Rhezo and Division

Attempted by: 1 / Accuracy: 100% /

Tag(s): Algorithms, Dynamic Programming, Medium, Z-algorithm

The problem asks to find all occurrences of a pattern in a text, and then find number of ways to choose one or more of the starting positions of these occurrences and tell if the number formed by product of these starting positions is divisible by all numbers from 1-9.

First we need to find all occurrences of a pattern in a text, this can be easily done through Z-Function. If you don't know about it, read it on e-maxx.ru or some other source. Once you have the index of all occurrences of pattern in the text, we can use Dynamic Programming to find the required good numbers.

The author's code uses a recursive DP, where at each step, we either take the number in our product or do not take it. One key observation is that we do not have to maintain the product in our state. For a number to be divisible by all numbers from 1-9, its prime factorization must have at least **three 2s**, **two 3s**, **one 5** and **one 7**. So we need to maintain only these. Also there is no need to keep more that three 2s or more than two 3s or more than one 5 or more than one 7. That is why there is a 'min' in the author's code.

Author Solution by Rishi Vikram

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e4+5;
long long dp[MAXN][5][5][3][3];
char P[2*MAXN];//Pattern
char T[MAXN];//Text
int ar[2*MAXN];//Array of positions of occurenes
int Z[2*MAXN];//Z-array as in Z-function
int counter;//number of occurences of pattern in text.
const int MOD = 1e9+7;
long long go(int idx, int two, int three, int five, int seven) {
        if(idx==counter+1 and two>=3 and three>=2 and five>=1 and seven>=1)
```

```
        else if(idx==counter+1) return 0;
        if(dp[idx][two][three][five][seven]!=-1) return dp[idx][two][three][
        long long ret=0;
        int cc=ar[idx];
        int tw=0;
        while(cc%2==0 and tw<=3) tw++,cc/=2;//Finding factors of 2
        cc=ar[idx];
        int thr=0;
        while(cc%3==0 and thr<=2) thr++,cc/=3;//Finding factors of 3
        cc=ar[idx];
        int fiv=0;
        while(cc%5==0 and fiv<=1) fiv++,cc/=5;//Finding factors of 5
        cc=ar[idx];
        int sev=0;
        while(cc%7==0 and sev<=1) sev++,cc/=7;//Finding factors of 7
        ret=go(idx+1,two,three,five,seven);//Not taking the current number
        ret+=go(idx+1,min(3,two+tw),min(2,three+thr),min(1,five+fiv),min(1,se
        if(ret>=MOD) ret-=MOD;//Be careful about overflows
        return dp[idx][two][three][five][seven]=ret;//Memoize
}
int main() {
//      freopen("TASK.in","r",stdin);
//      freopen("TASK.out","w",stdout);
        scanf("%s",P);
        scanf("%s",T);
        memset(dp,-1,sizeof(dp));
        /* Formation of Z-array starts where Z[i] denotes
           Longest Common Prefix between a Sting S, and its suffix starting
           To do that we have a new string S = pattern + '$' + text
           Rest you can read online.
        */
        int lenP=strlen(P);
        int cc=lenP;
        int lenT=strlen(T);
        P[lenP++]='$';
        for(int i=0;i<lenT;i++) P[lenP++]=T[i];
        P[lenP]='\0';
        Z[0]=0;
        int l=0,r=0;
        for(int i=1;i<lenP;i++) {
                Z[i]=0;
                if(i<=r) Z[i]=max(Z[i-l],r-i+1);
                while(i+Z[i]<lenP and P[i+Z[i]]==P[Z[i]]) ++Z[i];
                if(i+Z[i]-1>r) l=i,r=i+Z[i]-1;
        }
        /*Z-array formed*/
        counter=0;
        for(int i=cc+1;i<lenP;i++) if(Z[i]==cc) ar[++counter]=i-cc;
        printf("%lld\n",go(1,0,0,0,0));
        return 0;
}
```