题目描述

给定仅包含数字的字符串 S, T。

记 $S^\infty = S + S + \cdots$，$T^\infty = T + T + \cdots$，其中 $+$ 表示字符串的连接运算。

记号 $[P]$ 为 Iverson 括号，当命题 P 成立时 $[P] = 1$，否则 $[P] = 0$。

记号 S_i^∞ 表示字符串 S^∞ 的第 i 个字符，下标从 1 开始，T_i^∞ 同理。

给定 l, r，你需要求出 $\sum_{i=l}^{r}[S_i^\infty = T_i^\infty]$ 的值。

总之就是求这个：

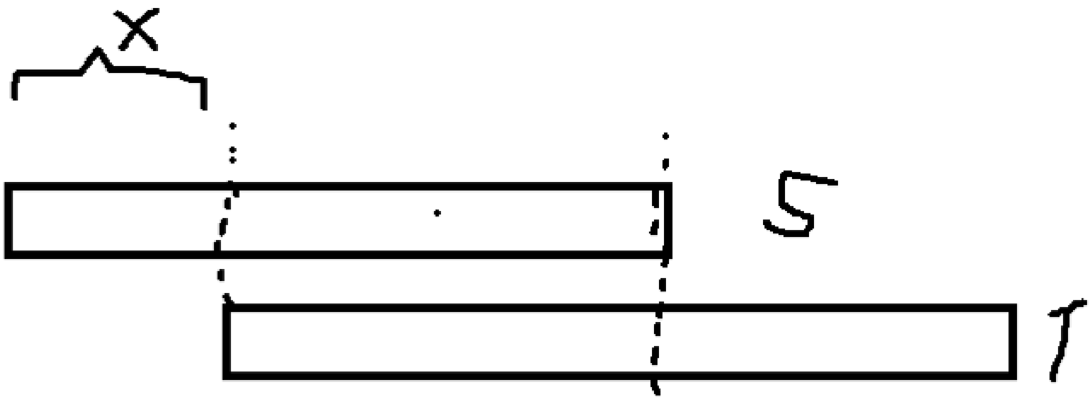
```
for (int i = l - 1; i <= r - 1; ++i) {
    ans += (s[i % m] == t[i % n]);
}
```

为了简化问题，我们先求出最小的$ll \geq l$使得$m|ll$，最小的$rr \leq r$使得$m|rr$

$ans(l, r) = ans(l, ll-1) + ans(ll, rr) + ans(rr+1, r)$

$ans(l, ll-1) + ans(rr+1, r)$可以使用原始方法计算，复杂度为$O(m)$

再考虑s相对于t偏移一定位置后，重叠区间的Iverson和，记为$c(x), (-m \leq x \leq n)$



$ans(ll, rr)$中共有$(rr - ll + 1)/m$段S字符串

考虑每段S字符串对答案的贡献，第i段S字符串相对T的偏移为$(ll + i * m) mod\ n$

若$[(ll + i * m) mod\ n] + m < n$，则第$i$段字符串$S$对答案的贡献就是$c[(ll + i * m) mod\ n]$

否则，第i段字符串S对答案的贡献是$c[(ll + i * m) mod\ n] + c[(ll + i * m) mod\ n - n]$

同时，还需注意到$ans(ll, ll + x)$以$lcm(n, m)$为周期，

$ans(ll, rr) = ans(0, lcm - 1) * k + ans(ll + k * lcm, rr)$

给定$c(x)$，根据每段S字符串的贡献，$ans(0, lcm-1)$与$ans(ll + k * lcm, rr)$的计算时间为$O(lcm/m)$，不会超过$O(n)$

在计算$c(x)$后，计算$ans(l, r)$的时间复杂度为$O(lcm/m + m)$

如何计算$c(x)$?

$c(x)=\Sigma_{i=0}^{min\{x+m,n\}}[S_{i-x}=T_i]$

由于字符集只有$\{0,...,9\}$，$[S_{i-x}=T_i]=\Sigma_{j=0}^{9}[S_{i-x}=j][T_i=j]$

可以预处理出$S_{i,j}=[S_i=j]$与$T_{i,j}=[T_i=j]$

$$c(x)=\Sigma_{i=0}^{min\{x+m,n\}}[S_{i-x}=T_i],$$
$$=\Sigma_{i=0}^{min\{x+m,n\}}\Sigma_{j=0}^{9}[S_{i-x}=j][T_i=j],$$
$$=\Sigma_{j=0}^{9}\Sigma_{i=0}^{min\{x+m,n\}}S_{i-x,j}T_{i,j}$$

