

# 模板汇总

---

模板汇总

编译命令

标准模板 (C++11)

快读

快速幂

## # 编译命令

---

```
1 | 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
```

```
1 | g++ test.cpp -o 1 -std=c++17 -pg -  
   | fsanitize=address,undefined,signed-integer-overflow,memory -fno-  
   | omit-frame-pointer  
2 |
```

## # 标准模板 (C++11)

---

```
1 | #define _USE_MATH_DEFINES  
2 | #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)){

```

```

14         ret *= 10;
15         ret += int(c - '0');
16         c = getchar();
17     }
18     ret *= flag;
19     return ret;
20 }

```

```

1  char buf[1<<23], *p1=buf, *p2=buf, obuf[1<<23], *0=obuf;
2  #define getchar() (p1==p2&&(p2=
    (p1=buf)+fread(buf, 1, 1<<21, stdin), p1==p2)?EOF:*p1++)
3  inline int read() {
4      int x=0, f=1; char ch=getchar();
5      while(!isdigit(ch)){if(ch=='-') f=-1; ch=getchar();}
6      while(isdigit(ch)) x=x*10+(ch^48), ch=getchar();
7      return x*f;
8  }
9  void print(long long x) {
10     if(x>9) print(x/10);
11     *0++=x%10+'0';
12 }
13 fwrite(obuf, 0-obuf, 1, stdout);

```

## # 快速幂

---

```

1  auto qpow = [](ll a, ll b)->ll{
2      ll ret(1), mul(a);
3      while(b){
4          if(b & 1) ret = (ret * mul) % MOD;
5          b >>= 1, mul = (mul * mul) % MOD;
6      } return ret;
7  };

```

```

1  #define _USE_MATH_DEFINES
2  #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 bool rnddd(int x){return rndd(1, 100) <= x;}
12
13 typedef unsigned int uint;
14 typedef unsigned long long ull;
15 typedef long long ll;
16 typedef long double ld;
17
18 template < typename T = int >
19 inline T read(void);
20
21 const ll MOD = 998244353ll;
22
23 auto qpow = [](ll a, ll b, ll mod = MOD)->ll{
24     if(b < 0)return 0;
25     ll ret(1), mul(a);
26     while(b){
27         if(b & 1)ret = ret * mul % mod;
28         b >>= 1;
29         mul = mul * mul % mod;
30     }return ret;
31 };
32
33 const ll g = 3;
34 const ll invg = qpow(g, MOD - 2);
35 const ll inv2 = qpow(2, MOD - 2);
36 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;

```

```

71     }
72 }
73 void Resize(int len){
74     this->poly.resize(len, 0);
75 }
76 void Derivate(void){
77     int len = poly.size();
78     if(len == 0)return;
79     poly[0] = 0;
80     for(int i = 1; i < len; ++i)poly[i - 1] = i * poly[i] %
MOD, poly[i] = 0;
81     Resize(len - 1);
82 }
83 void Integrate(void){
84     int len = poly.size();
85     if(len == 0)return;
86     Resize(len + 1);
87     for(int i = len - 1; i >= 0; --i)poly[i + 1] = poly[i] *
qpow(i + 1, MOD - 2) % MOD, poly[i] = 0;
88 }
89 auto Desc(void){
90     int len = poly.size();
91     // printf("Polynomial(len = %d): ", len);
92     for(int i = 0; i < len; ++i)printf("%lld%c", poly[i], i
== len - 1 ? '\n' : ' ');
93     return this;
94 }
95 };
96
97
98 auto Multiply = [](Polynomial* baseA, Polynomial* baseB)-
>Polynomial*{
99     auto A = new Polynomial(*baseA), B = new Polynomial(*baseB);
100     int len = A->poly.size() + B->poly.size() - 1;
101     int base(1); while(base < (len << 1))base <=< 1;
102     Polynomial* ret = new Polynomial(base);
103     A->Resize(base), B->Resize(base);
104     A->NTT(DFT), B->NTT(DFT);
105     for(int i = 0; i < base; ++i)

```

```

106         ret->poly[i] = (A->poly[i] * B->poly[i] % MOD);
107     ret->NTT(IDFT);
108     ret->Resize(len);
109     delete A; delete B;
110     return ret;
111 };
112
113 auto Inverse = [](auto&& self, Polynomial* baseF, int len)-
    >Polynomial*{
114     if(len == 1){
115         Polynomial *G = new Polynomial(1);
116         G->poly[0] = qpow(baseF->poly[0], MOD - 2);
117         return G;
118     }
119     auto *H = self(self, baseF, (len + 1) >> 1);
120     int base(1); while(base < (len << 1))base <= 1;
121     H->Resize(base);
122     Polynomial *G = new Polynomial(base), *F = new
    Polynomial(base);
123     for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F-
    >poly[i] = baseF->poly[i];
124     H->NTT(DFT), F->NTT(DFT);
125     for(int i = 0; i < base; ++i)
126         G->poly[i] = (2 * H->poly[i] % MOD - H->poly[i] * H-
    >poly[i] % MOD * F->poly[i] % MOD + MOD) % MOD;
127     G->NTT(IDFT), G->Resize(len);
128     delete H; delete F;
129     return G;
130 };
131
132 //Require A[0] == 1
133 auto Sqrt = [](auto&& self, Polynomial* baseF, int len)-
    >Polynomial*{
134     if(len == 1){
135         Polynomial *G = new Polynomial(1);
136         G->poly[0] = sqrt(baseF->poly[0]);
137         return G;
138     }
139     auto H = self(self, baseF, (len + 1) >> 1);

