ACM/ICPC Template Manaual

Harbin Institute of Technology

cycleke

September 22, 2019

Contents

1 Math

1.1 LinearSieve

```cpp
const int MAXN = 1e7 + 5;

bool vis[MAXN];
int prime[MAXN / 10], prime_cnt;
int fac[MAXN], e[MAXN], d[MAXN], mu[MAXN], phi[MAXN];

void sieve() {
  fac[1] = 1;
  e[1] = 0;
  d[1] = 1;
  mu[1] = 1;
  phi[1] = 1;
  for (int i = 2; i < MAXN; ++i) {
    if (!vis[i]) {
      prime[prime_cnt++] = i;
      fac[i] = i;
      e[i] = 1;
      d[i] = 2;
      mu[i] = -1;
      phi[i] = i - 1;
    }
    for (int j = 0; j < prime_cnt; ++j) {
      int t = prime[j] * i;
      if (t >= MAXN) { break; }
      vis[t] = true;
      fac[t] = prime[j];
      if (i % prime[j] == 0) {
        e[t] = e[i] + 1;
        d[t] = d[i] / (e[i] + 1) * (e[t] + 1);
        mu[t] = 0;
        phi[t] = phi[i] * prime[j];
        break;
      } else {
        e[t] = 1;
        d[t] = d[i] * 2;
        mu[t] = -mu[i];
        phi[t] = phi[i] * (prime[j] - 1);
      }
    }
  }
}
```

1.2 lucas

```cpp
// C(n, m) = C(n / p, m / p) * C(n % p, m % p) (mod p)
ll lucas(ll n, ll k, int p) {
  ll ret = 1;
  while (n && k) {
    ll nn = n % p, kk = k % p;
    if (nn < kk) return 0;
    ret = ret * f[nn] * mod_pow(f[kk] * f[nn - kk] % p, p - 2, p) % p;
    n /= p, k /= p;
  }
  return res;
}
```

1.3 Pollard rho

```
inline ll rand64(ll x) {
  return 1ll * ((rand() << 15 ^ rand()) << 30 ^ (rand() << 15 ^ rand())) % x;
}

inline ll Pollard_rho(const ll &x, const int &y) {
  ll v0 = rand64(x - 1) + 1, v = v0, d, s = 1;
  for (register int t = 0, k = 1;;) {
    if (v = (mod_mul(v, v, x) + y) % x, s = mod_mul(s, abs(v - v0), x),
        !(v ^ v0) || !s)
      return x;
    if (++t == k) {
      if ((d = __gcd(s, x)) ^ 1) return d;
      v0 = v, k <<= 1;
    }
  }
}

ll ans;
vector<ll> factor;
void findfac(ll n) {
  if (Miller_Rabin(n)) {
    factor.push_back(n);
    return;
  }
  ll p = n;
  while (p >= n) { p = Pollard_rho(p, rand64(n - 1) + 1); }
  findfac(p);
  findfac(n / p);
}
```

1.4 china

```
int china(int n, int *a, int *m) {
  int lcm = 1, res = 0;
  for (int i = 0; i < n; ++i) lcm *= m[i];
  for (int i = 0; i < n; ++i) {
    int t = lcm / m[i], x, y;
    exgcd(t, m[i], x, y);
    x = (x % m[i] + m[i]) % m[i];
    res = (res + 1LL * t * x) % lcm;
  }
  return res;
}
```

1.5 exctr

```
int exctr(int n, int *a, int *m) {
  int M = m[0], res = a[0];
  for (int i = 1; i < n; ++i) {
    int a = M, b = m[i], c = (a[i] - res % b + b) % b, x, y;
    int g = exgcd(a, b, x, y), bg = b / g;
    if (c % g != 0) return -1;
    x = 1LL * x * (c / g) % bg;
    res += x * M;
    M *= bg;
    res = (res % M + M) % M;
  }
  return res;
}
```

