Number of subsequences in a string divisible by n

Given a string consisting of digits 0-9, count the number of subsequences in it divisible by m.

**Examples:**

Input : str = "1234", n = 4

Output : 4

The subsequences 4, 12, 24 and 124 are

divisible by 4.

Input : str = "330", n = 6

Output : 4

The subsequences 30, 30, 330 and 0 are

divisible by n.

Input : str = "676", n = 6

Output : 3

The subsequences 6, 6 and 66

[**Recommended: Please solve it on “*PRACTICE* ” first, before moving on to the solution.**](https://practice.geeksforgeeks.org/problems/number-of-subsequences-in-a-string-divisible-by-n/0)

This problem can be recursively defined. Let remainder of a string with value x be ‘r’ when divided with n. Adding one more character to this string changes its remainder to (r\*10 + newdigit) % n. For every new character, we have two choices, either add it in all current subsequences or ignore it. Thus, we have an optimal substructure. Following shows the brute force version of this:

string str = "330";

int n = 6

// idx is value of current index in str

// rem is current remainder

int count(int idx, int rem)

{

// If last character reached

if (idx == n)

return (rem == 0)? 1 : 0;

int ans = 0;

// we exclude it, thus remainder

// remains the same

ans += count(idx+1, rem);

// we include it and thus new remainder

ans += count(idx+1, (rem\*10 + str[idx]-'0')%n);

return ans;

}

The above recursive solution has overlapping subproblems as shown in below recursion tree.

input string = "330"

(0,0) ===> at 0th index with 0 remainder

(exclude 0th / (include 0th character)

character) /

(1,0) (1,3) ======> at index 1 with 3 as

(E)/ (I) /(E) the current remainder

(2,0) (2,3) (2,3)

|-------|

These two subproblems overlap

Thus, we can apply Dynamic Programming. Below is implementation .

* C++
* Java
* Python 3
* C#

filter\_none

edit

play\_arrow

brightness\_5

|  |
| --- |
| // C++ program to count subsequences of a  // string divisible by n.  #include<bits/stdc++.h>  using namespace std;    // Returns count of subsequences of str  // divisible by n.  int countDivisibleSubseq(string str, int n)  {      int len = str.length();        // division by n can leave only n remainder      // [0..n-1]. dp[i][j] indicates number of      // subsequences in string [0..i] which leaves      // remainder j after division by n.      int dp[len][n];      memset(dp, 0, sizeof(dp));        // Filling value for first digit in str      dp[0][(str[0]-'0')%n]++;        for (int i=1; i<len; i++)      {          // start a new subsequence with index i          dp[i][(str[i]-'0')%n]++;            for (int j=0; j<n; j++)          {              // exclude i'th character from all the              // current subsequences of string [0...i-1]              dp[i][j] += dp[i-1][j];                // include i'th character in all the current              // subsequences of string [0...i-1]              dp[i][(j\*10 + (str[i]-'0'))%n] += dp[i-1][j];          }      }        return dp[len-1][0];  }    // Driver code  int main()  {      string str = "1234";      int n = 4;      cout << countDivisibleSubseq(str, n);      return 0;  } |

**Output:**

4

**Time Complexity :** O(len \* n)  
**Auxiliary Space :** O(len \* n)