```

```

140     auto invH = Inverse(Inverse, H, len);
141     int base(1); while(base < (len << 1))base <= 1;
142     auto G = new Polynomial(base), F = new Polynomial(len);
143     for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F-
>poly[i] = baseF->poly[i];
144     H->Resize(base), invH->Resize(base), F->Resize(base);
145     H->NTT(DFT), F->NTT(DFT), invH->NTT(DFT);
146     for(int i = 0; i < base; ++i)G->poly[i] = (F->poly[i] *
invH->poly[i] % MOD + H->poly[i]) % MOD * inv2 % MOD;
147     G->NTT(IDFT), G->Resize(len);
148     delete H; delete invH; delete F;
149     return G;
150 };
151 auto Ln = [] (Polynomial* baseF, int len)->Polynomial*{
152     auto F = new Polynomial(len);
153     for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)
154         F->poly[i] = baseF->poly[i];
155     auto invF = Inverse(Inverse, F, len);
156     F->Derivate();
157     int clen = F->poly.size() + invF->poly.size() - 1;
158     int base(1); while(base < clen)base <= 1;
159     Polynomial* G = new Polynomial(base);
160     F->Resize(base), invF->Resize(base);
161     F->NTT(DFT), invF->NTT(DFT);
162     for(int i = 0; i < base; ++i)
163         G->poly[i] = F->poly[i] * invF->poly[i] % MOD;
164     G->NTT(IDFT), G->Resize(len - 1);
165     G->Integrate();
166     delete invF; delete F;
167     return G;
168 };
169 auto Exp = [] (auto&& self, Polynomial* baseF, int len)-
>Polynomial*{
170     if(len == 1){
171         Polynomial* G = new Polynomial(1);
172         G->poly[0] = 1;
173         return G;
174     }
175     auto H = self(self, baseF, (len + 1) >> 1);

```



```

176     auto lnH = Ln(H, len);
177     int base(1); while(base < (len << 1))base <= 1;
178     auto F = new Polynomial(len), G = new Polynomial(base);
179     for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F-
>poly[i] = baseF->poly[i];
180     F->Resize(base), H->Resize(base), lnH->Resize(base);
181     F->NTT(DFT), H->NTT(DFT), lnH->NTT(DFT);
182     for(int i = 0; i < base; ++i)
183         G->poly[i] = (H->poly[i] * ((1 - lnH->poly[i] + MOD) %
MOD) % MOD + F->poly[i] * H->poly[i] % MOD) % MOD;
184     G->NTT(IDFT), G->Resize(len);
185     delete F; delete lnH; delete H;
186     return G;
187 };
188
189 auto Quickpow = [] (Polynomial* baseF, ll k1, ll k2, ll mx)-
>Polynomial*{
190     int len = baseF->poly.size();
191     if(baseF->poly[0] == 0 && mx >= len){
192         Polynomial* G = new Polynomial(len);
193         for(int i = 0; i < len; ++i)
194             G->poly[i] = 0;
195         return G;
196     }
197     if(len == 1){
198         Polynomial* G = new Polynomial(1);
199         G->poly[0] = qpow(baseF->poly[0], k2);
200         return G;
201     }
202     int offset(0);
203     while (offset < len && baseF->poly[offset] == 0) ++offset;
204     if((ll)offset * k1 >= len)return new Polynomial(len);
205     ll mul = qpow(baseF->poly[offset], k2), inv = qpow(baseF-
>poly[offset], MOD - 2);
206     auto F = new Polynomial(*baseF);
207     for(int i = 0; i + offset < len; ++i)F->poly[i] = F->poly[i
+ offset] * inv % MOD;
208     for(int i = len - offset; i < len; ++i)F->poly[i] = 0;
209     auto lnF = Ln(F, len);

