Kick Start 2020 - Round A

Analysis: Bundling

We need to maximise the sum of scores of each bundle. Let us consider a bundle and say longest prefix shared by all strings of this bundle is of length X. Now each prefix of length from 1 to X is shared by all of the strings in this bundle. Consider any prefix among these X prefixes, it is counted once in the score of this bundle. Thus the score of a bundle can be defined as number of prefixes that are shared by all of the strings in this bundle. Thus if a prefix is shared by all strings in Y bundles, then it will contribute Y to the total sum of scores.

Now instead of finding the total score, we find the contribution of each prefix in the total score. So for maximising the total score, we would want to maximize the contribution of each prefix in the total score. Let the contribution of each prefix PRE be contibution(PRE). We want to maximize ∑ contribution(PRE) where PRE comprises all possible prefixes of the given strings.

Let us say a prefix P_i is a prefix of S strings. The maximum number of bundles of size **K** formed by S strings is ⌊ S / **K** ⌋. In each of these ⌊ S / **K** ⌋ bundles, prefix P_i will add to their scores. Thus maximum value of contribution(P_i) can be ⌊ S / **K** ⌋. So a prefix P_i which occurs as a prefix of S strings will contribute ⌊ S / **K** ⌋ to the answer.

Let us see if we can achieve this maximum value for all the prefixes. Consider a prefix P of length L. It occurs as a prefix in CNT number of strings. Now consider there are C prefixes of length L + 1 which contain the prefix P as a prefix (P_1, P_2, ...,P_C). And we have stored the number of strings these prefixes are part of as (CNT_1, CNT_2, ...,CNT_C).

Let us say we divided the strings which have prefix P_i into ⌊ (CNT_i / **K**) ⌋ bundles. Now we have CNT_i%**K** strings remaining for each prefix that we need to arrange so that they make a bundle. For each of these remaining strings we cannot have a bundle of size **K** which would have a common prefix of length L + 1 because we have CNT_i%**K** remaining strings for each P_i. So, we can make bundles in any order using the remanining strings. Those bundles will still have a prefix of length L shared among them. Thus we would be left with CNT%**K** number of strings which are not in any bundle when we consider prefix P. We can continue this procedure till we are left with prefix of length 0. We would be able to bundle all the strings at this point because we would have **N** % **K** strings remaining, and as specified in the problem, **N** is divisible by **K**.

The problem is now reduced to finding number of times each prefix occurs in the given strings. Let this number be COUNT. We just need to add ⌊ COUNT / **K** ⌋ to the answer for each prefix.

Test set 1

The length of each string is at most 5. Thus we have total number of prefixes as **N** * 5 and each prefix can be of size at most 5. Store each prefix in a hashmap and increase the count for each prefix. In the end, we just need to add ⌊ (count(i) / **K**) ⌋ for each prefix i. The complexity of the solution would be O(**N** * 5 * 5).

Test set 2

Let the sum of length of all strings over all the test cases be SUM which is 10^6. For the large test set, the length of the string can be very large. So, we can't store all the prefixes in a hashmap. We need to store all the prefixes in an efficient manner along with the number of times they occur in given strings. We can use a data structure trie. The insertion cost would be

equal to sum of length of strings over the test cases which is O(SUM). Then finally we just need to traverse the trie and for each prefix, add its contribution to the answer. Time complexity of the solution would be O(SUM).