模板汇总

编译命令

```
g++ test.cpp -o 1 -std=c++17 -O2 -fsanitize=address,undefined,signed-integer-overflow,memory -fno-omit-frame-pointer && time ./1 < 1.in > 1.out
```

```
g++ test.cpp -o 1 -std=c++17 -pg -fsanitize=address,undefined,signed-integer-overflow,memory -fno-omit-frame-pointer

```

标准模板（C++11）

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

```
3
4   #define PI M_PI
5   #define E M_E
6
7   using namespace std;
8
9   mt19937 rnd(random_device{}());
10  int rndd(int l, int r){return rnd() % (r - l + 1) + l;}
11
12  typedef unsigned int uint;
13  typedef unsigned long long unll;
14  typedef long long ll;
15
16  int main(){
17
18
19      // fprintf(stderr, "Time: %.6lf\n", (double)clock() /
    CLOCKS_PER_SEC);
20      return 0;
21  }
```

快读

```
1   template<typename T = int>
2   inline T read(void);
3
4   int main(){}
5
6   template<typename T>
7   inline T read(void){
8       T ret(0);
9       short flag(1);
10      char c = getchar();
11      while(c != '-' && !isdigit(c))c = getchar();
12      if(c == '-')flag = -1, c = getchar();
13      while(isdigit(c)){
```

```
        ret *= 10;
        ret += int(c - '0');
        c = getchar();
    }
    ret *= flag;
    return ret;
}
```

```
char buf[1<<23],*p1=buf,*p2=buf,obuf[1<<23],*O=obuf;
#define getchar() (p1==p2&&(p2=
(p1=buf)+fread(buf,1,1<<21,stdin),p1==p2)?EOF:*p1++)
inline int read() {
    int x=0,f=1;char ch=getchar();
    while(!isdigit(ch)){if(ch=='-') f=-1;ch=getchar();}
    while(isdigit(ch)) x=x*10+(ch^48),ch=getchar();
    return x*f;
}
void print(long long x) {
    if(x>9) print(x/10);
    *O++=x%10+'0';
}
fwrite(obuf,O-obuf,1,stdout);
```