```

```

210     for(int i = 0; i < len; ++i) lnF->poly[i] = lnF->poly[i] * k1
% MOD;
211     auto eLnF = Exp(Exp, lnF, len);
212     ll shift = offset * k1;
213     for(int i = len - 1; i >= shift; --i)
214         eLnF->poly[i] = eLnF->poly[i - shift];
215     for(int i = 0; i < shift; ++i)
216         eLnF->poly[i] = 0;
217     for(auto i = 0; i < len; ++i) eLnF->poly[i] = eLnF->poly[i] *
mul % MOD;
218     delete lnF; delete F;
219     return eLnF;
220 };
221
222 struct Complex{
223     ll x, y;
224     static ll w;
225     friend Complex operator *(const Complex &a, const Complex
&b){
226         return Complex{
227             (a.x * b.x % MOD + w * a.y % MOD * b.y % MOD) % MOD,
228             (a.x * b.y % MOD + a.y * b.x % MOD) % MOD
229         };
230     }
231     static ll qpow(Complex a, ll b){
232         Complex ret{1, 0};
233         while(b){
234             if(b & 1) ret = ret * a;
235             a = a * a;
236             b >>= 1;
237         } return ret.x;
238     }
239 };
240 ll Complex::w;
241 auto Cipolla = [](ll x)->ll{
242     if(qpow(x, (MOD - 1) >> 1) == MOD - 1) return -1;
243     while(true){
244         ll a = (1ll * rnd() << 15 | rnd()) % MOD;
245         Complex::w = (a * a % MOD + MOD - x) % MOD;

```

```

246         if(qpow(Complex::w, (MOD - 1) >> 1) == MOD - 1) {
247             ll res = Complex::qpow(Complex{a, 1}, (MOD + 1) >>
1);
248             return min(res, MOD - res);
249         }
250     }
251 };
252
253 //Require A[0] is a quadratic residue modulo 998244353
254 auto ExSqrt = [](auto&& self, Polynomial* baseF, int len)-
>Polynomial*{
255     if(len == 1){
256         Polynomial *G = new Polynomial(1);
257         auto res = Cipolla(baseF->poly[0]);
258         G->poly[0] = min(res, MOD - res);
259         return G;
260     }
261     auto H = self(self, baseF, (len + 1) >> 1);
262     auto invH = Inverse(Inverse, H, len);
263     int base(1); while(base < (len << 1))base <= 1;
264     auto G = new Polynomial(base), F = new Polynomial(len);
265     for(int i = 0; i < min(len, (int)baseF->poly.size()); ++i)F-
>poly[i] = baseF->poly[i];
266     H->Resize(base), invH->Resize(base), F->Resize(base);
267     H->NTT(DFT), F->NTT(DFT), invH->NTT(DFT);
268     for(int i = 0; i < base; ++i)G->poly[i] = (F->poly[i] *
invH->poly[i] % MOD + H->poly[i]) % MOD * inv2 % MOD;
269     G->NTT(IDFT), G->Resize(len);
270     delete H; delete invH; delete F;
271     return G;
272 };
273
274 // auto CDQ = [](Polynomial* baseF, int l, int r)->Polynomial*{
275 //     if(l == r)return BuildPoly(l);
276 //     int mid = (l + r) >> 1;
277 //     Polynomial L = CDQ(l, mid, mx), R = CDQ(mid + 1, r, mx);
278 //     // for(int i = l; i <= r; ++i)pol[i].poly.clear(),
pol[i].poly.shrink_to_fit(), pol[i].len = 0;
279 //     return Mul(L, R, mx);

```

```

280 // }
281
282 namespace Tests{
283     auto ImplementMultiply = [] (void)->void{
284         int N = read() + 1, M = read() + 1;
285         Polynomial *A = new Polynomial(N), *B = new
Polynomial(M);
286         for(int i = 0; i < N; ++i)A->poly[i] = read();
287         for(int i = 0; i < M; ++i)B->poly[i] = read();
288         delete Multiply(A, B)->Desc();
289         delete A; delete B;
290     };
291     auto ImplementInverse = [] (void)->void{
292         int N = read();
293         Polynomial *A = new Polynomial(N);
294         for(int i = 0; i < N; ++i)A->poly[i] = read();
295         delete Inverse(Inverse, A, N)->Desc();
296         delete A;
297     };
298     auto ImplementLn = [] (void)->void{
299         int N = read();
300         Polynomial *A = new Polynomial(N);
301         for(int i = 0; i < N; ++i)A->poly[i] = read();
302         delete Ln(A, N)->Desc();
303         delete A;
304     };
305     auto ImplementExp = [] (void)->void{
306         int N = read();
307         Polynomial *A = new Polynomial(N);
308         for(int i = 0; i < N; ++i)A->poly[i] = read();
309         delete Exp(Exp, A, N)->Desc();
310         delete A;
311     };
312     auto ImplementSqrt = [] (void)->void{
313         int N = read();
314         Polynomial *A = new Polynomial(N);
315         for(int i = 0; i < N; ++i)A->poly[i] = read();
316         delete Sqrt(Sqrt, A, N)->Desc();
317         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() /
352     CLOCKS_PER_SEC);
353     return 0;
354 }
355 template < typename T >

```

```

356 inline T read(void){
357     T ret(0);
358     int flag(1);
359     char c = getchar();
360     while(c != '-' && !isdigit(c))c = getchar();
361     if(c == '-')flag = -1, c = getchar();
362     while(isdigit(c)){
363         ret *= 10;
364         ret += int(c - '0');
365         c = getchar();
366     }
367     ret *= flag;
368     return ret;
369 }

