
ACM/ICPC Template Manual

Harbin Institute of Technology

cycleke

September 2, 2019

Contents

1	Math	1
1.1	杜教筛	1
1.2	exctr	1
1.3	Miller Rabin	1
1.4	LinearSieve	2
1.5	china	3
1.6	BGS	3
1.7	Pollard rho	4
1.8	LinearRecurrence	4
1.9	类欧几里德算法	8
1.10	gauss	9
1.11	FFT	10
1.12	lucas	11
1.13	exgcd	11
2	Dynamic Programming	11
3	Data Structure	11
3.1	splay	11
3.2	zkw	14
4	String	15
4.1	hash	15
4.2	manacher	15
4.3	回文树	16
4.4	exkmp	17
4.5	kmp	18
5	Graph Theory	18
5.1	KM	18
6	Computational Geometry	19
7	Java	19
7.1	进制转换	19
8	Others	20
8.1	FastIO	20
8.2	head	22
8.3	vimrc	23
8.4	myalloc	24
8.5	duipai	24

1 Math

1.1 杜教筛

```

1 map<int, int> mp_mu;
2
3 int S_mu(int n) {
4     if (n < MAXN) return sum_mu[n];
5     if (mp_mu[n]) return mp_mu[n];
6     int ret = 1;
7     for (int i = 2, j; i <= n; i = j + 1) {
8         j = n / (n / i);
9         ret -= S_mu(n / i) * (j - i + 1);
10    }
11    return mp_mu[n] = ret;
12 }
13
14 ll S_phi(int n) {
15     ll res = 0;
16     for (int i = 1, j; i <= n; i = j + 1) {
17         j = n / (n / i);
18         res += 1LL * (S_mu(j) - S_mu(i - 1)) * (n / i) * (n / i);
19     }
20     return (res - 1) / 2 + 1;
21 }

```

1.2 exctr

```

1 int exctr(int n, int *a, int *m) {
2     int M = m[0], res = a[0];
3     for (int i = 1; i < n; ++i) {
4         int a = M, b = m[i], c = (a[i] - res % b + b) % b, x, y;
5         int g = exgcd(a, b, x, y), bg = b / g;
6         if (c % g != 0) return -1;
7         x = 1LL * x * (c / g) % bg;
8         res += x * M;
9         M *= bg;
10        res = (res % M + M) % M;
11    }
12    return res;
13 }

```

1.3 Miller Rabin

```

1 inline ll mod_mul(const ll &a, const ll &b, const ll &mod) {
2     ll k = (ll)((1.0L * a * b) / (1.0L * mod)), t = a * b - k * mod;
3     t -= mod;
4     while (t < 0) t += mod;
5     return t;
6 }
7 inline ll mod_pow(ll a, ll b, const ll &mod) {
8     ll res = 1;
9     for (; b; b >>= 1, a = mod_mul(a, a, mod))
10        (b & 1) && (res = mod_mul(res, a, mod));
11    return res;
12 }
13

```

```
14 inline bool check(const ll &x, const ll &p) {
15     if (!(x % p) || mod_pow(p % x, x - 1, x) ^ 1) return false;
16     ll k = x - 1, t;
17     while (~k & 1) {
18         if (((t = mod_pow(p % x, k >= 1, x)) ^ 1) && (t ^ (x - 1))) return false;
19         if (!(t ^ (x - 1))) return true;
20     }
21     return true;
22 }
23
24 inline bool Miller_Rabin(const ll &x) {
25     if (x < 2) return false;
26     static const int p[12] = {2, 3, 5, 7, 11, 13, 17, 19, 61, 2333, 4567, 24251};
27     for (register int i = 0; i < 12; ++i) {
28         if (!(x ^ p[i])) return true;
29         if (!check(x, p[i])) return false;
30     }
31     return true;
32 }
```

1.4 LinearSieve

```
1 const int MAXN = 1e7 + 5;
2
3 bool vis[MAXN];
4 int prime[MAXN / 10], prime_cnt;
5 int fac[MAXN], e[MAXN], d[MAXN], mu[MAXN], phi[MAXN];
6
7 void sieve() {
8     fac[1] = 1;
9     e[1] = 0;
10    d[1] = 1;
11    mu[1] = 1;
12    phi[1] = 1;
13    for (int i = 2; i < MAXN; ++i) {
14        if (!vis[i]) {
15            prime[prime_cnt++] = i;
16            fac[i] = i;
17            e[i] = 1;
18            d[i] = 2;
19            mu[i] = -1;
20            phi[i] = i - 1;
21        }
22        for (int j = 0; j < prime_cnt; ++j) {
23            int t = prime[j] * i;
24            if (t >= MAXN) { break; }
25            vis[t] = true;
26            fac[t] = prime[j];
27            if (i % prime[j] == 0) {
28                e[t] = e[i] + 1;
29                d[t] = d[i] / (e[i] + 1) * (e[t] + 1);
30                mu[t] = 0;
31                phi[t] = phi[i] * prime[j];
32                break;
33            } else {
34                e[t] = 1;
35                d[t] = d[i] * 2;
36                mu[t] = -mu[i];
37            }
38        }
39    }
40 }
```

```
37     phi[t] = phi[i] * (prime[j] - 1);
38     }
39     }
40     }
41 }
```

1.5 china

```
1 int china(int n, int *a, int *m) {
2     int lcm = 1, res = 0;
3     for (int i = 0; i < n; ++i) lcm *= m[i];
4     for (int i = 0; i < n; ++i) {
5         int t = lcm / m[i], x, y;
6         exgcd(t, m[i], x, y);
7         x = (x % m[i] + m[i]) % m[i];
8         res = (res + 1LL * t * x) % lcm;
9     }
10    return res;
11 }
```