1.6 exgcd

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

1.7 杜教筛

```
// e = mu x 1
// d = 1 x 1
// sigma = d x 1
// phi = mu x id
// id = phi x 1
// id^2 = (id * phi) x id

// S = sum(f)
// sum(fxg) = sum(g(i)S(n/i))
map<int, int> mp_mu;

int S_mu(int n) {
  if (n < MAXN) return sum_mu[n];
  if (mp_mu[n]) return mp_mu[n];
  int ret = 1;
  for (int i = 2, j; i <= n; i = j + 1) {
    j = n / (n / i);
    ret -= S_mu(n / i) * (j - i + 1);
  }
  return mp_mu[n] = ret;
}

ll S_phi(int n) {
  ll res = 0;
  for (int i = 1, j; i <= n; i = j + 1) {
    j = n / (n / i);
    res += 1LL * (S_mu(j) - S_mu(i - 1)) * (n / i) * (n / i);
  }
  return (res - 1) / 2 + 1;
}
```

1.8 FFT

```
const int MAXN = 4 * 1e5 + 3;
const double PI = acos(-1);
complex<double> a[MAXN], b[MAXN];

```

```
int n, bit;
int rev[MAXN];

void fft(complex<double> *a, int sign) {
  for (int i = 0; i < n; ++i)
    if (i < rev[i]) swap(a[i], a[rev[i]]);

  for (int j = 1; j < n; j <<= 1) {
    complex<double> wn(cos(2 * PI / (j << 1)), sign * sin(2 * PI / (j << 1)));
    for (int i = 0; i < n; i += (j << 1)) {
      complex<double> w(1, 0), t0, t1;
      FOR(k, 0, j) {
        t0 = a[i + k];
        t1 = w * a[i + j + k];
        a[i + k] = t0 + t1;
        a[i + j + k] = t0 - t1;
        w *= wn;
      }
    }
  }
  if (sign == -1)
    for (int i = 0; i < n; ++i) a[i] /= n;
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  int n, m, x;
  cin >> n >> m;
  for (int i = 0; i <= n; ++i) {
    cin >> x;
    a[i].real(x);
  }
  for (int i = 0; i <= m; ++i) {
    cin >> x;
    b[i].real(x);
  }

  ::n = 1;
  bit = 0;
  while (::n <= n + m) {
    ::n <<= 1;
    ++bit;
  }
  rev[0] = 0;
  FOR(i, 1, ::n) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
  fft(a, 1);
  fft(b, 1);
  FOR(i, 0, ::n) a[i] *= b[i];
  fft(a, -1);
  FOR(i, 0, n + m + 1) cout << int(a[i].real() + .5) << " ";
  cout << "\n";
  return 0;
}
```