快速幂

```
auto qpow = [](ll a, ll b)->ll{
    ll ret(1), mul(a);
    while(b){
        if(b & 1)ret = (ret * mul) % MOD;
        b >>= 1, mul = (mul * mul) % MOD;
    }return ret;
};
```

```cpp
#define _USE_MATH_DEFINES
#include <bits/stdc++.h>

#define PI M_PI
#define E M_E

using namespace std;

mt19937 rnd(random_device{}());
int rndd(int l, int r){return rnd() % (r - l + 1) + l;}
bool rnddd(int x){return rndd(1, 100) <= x;}

typedef unsigned int uint;
typedef unsigned long long unll;
typedef long long ll;
typedef long double ld;

template < typename T = int >
inline T read(void);

const ll MOD = 998244353ll;

auto qpow = [](ll a, ll b, ll mod = MOD)->ll{
    if(b < 0)return 0;
    ll ret(1), mul(a);
    while(b){
        if(b & 1)ret = ret * mul % mod;
        b >>= 1;
        mul = mul * mul % mod;
    }return ret;
};

const ll g = 3;
const ll invg = qpow(g, MOD - 2);
const ll inv2 = qpow(2, MOD - 2);
vector < int > pos;
```

```
37
38  enum Pattern{DFT, IDFT};
39
40  class Polynomial{
41  private:
42  public:
43      vector < ll > poly;
44      Polynomial(void){this->poly.resize(0);}
45      Polynomial(int len){this->poly.assign(len, 0);}
46      void Reverse(void){
47          int len = poly.size();
48          pos.resize(len);
49          if(len > 0)pos[0] = 0;
50          for(int i = 1; i < len; ++i)
51              pos[i] = (pos[i >> 1] >> 1) | (i & 1 ? len >> 1 :
    0);
52          for(int i = 0; i < len; ++i)if(i < pos[i])swap(poly[i],
    poly[pos[i]]);
53      }
54      void NTT(Pattern pat){
55          int len = poly.size();
56          Reverse();
57          for(int siz = 2; siz <= len; siz <<= 1){
58              ll gn = qpow(pat == DFT ? g : invg, (MOD - 1) /
    siz);
59              for(auto p = poly.begin(); p < next(poly.begin(),
    len); advance(p, siz)){
60                  int mid = siz >> 1; ll g(1);
61                  for(int i = 0; i < mid; ++i, (g *= gn) %= MOD){
62                      auto tmp = g * p[i + mid] % MOD;
63                      p[i + mid] = (p[i] - tmp + MOD) % MOD;
64                      p[i] = (p[i] + tmp) % MOD;
65                  }
66              }
67          }
68          if(pat == IDFT){
69              ll inv_len = qpow(len, MOD - 2);
70              for(int i = 0; i < len; ++i)(poly[i] *= inv_len) %=
    MOD;
```

```
        }
    }
    void Resize(int len){
        this->poly.resize(len, 0);
    }
    void Derivate(void){
        int len = poly.size();
        if(len == 0)return;
        poly[0] = 0;
        for(int i = 1; i < len; ++i)poly[i - 1] = i * poly[i] %
MOD, poly[i] = 0;
        Resize(len - 1);
    }
    void Integrate(void){
        int len = poly.size();
        if(len == 0)return;
        Resize(len + 1);
        for(int i = len - 1; i >= 0; --i)poly[i + 1] = poly[i] *
qpow(i + 1, MOD - 2) % MOD, poly[i] = 0;
    }
    auto Desc(void){
        int len = poly.size();
        // printf("Polynomial(len = %d): ", len);
        for(int i = 0; i < len; ++i)printf("%lld%c", poly[i], i
== len - 1 ? '\n' : ' ');
        return this;
    }
};


auto Multiply = [](Polynomial* baseA, Polynomial* baseB)-
>Polynomial*{
    auto A = new Polynomial(*baseA), B = new Polynomial(*baseB);
    int len = A->poly.size() + B->poly.size() - 1;
    int base(1); while(base < (len << 1))base <<= 1;
    Polynomial* ret = new Polynomial(base);
    A->Resize(base), B->Resize(base);
    A->NTT(DFT), B->NTT(DFT);
    for(int i = 0; i < base; ++i)
```

```
        ret->poly[i] = (A->poly[i] * B->poly[i] % MOD);
    ret->NTT(IDFT);
    ret->Resize(len);
    delete A; delete B;
    return ret;
};

auto Inverse = [](auto&& self, Polynomial* baseF, int len)->Polynomial*{
    if(len == 1){
        Polynomial *G = new Polynomial(1);
        G->poly[0] = qpow(baseF->poly[0], MOD - 2);
        return G;
    }
    auto *H = self(self, baseF, (len + 1) >> 1);
    int base(1); while(base < (len << 1))base <<= 1;
    H->Resize(base);
    Polynomial *G = new Polynomial(base), *F = new Polynomial(base);
    for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F->poly[i] = baseF->poly[i];
    H->NTT(DFT), F->NTT(DFT);
    for(int i = 0; i < base; ++i)
        G->poly[i] = (2 * H->poly[i] % MOD - H->poly[i] * H->poly[i] % MOD * F->poly[i] % MOD + MOD) % MOD;
    G->NTT(IDFT), G->Resize(len);
    delete H; delete F;
    return G;
};

//Require A[0] == 1
auto Sqrt = [](auto&& self, Polynomial* baseF, int len)->Polynomial*{
    if(len == 1){
        Polynomial *G = new Polynomial(1);
        G->poly[0] = sqrt(baseF->poly[0]);
        return G;
    }
    auto H = self(self, baseF, (len + 1) >> 1);
```

```
    auto invH = Inverse(Inverse, H, len);
    int base(1); while(base < (len << 1))base <<= 1;
    auto G = new Polynomial(base), F = new Polynomial(len);
    for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F->poly[i] = baseF->poly[i];
    H->Resize(base), invH->Resize(base), F->Resize(base);
    H->NTT(DFT), F->NTT(DFT), invH->NTT(DFT);
    for(int i = 0; i < base; ++i)G->poly[i] = (F->poly[i] * invH->poly[i] % MOD + H->poly[i]) % MOD * inv2 % MOD;
    G->NTT(IDFT), G->Resize(len);
    delete H; delete invH; delete F;
    return G;
};
auto Ln = [](Polynomial* baseF, int len)->Polynomial*{
    auto F = new Polynomial(len);
    for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)
        F->poly[i] = baseF->poly[i];
    auto invF = Inverse(Inverse, F, len);
    F->Derivate();
    int clen = F->poly.size() + invF->poly.size() - 1;
    int base(1); while(base < clen)base <<= 1;
    Polynomial* G = new Polynomial(base);
    F->Resize(base), invF->Resize(base);
    F->NTT(DFT), invF->NTT(DFT);
    for(int i = 0; i < base; ++i)
        G->poly[i] = F->poly[i] * invF->poly[i] % MOD;
    G->NTT(IDFT), G->Resize(len - 1);
    G->Integrate();
    delete invF; delete F;
    return G;
};
auto Exp = [](auto&& self, Polynomial* baseF, int len)->Polynomial*{
    if(len == 1){
        Polynomial* G = new Polynomial(1);
        G->poly[0] = 1;
        return G;
    }
    auto H = self(self, baseF, (len + 1) >> 1);
```

```
    auto lnH = Ln(H, len);
    int base(1); while(base < (len << 1))base <<= 1;
    auto F = new Polynomial(len), G = new Polynomial(base);