为卷积形式，可以使用fft来进行计算，时间复杂度为$O(10(n+m)log(n+m))$

为防止MLE和TLE，可以先累加，最后一次再进行逆变换，由于累加过程中可能会产生累计误差导致WA，可能需要使用long double

总共需要20次dft+1次idft

总的复杂度为$O(10(n+m)log(n+m))$

代码：

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

using namespace std;
using ll = long long;

const double PI = acos(-1);

struct cd {
    long double real, imag;
    cd() {}
    cd(long double real, long double imag) : real(real), imag(imag) {}

    cd operator * (cd const& a) {
        return {a.real * real - a.imag * imag, a.real * imag + a.imag * real};
    }
    cd operator + (cd const& a) {
        return {a.real + real, a.imag + imag};
    }
    cd operator - (cd const& a) {
        return {real - a.real, imag - a.imag};
    }
    cd operator / (long double n) {
        return {real / n, imag / n};
    }
};

cd a[1 << 20], b[1 << 20], c[1 << 20];
int rev[1 << 20];
int lg_n, rn;

ll gcd(ll a, ll b) {
    return b == 0 ? a : gcd(b, a % b);
}

int reverse(int num, int lg_n) {
```

```
    int res = 0;
    for (int i = 0; i < lg_n; i++) {
        if (num & (1 << i))
            res |= 1 << (lg_n - 1 - i);
    }
    return res;
}

void fft(cd a[], int n, bool invert) {


    for (int i = 0; i < n; i++) {
        if (i < rev[i])
            swap(a[i], a[rev[i]]);
    }

    for (int len = 2; len <= n; len <<= 1) {
        long double ang = 2 * PI / len * (invert ? -1 : 1);
        cd wlen(cos(ang), sin(ang));
        for (int i = 0; i < n; i += len) {
            cd w(1, 0);
            for (int j = 0; j < len / 2; j++) {
                cd u = a[i + j], v = a[i + j + len / 2] * w;
                a[i + j] = u + v;
                a[i + j + len / 2] = u - v;
                w = w * wlen;
            }
        }
    }

    if (invert) {
        for (int i = 0; i < n; ++i)
            a[i] = a[i] / n;
    }
}

void multiply(cd A[], cd B[]) {

    fft(A, rn, false);
    fft(B, rn, false);

    for (int k = 0; k < rn; ++k) {
        c[k] = c[k] + A[k] * B[k];
    }
}

int main() {
    cin.tie(NULL);
    ios::sync_with_stdio(false);

    ll l, r;
    cin >> l >> r;
    --l, --r;
    string s, t;
    cin >> s >> t;
    if (s.length() > t.length()) {
```

```
        swap(s, t);
    }
    ll m = s.length(), n = t.length();

    rn = 1;
    while (rn < n + m - 1) rn <<= 1;
    while ((1 << lg_n) < rn)
        lg_n++;
    for (int i = 0; i < rn; ++i) {
        rev[i] = reverse(i, lg_n);
    }


    for (int num = 0; num < 10; ++num) {
        for (int i = 0; i < rn; ++i) {
            a[i] = b[i] = cd(0, 0);
        }
        for (int i = 0; s[i]; ++i) {
            a[i] = cd(s[m - i - 1] - '0' == num, 0);
        }
        for (int i = 0; t[i]; ++i) {
            b[i] = cd(t[i] - '0' == num, 0);
        }

        multiply(a, b);

    }
    fft(c, rn, true);

    ll lcm = n * m / gcd(n, m);

    ll tot = 0;
    ll cyc = 0;

    // alignment
    while (l <= r && l % m != 0) {
        tot += (s[l % m] == t[l % n]);
        ++l;
    }

    while (l <= r && (r + 1) % m != 0) {
        tot += (s[r % m] == t[r % n]);
        --r;
    }

    ll period = (r - l + 1) / lcm;

    for (int i = 0; i < lcm / m; ++i) {
        int off = (i * m) % n;
        cyc += round(c[(m + off) - 1].real);
        if (off + m > n) {
            cyc += round(c[(m + off - n) - 1].real);
        }
    }

    tot += period * cyc;
```

```
    l += period * lcm;


    for (int i = 0; i < (r - l + 1) / m; ++i) {
        int off = (l + i * m) % n;
        tot += round(c[(m + off) - 1].real);
        if (off + m > n) {
            tot += round(c[(m + off - n) - 1].real);
        }
    }

    cout << tot << endl;

    return 0;
}
```