1.9 LinearRecurrence

```cpp
struct LinearRecurrence {
  using int64 = long long;
  using vec = std::vector<int64>;

  static void extand(vec &a, size_t d, int64 value = 0) {
    if (d <= a.size()) return;
    a.resize(d, value);
  }

  static vec BerlekampMassey(const vec &s, int64 mod) {
    std::function<int64(int64)> inverse = [&](int64 a) {
      return a == 1 ? 1 : (int64)(mod - mod / a) * inverse(mod % a) % mod;
    };
    vec A = {1}, B = {1};
    int64 b = s[0];
    for (size_t i = 1, m = 1; i < s.size(); ++i, m++) {
      int64 d = 0;
      for (size_t j = 0; j < A.size(); ++j) { d += A[j] * s[i - j] % mod; }
      if (!(d %= mod)) continue;
      if (2 * (A.size() - 1) <= i) {
        auto temp = A;
        extand(A, B.size() + m);
        int64 coef = d * inverse(b) % mod;
        for (size_t j = 0; j < B.size(); ++j) {
          A[j + m] -= coef * B[j] % mod;
          if (A[j + m] < 0) A[j + m] += mod;
        }
        B = temp, b = d, m = 0;
      } else {
        extand(A, B.size() + m);
        int64 coef = d * inverse(b) % mod;
        for (size_t j = 0; j < B.size(); ++j) {
          A[j + m] -= coef * B[j] % mod;
          if (A[j + m] < 0) A[j + m] += mod;
        }
      }
    }
    return A;
  }

  static void exgcd(int64 a, int64 b, int64 &g, int64 &x, int64 &y) {
    if (!b)
      x = 1, y = 0, g = a;
    else {
      exgcd(b, a % b, g, y, x);
      y -= x * (a / b);
    }
  }

  static int64 crt(const vec &c, const vec &m) {
    int n = c.size();
    int64 M = 1, ans = 0;
    for (int i = 0; i < n; ++i) M *= m[i];
    for (int i = 0; i < n; ++i) {
      int64 x, y, g, tm = M / m[i];
      exgcd(tm, m[i], g, x, y);
      ans = (ans + tm * x * c[i] % M) % M;
    }
    return (ans + M) % M;
```

```
}

static vec ReedsSloane(const vec &s, int64 mod) {
  auto inverse = [](int64 a, int64 m) {
    int64 d, x, y;
    exgcd(a, m, d, x, y);
    return d == 1 ? (x % m + m) % m : -1;
  };
  auto L = [](const vec &a, const vec &b) {
    int da = (a.size() > 1 || (a.size() == 1 && a[0])) ? a.size() - 1 : -1000;
    int db = (b.size() > 1 || (b.size() == 1 && b[0])) ? b.size() - 1 : -1000;
    return std::max(da, db + 1);
  };
  auto prime_power = [&](const vec &s, int64 mod, int64 p, int64 e) {
    // linear feedback shift register mod p^e, p is prime
    std::vector<vec> a(e), b(e), an(e), bn(e), ao(e), bo(e);
    vec t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
    ;
    pw[0] = 1;
    for (int i = pw[0] = 1; i <= e; ++i) pw[i] = pw[i - 1] * p;
    for (int64 i = 0; i < e; ++i) {
      a[i] = {pw[i]}, an[i] = {pw[i]};
      b[i] = {0}, bn[i] = {s[0] * pw[i] % mod};
      t[i] = s[0] * pw[i] % mod;
      if (t[i] == 0) {
        t[i] = 1, u[i] = e;
      } else {
        for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i])
          ;
      }
    }
    for (size_t k = 1; k < s.size(); ++k) {
      for (int g = 0; g < e; ++g) {
        if (L(an[g], bn[g]) > L(a[g], b[g])) {
          ao[g] = a[e - 1 - u[g]];
          bo[g] = b[e - 1 - u[g]];
          to[g] = t[e - 1 - u[g]];
          uo[g] = u[e - 1 - u[g]];
          r[g] = k - 1;
        }
      }
      a = an, b = bn;
      for (int o = 0; o < e; ++o) {
        int64 d = 0;
        for (size_t i = 0; i < a[o].size() && i <= k; ++i) {
          d = (d + a[o][i] * s[k - i]) % mod;
        }
        if (d == 0) {
          t[o] = 1, u[o] = e;
        } else {
          for (u[o] = 0, t[o] = d; t[o] % p == 0; t[o] /= p, ++u[o])
            ;
          int g = e - 1 - u[o];
          if (L(a[g], b[g]) == 0) {
            extand(bn[o], k + 1);
            bn[o][k] = (bn[o][k] + d) % mod;
          } else {
            int64 coef =
                t[o] * inverse(to[g], mod) % mod * pw[u[o] - uo[g]] % mod;
            int m = k - r[g];
            extand(an[o], ao[g].size() + m);
            extand(bn[o], bo[g].size() + m);
            for (size_t i = 0; i < ao[g].size(); ++i) {
              an[o][i + m] -= coef * ao[g][i] % mod;
              if (an[o][i + m] < 0) an[o][i + m] += mod;
            }
            while (an[o].size() && an[o].back() == 0) an[o].pop_back();
            for (size_t i = 0; i < bo[g].size(); ++i) {
              bn[o][i + m] -= coef * bo[g][i] % mod;
              if (bn[o][i + m] < 0) bn[o][i + m] -= mod;
            }
            while (bn[o].size() && bn[o].back() == 0) bn[o].pop_back();
          }
        }
      }
    }
    return std::make_pair(an[0], bn[0]);
  };

  std::vector<std::tuple<int64, int64, int>> fac;
  for (int64 i = 2; i * i <= mod; ++i)
    if (mod % i == 0) {
      int64 cnt = 0, pw = 1;
      while (mod % i == 0) mod /= i, ++cnt, pw *= i;
      fac.emplace_back(pw, i, cnt);
    }
  if (mod > 1) fac.emplace_back(mod, mod, 1);
  std::vector<vec> as;
  size_t n = 0;
  for (auto &&x : fac) {
    int64 mod, p, e;
    vec a, b;
    std::tie(mod, p, e) = x;
    auto ss = s;
    for (auto &&x : ss) x %= mod;
    std::tie(a, b) = prime_power(ss, mod, p, e);
    as.emplace_back(a);
    n = std::max(n, a.size());
  }
  vec a(n), c(as.size()), m(as.size());
  for (size_t i = 0; i < n; ++i) {
    for (size_t j = 0; j < as.size(); ++j) {
      m[j] = std::get<0>(fac[j]);
      c[j] = i < as[j].size() ? as[j][i] : 0;
    }
    a[i] = crt(c, m);
  }
  return a;
}

LinearRecurrence(const vec &s, const vec &c, int64 mod)
    : init(s), trans(c), mod(mod), m(s.size()) {}

LinearRecurrence(const vec &s, int64 mod, bool is_prime = true) : mod(mod) {
  vec A;
  if (is_prime)
    A = BerlekampMassey(s, mod);
  else
```

```
      A = ReedsSloane(s, mod);
    if (A.empty()) A = {0};
    m = A.size() - 1;
    trans.resize(m);
    for (int i = 0; i < m; ++i) { trans[i] = (mod - A[i + 1]) % mod; }
    std::reverse(trans.begin(), trans.end());
    init = {s.begin(), s.begin() + m};
  }

  int64 calc(int64 n) {
    if (mod == 1) return 0;
    if (n < m) return init[n];
    vec v(m), u(m << 1);
    int msk = !!n;
    for (int64 m = n; m > 1; m >>= 1) msk <<= 1;
    v[0] = 1 % mod;
    for (int x = 0; msk; msk >>= 1, x <<= 1) {
      std::fill_n(u.begin(), m * 2, 0);
      x |= !!(n & msk);
      if (x < m)
        u[x] = 1 % mod;
      else { // can be optimized by fft/ntt
        for (int i = 0; i < m; ++i) {
          for (int j = 0, t = i + (x & 1); j < m; ++j, ++t) {
            u[t] = (u[t] + v[i] * v[j]) % mod;
          }
        }
        for (int i = m * 2 - 1; i >= m; --i) {
          for (int j = 0, t = i - m; j < m; ++j, ++t) {
            u[t] = (u[t] + trans[j] * u[i]) % mod;
          }
        }
      }
      v = {u.begin(), u.begin() + m};
    }
    int64 ret = 0;
    for (int i = 0; i < m; ++i) { ret = (ret + v[i] * init[i]) % mod; }
    return ret;
  }

  vec init, trans;
  int64 mod;
  int m;
};
```