    for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F->poly[i] = baseF->poly[i];
    F->Resize(base), H->Resize(base), lnH->Resize(base);
    F->NTT(DFT), H->NTT(DFT), lnH->NTT(DFT);
    for(int i = 0; i < base; ++i)
        G->poly[i] = (H->poly[i] * ((1 - lnH->poly[i] + MOD) % MOD) % MOD + F->poly[i] * H->poly[i] % MOD) % MOD;
    G->NTT(IDFT), G->Resize(len);
    delete F; delete lnH; delete H;
    return G;
};

auto Quickpow = [](Polynomial* baseF, ll k1, ll k2, ll mx)->Polynomial*{
    int len = baseF->poly.size();
    if(baseF->poly[0] == 0 && mx >= len){
        Polynomial* G = new Polynomial(len);
        for(int i = 0; i < len; ++i)
            G->poly[i] = 0;
        return G;
    }
    if(len == 1){
        Polynomial* G = new Polynomial(1);
        G->poly[0] = qpow(baseF->poly[0], k2);
        return G;
    }
    int offset(0);
    while (offset < len && baseF->poly[offset] == 0) ++offset;
    if((ll)offset * k1 >= len)return new Polynomial(len);
    ll mul = qpow(baseF->poly[offset], k2), inv = qpow(baseF->poly[offset], MOD - 2);
    auto F = new Polynomial(*baseF);
    for(int i = 0; i + offset < len; ++i)F->poly[i] = F->poly[i + offset] * inv % MOD;
    for(int i = len - offset; i < len; ++i)F->poly[i] = 0;
    auto lnF = Ln(F, len);
```

```
    for(int i = 0; i < len; ++i)lnF->poly[i] = lnF->poly[i] * k1 % MOD;
    auto eLnF = Exp(Exp, lnF, len);
    ll shift = offset * k1;
    for(int i = len - 1; i >= shift; --i)
        eLnF->poly[i] = eLnF->poly[i - shift];
    for(int i = 0; i < shift; ++i)
        eLnF->poly[i] = 0;
    for(auto i = 0; i < len; ++i)eLnF->poly[i] = eLnF->poly[i] * mul % MOD;
    delete lnF; delete F;
    return eLnF;
};

struct Complex{
    ll x, y;
    static ll w;
    friend Complex operator *(const Complex &a, const Complex &b){
        return Complex{
            (a.x * b.x % MOD + w * a.y % MOD * b.y % MOD) % MOD,
            (a.x * b.y % MOD + a.y * b.x % MOD) % MOD
        };
    }
    static ll qpow(Complex a, ll b){
        Complex ret{1, 0};
        while(b){
            if(b & 1)ret = ret * a;
            a = a * a;
            b >>= 1;
        }return ret.x;
    }
};
ll Complex::w;
auto Cipolla = [](ll x)->ll{
    if(qpow(x, (MOD - 1) >> 1) == MOD - 1)return -1;
    while(true){
        ll a = (1ll * rnd() << 15 | rnd()) % MOD;
        Complex::w = (a * a % MOD + MOD - x) % MOD;
```

```
        if(qpow(Complex::w, (MOD - 1) >> 1) == MOD - 1) {
            ll res = Complex::qpow(Complex{a, 1}, (MOD + 1) >> 1);
            return min(res, MOD - res);
        }
    }
};

//Require A[0] is a quadratic residue modulo 998244353
auto ExSqrt = [](auto&& self, Polynomial* baseF, int len)->Polynomial*{
    if(len == 1){
        Polynomial *G = new Polynomial(1);
        auto res = Cipolla(baseF->poly[0]);
        G->poly[0] = min(res, MOD - res);
        return G;
    }
    auto H = self(self, baseF, (len + 1) >> 1);
    auto invH = Inverse(Inverse, H, len);
    int base(1); while(base < (len << 1))base <<= 1;
    auto G = new Polynomial(base), F = new Polynomial(len);
    for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F->poly[i] = baseF->poly[i];
    H->Resize(base), invH->Resize(base), F->Resize(base);
    H->NTT(DFT), F->NTT(DFT), invH->NTT(DFT);
    for(int i = 0; i < base; ++i)G->poly[i] = (F->poly[i] * invH->poly[i] % MOD + H->poly[i]) % MOD * inv2 % MOD;
    G->NTT(IDFT), G->Resize(len);
    delete H; delete invH; delete F;
    return G;
};

// auto CDQ = [](Polynomial* baseF, int l, int r)->Polynomial*{
//     if(l == r)return BuildPoly(l);
//     int mid = (l + r) >> 1;
//     Polynomial L = CDQ(l, mid, mx), R = CDQ(mid + 1, r, mx);
//     // for(int i = l; i <= r; ++i)pol[i].poly.clear(), pol[i].poly.shrink_to_fit(), pol[i].len = 0;
//     return Mul(L, R, mx);
```

```
// }

namespace Tests{
    auto ImplementMultiply = [](void)->void{
        int N = read() + 1, M = read() + 1;
        Polynomial *A = new Polynomial(N), *B = new
Polynomial(M);
        for(int i = 0; i < N; ++i)A->poly[i] = read();
        for(int i = 0; i < M; ++i)B->poly[i] = read();
        delete Multiply(A, B)->Desc();
        delete A; delete B;
    };
    auto ImplementInverse = [](void)->void{
        int N = read();
        Polynomial *A = new Polynomial(N);
        for(int i = 0; i < N; ++i)A->poly[i] = read();
        delete Inverse(Inverse, A, N)->Desc();
        delete A;
    };
    auto ImplementLn = [](void)->void{
        int N = read();
        Polynomial *A = new Polynomial(N);
        for(int i = 0; i < N; ++i)A->poly[i] = read();
        delete Ln(A, N)->Desc();
        delete A;
    };
    auto ImplementExp = [](void)->void{
        int N = read();
        Polynomial *A = new Polynomial(N);
        for(int i = 0; i < N; ++i)A->poly[i] = read();
        delete Exp(Exp, A, N)->Desc();
        delete A;
    };
    auto ImplementSqrt = [](void)->void{
        int N = read();
        Polynomial *A = new Polynomial(N);
        for(int i = 0; i < N; ++i)A->poly[i] = read();
        delete Sqrt(Sqrt, A, N)->Desc();
        delete A;
```

```
318     };
319     auto ImplementExSqrt = [](void)->void{
320         int N = read();
321         Polynomial *A = new Polynomial(N);
322         for(int i = 0; i < N; ++i)A->poly[i] = read();
323         delete ExSqrt(ExSqrt, A, N)->Desc();
324         delete A;
325     };
326     auto ImplementQuickPow = [](void)->void{
327         auto ReadIndex = [](void)->tuple < ll, ll, ll >{
328             ll ret1(0), ret2(0), mx(0);
329             char c = getchar(); while(!isdigit(c))c = getchar();
330             while(isdigit(c)){
331                 ((ret1 *= 10) += c - '0') %= MOD;
332                 ((ret2 *= 10) += c - '0') %= MOD - 1;
333                 if(mx < 10000000)
334                     mx = mx * 10 + c - '0';
335                 c = getchar();
336             }return {ret1, ret2, mx};
337         };
338         int N = read();
339         Polynomial *A = new Polynomial(N);
340         auto [k1, k2, mx] = ReadIndex();
341         for(int i = 0; i < N; ++i)A->poly[i] = read();
342         delete Quickpow(A, k1, k2, mx)->Desc();
343         delete A;
344     };
345 
346 }
347 
348 int main(){
349     Tests::ImplementExSqrt();
350 
351     // fprintf(stderr, "Time: %.6lf\n", (double)clock() /