1.6 BSGS

```
1 // Finds the primitive root modulo p
2 int generator(int p) {
3     vector<int> fact;
4     int phi = p - 1, n = phi;
5     for (int i = 2; i * i <= n; ++i) {
6         if (n % i == 0) {
7             fact.push_back(i);
8             while (n % i == 0) n /= i;
9         }
10    }
11    if (n > 1) fact.push_back(n);
12    for (int res = 2; res <= p; ++res) {
13        bool ok = true;
14        for (int factor : fact)
15            if (mod_pow(res, phi / factor, p) == 1) {
16                ok = false;
17                break;
18            }
19        if (ok) return res;
20    }
21    return -1;
22 }
23 // This program finds all numbers x such that x^k=a (mod n)
24 vector<int> BSGS(int n, int k, int a) {
25     if (a == 0) return vector<int>({0});
26     int g = generator(n);
27     // Baby-step giant-step discrete logarithm algorithm
28     int sq = (int)sqrt(n + .0) + 1;
29     vector<pair<int, int>> dec(sq);
30     for (int i = 1; i <= sq; ++i)
31         dec[i - 1] = {mod_pow(g, i * sq * k % (n - 1), n), i};
32     sort(dec.begin(), dec.end());
33 }
```

```

36  int any_ans = -1;
37  for (int i = 0; i < sq; ++i) {
38      int my = mod_pow(g, i * k % (n - 1), n) * a % n;
39      auto it = lower_bound(dec.begin(), dec.end(), make_pair(my, 0));
40      if (it != dec.end() && it->first == my) {
41          any_ans = it->second * sq - i;
42          break;
43      }
44  }
45  if (any_ans == -1) return vector<int>();
46  // Print all possible answers
47  int delta = (n - 1) / __gcd(k, n - 1);
48  vector<int> ans;
49  for (int cur = any_ans % delta; cur < n - 1; cur += delta)
50      ans.push_back(mod_pow(g, cur, n));
51  sort(ans.begin(), ans.end());
52  return ans;
53 }

```

1.7 Pollard rho

```

1  inline ll rand64(ll x) {
2      return 1ll * ((rand() << 15 ^ rand()) << 30 ^ (rand() << 15 ^ rand())) % x;
3  }
4
5  inline ll Pollard_rho(const ll &x, const int &y) {
6      ll v0 = rand64(x - 1) + 1, v = v0, d, s = 1;
7      for (register int t = 0, k = 1;;) {
8          if (v = (mod_mul(v, v, x) + y) % x, s = mod_mul(s, abs(v - v0), x),
9              !(v ^ v0) || !s)
10             return x;
11          if (++t == k) {
12              if ((d = __gcd(s, x)) ^ 1) return d;
13              v0 = v, k <= 1;
14          }
15      }
16  }
17
18  ll ans;
19  vector<ll> factor;
20  void findfac(ll n) {
21      if (Miller_Rabin(n)) {
22          factor.push_back(n);
23          return;
24      }
25      ll p = n;
26      while (p >= n) { p = Pollard_rho(p, rand64(n - 1) + 1); }
27      findfac(p);
28      findfac(n / p);
29  }

```

1.8 LinearRecurrence

```

1  struct LinearRecurrence {
2      using int64 = long long;
3      using vec = std::vector<int64>;
4

```

```

5 static void extend(vec &a, size_t d, int64 value = 0) {
6     if (d <= a.size()) return;
7     a.resize(d, value);
8 }
9
10 static vec BerlekampMassey(const vec &s, int64 mod) {
11     std::function<int64(int64)> inverse = [&](int64 a) {
12         return a == 1 ? 1 : (int64)(mod - mod / a) * inverse(mod % a) % mod;
13     };
14     vec A = {1}, B = {1};
15     int64 b = s[0];
16     for (size_t i = 1, m = 1; i < s.size(); ++i, m++) {
17         int64 d = 0;
18         for (size_t j = 0; j < A.size(); ++j) { d += A[j] * s[i - j] % mod; }
19         if (!(d % mod)) continue;
20         if (2 * (A.size() - 1) <= i) {
21             auto temp = A;
22             extend(A, B.size() + m);
23             int64 coef = d * inverse(b) % mod;
24             for (size_t j = 0; j < B.size(); ++j) {
25                 A[j + m] -= coef * B[j] % mod;
26                 if (A[j + m] < 0) A[j + m] += mod;
27             }
28             B = temp, b = d, m = 0;
29         } else {
30             extend(A, B.size() + m);
31             int64 coef = d * inverse(b) % mod;
32             for (size_t j = 0; j < B.size(); ++j) {
33                 A[j + m] -= coef * B[j] % mod;
34                 if (A[j + m] < 0) A[j + m] += mod;
35             }
36         }
37     }
38     return A;
39 }
40
41 static void exgcd(int64 a, int64 b, int64 &g, int64 &x, int64 &y) {
42     if (!b)
43         x = 1, y = 0, g = a;
44     else {
45         exgcd(b, a % b, g, y, x);
46         y -= x * (a / b);
47     }
48 }
49
50 static int64 crt(const vec &c, const vec &m) {
51     int n = c.size();
52     int64 M = 1, ans = 0;
53     for (int i = 0; i < n; ++i) M *= m[i];
54     for (int i = 0; i < n; ++i) {
55         int64 x, y, g, tm = M / m[i];
56         exgcd(tm, m[i], g, x, y);
57         ans = (ans + tm * x * c[i] % M) % M;
58     }
59     return (ans + M) % M;
60 }
61
62 static vec ReedsSloane(const vec &s, int64 mod) {
63     auto inverse = [&](int64 a, int64 m) {