1.10 Miller Rabin

```
inline ll mod_mul(const ll &a, const ll &b, const ll &mod) {
  ll k = (ll)((1.0L * a * b) / (1.0L * mod)), t = a * b - k * mod;
  t -= mod;
  while (t < 0) t += mod;
  return t;
}
inline ll mod_pow(ll a, ll b, const ll &mod) {
  ll res = 1;
  for (; b; b >>= 1, a = mod_mul(a, a, mod))
    (b & 1) && (res = mod_mul(res, a, mod));
  return res;
}

inline bool check(const ll &x, const ll &p) {
  if (!(x % p) || mod_pow(p % x, x - 1, x) ^ 1) return false;
  ll k = x - 1, t;
  while (~k & 1) {
    if (((t = mod_pow(p % x, k >>= 1, x)) ^ 1) && (t ^ (x - 1))) return false;
    if (!(t ^ (x - 1))) return true;
  }
  return true;
}

inline bool Miller_Rabin(const ll &x) {
  if (x < 2) return false;
  static const int p[12] = {2, 3, 5, 7, 11, 13, 17, 19, 61, 2333, 4567, 24251};
  for (int i = 0; i < 12; ++i) {
    if (!(x ^ p[i])) return true;
    if (!check(x, p[i])) return false;
  }
  return true;
}
```