    CLOCKS_PER_SEC);
352     return 0;
353 }
354 
355 template < typename T >
```

```cpp
inline T read(void){
    T ret(0);
    int flag(1);
    char c = getchar();
    while(c != '-' && !isdigit(c))c = getchar();
    if(c == '-')flag = -1, c = getchar();
    while(isdigit(c)){
        ret *= 10;
        ret += int(c - '0');
        c = getchar();
    }
    ret *= flag;
    return ret;
}
```

```
g++ ./Poly.cpp -o ./1 -fsanitize=undefined,signed-integer-overflow,address -Wall -std=c++17 \
&& time ./1 < ./1.in > ./1.out
```

```cpp
#define _USE_MATH_DEFINES
#include <bits/stdc++.h>

#define PI M_PI
#define E M_E

using namespace std;

mt19937 rnd(random_device{}());
int rndd(int l, int r){return rnd() % (r - l + 1) + l;}

typedef unsigned int uint;
typedef unsigned long long unll;
typedef long long ll;

#define EPS (1e-9)

template<typename T = int>
inline T read(void);
```

```cpp

struct Fraction{//non-negative
    __int128_t a, b;
    Fraction Shrink(void){
        __int128_t div = __gcd(a, b);
        a /= div, b /= div;
        return *this;
    }
    friend const Fraction operator + (const Fraction &x, const Fraction &y){
        __int128_t below = x.b * y.b / __gcd(x.b, y.b);
        return Fraction{below / x.b * x.a + below / y.b * y.a, below}.Shrink();
    }
    friend const Fraction operator / (const Fraction &x, const int &v){
        return Fraction{x.a, x.b * v}.Shrink();
    }
    friend const Fraction operator / (const Fraction &x, const Fraction &y){
        return Fraction{x.a * y.b, x.b * y.a}.Shrink();
    }
    friend const Fraction operator * (const Fraction &x, const Fraction &y){
        return Fraction{x.a * y.a, x.b * y.b}.Shrink();
    }
    friend const bool operator <= (const Fraction &x, const Fraction &y){
        return x.a * y.b <= y.a * x.b;
    }
    friend const bool operator >= (const Fraction &x, const Fraction &y){
        return x.a * y.b > y.a * x.b;
    }
    friend const bool operator < (const Fraction &x, const Fraction &y){
        return x.a * y.b < y.a * x.b;
    }
```

```
    friend const bool operator > (const Fraction &x, const Fraction &y){
        return x.a * y.b > y.a * x.b;
    }
    void Desc(void){
        this->Shrink();
        printf("%lld/%lld\n", (ll)this->a, (ll)this->b);
    }
};

int main(){

    // fprintf(stderr, "Time: %.6lf\n", (double)clock() / CLOCKS_PER_SEC);
    return 0;
}



template<typename T>
inline T read(void){
    T ret(0);
    short flag(1);
    char c = getchar();
    while(c != '-' && !isdigit(c))c = getchar();
    if(c == '-')flag = -1, c = getchar();
    while(isdigit(c)){
        ret *= 10;
        ret += int(c - '0');
        c = getchar();
    }
    ret *= flag;
    return ret;
}
```

```
/*
 * LG-P3804 【模板】后缀自动机（SAM）
 * https://www.luogu.com.cn/problem/P3804
```

```
 */

#define _USE_MATH_DEFINES
#include <bits/stdc++.h>

#define PI M_PI
#define E M_E

using namespace std;

mt19937 rnd(random_device{}());
int rndd(int l, int r){return rnd() % (r - l + 1) + l;}
bool rnddd(int x){return rndd(1, 100) <= x;}

typedef unsigned int uint;
typedef unsigned long long unll;
typedef long long ll;
typedef long double ld;

template < typename T = int >
inline T read(void);

#define d(c) (c - 'a')
#define npt nullptr
#define SON i->to

struct Edge;
struct Node{
    unordered_map < char, Node* > trans;
    Node* link;
    int len;
    int siz;
    Edge* head;
    void* operator new(size_t);
}nd[2100000];
void* Node::operator new(size_t){static Node* P = nd; return
P++;}

struct Edge{
```

```
    Edge* nxt;
    Node* to;
    void* operator new(size_t);
}ed[4100000];
void* Edge::operator new(size_t){static Edge* P = ed; return P++;}

class SAM{
private:
public:
    Node* root;
    void Insert(int c){
        static Node* lst = root;
        Node* p = lst; Node* cp = lst = new Node; cp->siz = 1;
        cp->len = p->len + 1;
        while(p && !p->trans[c])p->trans[c] = cp, p = p->link;
        if(!p)cp->link = root;
        else if(p->trans[c]->len == p->len + 1)cp->link = p->trans[c];
        else{
            auto q = p->trans[c], sq = new Node(*q); sq->siz = 0;
            sq->len = p->len + 1;
            cp->link = q->link = sq;
            while(p && p->trans[c] == q)p->trans[c] = sq, p = p->link;
        }
    }
    void Link(void){
        auto endp = new Node();
        for(auto p = nd; p != endp;++p)
            if(p->link)
                p->head = new Edge{p->head, p->link},
                p->link->head = new Edge{p->link->head, p};
    }

}sam;

ll ans(0);
```

```
string S;

int main(){
    sam.root = new Node();
    cin >> S;
    for(auto c : S)sam.Insert(d(c));
    sam.Link();
    auto dfs = [](auto&& self, Node* p = sam.root, Node* fa = npt)->void{
        for(auto i = p->head; i; i = i->nxt)
            if(SON != fa)self(self, SON, p), p->siz += SON->siz;
        if(p->siz > 1)ans = max(ans, (ll)p->siz * p->len);
    }; dfs(dfs);

    printf("%lld\n", ans);

    // fprintf(stderr, "Time: %.6lf\n", (double)clock() / CLOCKS_PER_SEC);
    return 0;
}



template<typename T>
inline T read(void){
    T ret(0);
    short flag(1);
    char c = getchar();
    while(c != '-' && !isdigit(c))c = getchar();
    if(c == '-')flag = -1, c = getchar();
    while(isdigit(c)){
        ret *= 10;
        ret += int(c - '0');
        c = getchar();
    }
    ret *= flag;
    return ret;
}
```

```cpp
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <climits>
#include <iostream>
#include <string>
#include <unistd.h>
#define BASE 1000
#define MOD 10000
#define RANGE 9999
#define pow10(n) int(pow(10.0, double(n)))
//#define nxt(i, len) ((i < len && i) ? (++i) : (i = 0))
using namespace std;
typedef unsigned long long unll;
typedef long long ll;
template <typename T = int>
inline T read(void);
int c2d(char);
void PrintInt(char*, int*, int);
inline void nxt(int&, const int&, bool&);
class Integer{
    public:
        Integer(char*, int);
        Integer(vector<int>);
        void Init(void);
        Integer operator+(const Integer&);
        Integer operator-(const Integer&);//TODO Completion Required -*/
        void PrintInt(void);
    protected:
    private:
        int value_4[1100];
        int len;
        int real_len;
};