```

```

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

```

```

1  #define _USE_MATH_DEFINES
2  #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 ull;
14 typedef long long ll;
15
16 #define EPS (1e-9)
17
18 template<typename T = int>
19 inline T read(void);

```

```

20
21 struct Fraction{//non-negative
22     __int128_t a, b;
23     Fraction Shrink(void){
24         __int128_t div = __gcd(a, b);
25         a /= div, b /= div;
26         return *this;
27     }
28     friend const Fraction operator + (const Fraction &x, const
Fraction &y){
29         __int128_t below = x.b * y.b / __gcd(x.b, y.b);
30         return Fraction{below / x.b * x.a + below / y.b * y.a,
below}.Shrink();
31     }
32     friend const Fraction operator / (const Fraction &x, const
int &v){
33         return Fraction{x.a, x.b * v}.Shrink();
34     }
35     friend const Fraction operator / (const Fraction &x, const
Fraction &y){
36         return Fraction{x.a * y.b, x.b * y.a}.Shrink();
37     }
38     friend const Fraction operator * (const Fraction &x, const
Fraction &y){
39         return Fraction{x.a * y.a, x.b * y.b}.Shrink();
40     }
41     friend const bool operator <= (const Fraction &x, const
Fraction &y){
42         return x.a * y.b <= y.a * x.b;
43     }
44     friend const bool operator >= (const Fraction &x, const
Fraction &y){
45         return x.a * y.b > y.a * x.b;
46     }
47     friend const bool operator < (const Fraction &x, const
Fraction &y){
48         return x.a * y.b < y.a * x.b;
49     }

```

```

50     friend const bool operator > (const Fraction &x, const
Fraction &y){
51         return x.a * y.b > y.a * x.b;
52     }
53     void Desc(void){
54         this->Shrink();
55         printf("%lld/%lld\n", (ll)this->a, (ll)this->b);
56     }
57 };
58
59 int main(){
60
61     // fprintf(stderr, "Time: %.6lf\n", (double)clock() /
CLOCKS_PER_SEC);
62     return 0;
63 }
64
65
66
67 template<typename T>
68 inline T read(void){
69     T ret(0);
70     short flag(1);
71     char c = getchar();
72     while(c != '-' && !isdigit(c))c = getchar();
73     if(c == '-')flag = -1, c = getchar();
74     while(isdigit(c)){
75         ret *= 10;
76         ret += int(c - '0');
77         c = getchar();
78     }
79     ret *= flag;
80     return ret;
81 }

```

```

1  /*
2   * LG-P3804 【模板】后缀自动机 (SAM)
3   * https://www.luogu.com.cn/problem/P3804

```



```

4      */
5
6      #define _USE_MATH_DEFINES
7      #include <bits/stdc++.h>
8
9      #define PI M_PI
10     #define E M_E
11
12     using namespace std;
13
14     mt19937 rnd(random_device{}());
15     int rndd(int l, int r){return rnd() % (r - l + 1) + l;}
16     bool rnddd(int x){return rndd(1, 100) <= x;}
17
18     typedef unsigned int uint;
19     typedef unsigned long long unll;
20     typedef long long ll;
21     typedef long double ld;
22
23     template < typename T = int >
24     inline T read(void);
25
26     #define d(c) (c - 'a')
27     #define npt nullptr
28     #define SON i->to
29
30     struct Edge;
31     struct Node{
32         unordered_map < char, Node* > trans;
33         Node* link;
34         int len;
35         int siz;
36         Edge* head;
37         void* operator new(size_t);
38     }nd[2100000];
39     void* Node::operator new(size_t){static Node* P = nd; return
    P++;}
40
41     struct Edge{

```

```

42     Edge* nxt;
43     Node* to;
44     void* operator new(size_t);
45 }ed[4100000];
46 void* Edge::operator new(size_t){static Edge* P = ed; return
P++;}
47
48 class SAM{
49 private:
50 public:
51     Node* root;
52     void Insert(int c){
53         static Node* lst = root;
54         Node* p = lst; Node* cp = lst = new Node; cp->siz = 1;
55         cp->len = p->len + 1;
56         while(p && !p->trans[c])p->trans[c] = cp, p = p->link;
57         if(!p)cp->link = root;
58         else if(p->trans[c]->len == p->len + 1)cp->link = p-
>trans[c];
59         else{
60             auto q = p->trans[c], sq = new Node(*q); sq->siz =
0;
61             sq->len = p->len + 1;
62             cp->link = q->link = sq;
63             while(p && p->trans[c] == q)p->trans[c] = sq, p = p-
>link;
64         }
65     }
66     void Link(void){
67         auto endp = new Node();
68         for(auto p = nd; p != endp;++p)
69             if(p->link)
70                 p->head = new Edge{p->head, p->link},
71                 p->link->head = new Edge{p->link->head, p};
72     }
73
74 }sam;
75
76 ll ans(0);