```

```

64     int64 d, x, y;
65     exgcd(a, m, d, x, y);
66     return d == 1 ? (x % m + m) % m : -1;
67 };
68 auto L = [](const vec &a, const vec &b) {
69     int da = (a.size() > 1 || (a.size() == 1 && a[0])) ? a.size() - 1 : -1000;
70     int db = (b.size() > 1 || (b.size() == 1 && b[0])) ? b.size() - 1 : -1000;
71     return std::max(da, db + 1);
72 };
73 auto prime_power = [&](const vec &s, int64 mod, int64 p, int64 e) {
74     // linear feedback shift register mod p^e, p is prime
75     std::vector<vec> a(e), b(e), an(e), bn(e), ao(e), bo(e);
76     vec t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
77     ;
78     pw[0] = 1;
79     for (int i = pw[0] = 1; i <= e; ++i) pw[i] = pw[i - 1] * p;
80     for (int64 i = 0; i < e; ++i) {
81         a[i] = {pw[i]}, an[i] = {pw[i]};
82         b[i] = {0}, bn[i] = {s[0] * pw[i] % mod};
83         t[i] = s[0] * pw[i] % mod;
84         if (t[i] == 0) {
85             t[i] = 1, u[i] = e;
86         } else {
87             for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i])
88                 ;
89         }
90     }
91     for (size_t k = 1; k < s.size(); ++k) {
92         for (int g = 0; g < e; ++g) {
93             if (L(an[g], bn[g]) > L(a[g], b[g])) {
94                 ao[g] = a[e - 1 - u[g]];
95                 bo[g] = b[e - 1 - u[g]];
96                 to[g] = t[e - 1 - u[g]];
97                 uo[g] = u[e - 1 - u[g]];
98                 r[g] = k - 1;
99             }
100         }
101         a = an, b = bn;
102         for (int o = 0; o < e; ++o) {
103             int64 d = 0;
104             for (size_t i = 0; i < a[o].size() && i <= k; ++i) {
105                 d = (d + a[o][i] * s[k - i]) % mod;
106             }
107             if (d == 0) {
108                 t[o] = 1, u[o] = e;
109             } else {
110                 for (u[o] = 0, t[o] = d; t[o] % p == 0; t[o] /= p, ++u[o])
111                     ;
112                 int g = e - 1 - u[o];
113                 if (L(a[g], b[g]) == 0) {
114                     extend(bn[o], k + 1);
115                     bn[o][k] = (bn[o][k] + d) % mod;
116                 } else {
117                     int64 coef =
118                         t[o] * inverse(to[g], mod) % mod * pw[u[o] - uo[g]] % mod;
119                     int m = k - r[g];
120                     extend(an[o], ao[g].size() + m);
121                     extend(bn[o], bo[g].size() + m);
122                     for (size_t i = 0; i < ao[g].size(); ++i) {

```



```

123         an[o][i + m] -= coef * ao[g][i] % mod;
124         if (an[o][i + m] < 0) an[o][i + m] += mod;
125     }
126     while (an[o].size() && an[o].back() == 0) an[o].pop_back();
127     for (size_t i = 0; i < bo[g].size(); ++i) {
128         bn[o][i + m] -= coef * bo[g][i] % mod;
129         if (bn[o][i + m] < 0) bn[o][i + m] += mod;
130     }
131     while (bn[o].size() && bn[o].back() == 0) bn[o].pop_back();
132 }
133 }
134 }
135 }
136 return std::make_pair(an[0], bn[0]);
137 };
138
139 std::vector<std::tuple<int64, int64, int>> fac;
140 for (int64 i = 2; i * i <= mod; ++i)
141     if (mod % i == 0) {
142         int64 cnt = 0, pw = 1;
143         while (mod % i == 0) mod /= i, ++cnt, pw *= i;
144         fac.emplace_back(pw, i, cnt);
145     }
146 if (mod > 1) fac.emplace_back(mod, mod, 1);
147 std::vector<vec> as;
148 size_t n = 0;
149 for (auto &&x : fac) {
150     int64 mod, p, e;
151     vec a, b;
152     std::tie(mod, p, e) = x;
153     auto ss = s;
154     for (auto &&x : ss) x %= mod;
155     std::tie(a, b) = prime_power(ss, mod, p, e);
156     as.emplace_back(a);
157     n = std::max(n, a.size());
158 }
159 vec a(n), c(as.size()), m(as.size());
160 for (size_t i = 0; i < n; ++i) {
161     for (size_t j = 0; j < as.size(); ++j) {
162         m[j] = std::get<0>(fac[j]);
163         c[j] = i < as[j].size() ? as[j][i] : 0;
164     }
165     a[i] = crt(c, m);
166 }
167 return a;
168 }
169
170 LinearRecurrence(const vec &s, const vec &c, int64 mod)
171     : init(s), trans(c), mod(mod), m(s.size()) {}
172
173 LinearRecurrence(const vec &s, int64 mod, bool is_prime = true) : mod(mod) {
174     vec A;
175     if (is_prime)
176         A = BerlekampMassey(s, mod);
177     else
178         A = ReedsSloane(s, mod);
179     if (A.empty()) A = {0};
180     m = A.size() - 1;
181     trans.resize(m);

```

```

182     for (int i = 0; i < m; ++i) { trans[i] = (mod - A[i + 1]) % mod; }
183     std::reverse(trans.begin(), trans.end());
184     init = {s.begin(), s.begin() + m};
185 }
186
187 int64 calc(int64 n) {
188     if (mod == 1) return 0;
189     if (n < m) return init[n];
190     vec v(m), u(m << 1);
191     int msk = !!n;
192     for (int64 m = n; m > 1; m >>= 1) msk <<= 1;
193     v[0] = 1 % mod;
194     for (int x = 0; msk; msk >>= 1, x <<= 1) {
195         std::fill_n(u.begin(), m * 2, 0);
196         x |= !(n & msk);
197         if (x < m)
198             u[x] = 1 % mod;
199         else { // can be optimized by fft/ntt
200             for (int i = 0; i < m; ++i) {
201                 for (int j = 0, t = i + (x & 1); j < m; ++j, ++t) {
202                     u[t] = (u[t] + v[i] * v[j]) % mod;
203                 }
204             }
205             for (int i = m * 2 - 1; i >= m; --i) {
206                 for (int j = 0, t = i - m; j < m; ++j, ++t) {
207                     u[t] = (u[t] + trans[j] * u[i]) % mod;
208                 }
209             }
210         }
211         v = {u.begin(), u.begin() + m};
212     }
213     int64 ret = 0;
214     for (int i = 0; i < m; ++i) { ret = (ret + v[i] * init[i]) % mod; }
215     return ret;
216 }
217
218 vec init, trans;
219 int64 mod;
220 int m;
221 };

