Digit Strings - Hackerearth

Wednesday, March 25, 2020 8:42 PM

The Problem:

You are given a string S of length N. You are also given an integer K. The string consists of digits from 1−9 only. Determine the number of ways to partition the string S such that each segment value is less than K.
If there is no way to perform partition on the string, then print 0.
Since the answer could be a large, print the answer as 109+7.

Input format

- First line: of input contains a single integer T denoting the number of test cases.
- For each test case:
    - First line: Two space-separated integers N and K
    - Second line: A string S of size N

Output format
Print the required answer.

Constraints
1≤T≤5
1≤N≤10^5
1≤K≤10^18

SAMPLE INPUT

```
2
5 6
34212
2 21
11
```

SAMPLE OUTPUT

```
1
2
```

Explanation
In first testcase, We have only one way to partition i.e 3,4,2,1,2.
In second testcase, we can partition in it two ways: 1,1 and 11

The Code:

```
/* Take 1234. Traverse from the end. Iteration-wise you'll need to check for
* (4) = [4] ( Just a notation for all possibilities )
* (3,[4]), (34) = [34]
* (2,[34]), (23, [4]), (234) = [234]
* (1, [234]), (12, [34]), (123, [4]), (1234) = [1234]
* In iterations when you're moving left, you're just using already calculated
values from the right.
* Memoize them and re-use. That's it.
* One trick here is, we'll need not go to the right extreme everytime.
* We'll only need to go until the value encountered so far is < K. Cause if we go
further,
* we'll only increase the value further. Worst-case scenario, we'll go till 18
digits, cause that's K's
* limit.
*/
#include <stdio.h>
#include <stdlib.h>
#define MOD 1000000007
long long int custom_strtoll(char* arr, int left, int right) {
    int i;
    long long int result = 0;
    for(i=left; i<=right; i++) {
        result = result*10 + (arr[i]-48);
    }
    return result;
}
int main(){
    int caseCount, len, i, j;
    long long int K;
    char *eptr;
    scanf("%d", &caseCount);

    char* string = (char*)malloc(100000 * sizeof(char));
    int* substrMax = (int*)malloc(100000 * sizeof(int));
    while(caseCount > 0) {
        scanf("%d %lld", &len, &K);
        scanf("%s", string);
        for(i=0; i<100000; i++)
            substrMax[i] = 0;
        for(i=len-1; i>=0; i--) {

            for(j=i; j<len; j++) {

                if(custom_strtoll(string, i, j) < K ) {
                    if(j == (len-1)) {
                        substrMax[i] = (substrMax[i] + 1) % MOD;
                    } else {
                        substrMax[i] = (substrMax[i] + substrMax[j+1]) % MOD;
                    }
                } else {
                    break;
                }
            }
        }
        printf("%d\n", substrMax[0]);
        caseCount--;
    }
}
```

The Stats:

Score	30.0
Time (sec)	0.50852
Memory (KiB)	64
Language	C