Kth Character

Problem Explanation

We are given a string *s* which consists of only lowercase English Letters. For each query q, we are given 3 integers *l*, *r* and *k.* We have to find the kth smallest letter(lexicographically) in the subarray from *l* to *r.*
A letter x is lexicographically smaller than letter y if it comes before y in the dictionary.
Note we are considering all the letters which occur in the subarray from *l* to *r* and not distinct letters.

Approach 1 (Brute force)

For each query, we traverse the string from l to r and store their count (i.e. number of times a particular character occurs) in frequency array.
Now, after traversing the entire subarray we start traversing the frequency array in ascending order of characters. The moment when the total sum of all the counts increases the required count k, that character will be the required character.

However, the time complexity is O(N Q) where N is the length of the string and Q is the number of queries.

This will lead to TLE (Time Limit Exceeded) as the number of queries and length are both of order 10^4.

Approach 2 (Optimised Approach)

A better approach would be to store the frequency of all the characters for every index of the string, i.e. create a 2 Dimensional Arrays of size (N X 26) where N is the size of the string and 26 is the number of lowercase English characters.

Hence, each query would be solved in O(1) because the frequency of all characters *i* for a particular subarray from *l* to *r* can be obtained by (freq[r][i] - freq[l-1][i]).

Code snippet to calculate the frequency of subarray from *l* to *r.*

```
for ( j = 0; j < 26; j++) {
          k -= freq [r][j];
          if ( l-1 >= 0)
             k += freq [l-1][j];
          if(k<=0) break;
 }
```