```

1.9 类欧几里德算法

```

1 //求 f=sum((a*i+b)/c),g=sum((a*i+b)/c*i),h=sum(((a*i+b)/c)^2), for i in [0..n],
2 //整除向下
3 #include <bits/stdc++.h>
4 #define int long long
5 using namespace std;
6 const int P = 998244353;
7 int i2 = 499122177, i6 = 166374059;
8 struct data {
9     data() { f = g = h = 0; }
10    int f, g, h;
11 }; // 三个函数打包
12 data calc(int n, int a, int b, int c) {
13     int ac = a / c, bc = b / c, m = (a * n + b) / c, n1 = n + 1, n21 = n * 2 + 1;
14     data d;
15     if (a == 0) { // 迭代到底层

```

```

16     d.f = bc * n1 % P;
17     d.g = bc * n % P * n1 % P * i2 % P;
18     d.h = bc * bc % P * n1 % P;
19     return d;
20 }
21 if (a >= c || b >= c) { // 取模
22     d.f = n * n1 % P * i2 % P * ac % P + bc * n1 % P;
23     d.g = ac * n % P * n1 % P * n21 % P * i6 % P + bc * n % P * n1 % P * i2 % P;
24     d.h = ac * ac % P * n % P * n1 % P * n21 % P * i6 % P +
25         bc * bc % P * n1 % P + ac * bc % P * n % P * n1 % P;
26     d.f %= P, d.g %= P, d.h %= P;
27
28     data e = calc(n, a % c, b % c, c); // 迭代
29
30     d.h += e.h + 2 * bc % P * e.f % P + 2 * ac % P * e.g % P;
31     d.g += e.g, d.f += e.f;
32     d.f %= P, d.g %= P, d.h %= P;
33     return d;
34 }
35 data e = calc(m - 1, c, c - b - 1, a);
36 d.f = n * m % P - e.f, d.f = (d.f % P + P) % P;
37 d.g = m * n % P * n1 % P - e.h - e.f, d.g = (d.g * i2 % P + P) % P;
38 d.h = n * m % P * (m + 1) % P - 2 * e.g - 2 * e.f - d.f;
39 d.h = (d.h % P + P) % P;
40 return d;
41 }
42
43 int T, n, a, b, c;
44 signed main() {
45     scanf("%lld", &T);
46     while (T--) {
47         scanf("%lld%lld%lld%lld", &n, &a, &b, &c);
48         data ans = calc(n, a, b, c);
49         printf("%lld %lld %lld\n", ans.f, ans.h, ans.g);
50     }
51     return 0;
52 }

```

1.10 gauss

```

1  const double EPS = 1e-9;
2  const int MAXN = 220;
3  double a[MAXN][MAXN], x[MAXN];
4  int equ, var;
5
6  int Gauss() {
7      int i, j, k, col, max_r;
8      for (k = 0, col = 0; k < equ && col < var; k++, col++) {
9          max_r = k;
10         for (i = k + 1; i < equ; i++)
11             if (fabs(a[i][col]) > fabs(a[max_r][col])) max_r = i;
12         if (fabs(a[max_r][col]) < eps) return 0;
13
14         if (k != max_r) {
15             for (j = col; j < var; j++) swap(a[k][j], a[max_r][j]);
16             swap(x[k], x[max_r]);
17         }
18     }

```

```

19     x[k] /= a[k][col];
20     for (j = col + 1; j < var; j++) a[k][j] /= a[k][col];
21     a[k][col] = 1;
22
23     for (i = 0; i < equ; i++)
24         if (i != k) {
25             x[i] -= x[k] * a[i][col];
26             for (j = col + 1; j < var; j++) a[i][j] -= a[k][j] * a[i][col];
27             a[i][col] = 0;
28         }
29     }
30     return 1;
31 }

```

1.11 FFT

```

1  const int MAXN = 4 * 1e5 + 3;
2  const double PI = acos(-1);
3  complex<double> a[MAXN], b[MAXN];
4
5  int n, bit;
6  int rev[MAXN];
7
8  void fft(complex<double> *a, int sign) {
9      for (int i = 0; i < n; ++i)
10         if (i < rev[i]) swap(a[i], a[rev[i]]);
11
12     for (int j = 1; j < n; j <= 1) {
13         complex<double> wn(cos(2 * PI / (j <= 1)), sign * sin(2 * PI / (j <= 1)));
14         for (int i = 0; i < n; i += (j <= 1)) {
15             complex<double> w(1, 0), t0, t1;
16             FOR(k, 0, j) {
17                 t0 = a[i + k];
18                 t1 = w * a[i + j + k];
19                 a[i + k] = t0 + t1;
20                 a[i + j + k] = t0 - t1;
21                 w *= wn;
22             }
23         }
24     }
25     if (sign == -1)
26         for (int i = 0; i < n; ++i) a[i] /= n;
27 }
28
29 int main() {
30     ios::sync_with_stdio(false);
31     cin.tie(0);
32     cout.tie(0);
33
34     int n, m, x;
35     cin >> n >> m;
36     for (int i = 0; i <= n; ++i) {
37         cin >> x;
38         a[i].real(x);
39     }
40     for (int i = 0; i <= m; ++i) {
41         cin >> x;
42         b[i].real(x);

```

```
43 }
44
45 ::n = 1;
46 bit = 0;
47 while (::n <= n + m) {
48     ::n <= 1;
49     ++bit;
50 }
51 rev[0] = 0;
52 FOR(i, 1, ::n) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
53 fft(a, 1);
54 fft(b, 1);
55 FOR(i, 0, ::n) a[i] *= b[i];
56 fft(a, -1);
57 FOR(i, 0, n + m + 1) cout << int(a[i].real() + .5) << " ";
58 cout << "\n";
59 return 0;
60 }
```

1.12 lucas

```
1 // C(n, m) = C(n / p, m / p) * C(n % p, m % p) (mod p)
2 ll lucas(ll n, ll k, int p) {
3     ll ret = 1;
4     while (n && k) {
5         ll nn = n % p, kk = k % p;
6         if (nn < kk) return 0;
7         ret = ret * f[nn] * mod_pow(f[kk] * f[nn - kk] % p, p - 2, p) % p;
8         n /= p, k /= p;
9     }
10    return res;
11 }
```

1.13 exgcd

```
1 int exgcd(int a, int b, int &x, int &y) {
2     if (b == 0) return x = 1, y = 0, a;
3     int g = exgcd(b, a % b, y, x);
4     y -= a / b * x;
5     return g;
6 }
```