```

```
39
40  char c1[1100], c2[1100];
41  int main(){
42      scanf("%s%s", ::c1, ::c2);
43      Integer a(::c1, strlen(::c1)), b(::c2, strlen(::c2));
44      (a + b).PrintInt();
45
46
47      pause();
48      return 0;
49  }
50  Integer Integer::operator+(const Integer& addend){
51      vector<int>answer;
52      int adv(0); //advance
53      /*
54      //check if value_4[1] is less than BASE
55      int ans_1 = this -> value_4[1] + addend.value_4[1];
56      int len_1 = max(int(log10(this -> value_4[1])),
    int(log10(addend.value_4[1]))) + 1;
57      if(ans_1 >= pow10((len_1))){ans_1 %= (pow10((len_1))); adv =
    1;}
58      answer.push_back(ans_1);
59      printf("Add: get ans_1: %d\n", ans_1);*/
60      bool flag_i(true), flag_j(true);
61      for(int i = 1, j = 1; flag_i || flag_j; nxt(i, this -> len,
    flag_i), nxt(j, addend.len, flag_j)){
62          int ans = (flag_i ? this -> value_4[i] : 0) + (flag_j ?
    addend.value_4[j]: 0);
63          if(adv){ans += adv; adv = 0;}
64          if(ans > RANGE){adv = 1; ans %= MOD;}
65          answer.push_back(ans);
66          // printf("Add: get ans: %d\n", ans);
67      }
68      if(adv)answer.push_back(adv);
69      // printf("vector.ans: "); for(auto i : answer)printf("%d ",
    i);printf("\n");
70  //    int len = max(this -> len, adden.len) + adv;
71      reverse(answer.begin(), answer.end());
72      Integer ret(answer);
```

```
73      // ::PrintInt("Get Answer: ", ret.value_4, ret.len);
74      return ret;
75  }
76  void Integer::PrintInt(void){
77      for(int i = 1; i <= this -> len; ++i){
78          if(i != 1){
79              int num_0 = this -> value_4[i] ? (4 - int(log10(this
    -> value_4[i])) - 1) : 3;
80              for(int j = 1; j <= num_0; ++j)printf("0");
81          }
82          printf("%d", this -> value_4[i]);
83      }
84      printf("\n");
85  }
86  void Integer::Init(void){
87      reverse(this -> value_4 + 1, this -> value_4 + this -> len +
    1);
88      ::PrintInt("After init values : ", this -> value_4, this ->
    len);
89  }
90  Integer::Integer(vector<int> v){
91      memset(this -> value_4, 0, sizeof(this -> value_4));
92      this -> len = this -> real_len = v.size();
93      int cnt(0);
94      for(auto itea = v.begin(); itea != v.end(); ++itea)this ->
    value_4[++cnt] = *itea;
95  }
96  Integer::Integer(char *c, int len){
97      memset(this -> value_4, 0, sizeof(this -> value_4));
98      this -> len = int(ceil(len / 4.00));
99      this -> real_len = len;
100     /*
101     int nowPos(0);
102     for(int i = 1; i <= this -> len; ++i){
103         int num(0);
104         int base(BASE);
105         if(nowPos + 4 >= len)base = pow10((len - nowPos - 1));
106         for(int count = 1; count <= 4 && nowPos < len; ++count){
107             num += c2d(c[nowPos++]) * base;
```

```
108             base /= 10;
109         }
110 //        printf("Get number: %d\n", num);
111         this -> value_4[i] = num;
112     }*/
113     int nowPos(len - 1);//nowPos: 0 ~ len-1
114     for(int i = 1; i <= this -> len; ++i){
115         int num(0);
116         int base(1);
117 //        if(nowPos - 3 < 1)base = pow10((nowPos - 1));
118         for(int count = 1; count <= 4 && nowPos >= 0; ++count){
119             num += c2d(c[nowPos--]) * base;
120             base *= 10;
121         }
122 //        printf("Get number: %d\n", num);
123         this -> value_4[i] = num;
124     }
125     //  ::PrintInt("Read str to int: ", this -> value_4, this ->
    len);
126 //    this -> Init();
127 }
128 inline void nxt(int& i, const int& len, bool& flag){
129     if(!flag)return;
130     if(++i > len){flag = false; return;}
131     return;
132 }
133 void PrintInt(char *note, int *v, int len){
134     printf(note);
135     for(int i = 1; i <= len; ++i)
136         printf("%d ", v[i]);
137     printf("  len = %d\n", len);
138 }
139 int c2d(char c){
140     return int(c) - int('0');
141 }
142 template <typename T = int>
143 inline T read(void)
144 {
145     T ret(0);
```

```
    short flag(1);
    char c = getchar();
    while (c < '0' || c > '9') {
        if (c == '-')flag = -1;
        c = getchar();
    }
    while (c >= '0' && c <= '9') {
        ret *= 10, ret += (c - '0');
        c = getchar();
    }
    ret *= flag;
    return ret;
}
```

