Super Six Substrings

locked

by nikasvanidze

Editorial by Shafaet

Observation

A number is divisible by 6 only if it's divisible by its two prime factors, 3 and 2, so we can reframe this problem as "How many substrings are divisible by both 3 and 2?" Recall that a number is divisible by 3 if the sum of all of its digits are divisible by 3. A number is divisible by 2 if the last digit of the number is divisible by 2. We mention these things because the substrings become too massive to be converted into numeric data types. If we were working with smaller numbers we could simply convert the substrings into integers and use modular arithmetic, but unfortunately that's not possible.

Solution

We solve this problem using dynamic programming, which enables us to compute answers quickly and efficiently by tracking and previously computed answers in a memory structure and using these stored answers instead of recomputing values.

Let $f(i, m)$ be the number of substrings starting at index i and the sum of their digits modulo 3 is m, and the number it represents is even. Our answer will be $\sum_{i}^{n-1} f(i, 0)$

Let x be the i^{th} digit in the string. From $f(i, m)$ we need to find all the even substrings that starts in $i + 1$ and sum of their digits modulo 3 is $(m + x)$. Also we will get an extra substring if $(x + m)$ itself is divisible by 3 and x is even. So we get the recurrence relation $f(i, m) = f(i + 1, (m + x)\%3) + ((x + m)\%3 == 0$ and $x\%2 == 0)$. By memorizing the states, we get a $O(n)$ solution. See the code below for better understanding:

C++

```
#include <bits/stdc++.h>

using namespace std;

string s;

long long memoize[100002][5];
long long f(int i, int m){
    if(i == (int)s.size()){
        return 0;
    }
    if(memoize[i][m] != -1) {
        return memoize[i][m];
        }
    int x=s[i]-'0';
    int ans = f(i + 1, (m + x) % 3) + ((x + m) % 3 == 0 and x % 2 == 0);

    return memoize[i][m] = ans;
}

int main(){
    //freopen("in","r",stdin);
    memset(memoize, -1, sizeof memoize);
    long long ans=0;
    cin >> s;
```

Statistics

Difficulty: Medium
Time Complexity: O(n)
Required Knowledge: Dynamic Programming, Divisibility, Modular Arithmatic
Publish Date: Feb 18 2017

```
    for(int i=0; i<(int)s.size(); i++) {
        if(s[i] == '0') ans++;
        else ans = ans + f(i,0);

    }
    cout << ans << endl;

    return 0;
}
```

Solution #2

Let's iterate through all string and remember values of sum modulo 3 for all the prefixes, that are ending in even digit. We will save this values in array a[3]. After that iterate through all $i = 0 \dots n-1$ and find the number of valid strings starting in position i:

- if $s[i] =' 0'$, then only 1 string is valid starting in this digit
- in other case we need to find all the prefixes with position $\geq i$, that have same value of sum modulo 3, as $sum[0, i)$ - we can use precalculated array above for it.

Code by wild_hamster:

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <vector>
#include <cstring>
#include <stdio.h>
#include <map>
#include <assert.h>
#define pdd pair<double,double>
#define pii pair<ll,ll>
#define MOD2 1000000009
#define INF ((ll)1e+18)


typedef long long ll;
typedef long double ld;
using namespace std;
ll i,j,n, flag,r,x,y,ans;
string s,t;
ll a[4];

void solveN()
{
    cin >> s;
    n = s.size();
    for (i = 0; i < n; i++)
    {
        x = (x+s[i])%3;
        if (s[i] % 2 == 0)
            a[x]++;
    }
    x = 0;
    for (i = 0; i < n; i++)
    {

        if (s[i] == '0')
            ans++;
        else
            ans += a[x];
        x = (x+s[i])%3;
        if (s[i] % 2 == 0)
            a[x]--;
    }
    cout << ans << endl;
}
int main() {

        solveN();
    return 0;
}
```