2 Dynamic Programming

3 Data Structure

3.1 splay

```
1 #include <algorithm>
2 #include <cstdio>
3 #include <cstring>
4 #include <iostream>
5 using namespace std;
6
7 const int MAXN = 2e5 + 10;
```

```

8
9 struct Node {
10     long long sum;
11     int id, val, lazy, size;
12     Node *fa, *ch[2];
13 } node_pool[MAXN], *pool_it, *root, *nil;
14
15 Node *newnode(int id, int val) {
16     pool_it->id = id;
17     pool_it->lazy = 0;
18     pool_it->size = 1;
19     pool_it->sum = pool_it->val = val;
20     pool_it->fa = pool_it->ch[0] = pool_it->ch[1] = nil;
21     return pool_it++;
22 }
23
24 void maintain(Node *u) {
25     if (u == nil) { return; }
26     u->size = u->ch[0]->size + u->ch[1]->size + 1;
27     u->sum = u->ch[0]->sum + u->ch[1]->sum + u->val;
28 }
29
30 void push_down(Node *u) {
31     if (u->lazy) {
32         if (u->ch[0] != nil) {
33             u->ch[0]->val += u->lazy;
34             u->ch[0]->sum += 1LL * u->ch[0]->size * u->lazy;
35             u->ch[0]->lazy += u->lazy;
36         }
37         if (u->ch[1] != nil) {
38             u->ch[1]->val += u->lazy;
39             u->ch[1]->sum += 1LL * u->ch[1]->size * u->lazy;
40             u->ch[1]->lazy += u->lazy;
41         }
42         u->lazy = 0;
43     }
44 }
45
46 inline void rot(Node *u) {
47     Node *f = u->fa, *ff = f->fa;
48     int d = u == f->ch[1];
49     push_down(f);
50     push_down(u);
51     if ((f->ch[d] = u->ch[d ^ 1]) != nil) f->ch[d]->fa = f;
52     if ((u->fa = ff) != nil) ff->ch[f == ff->ch[1]] = u;
53     f->fa = u;
54     u->ch[d ^ 1] = f;
55     maintain(f);
56     maintain(u);
57 }
58
59 void splay(Node *u, Node *target) {
60     for (Node *f; u->fa != target; rot(u))
61         if ((f = u->fa)->fa != target) {
62             ((u == f->ch[1]) ^ (f == f->fa->ch[1])) ? rot(u) : rot(f);
63         }
64     if (target == nil) root = u;
65 }
66

```

```
67 inline void insert(int id, int val) {
68     if (root == nil) {
69         root = newnode(id, val);
70         return;
71     }
72     Node *u = root;
73     while (u != nil) {
74         int d = id >= u->id;
75         ++u->size;
76         push_down(u);
77         u->sum += val;
78         if (u->ch[d] != nil) {
79             u = u->ch[d];
80         } else {
81             u->ch[d] = newnode(id, val);
82             u->ch[d]->fa = u;
83             u = u->ch[d];
84             break;
85         }
86     }
87     splay(u, nil);
88 }
89
90 inline Node *find_pred(int id) {
91     Node *u = root, *ret = nil;
92     while (u != nil) {
93         push_down(u);
94         if (u->id < id) {
95             ret = u;
96             u = u->ch[1];
97         } else {
98             u = u->ch[0];
99         }
100     }
101     return ret;
102 }
103
104 inline Node *find_succ(int id) {
105     Node *u = root, *ret = nil;
106     while (u != nil) {
107         push_down(u);
108         if (u->id > id) {
109             ret = u;
110             u = u->ch[0];
111         } else {
112             u = u->ch[1];
113         }
114     }
115     return ret;
116 }
117
118 Node *find_kth(int k) {
119     Node *u = root;
120     while (u != nil) {
121         push_down(u);
122         if (u->ch[0]->size + 1 == k) {
123             splay(u, nil);
124             return u;
125         }
126     }
```

```

126     if (u->ch[0]->size >= k) {
127         u = u->ch[0];
128     } else {
129         k -= u->ch[0]->size + 1;
130         u = u->ch[1];
131     }
132 }
133 return nil;
134 }
135
136 Node *range(int l, int r) {
137     Node *pred = find_pred(l);
138     Node *succ = find_succ(r);
139
140     splay(pred, nil);
141     splay(succ, root);
142     push_down(pred);
143     push_down(succ);
144     return root->ch[1]->ch[0];
145 }
146
147 int main() {
148
149     // freopen("input.txt", "r", stdin);
150
151     ios::sync_with_stdio(false);
152     cin.tie(0);
153     cout.tie(0);
154
155     int n;
156     cin >> n;
157
158     pool_it = node_pool;
159     nil = pool_it++;
160     nil->ch[0] = nil->ch[1] = nil->fa = nil;
161     nil->id = -1;
162     nil->val = 0;
163     root = nil;
164
165     insert(-0x3fffffff, 0);
166     insert(0x3fffffff, 0);
167
168     return 0;
169 }

```

3.2 zkw

```

1  int tree[MAXN * 2], pre;
2
3  void init(int n, int *a) {
4      memset(tree, 0, sizeof(tree));
5      for (pre = 1; pre <= n; pre <= 1) {}
6      for (int i = 1; i <= n; ++i) tree[i + pre] = i;
7      for (int i = pre; i; --i) tree[i] = max(tree[i << 1], tree[i << 1 | 1]);
8  }
9
10 void update(int pos, const int &val) {
11     tree[pos += pre] = val;

```



```
12     for (pos >= 1; pos; pos >= 1)
13         tree[pos] = max(tree[pos << 1], tree[pos << 1 | 1]);
14 }
15
16 int query(int s, int t) {
17     int res = 0;
18     for (s += pre - 1, t += pre + 1; s ^ t ^ 1; s >>= 1, t >>= 1) {
19         if (~s & 1) res = max(res, tree[s ^ 1]);
20         if (t & 1) res = max(res, tree[t ^ 1]);
21     }
22     return res;
23 }
```