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

#define PI M_PI
#define E M_E

using namespace std;

mt19937 rnd(random_device{}());
int rndd(int l, int r){return rnd() % (r - l + 1) + l;}
bool rnddd(int x){return rndd(1, 100) <= x;}

typedef unsigned int uint;
typedef unsigned long long unll;
typedef long long ll;
typedef long double ld;

template < typename T = int >
inline T read(void);

int N;
const ll MOD = 998244353ll;
```

```cpp

ll qpow(ll a, ll b, ll mod = MOD){
    if(b < 0)return 0;
    ll ret(1), mul(a);
    while(b){
        if(b & 1)ret = ret * mul % mod;
        b >>= 1;
        mul = mul * mul % mod;
    }return ret;
}

const ll g(3), inv_g(qpow(g, MOD - 2));

enum Pattern{DFT, IDFT};

class Polynomial{
private:
    vector < int > pos;
public:
    int len;
    vector < ll > poly;
    Polynomial(void){
        this->len = 0;
        this->poly.resize(0), this->poly.shrink_to_fit();
    }
    Polynomial(int len){
        this->len = len;
        this->poly.assign(len, 0);
    }
    void Reverse(void){
        pos.resize(len);
        if(len > 0)pos[0] = 0;
        for(int i = 0; i < len; ++i)
            pos[i] = (pos[i >> 1] >> 1) | (i & 1 ? len >> 1 :
0);
        for(int i = 0; i < len; ++i)if(i < pos[i])swap(poly[i],
poly[pos[i]]);
    }
    void NTT(Pattern pat){

```

```
        Reverse();
        for(int siz = 2; siz <= len; siz <<= 1){
            ll gn = qpow(pat == DFT ? g : inv_g, (MOD - 1) / siz);
            for(auto p = poly.begin(); p < next(poly.begin(), len); advance(p, siz)){
                int mid = siz >> 1; ll g(1);
                for(int i = 0; i < mid; ++i, (g *= gn) %= MOD){
                    auto tmp = g * p[i + mid] % MOD;
                    p[i + mid] = (p[i] - tmp + MOD) % MOD;
                    p[i] = (p[i] + tmp) % MOD;
                }
            }
        }
        if(pat == IDFT){
            ll inv_len = qpow(len, MOD - 2);
            for(int i = 0; i < len; ++i)(poly[i] *= inv_len) %= MOD;
        }
    }
    void Resize(int len){
        this->poly.resize(len), this->len = len;
    }
};


class Bignum{
private:
public:
    basic_string < int > nums;
    friend Bignum operator + (Bignum a, Bignum b){
        // reverse(a.nums.begin(), a.nums.end());
        // reverse(b.nums.begin(), b.nums.end());
        while(a.nums.size() < b.nums.size())a.nums += 0;
        while(b.nums.size() < a.nums.size())b.nums += 0;
        Bignum ret; bool plus(false);
        for(int i = 0; i < (int)a.nums.size(); ++i){
            a.nums.at(i) += b.nums.at(i) + plus;
            plus = false;
```

```
            if(a.nums.at(i) >= 10)
                plus = true, a.nums.at(i) %= 10;
        }
        if(plus)a.nums += 1;
        // reverse(a.nums.begin(), a.nums.end());
        return a;
    }
    friend Bignum operator * (Bignum a, Bignum b){
        // reverse(a.nums.begin(), a.nums.end());
        // reverse(b.nums.begin(), b.nums.end());
        Bignum ret;
        for(int i = 1; i <= (int)(a.nums.size() +
b.nums.size()); ++i)ret.nums += 0;
        for(auto i = 0; i < (int)a.nums.size(); ++i)
            for(int j = 0; j < (int)b.nums.size(); ++j)
                ret.nums.at(i + j) += a.nums.at(i) *
b.nums.at(j);
        for(int i = 0; i < (int)ret.nums.size() - 1; ++i)
            ret.nums.at(i + 1) += ret.nums.at(i) / 10,
ret.nums.at(i) %= 10;
        if(ret.nums.back() >= 10)ret.nums += ret.nums.back() /
10, *prev(ret.nums.end(), 2) %= 10;
        while(ret.nums.size() > 1 && ret.nums.back() ==
0)ret.nums.pop_back();
        // reverse(ret.nums.begin(), ret.nums.end());
        return ret;
    }
    friend Bignum operator / (Bignum a, ll div){
        Bignum ret;
        ll cur(0); bool flag(false);
        for(auto i : a.nums){
            cur *= 10, cur += i;
            if(cur < div && !flag)continue;
            flag = true, ret.nums += cur / div, cur %= div;
        }return ret;
    }
    void Print(void){
        for(auto v : nums)printf("%d", v);
        printf("\n");
```

```
    }
};