1.11 BGSG

```
// Finds the primitive root modulo p
int generator(int p) {
  vector<int> fact;
  int phi = p - 1, n = phi;
  for (int i = 2; i * i <= n; ++i) {
    if (n % i == 0) {
      fact.push_back(i);
      while (n % i == 0) n /= i;
    }
  }
  if (n > 1) fact.push_back(n);
  for (int res = 2; res <= p; ++res) {
    bool ok = true;
    for (int factor : fact)
      if (mod_pow(res, phi / factor, p) == 1) {
        ok = false;
        break;
      }

    if (ok) return res;
  }
  return -1;
}
// This program finds all numbers x such that x^k=a (mod n)
vector<int> BSGS(int n, int k, int a) {
  if (a == 0) return vector<int>({0});

  int g = generator(n);
  // Baby-step giant-step discrete logarithm algorithm
  int sq = (int)sqrt(n + .0) + 1;
  vector<pair<int, int>> dec(sq);
  for (int i = 1; i <= sq; ++i)
    dec[i - 1] = {mod_pow(g, i * sq * k % (n - 1), n), i};

  sort(dec.begin(), dec.end());
  int any_ans = -1;
  for (int i = 0; i < sq; ++i) {
    int my = mod_pow(g, i * k % (n - 1), n) * a % n;
    auto it = lower_bound(dec.begin(), dec.end(), make_pair(my, 0));
    if (it != dec.end() && it->first == my) {
      any_ans = it->second * sq - i;
      break;
    }
  }
  if (any_ans == -1) return vector<int>();
  // Print all possible answers
  int delta = (n - 1) / __gcd(k, n - 1);
  vector<int> ans;
  for (int cur = any_ans % delta; cur < n - 1; cur += delta)
    ans.push_back(mod_pow(g, cur, n));
  sort(ans.begin(), ans.end());
  return ans;
}
```

1.12 gauss

```

const double EPS = 1e-9;
const int MAXN = MAX_NODE;
double a[MAXN][MAXN], x[MAXN];
int equ, var;

int gauss() {
  int i, j, k, col, max_r;
  for (k = 0, col = 0; k < equ && col < var; k++, col++) {
    max_r = k;
    for (i = k + 1; i < equ; i++)
      if (fabs(a[i][col]) > fabs(a[max_r][col])) max_r = i;
    if (fabs(a[max_r][col]) < EPS) return 0;

    if (k != max_r) {
      for (j = col; j < var; j++) swap(a[k][j], a[max_r][j]);
      swap(x[k], x[max_r]);
    }

    x[k] /= a[k][col];
    for (j = col + 1; j < var; j++) a[k][j] /= a[k][col];
    a[k][col] = 1;

    for (i = k + 1; i < equ; i++)
      if (i != k) {
        x[i] -= x[k] * a[i][col];
        for (j = col + 1; j < var; j++) a[i][j] -= a[k][j] * a[i][col];
        a[i][col] = 0;
      }
  }

  for (col = equ - 1, k = var - 1; ~col; --col, --k) {
    if (fabs(a[col][k]) > 0) {
      for (i = 0; i < k; ++i) {
        x[i] -= x[k] * a[i][col];
        for (j = col + 1; j < var; j++) a[i][j] -= a[k][j] * a[i][col];
        a[i][col] = 0;
      }
    }
  }

  return 1;
}
```

1.13 类欧几里德算法

```
//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],
//整除向下
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int P = 998244353;
int i2 = 499122177, i6 = 166374059;
struct data {
  data() { f = g = h = 0; }
  int f, g, h;
}; // 三个函数打包
data calc(int n, int a, int b, int c) {
  int ac = a / c, bc = b / c, m = (a * n + b) / c, n1 = n + 1, n21 = n * 2 + 1;
  data d;
  if (a == 0) { // 迭代到最底层
    d.f = bc * n1 % P;
    d.g = bc * n % P * n1 % P * i2 % P;
    d.h = bc * bc % P * n1 % P;
    return d;
  }
  if (a >= c || b >= c) { // 取模
    d.f = n * n1 % P * i2 % P * ac % P + bc * n1 % P;
    d.g = ac * n % P * n1 % P * n21 % P * i6 % P + bc * n % P * n1 % P * i2 % P;
    d.h = ac * ac % P * n % P * n1 % P * n21 % P * i6 % P +
          bc * bc % P * n1 % P + ac * bc % P * n % P * n1 % P;
    d.f %= P, d.g %= P, d.h %= P;

    data e = calc(n, a % c, b % c, c); // 迭代

    d.h += e.h + 2 * bc % P * e.f % P + 2 * ac % P * e.g % P;
    d.g += e.g, d.f += e.f;
    d.f %= P, d.g %= P, d.h %= P;
    return d;
  }
  data e = calc(m - 1, c, c - b - 1, a);
  d.f = n * m % P - e.f, d.f = (d.f % P + P) % P;
  d.g = m * n % P * n1 % P - e.h - e.f, d.g = (d.g * i2 % P + P) % P;
  d.h = n * m % P * (m + 1) % P - 2 * e.g - 2 * e.f - d.f;
  d.h = (d.h % P + P) % P;
  return d;
}

int T, n, a, b, c;
signed main() {
  scanf("%lld", &T);
  while (T--) {
    scanf("%lld%lld%lld%lld", &n, &a, &b, &c);
    data ans = calc(n, a, b, c);
    printf("%lld %lld %lld\n", ans.f, ans.h, ans.g);
  }
  return 0;
}
```