4 String

4.1 hash

```
1
2 const unsigned int KEY = 6151;
3 const unsigned int MOD = 1610612741;
4
5 unsigned int hash[MAXN], p[MAXN];
6
7 inline unsigned int get_hash(int l, int r) {
8     return (hash[r] + MOD - 1ULL * hash[l - 1] * p[r - l + 1] % MOD) % MOD;
9 }
10
11 void init(char *s, int n) {
12     p[0] = 1;
13     for (int i = 1; i <= n; ++i) {
14         p[i] = p[i - 1] * KEY % MOD;
15         hash[i] = (1LL * hash[i - 1] * KEY + s[i]) % MOD;
16     }
17 }
```

4.2 mancher

```
1 void mancher(char *s, int n) {
2     str[0] = '~';
3     str[1] = '!';
4     for (int i = 1; i <= n; ++i) {
5         str[i * 2] = s[i];
6         str[i * 2 + 1] = '!';
7     }
8     for (int i = 1, j = 0; i <= n; ++i) {
9         if (p[j] + j > i) {
10             p[i] = min(p[2 * j - i], p[j] + j - i);
11         } else {
12             p[i] = 1;
13         }
14         while (str[i + p[i]] == str[i - p[i]]) {
15             ++p[i];
16         }
17         if (i + p[i] > j + p[j]) {
18             j = i;
19         }
20     }
21 }
```

21 }

4.3 回文树

```

1 //最长双回文串
2 struct PT {
3     char s[MAXL];
4     int fail[MAXL], ch[26][MAXL], l[MAXL], dep[MAXL], lst, nc, n;
5     void init() {
6         l[0] = 0;
7         l[1] = -1;
8         fail[0] = fail[1] = 1;
9         for (int i = 0; i < 26; ++i) {
10             for (int j = 0; j < nc; ++j) {
11                 ch[i][j] = 0;
12             }
13         }
14         for (int i = 2; i < nc; ++i) {
15             l[i] = 0;
16             fail[i] = 0;
17         }
18         lst = 0;
19         nc = 2;
20         n = 0;
21         s[0] = '#';
22     }
23 }
24
25 int insert(char c) {
26     int id = c - 'a';
27     s[++n] = c;
28     while (s[n - l[lst] - 1] != s[n]) {
29         lst = fail[lst];
30     }
31     if (ch[id][lst] == 0) {
32         l[nc] = l[lst] + 2;
33         int f = fail[lst];
34         while (s[n - l[f] - 1] != s[n]) {
35             f = fail[f];
36         }
37         fail[nc] = ch[id][f];
38         dep[nc] = dep[fail[nc]] + 1;
39         ch[id][lst] = nc;
40         ++nc;
41     }
42     lst = ch[id][lst];
43     return lst;
44 }
45 } pt;
46
47 char S[MAXL];
48 int len[MAXL];
49 int main() {
50     ios::sync_with_stdio(false);
51     cin.tie(0);
52     cout.tie(0);
53
54     cin >> S;

```

```

55  int n = strlen(S);
56  pt.init();
57  for (int i = 0; i < n; ++i) {
58      len[i] = pt.l[pt.insert(S[i])];
59  }
60  pt.init();
61  int ans = 0;
62  for (int i = n - 1; i; --i) {
63      ans = max(ans, len[i - 1] + pt.l[pt.insert(S[i])]);
64  }
65  cout << ans << "\n";
66
67  return 0;
68 }

```

4.4 exkmp

```

1  // next[i]:x[i...m-1] 与 x[0...m-1] 的最长公共前缀
2  // extend[i]:y[i...n-1] 与 x[0...m-1] 的最长公共前缀
3  void prework(char x[], int m, int next[]) {
4      next[0] = m;
5      int j = 0;
6      while (j + 1 < m && x[j] == x[j + 1])
7          j++;
8      next[1] = j;
9      int k = 1;
10     for (int i = 2; i < m; i++) {
11         int p = next[k] + k - 1;
12         int L = next[i - k];
13         if (i + L < p + 1)
14             next[i] = L;
15         else {
16             j = max(0, p - i + 1);
17             while (i + j < m && x[i + j] == x[j])
18                 j++;
19             next[i] = j;
20             k = i;
21         }
22     }
23 }
24 void exkmp(char x[], int m, char y[], int n, int next[], int extend[]) {
25     prework(x, m, next);
26     int j = 0;
27     while (j < n && j < m && x[j] == y[j])
28         j++;
29     extend[0] = j;
30     int k = 0;
31     for (int i = 1; i < n; i++) {
32         int p = extend[k] + k - 1;
33         int L = next[i - k];
34         if (i + L < p + 1)
35             extend[i] = L;
36         else {
37             j = max(0, p - i + 1);
38             while (i + j < n && j < m && y[i + j] == x[j])
39                 j++;
40             extend[i] = j;
41             k = i;

```

```

42     }
43 }
44 }

```

4.5 kmp

```

1 void get_next(char *S, int *nxt, int n) {
2     nxt[0] = -1;
3     int j = -1;
4     for (int i = 1; i < n; ++i) {
5         while ((~j) && S[j + 1] != S[i]) {
6             j = nxt[j];
7         }
8         nxt[i] = (S[j + 1] == S[i]) ? (++j) : j;
9     }
10 }
11
12 int pattern(char *S, char *T, int *nxt, int n, int m) {
13     int j = -1;
14     for (int i = 0; i < m; ++i) {
15         while ((~j) && S[j + 1] != T[i]) {
16             j = nxt[j];
17         }
18         j += S[j + 1] == T[i];
19         if (j == n - 1) {
20             return i - n + 1;
21         }
22     }
23     return -1;
24 }

```

5 Graph Theory

5.1 KM

```

1 int n, m, match[MAXN];
2 int adj[MAXN][MAXN], lx[MAXN], ly[MAXN], slack[MAXN];
3 int visx[MAXN], visx_c, visy[MAXN], visy_c;
4
5 bool dfs(int x) {
6     visx[x] = visx_c;
7     for (int y = 0; y < m; ++y)
8         if (visy[y] ^ visy_c) {
9             int t = lx[x] + ly[y] - adj[x][y];
10            if (!t) {
11                visy[y] = visy_c;
12                if (match[y] < 0 || dfs(match[y])) return match[y] = x, true;
13            } else
14                (slack[y] > t) && (slack[y] = t);
15        }
16     return false;
17 }
18
19 int KM() {
20     memset(match, -1, sizeof(int) * m);
21     memset(ly, 0, sizeof(int) * m);
22     for (int i = 0; i < n; ++i) {