Bignum qpow(Bignum a, ll b){
    Bignum ret, mul(a);
    ret.nums += 1;
    while(b){
        if(b & 1)ret = ret * mul;
        b >>= 1;
        mul = mul * mul;
    }return ret;
}

int main(){

    int T = read();
    while(T--){
        string SA, SB; cin >> SA >> SB;
        Polynomial A(SA.length()), B(SB.length());
        for(int i = 0; i < A.len; ++i)A.poly[i] = int(SA[A.len - i - 1] - '0');
        for(int i = 0; i < B.len; ++i)B.poly[i] = int(SB[B.len - i - 1] - '0');
        int clen = A.len + B.len - 1;
        int base(1); while(base < clen)base <<= 1;
        A.Resize(base), B.Resize(base);
        A.NTT(DFT), B.NTT(DFT);
        for(int i = 0; i < A.len; ++i)A.poly[i] = A.poly[i] * B.poly[i] % MOD;
        A.NTT(IDFT);

        int fst1 = A.len - 1;
        while(fst1 >= 0 && A.poly[fst1] == 0)--fst1;

        vector < ll > ans;
        for(auto it = A.poly.begin(); it != next(A.poly.begin(), fst1 + 1); advance(it, 1))ans.emplace_back(*it);

```

```
164         // for(int i = fst1; i >= 0; --i)printf("%lld",
    A.poly[i]);
165         // printf("\n");
166 
167         // if(ans.size() == 0){printf("0\n"); continue;}
168 
169         // for(int i = 0; i < ans.size(); ++i){
170         //     printf("i = %d, ans = %lld\n", i, ans[i]);
    fflush(stdout);
171         //     if(ans[i] <= 1)continue;
172         //     if(i + 4 > ans.size())ans.resize(i + 4, 0);
173         //     ans[i + 4] += ans[i] >> 1;
174         //     ans[i + 2] += ans[i] >> 1;
175         //     ans[i] %= 2;
176         // }
177         auto ToNegBinary = [](vector < ll >& d)->void{
178             for(int i = 0; i < d.size(); ++i){
179                 while(d[i] < 0 || d[i] > 1){
180                     ll r = ((d[i] % 2) + 2) % 2;
181                     ll q = (d[i] - r) / -2;
182                     d[i] = r;
183                     if(i + 1 >= d.size())d.resize(i + 2, 0);
184                     d[i + 1] += q;
185                 }
186             }
187             while(d.size() > 1 && d.back() == 0)d.pop_back();
188         };
189 
190         vector < ll > even, odd;
191         for(int i = 0; i < ans.size(); ++i)
192             (i & 1 ? odd : even).emplace_back(ans[i]);
193 
194         ToNegBinary(even);
195         ToNegBinary(odd);
196 
197         int M(max(even.size(), odd.size()));
198         vector < ll > res(2 * M, 0);
199         for(int i = 0; i < M; ++i){
200             if(i < even.size())res[i << 1] = even[i];
```

```
            if(i < odd.size())res[(i << 1) | 1] = odd[i];
        }
        while(res.size() > 1 && res.back() == 0) res.pop_back();

        if(res.empty()) puts("0");
        else{
            for(auto it = res.rbegin(); it != res.rend(); ++it)
                printf("%lld", *it);
            printf("\n");
        }

        // for(auto it = ans.rbegin(); it != ans.rend();
++it)printf("%lld", *it);



    }

    // fprintf(stderr, "Time: %.6lf\n", (double)clock() /
CLOCKS_PER_SEC);
    return 0;
}

template < typename T >
inline T read(void){
    T ret(0);
    int flag(1);
    char c = getchar();
    while(c != '-' && !isdigit(c))c = getchar();
    if(c == '-')flag = -1, c = getchar();
    while(isdigit(c)){
        ret *= 10;
        ret += int(c - '0');
        c = getchar();
    }
    ret *= flag;
    return ret;
}
```

```cpp
#define _USE_MATH_DEFINES
#include <bits/stdc++.h>

#define PI M_PI
#define E M_E

using namespace std;

mt19937 rnd(random_device{}());
int rndd(int l, int r){return rnd() % (r - l + 1) + l;}

typedef unsigned int uint;
typedef unsigned long long unll;
typedef long long ll;

template<typename T = int>
inline T read(void);

auto qpow = [](ll a, ll b, ll mod)->ll{
    ll ret(1), mul(a);
    while(b){
        if(b & 1)ret = (ret * mul) % mod;
        b >>= 1, mul = (mul * mul) % mod;
    }return ret;
};

int N, Q, B;

class LengthContainer{
private:
    multiset < int > lens;
    vector < ll > curG;
    ll len0;

public:
    int prel, sufl, mxl;
```

```
    ll Cal(int b, int len){
        if(len <= (b << 1))return 0;
        int num = len - (b << 1), div = (b << 1) | 1;
        return (num + div - 1) / div;
    }
    void InsertLen(int len){
        if(len <= 0)return;
        len0 += len;
        lens.insert(len);
        if(lens.size()){
            auto it = prev(lens.end());
            if(!lens.empty())mxl = max(mxl, *it);
            mxl = max(mxl, len);
        }else mxl = max(mxl, len);
        for(int b = 0; b <= B; ++b)curG[b] += Cal(b, len);
    }
    void RemoveLen(int len){
        if(len <= 0)return;
        len0 -= len;
        auto it = lens.find(len);
        if(it != lens.end())lens.erase(it);
        for(int b = 0; b <= B; ++b)curG[b] -= Cal(b, len);
        mxl = lens.empty() ? 0 : *prev(lens.end());
    }
    ll Query(void){
        if(!len0)return 0;
        ll ans(LONG_LONG_MAX >> 2);
        int lmx = max(mxl, prel + sufl);
        ans = min(ans, (ll)((lmx + 1) >> 1));
        if(len0 == N){
            for(int b = 0; b <= B; ++b){
                ll val = b + 1 + Cal(b, N - 1);
                if(val < ans)ans = val;
            }
            return ans;
        }
        for(int b = 0; b <= B; ++b){
            ll res(0);
```