```

```

77 string S;
78
79 int main(){
80     sam.root = new Node();
81     cin >> S;
82     for(auto c : S)sam.Insert(d(c));
83     sam.Link();
84     auto dfs = [](auto&& self, Node* p = sam.root, Node* fa =
npt)->void{
85         for(auto i = p->head; i; i = i->nxt)
86             if(SON != fa)self(self, SON, p), p->siz += SON->siz;
87             if(p->siz > 1)ans = max(ans, (ll)p->siz * p->len);
88     }; dfs(dfs);
89
90     printf("%lld\n", ans);
91
92     // fprintf(stderr, "Time: %.6lf\n", (double)clock() /
CLOCKS_PER_SEC);
93     return 0;
94 }
95
96
97
98 template<typename T>
99 inline T read(void){
100     T ret(0);
101     short flag(1);
102     char c = getchar();
103     while(c != '-' && !isdigit(c))c = getchar();
104     if(c == '-')flag = -1, c = getchar();
105     while(isdigit(c)){
106         ret *= 10;
107         ret += int(c - '0');
108         c = getchar();
109     }
110     ret *= flag;
111     return ret;
112 }

```

```

1  #include <cstdio>
2  #include <algorithm>
3  #include <cstring>
4  #include <cstdlib>
5  #include <cmath>
6  #include <vector>
7  #include <climits>
8  #include <iostream>
9  #include <string>
10 #include <unistd.h>
11 #define BASE 1000
12 #define MOD 10000
13 #define RANGE 9999
14 #define pow10(n) int(pow(10.0, double(n)))
15 // #define nxt(i, len) ((i < len && i) ? (++i) : (i = 0))
16 using namespace std;
17 typedef unsigned long long ull;
18 typedef long long ll;
19 template <typename T = int>
20 inline T read(void);
21 int c2d(char);
22 void PrintInt(char*, int*, int);
23 inline void nxt(int&, const int&, bool&);
24 class Integer{
25     public:
26         Integer(char*, int);
27         Integer(vector<int>);
28         void Init(void);
29         Integer operator+(const Integer&);
30         Integer operator-(const Integer&); // TODO Completion
31         Required -*/
32         void PrintInt(void);
33     protected:
34     private:
35         int value_4[1100];
36         int len;
37         int real_len;
38 };

```

```

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])),
57 int(log10(addend.value_4[1]))) + 1;
58     if(ans_1 >= pow10((len_1))){ans_1 %= (pow10((len_1))); adv =
59 1;}
60     answer.push_back(ans_1);
61     printf("Add: get ans_1: %d\n", ans_1);*/
62     bool flag_i(true), flag_j(true);
63     for(int i = 1, j = 1; flag_i || flag_j; nxt(i, this -> len,
64 flag_i), nxt(j, addend.len, flag_j)){
65         int ans = (flag_i ? this -> value_4[i] : 0) + (flag_j ?
66 addend.value_4[j] : 0);
67         if(adv){ans += adv; adv = 0;}
68         if(ans > RANGE){adv = 1; ans %= MOD;}
69         answer.push_back(ans);
70         // printf("Add: get ans: %d\n", ans);
71     }
72     if(adv)answer.push_back(adv);
73     // printf("vector.ans: "); for(auto i : answer)printf("%d ",
74 i);printf("\n");
75     // int len = max(this -> len, addend.len) + adv;
76     reverse(answer.begin(), answer.end());
77     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);

```

```

146     short flag(1);
147     char c = getchar();
148     while (c < '0' || c > '9') {
149         if (c == '-')flag = -1;
150         c = getchar();
151     }
152     while (c >= '0' && c <= '9') {
153         ret *= 10, ret += (c - '0');
154         c = getchar();
155     }
156     ret *= flag;
157     return ret;
158 }

```

```

1  #define _USE_MATH_DEFINES
2  #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 bool rnddd(int x){return rndd(1, 100) <= x;}
12
13 typedef unsigned int uint;
14 typedef unsigned long long ull;
15 typedef long long ll;
16 typedef long double ld;
17
18 template < typename T = int >
19 inline T read(void);
20
21 int N;
22 const ll MOD = 99824435311;

```



```

23
24 ll qpow(ll a, ll b, ll mod = MOD){
25     if(b < 0)return 0;
26     ll ret(1), mul(a);
27     while(b){
28         if(b & 1)ret = ret * mul % mod;
29         b >>= 1;
30         mul = mul * mul % mod;
31     }return ret;
32 }
33
34 const ll g(3), inv_g(qpow(g, MOD - 2));
35
36 enum Pattern{DFT, IDFT};
37
38 class Polynomial{
39 private:
40     vector < int > pos;
41 public:
42     int len;
43     vector < ll > poly;
44     Polynomial(void){
45         this->len = 0;
46         this->poly.resize(0), this->poly.shrink_to_fit();
47     }
48     Polynomial(int len){
49         this->len = len;
50         this->poly.assign(len, 0);
51     }
52     void Reverse(void){
53         pos.resize(len);
54         if(len > 0)pos[0] = 0;
55         for(int i = 0; i < len; ++i)
56             pos[i] = (pos[i >> 1] >> 1) | (i & 1 ? len >> 1 :
0);
57         for(int i = 0; i < len; ++i)if(i < pos[i])swap(poly[i],
poly[pos[i]]);
58     }
59     void NTT(Pattern pat){