2 Dynamic Programming

2.1 斜率优化

```
// 树上斜率优化
// 定义dpi 表示i节点传递到根节点的最短耗时, 规定dproot=-P。
// 有如下转移方程dpu=dpv+dist(u,v)^2+P,v is an ancestor of u.

#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
const int MAXN = 1e5 + 5;

vector<pii> adj[MAXN];
ll dp[MAXN], d[MAXN];
int n, p, q[MAXN], head, tail;

inline ll S(int a, int b) { return (d[b] - d[a]) << 1; }
inline ll G(int a, int b) { return dp[b] - dp[a] + d[b] * d[b] - d[a] * d[a]; }

void dfs(int u, int from) {
  vector<int> dhead, dtail;
  if (u ^ 1) {
    while (head + 2 <= tail &&
           S(q[head + 1], q[head]) * d[u] <= G(q[head + 1], q[head]))
      dhead.push_back(q[head++]);
    int v = q[head];
    dp[u] = dp[v] + p + (d[u] - d[v]) * (d[u] - d[v]);
  }
  while (head + 2 <= tail &&
         G(u, q[tail - 1]) * S(q[tail - 1], q[tail - 2]) <=
             G(q[tail - 1], q[tail - 2]) * S(u, q[tail - 1]))
    dtail.push_back(q[--tail]);
  q[tail++] = u;
  for (pii &e : adj[u]) {
    if (e.first == from) continue;
    d[e.first] = d[u] + e.second;
    dfs(e.first, u);
  }
  --tail;
  for (int i = dtail.size() - 1; ~i; --i) q[tail++] = dtail[i];
  for (int i = dhead.size() - 1; ~i; --i) q[--head] = dhead[i];
}

void solve() {
  cin >> n >> p;
  for (int i = 1; i <= n; ++i) adj[i].clear();
  for (int i = 1, u, v, w; i < n; ++i) {
    cin >> u >> v >> w;
    adj[u].emplace_back(v, w);
    adj[v].emplace_back(u, w);
  }
  dp[1] = -p;
  head = tail = 0;
  dfs(1, 1);

  ll ans = 0;
  for (int i = 1; i <= n; ++i)
    if (dp[i] > ans) ans = dp[i];
  cout << ans << '\n';
}

int main() {
  // freopen("in.txt", "r", stdin);
  ios::sync_with_stdio(false);
  cin.tie(0);

  int o_o;
  for (cin >> o_o; o_o; --o_o) solve();

  return 0;
}
```

3 Data Structure

3.1 zkw

```
int tree[MAXN * 2], pre;

void init(int n, int *a) {
  memset(tree, 0, sizeof(tree));
  for (pre = 1; pre <= n; pre <<= 1) {}
  for (int i = 1; i <= n; ++i) tree[i + pre] = a[i];
  for (int i = pre; i; --i) tree[i] = max(tree[i << 1], tree[i << 1 | 1]);
}

void update(int pos, const int &val) {
  tree[pos += pre] = val;
  for (pos >>= 1; pos; pos >>= 1)
    tree[pos] = max(tree[pos << 1], tree[pos << 1 | 1]);
}

int query(int s, int t) {
  int res = 0;
  for (s += pre - 1, t += pre + 1; s ^ t ^ 1; s >>= 1, t >>= 1) {
    if (~s & 1) res = max(res, tree[s ^ 1]);
    if (t & 1) res = max(res, tree[t ^ 1]);
  }
  return res;
}
```

3.2 splay

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
```

```

const int MAXN = 2e5 + 10;

struct Node {
  long long sum;
  int id, val, lazy, size;
  Node *fa, *ch[2];
} node_pool[MAXN], *pool_it, *root, *nil;