```
            if(prel > 0 && sufl > 0) res = Cal(b, prel + sufl) - Cal(b, prel) - Cal(b, sufl);
            ll val = (ll)b + curG[b] + res;
            if(val < ans)ans = val;
        }return ans;
    }
    void Clear(void){
        lens.clear();
        curG.assign(B + 1, 0);
        len0 = prel = sufl = mxl = 0;
    }
}lc;


struct Node{
    int l, r;
    int size(void)const{return r - l + 1;}
    mutable ll val;
    friend const bool operator < (const Node &x, const Node &y) {return x.l < y.l;}
};

class ODT{
private:
    set < Node > tr;
public:
    auto Insert(Node p){return tr.insert(p);}
    auto Split(int p){
        auto it = tr.lower_bound(Node{p});
        if(it != tr.end() && p == it->l)return it;
        if(it == tr.begin()) return tr.end();
        --it;
        if(p > it->r)return tr.end();
        int l = it->l, r = it->r;
        ll val = it->val;
        if(!val) lc.RemoveLen(r - l + 1);
        tr.erase(it);
        if(l <= p - 1){
            Insert(Node{l, p - 1, val});
```

```
            if(!val) lc.InsertLen(p - 1 - l + 1);
        }
        auto ret = Insert(Node{p, r, val}).first;
        if(!val)lc.InsertLen(r - p + 1);
        return ret;
    }
    void Modify(int l, int r, ll val){
        auto itR = Split(r + 1), itL = Split(l);
        for(auto it = itL; it != itR; ++it)it->val += val;
    }
    void Assign(int l, int r, ll val){
        auto itR = Split(r + 1), itL = Split(l);
        tr.erase(itL, itR);
        Insert(Node{l, r, val});
    }
    ll QueryKth(int l, int r, int k){
        vector < Node > rnk;
        auto itR = Split(r + 1), itL = Split(l);
        for(auto it = itL; it != itR; ++it)rnk.push_back(*it);
        sort(rnk.begin(), rnk.end(), [](const Node x, const Node
y)->bool{return x.val < y.val;});
        int cur(0);
        for(auto i : rnk){
            cur += i.size();
            if(cur >= k)return i.val;
        }
        return -1;
    }
    ll QuerySum(int l, int r, ll k, ll mod){
        ll ret(0);
        auto itR = Split(r + 1), itL = Split(l);
        for(auto it = itL; it != itR; ++it)
            ret = (ret + qpow(it->val, k, mod) * it->size() %
mod) % mod;
        return ret;
    }
    void Build(const vector < ll > &A){
        tr.clear();
        for(int i = 1; i <= N;){
```

```
            int cur(i);
            while(cur + 1 <= N && A[cur + 1] == A[i])++cur;
            Insert(Node{i, cur, A[i]});
            i = cur + 1;
        }
    }

    void Assign01(int l, int r, int val){
        auto itR = Split(r + 1), itL = Split(l);
        for(auto it = itL; it != itR; ++it)
            if(!it->val)lc.RemoveLen(it->size());
        tr.erase(itL,itR);
        if(val == 1){
            auto it = tr.lower_bound(Node{l});
            int L(l), R(r);
            if(it != tr.begin() && prev(it)->r == l - 1 &&
prev(it)->val == 1)
                L = prev(it)->l, tr.erase(prev(it));
            it = tr.lower_bound(Node{r + 1});
            if(it != tr.end() && it->l == r + 1 && it->val == 1)
                R = it->r, tr.erase(it);
            Insert(Node{L, R, 1});
        }else{
            auto it = tr.lower_bound(Node{l});
            int L(l), R(r);
            if(it != tr.begin() && prev(it)->r == l - 1 &&
prev(it)->val == 0)
                lc.RemoveLen(prev(it)->size()), L = prev(it)->l,
tr.erase(prev(it));
            it = tr.lower_bound(Node{r + 1});
            if(it != tr.end() && it->l == r + 1 && it->val == 0)
                lc.RemoveLen(it->size()), R = it->r,
tr.erase(it);
            Insert(Node{L, R, 0});
            lc.InsertLen(R - L + 1);
        }
    }
    void Maintain(){
        if(tr.empty()){lc.prel = lc.sufl = 0; return;}
```

```
        lc.prel = (tr.begin()->l == 1 && tr.begin()->val == 0) ? tr.begin()->size() : 0;
        lc.sufl = (prev(tr.end())->r == N && prev(tr.end())->val == 0) ? prev(tr.end())->size() : 0;
    }
    void Initialize(void){
        for(auto i : tr)
            if(i.val == 0)lc.InsertLen(i.size());
    }
}odt;

int main(){
    int T = read();
    while(T--){
        N = read(), Q = read();
        vector < ll > A(N + 10);
        string S; cin >> S;
        for(int i = 1; i <= N; ++i)A[i] = S.at(i - 1) - '0';

        odt.Build(A);
        B = (int)sqrt((double)N) + 1;

        lc.Clear();
        odt.Initialize();
        odt.Maintain();

        while(Q--){
            int opt = read(), l = read(), r = read();
            odt.Assign01(l, r, opt == 1 ? 1 : 0);
            odt.Maintain();
            printf("%lld\n", lc.Query());
        }
    }


    return 0;
}

template<typename T>
```

```
inline T read(void){
    T ret(0);
    short flag(1);
    char c = getchar();
    while(c != '-' && !isdigit(c))c = getchar();
    if(c == '-')flag = -1, c = getchar();
    while(isdigit(c)){
        ret *= 10;
        ret += int(c - '0');
        c = getchar();
    }
    ret *= flag;
    return ret;
}

```