```

```

23     lx[i] = -INF;
24     for (int j = 0; j < m; ++j) (adj[i][j] > lx[i]) && (lx[i] = adj[i][j]);
25 }
26 for (int x = 0; x < n; ++x) {
27     fill(slack, slack + m, INF);
28     for (;;) {
29         ++visx_c, ++visy_c;
30         if (dfs(x)) break;
31         int d = INF;
32         for (int i = 0; i < m; ++i)
33             (visy[i] ^ visy_c) && (d > slack[i]) && (d = slack[i]);
34         for (int i = 0; i < n; ++i) (visx[i] == visx_c) && (lx[i] -= d);
35         for (int i = 0; i < m; ++i)
36             (visy[i] ^ visy_c) ? slack[i] -= d : ly[i] += d;
37     }
38 }
39 int res = 0;
40 for (int i = 0; i < m; ++i) (~match[i]) && (res += adj[match[i]][i]);
41 return res;
42 }

```

6 Computational Geometry

7 Java

7.1 进制转换

```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4
5 /**
6  * Built using CHelper plug-in
7  * Actual solution is at the top
8  */
9 public class Main {
10     public static void main(String[] args) {
11         InputStream inputStream = System.in;
12         OutputStream outputStream = System.out;
13         Scanner in = new Scanner(inputStream);
14         PrintWriter out = new PrintWriter(outputStream);
15         Solver solver = new Solver();
16         int testCount = Integer.parseInt(in.next());
17         for (int i = 1; i <= testCount; i++)
18             solver.solve(i, in, out);
19         out.close();
20     }
21
22     static class Solver {
23         public void solve(int testNumber, Scanner in, PrintWriter out) {
24             int a = in.nextInt();
25             int b = in.nextInt();
26             String num = in.next();
27
28             BigInteger value = BigInteger.ZERO;
29             for (int i = 0; i < num.length(); ++i) {
30                 value = value.multiply(BigInteger.valueOf(a));

```

```

31         value = BigInteger.valueOf(getValue(num.charAt(i))).add(value);
32     }
33     out.println(a + " " + num);
34
35     if (value.equals(BigInteger.ZERO)) {
36         out.println(b + " 0");
37         out.println();
38         return;
39     }
40
41     out.print(b + " ");
42
43     char[] ans = new char[1000];
44     int length = 0;
45     while (!value.equals(BigInteger.ZERO)) {
46         int digit = value.mod(BigInteger.valueOf(b)).intValue();
47         value = value.divide(BigInteger.valueOf(b));
48         ans[length] = getChar(digit);
49         ++length;
50     }
51
52     for (int i = length - 1; i >= 0; --i) {
53         out.print(ans[i]);
54     }
55     out.println("\n");
56 }
57
58 private int getValue(char ch) {
59     if (ch >= 'A' && ch <= 'Z') {
60         return ch - 'A' + 10;
61     }
62     if (ch >= 'a' && ch <= 'z') {
63         return ch - 'a' + 36;
64     }
65     return ch - '0';
66 }
67
68 private char getChar(int x) {
69     if (x < 10) {
70         return (char) ('0' + x);
71     } else if (x < 36) {
72         return (char) ('A' + x - 10);
73     } else {
74         return (char) ('a' + x - 36);
75     }
76 }
77
78 }
79 }

```

8 Others

8.1 FastIO

```

1 namespace FastIO {
2 struct Control {
3     int ct, val;
4     Control(int Ct, int Val = -1) : ct(Ct), val(Val) {}

```

```

5  inline Control operator()(int Val) { return Control(ct, Val); }
6  } _endl(0), _prs(1), _setprecision(2);
7
8  const int IO_SIZE = 1 << 16 | 127;
9
10 struct FastIO {
11     char in[IO_SIZE], *p, *pp, out[IO_SIZE], *q, *qq, ch[20], *t, b, K, prs;
12     FastIO() : p(in), pp(in), q(out), qq(out + IO_SIZE), t(ch), b(1), K(6) {}
13     ~FastIO() { fwrite(out, 1, q - out, stdout); }
14     inline char getc() {
15         return p == pp && (pp = (p = in) + fread(in, 1, IO_SIZE, stdin), p == pp)
16             ? (b = 0, EOF)
17             : *p++;
18     }
19     inline void putc(char x) {
20         q == qq && (fwrite(out, 1, q - out, stdout), q = out), *q++ = x;
21     }
22     inline void puts(const char str[]) {
23         fwrite(out, 1, q - out, stdout), fwrite(str, 1, strlen(str), stdout),
24             q = out;
25     }
26     inline void getline(string &s) {
27         s = "";
28         for (char ch; (ch = getc()) != '\n' && b;) s += ch;
29     }
30 #define indef(T) \
31     inline FastIO &operator>>(T &x) { \
32         x = 0; \
33         char f = 0, ch; \
34         while (!isdigit(ch = getc()) && b) f |= ch == '-'; \
35         while (isdigit(ch)) x = (x << 1) + (x << 3) + (ch ^ 48), ch = getc(); \
36         return x = f ? -x : x, *this; \
37     }
38 indef(int);
39 indef(long long);
40
41 inline FastIO &operator>>(string &s) {
42     s = "";
43     char ch;
44     while (isspace(ch = getc()) && b) {}
45     while (!isspace(ch) && b) s += ch, ch = getc();
46     return *this;
47 }
48 inline FastIO &operator>>(double &x) {
49     x = 0;
50     char f = 0, ch;
51     double d = 0.1;
52     while (!isdigit(ch = getc()) && b) f |= (ch == '-');
53     while (isdigit(ch)) x = x * 10 + (ch ^ 48), ch = getc();
54     if (ch == '.')
55         while (isdigit(ch = getc())) x += d * (ch ^ 48), d *= 0.1;
56     return x = f ? -x : x, *this;
57 }
58 #define outdef(_T) \
59     inline FastIO &operator<<(_T x) { \
60         !x && (putc('0'), 0), x < 0 && (putc('-'), x = -x); \
61         while (x) *t++ = x % 10 + 48, x /= 10; \
62         while (t != ch) *q++ = *--t; \
63         return *this; \