Node *newnode(int id, int val) {
  pool_it->id = id;
  pool_it->lazy = 0;
  pool_it->size = 1;
  pool_it->sum = pool_it->val = val;
  pool_it->fa = pool_it->ch[0] = pool_it->ch[1] = nil;
  return pool_it++;
}

void maintain(Node *u) {
  if (u == nil) { return; }
  u->size = u->ch[0]->size + u->ch[1]->size + 1;
  u->sum = u->ch[0]->sum + u->ch[1]->sum + u->val;
}

void push_down(Node *u) {
  if (u->lazy) {
    if (u->ch[0] != nil) {
      u->ch[0]->val += u->lazy;
      u->ch[0]->sum += 1LL * u->ch[0]->size * u->lazy;
      u->ch[0]->lazy += u->lazy;
    }
    if (u->ch[1] != nil) {
      u->ch[1]->val += u->lazy;
      u->ch[1]->sum += 1LL * u->ch[1]->size * u->lazy;
      u->ch[1]->lazy += u->lazy;
    }
    u->lazy = 0;
  }
}

inline void rot(Node *u) {
  Node *f = u->fa, *ff = f->fa;
  int d = u == f->ch[1];
  push_down(f);
  push_down(u);
  if ((f->ch[d] = u->ch[d ^ 1]) != nil) f->ch[d]->fa = f;
  if ((u->fa = ff) != nil) ff->ch[f == ff->ch[1]] = u;
  f->fa = u;
  u->ch[d ^ 1] = f;
  maintain(f);
  maintain(u);
}

void splay(Node *u, Node *target) {
  for (Node *f; u->fa != target; rot(u))
    if ((f = u->fa)->fa != target) {
      ((u == f->ch[1]) ^ (f == f->fa->ch[1])) ? rot(u) : rot(f);
    }
  if (target == nil) root = u;
```

```
}

inline void insert(int id, int val) {
  if (root == nil) {
    root = newnode(id, val);
    return;
  }
  Node *u = root;
  while (u != nil) {
    int d = id >= u->id;
    ++u->size;
    push_down(u);
    u->sum += val;
    if (u->ch[d] != nil) {
      u = u->ch[d];
    } else {
      u->ch[d] = newnode(id, val);
      u->ch[d]->fa = u;
      u = u->ch[d];
      break;
    }
  }
  splay(u, nil);
}

inline Node *find_pred(int id) {
  Node *u = root, *ret = nil;
  while (u != nil) {
    push_down(u);
    if (u->id < id) {
      ret = u;
      u = u->ch[1];
    } else {
      u = u->ch[0];
    }
  }
  return ret;
}

inline Node *find_succ(int id) {
  Node *u = root, *ret = nil;
  while (u != nil) {
    push_down(u);
    if (u->id > id) {
      ret = u;
      u = u->ch[0];
    } else {
      u = u->ch[1];
    }
  }
  return ret;
}

Node *find_kth(int k) {
  Node *u = root;
  while (u != nil) {
    push_down(u);
    if (u->ch[0]->size + 1 == k) {
      splay(u, nil);
      return u;
    }
    if (u->ch[0]->size >= k) {
      u = u->ch[0];
    } else {
      k -= u->ch[0]->size + 1;
      u = u->ch[1];
    }
  }
  return nil;
}

Node *range(int l, int r) {
  Node *pred = find_pred(l);
  Node *succ = find_succ(r);

  splay(pred, nil);
  splay(succ, root);
  push_down(pred);
  push_down(succ);
  return root->ch[1]->ch[0];
}

int main() {

  // freopen("input.txt", "r", stdin);

  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  int n;
  cin >> n;

  pool_it = node_pool;
  nil = pool_it++;
  nil->ch[0] = nil->ch[1] = nil->fa = nil;
  nil->id = -1;
  nil->val = 0;
  root = nil;

  insert(-0x3fffffff, 0);
  insert(0x3fffffff, 0);

  return 0;
}
```

4 String

4.1 da

```
char s[MAXN];
int sa[MAXN], x[MAXN], y[MAXN], c[MAXN];
int rk[MAXN], height[MAXN], st[17][MAXN], lg[MAXN];

bool cmp(int *r, int i, int j, int l) {
  return r[i] == r[j] && r[i + l] == r[j + l];
}
```