```

```

60         Reverse();
61         for(int siz = 2; siz <= len; siz <= 1){
62             ll gn = qpow(pat == DFT ? g : inv_g, (MOD - 1) /
siz);
63             for(auto p = poly.begin(); p < next(poly.begin(),
len); advance(p, siz)){
64                 int mid = siz >> 1; ll g(1);
65                 for(int i = 0; i < mid; ++i, (g *= gn) %= MOD){
66                     auto tmp = g * p[i + mid] % MOD;
67                     p[i + mid] = (p[i] - tmp + MOD) % MOD;
68                     p[i] = (p[i] + tmp) % MOD;
69                 }
70             }
71         }
72         if(pat == IDFT){
73             ll inv_len = qpow(len, MOD - 2);
74             for(int i = 0; i < len; ++i)(poly[i] *= inv_len) %=
MOD;
75         }
76     }
77     void Resize(int len){
78         this->poly.resize(len), this->len = len;
79     }
80 };
81
82
83 class Bignum{
84 private:
85 public:
86     basic_string < int > nums;
87     friend Bignum operator + (Bignum a, Bignum b){
88         // reverse(a.nums.begin(), a.nums.end());
89         // reverse(b.nums.begin(), b.nums.end());
90         while(a.nums.size() < b.nums.size())a.nums += 0;
91         while(b.nums.size() < a.nums.size())b.nums += 0;
92         Bignum ret; bool plus(false);
93         for(int i = 0; i < (int)a.nums.size(); ++i){
94             a.nums.at(i) += b.nums.at(i) + plus;
95             plus = false;

```

```

96         if(a.nums.at(i) >= 10)
97             plus = true, a.nums.at(i) %= 10;
98     }
99     if(plus)a.nums += 1;
100    // reverse(a.nums.begin(), a.nums.end());
101    return a;
102 }
103 friend Bignum operator * (Bignum a, Bignum b){
104     // reverse(a.nums.begin(), a.nums.end());
105     // reverse(b.nums.begin(), b.nums.end());
106     Bignum ret;
107     for(int i = 1; i <= (int)(a.nums.size() +
b.nums.size()); ++i)ret.nums += 0;
108     for(auto i = 0; i < (int)a.nums.size(); ++i)
109         for(int j = 0; j < (int)b.nums.size(); ++j)
110             ret.nums.at(i + j) += a.nums.at(i) *
b.nums.at(j);
111     for(int i = 0; i < (int)ret.nums.size() - 1; ++i)
112         ret.nums.at(i + 1) += ret.nums.at(i) / 10,
ret.nums.at(i) %= 10;
113     if(ret.nums.back() >= 10)ret.nums += ret.nums.back() /
10, *prev(ret.nums.end(), 2) %= 10;
114     while(ret.nums.size() > 1 && ret.nums.back() ==
0)ret.nums.pop_back();
115     // reverse(ret.nums.begin(), ret.nums.end());
116     return ret;
117 }
118 friend Bignum operator / (Bignum a, ll div){
119     Bignum ret;
120     ll cur(0); bool flag(false);
121     for(auto i : a.nums){
122         cur *= 10, cur += i;
123         if(cur < div && !flag)continue;
124         flag = true, ret.nums += cur / div, cur %= div;
125     }return ret;
126 }
127 void Print(void){
128     for(auto v : nums)printf("%d", v);
129     printf("\n");

```

```

130     }
131 };
132
133 Bignum qpow(Bignum a, ll b){
134     Bignum ret, mul(a);
135     ret.nums += 1;
136     while(b){
137         if(b & 1)ret = ret * mul;
138         b >>= 1;
139         mul = mul * mul;
140     }return ret;
141 }
142
143 int main(){
144
145     int T = read();
146     while(T--){
147         string SA, SB; cin >> SA >> SB;
148         Polynomial A(SA.length()), B(SB.length());
149         for(int i = 0; i < A.len; ++i)A.poly[i] = int(SA[A.len -
150 i - 1] - '0');
151         for(int i = 0; i < B.len; ++i)B.poly[i] = int(SB[B.len -
152 i - 1] - '0');
153         int clen = A.len + B.len - 1;
154         int base(1); while(base < clen)base <= 1;
155         A.Resize(base), B.Resize(base);
156         A.NTT(DFT), B.NTT(DFT);
157         for(int i = 0; i < A.len; ++i)A.poly[i] = A.poly[i] *
158 B.poly[i] % MOD;
159         A.NTT(IDFT);
160
161         int fst1 = A.len - 1;
162         while(fst1 >= 0 && A.poly[fst1] == 0)--fst1;
163
164         vector < ll > ans;
165         for(auto it = A.poly.begin(); it != next(A.poly.begin(),
166 fst1 + 1); advance(it, 1))ans.emplace_back(*it);
167