```

```

64 }
65 outdef(int);
66 outdef(long long);
67 inline FastIO &operator<<(char ch) { return putc(ch), *this; }
68 inline FastIO &operator<<(const char str[]) { return puts(str), *this; }
69 inline FastIO &operator<<(const string &s) { return puts(s.c_str()), *this; }
70 inline FastIO &operator<<(double x) {
71     int k = 0;
72     this->operator<<(int(x));
73     putc('.');
74     x -= int(x);
75     prs && (x += 5 * pow(10, -K - 1));
76     while (k < K) putc(int(x *= 10) ^ 48), x -= int(x), ++k;
77     return *this;
78 }
79 inline FastIO &operator<<(const Control &cl) {
80     switch (cl.ct) {
81     case 0: putc('\n'); break;
82     case 1: prs = cl.val; break;
83     case 2: K = cl.val; break;
84     }
85     return *this;
86 }
87 inline operator bool() { return b; }
88 };
89 } // namespace FastIO

```

8.2 head

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef long long ll;
5  typedef unsigned long long ull;
6  typedef long double ld;
7  typedef pair<int, int> pii;
8  typedef tuple<int, int, int> tiii;
9  typedef vector<int> vi;
10 typedef vector<vi> vvi;
11 typedef vector<long long> vll;
12 typedef vector<pii> vpii;
13
14 #define SZ(a) int((a).size())
15 #define ALL(a) (a).begin(), (a).end()
16 #define EXIST(s, x) ((s).find(x) != (s).end())
17 #define A_EXIST(a, x) (find(ALL(a), x) != (a).end())
18 #define ZERO(a) memset((a), 0, sizeof(a))
19
20 #define FOR(i, a, b) for (int i = int(a); i < int(b); ++i)
21 #define REP(i, a, b) for (int i = int(b) - 1; i >= a; --i)
22 #define FOR2(i, a, b, j, c, d) FOR(i, a, b) FOR(j, c, d)
23 #define REP2(i, a, b, j, c, d) REP(i, a, b) REP(j, c, d)
24 #define EACH(i, s) for (auto i = (s).begin(); i != (s).end(); ++i)
25 #define debug(...) fprintf(stderr, __VA_ARGS__)
26 #define dbg(x) \
27     cerr << "debug: " << __FUNCTION__ << "()" @ " << __TIMESTAMP__ << "\n" \
28     << __FILE__ << " L" << __LINE__ << "\n" \
29     << #x " = " << (x) << endl

```



```
30
31 const int INF = 0x3fffffff;
32 const ll LL_INF = 0x3ffffffffffffffffll;
33 const int MOD = 1e9 + 7;
34 const ll HASH_KEY = 6151;
35 const ll HASH_MOD = 1610612741;
36
37 // mt19937 rdm(chrono::steady_clock::now().time_since_epoch().count());
38 // uniform_int_distribution<int> u_int(begin, end);
39 // uniform_real_distribution<double> u_read(begin, end);
40 /* -- HEAD END -- */
41
42 void solve() {}
43
44 int main(int argc, char *argv[]) {
45     ios::sync_with_stdio(false);
46     cin.tie(nullptr);
47     cout << fixed << setprecision(10);
48
49     int o_o;
50     for (o_o = 1; o_o; --o_o) solve();
51
52     return 0;
53 }
```

8.3 vimrc

```
1 syntax enable
2 set syntax=on
3 set nobackup
4 set noswapfile
5 set noundofile
6 set nu
7 set smartindent
8 set cindent
9 set foldmethod=marker
10 set foldlevel=3
11 set foldenable
12 set autowrite
13 set noeb
14 set tabstop=2
15 set softtabstop=2
16 set shiftwidth=2
17 set expandtab
18 set anti enc=utf-8
19 set guifont=GoMono\ Nerd\ Font\ 13
20
21 :imap jk <Esc>
22
23 map <F5> : call Complie() <CR>
24
25 func Complie()
26     exec "w"
27     exec "!g++ % -o %< -g -Wall -std=c++11"
28 endfunc
29
30 map <F6> : call Run() <CR>
31
```

```
32 func Run()
33     exec "!./%<"
34 endfunc
35
36 map <F9> : call DeBug() <CR>
37
38 func DeBug()
39     exec "!gdb %<"
40 endfunc
41
42 set guioptions-=T
43 set guioptions-=m
44 set guioptions-=r
45 set guioptions-=egrL
46 set cursorline
47 :nn <M-1> 1gt
48 :nn <M-2> 2gt
49 :nn <M-3> 3gt
50 :nn <M-4> 4gt
51 :nn <M-5> 5gt
52 :nn <M-6> 6gt
53 :nn <M-7> 7gt
54 :nn <M-8> 8gt
55 :nn <M-9> 9gt
56 :nn <M-t> :tabnew<CR>
57 :nn <M-w> :close<CR>
58 :nn <C-Tab> :tabnext<CR>
```

8.4 myalloc

```
1 // usage: vector<int, myalloc<int>> L;
2 static char space[10000000], *sp = space;
3 template <typename T> struct myalloc : allocator<T> {
4     myalloc() {}
5     template <typename T2> myalloc(const myalloc<T2> &a) {}
6     template <typename T2> myalloc<T> &operator=(const myalloc<T2> &a) {
7         return *this;
8     }
9     template <typename T2> struct rebind { typedef myalloc<T2> other; };
10    inline T *allocate(size_t n) {
11        T *result = (T *)sp;
12        sp += n * sizeof(T);
13        return result;
14    }
15    inline void deallocate(T *p, size_t n) {}
16 };
```

8.5 duipai

```
1 #/usr/bin/bash
2
3 while true; do
4     python gen_data.py
5     ./E < input.txt > output.txt
6     ./E_r <input.txt > r.txt
7     if diff output.txt r.txt; then
8         printf AC
```

```
9  else
10     echo WA
11     exit 0
12 fi
13 done
```