater than k; otherwise, change it to 0.

```
for Every-Value-In-Array do
  if a[i] > K then
    a[i] ← 1
  else
    a[i] ← 0
  end if
end for
```

Figure 2: 1st array after the previous operation

91	92	93	94	95	96	97	98	99
0	1	1	1	0	0	1	0	0

After that all we will do is iterate and sum, So the total **Time complexity** = $O(n^2) + O(n)$.
Time complexity = $O(n^2)$.

3rd Possible Solution

Require: For better **Time complexity** we can solve this problem in $O(n)$ by using prefix sum algorithm.

1: And we will do that by make $a[l] + 1$ and $a[r + 1] - 1$ Such : l is the first digit in the query, and r is the second digit in the query.

Figure 3: Array by doing prefix sum this way for example 92:94

91	92	93	94	95	96	97	98	99
0	1	0	0	-1	0	0	0	0
0	1	1	1	0	0	0	0	0

At the end, by doing this little trick we can solve the problem in **Time complexity** $= O(n)$.