```

```

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];

```

```

201         if(i < odd.size())res[(i << 1) | 1] = odd[i];
202     }
203     while(res.size() > 1 && res.back() == 0) res.pop_back();
204
205     if(res.empty()) puts("0");
206     else{
207         for(auto it = res.rbegin(); it != res.rend(); ++it)
208             printf("%lld", *it);
209         printf("\n");
210     }
211
212     // for(auto it = ans.rbegin(); it != ans.rend();
213     ++it)printf("%lld", *it);
214
215     }
216
217     // fprintf(stderr, "Time: %.6lf\n", (double)clock() /
218     CLOCKS_PER_SEC);
219     return 0;
220 }
221
222 template < typename T >
223 inline T read(void){
224     T ret(0);
225     int flag(1);
226     char c = getchar();
227     while(c != '-' && !isdigit(c))c = getchar();
228     if(c == '-')flag = -1, c = getchar();
229     while(isdigit(c)){
230         ret *= 10;
231         ret += int(c - '0');
232         c = getchar();
233     }
234     ret *= flag;
235     return ret;
236 }

```

```

1  #define _USE_MATH_DEFINES
2  #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) + 1;}
11
12 typedef unsigned int uint;
13 typedef unsigned long long ull;
14 typedef long long ll;
15
16 template<typename T = int>
17 inline T read(void);
18
19 auto qpow = [](ll a, ll b, ll mod)->ll{
20     ll ret(1), mul(a);
21     while(b){
22         if(b & 1)ret = (ret * mul) % mod;
23         b >>= 1, mul = (mul * mul) % mod;
24     }return ret;
25 };
26
27 int N, Q, B;
28
29 class LengthContainer{
30 private:
31     multiset < int > lens;
32     vector < ll > curG;
33     ll len0;
34
35 public:
36     int prel, sufl, mxl;

```

```

37 ll Cal(int b, int len){
38     if(len <= (b << 1))return 0;
39     int num = len - (b << 1), div = (b << 1) | 1;
40     return (num + div - 1) / div;
41 }
42 void InsertLen(int len){
43     if(len <= 0)return;
44     len0 += len;
45     lens.insert(len);
46     if(lens.size()){
47         auto it = prev(lens.end());
48         if(!lens.empty())mxl = max(mx1, *it);
49         mx1 = max(mx1, len);
50     }else mx1 = max(mx1, len);
51     for(int b = 0; b <= B; ++b)curG[b] += Cal(b, len);
52 }
53 void RemoveLen(int len){
54     if(len <= 0)return;
55     len0 -= len;
56     auto it = lens.find(len);
57     if(it != lens.end())lens.erase(it);
58     for(int b = 0; b <= B; ++b)curG[b] -= Cal(b, len);
59     mx1 = lens.empty() ? 0 : *prev(lens.end());
60 }
61 ll Query(void){
62     if(!len0)return 0;
63     ll ans(LONG_LONG_MAX >> 2);
64     int lmx = max(mx1, prel + sufl);
65     ans = min(ans, (ll)((lmx + 1) >> 1));
66     if(len0 == N){
67         for(int b = 0; b <= B; ++b){
68             ll val = b + 1 + Cal(b, N - 1);
69             if(val < ans)ans = val;
70         }
71         return ans;
72     }
73     for(int b = 0; b <= B; ++b){
74         ll res(0);

```



```

75         if(prel > 0 && sufl > 0) res = Cal(b, prel + sufl) -
Cal(b, prel) - Cal(b, sufl);
76         ll val = (ll)b + curG[b] + res;
77         if(val < ans)ans = val;
78     }return ans;
79 }
80 void Clear(void){
81     lens.clear();
82     curG.assign(B + 1, 0);
83     len0 = prel = sufl = mxl = 0;
84 }
85 }lc;
86
87
88 struct Node{
89     int l, r;
90     int size(void)const{return r - l + 1;}
91     mutable ll val;
92     friend const bool operator < (const Node &x, const Node &y)
{return x.l < y.l;}
93 };
94
95 class ODT{
96 private:
97     set < Node > tr;
98 public:
99     auto Insert(Node p){return tr.insert(p);}
100     auto Split(int p){
101         auto it = tr.lower_bound(Node{p});
102         if(it != tr.end() && p == it->l)return it;
103         if(it == tr.begin()) return tr.end();
104         --it;
105         if(p > it->r)return tr.end();
106         int l = it->l, r = it->r;
107         ll val = it->val;
108         if(!val) lc.RemoveLen(r - l + 1);
109         tr.erase(it);
110         if(l <= p - 1){
111             Insert(Node{l, p - 1, val});

```

```

112         if(!val) lc.InsertLen(p - 1 - l + 1);
113     }
114     auto ret = Insert(Node{p, r, val}).first;
115     if(!val)lc.InsertLen(r - p + 1);
116     return ret;
117 }
118 void Modify(int l, int r, ll val){
119     auto itR = Split(r + 1), itL = Split(l);
120     for(auto it = itL; it != itR; ++it)it->val += val;
121 }
122 void Assign(int l, int r, ll val){
123     auto itR = Split(r + 1), itL = Split(l);
124     tr.erase(itL, itR);
125     Insert(Node{l, r, val});
126 }
127 ll QueryKth(int l, int r, int k){
128     vector < Node > rnk;
129     auto itR = Split(r + 1), itL = Split(l);
130     for(auto it = itL; it != itR; ++it)rnk.push_back(*it);
131     sort(rnk.begin(), rnk.end(), [](const Node x, const Node
y)->bool{return x.val < y.val;});
132     int cur(0);
133     for(auto i : rnk){
134         cur += i.size();
135         if(cur >= k)return i.val;
136     }
137     return -1;
138 }
139 ll QuerySum(int l, int r, ll k, ll mod){
140     ll ret(0);
141     auto itR = Split(r + 1), itL = Split(l);
142     for(auto it = itL; it != itR; ++it)
143         ret = (ret + qpow(it->val, k, mod) * it->size() %
mod) % mod;
144     return ret;
145 }
146 void Build(const vector < ll > &A){
147     tr.clear();
148     for(int i = 1; i <= N;){

```

```

149         int cur(i);
150         while(cur + 1 <= N && A[cur + 1] == A[i])++cur;
151         Insert(Node{i, cur, A[i]});
152         i = cur + 1;
153     }
154 }
155
156 void Assign01(int l, int r, int val){
157     auto itR = Split(r + 1), itL = Split(l);
158     for(auto it = itL; it != itR; ++it)
159         if(!it->val)lc.RemoveLen(it->size());
160     tr.erase(itL,itR);
161     if(val == 1){
162         auto it = tr.lower_bound(Node{l});
163         int L(l), R(r);
164         if(it != tr.begin() && prev(it)->r == l - 1 &&
prev(it)->val == 1)
165             L = prev(it)->l, tr.erase(prev(it));
166         it = tr.lower_bound(Node{r + 1});
167         if(it != tr.end() && it->l == r + 1 && it->val == 1)
168             R = it->r, tr.erase(it);
169         Insert(Node{L, R, 1});
170     }else{
171         auto it = tr.lower_bound(Node{l});
172         int L(l), R(r);
173         if(it != tr.begin() && prev(it)->r == l - 1 &&
prev(it)->val == 0)
174             lc.RemoveLen(prev(it)->size()), L = prev(it)->l,
tr.erase(prev(it));
175         it = tr.lower_bound(Node{r + 1});
176         if(it != tr.end() && it->l == r + 1 && it->val == 0)
177             lc.RemoveLen(it->size()), R = it->r,
tr.erase(it);
178         Insert(Node{L, R, 0});
179         lc.InsertLen(R - L + 1);
180     }
181 }
182 void Maintain(){
183     if(tr.empty()){lc.prel = lc.suf1 = 0; return;}

```

```

184         lc.prel = (tr.begin()->l == 1 && tr.begin()->val == 0) ?
tr.begin()->size() : 0;
185         lc.sufl = (prev(tr.end())->r == N && prev(tr.end())->val
== 0) ? prev(tr.end())->size() : 0;
186     }
187     void Initialize(void){
188         for(auto i : tr)
189             if(i.val == 0)lc.InsertLen(i.size());
190     }
191 }odt;
192
193 int main(){
194     int T = read();
195     while(T--){
196         N = read(), Q = read();
197         vector<ll> A(N + 10);
198         string S; cin >> S;
199         for(int i = 1; i <= N; ++i)A[i] = S.at(i - 1) - '0';
200
201         odt.Build(A);
202         B = (int)sqrt((double)N) + 1;
203
204         lc.Clear();
205         odt.Initialize();
206         odt.Maintain();
207
208         while(Q--){
209             int opt = read(), l = read(), r = read();
210             odt.Assign01(l, r, opt == 1 ? 1 : 0);
211             odt.Maintain();
212             printf("%lld\n", lc.Query());
213         }
214     }
215
216
217     return 0;
218 }
219
220 template<typename T>

```

```
221 inline T read(void){
222     T ret(0);
223     short flag(1);
224     char c = getchar();
225     while(c != '-' && !isdigit(c))c = getchar();
226     if(c == '-')flag = -1, c = getchar();
227     while(isdigit(c)){
228         ret *= 10;
229         ret += int(c - '0');
230         c = getchar();
231     }
232     ret *= flag;
233     return ret;
234 }
235
```