```cpp
void da(char *s, int n, int m) {
  int i, j, p;
  for (i = 0; i < m; ++i) c[i] = 0;
  for (i = 0; i < n; ++i) ++c[x[i] = s[i]];
  for (i = 1; i < m; ++i) c[i] += c[i - 1];
  for (i = n - 1; ~i; --i) sa[--c[x[i]]] = i;
  for (p = j = 1; p < n; j <<= 1, m = p) {
    for (p = 0, i = n - j; i < n; ++i) y[p++] = i;
    for (i = 0; i < n; ++i)
      if (sa[i] >= j) y[p++] = sa[i] - j;
    for (i = 0; i < m; ++i) c[i] = 0;
    for (i = 0; i < n; ++i) ++c[x[y[i]]];
    for (i = 1; i < m; ++i) c[i] += c[i - 1];
    for (i = n - 1; ~i; --i) sa[--c[x[y[i]]]] = y[i];
    for (swap(x, y), p = 1, x[sa[0]] = 0, i = 1; i < n; ++i)
      x[sa[i]] = cmp(y, sa[i], sa[i - 1], j) ? p - 1 : p++;
  }
}

void get_height(char *s, int n) {
  int i, j, k;
  for (i = 0; i < n; ++i) rk[sa[i]] = i;
  for (i = k = height[rk[0]] = 0; i < n; height[rk[i++]] = k)
    if (rk[i])
      for (k > 0 ? --k : 0, j = sa[rk[i] - 1]; s[i + k] == s[j + k]; ++k) {}
}

void init_st_table(int n) {
  int lgn = lg[n];
  for (int i = 0; i < n; ++i) st[0][i] = height[i];
  for (int i = 1; i <= lgn; ++i)
    for (int j = 0; j + (1 << i - 1) < n; ++j)
      st[i][j] = min(st[i - 1][j], st[i - 1][j + (1 << i - 1)]);
}

int lcp(int i, int j) {
  if (i > j) swap(i, j);
  ++i;
  int lgl = lg[j - i + 1];
  return min(st[lgl][i], st[lgl][j - (1 << lgl) + 1]);
}
```

4.2 exkmp

```cpp
// next[i]:x[i...m-1] 与 x[0...m-1] 的最长公共前缀
// extend[i]:y[i...n-1] 与 x[0...m-1] 的最长公共前缀
void prework(char x[], int m, int next[]) {
  next[0] = m;
  int j = 0;
  while (j + 1 < m && x[j] == x[j + 1])
    j++;
  next[1] = j;
  int k = 1;
  for (int i = 2; i < m; i++) {
    int p = next[k] + k - 1;
    int L = next[i - k];
    if (i + L < p + 1)
      next[i] = L;
    else {
      j = max(0, p - i + 1);
      while (i + j < m && x[i + j] == x[j])
        j++;
      next[i] = j;
      k = i;
    }
  }
}
void exkmp(char x[], int m, char y[], int n, int next[], int extend[]) {
  prework(x, m, next);
  int j = 0;
  while (j < n && j < m && x[j] == y[j])
    j++;
  extend[0] = j;
  int k = 0;
  for (int i = 1; i < n; i++) {
    int p = extend[k] + k - 1;
    int L = next[i - k];
    if (i + L < p + 1)
      extend[i] = L;
    else {
      j = max(0, p - i + 1);
      while (i + j < n && j < m && y[i + j] == x[j])
        j++;
      extend[i] = j;
      k = i;
    }
  }
}
```

4.3 回文树

```
//最长双回文串
struct PT {
  char s[MAXL];
  int fail[MAXL], ch[26][MAXL], l[MAXL], dep[MAXL], lst, nc, n;
  void init() {
    l[0] = 0;
    l[1] = -1;
    fail[0] = fail[1] = 1;
    for (int i = 0; i < 26; ++i) {
      for (int j = 0; j < nc; ++j) {
        ch[i][j] = 0;
      }
    }
    for (int i = 2; i < nc; ++i) {
      l[i] = 0;
      fail[i] = 0;
    }

    lst = 0;
    nc = 2;
    n = 0;
    s[0] = '#';
  }

  int insert(char c) {
    int id = c - 'a';
    s[++n] = c;
    while (s[n - l[lst] - 1] != s[n]) {
      lst = fail[lst];
    }
    if (ch[id][lst] == 0) {
      l[nc] = l[lst] + 2;
      int f = fail[lst];
      while (s[n - l[f] - 1] != s[n]) {
        f = fail[f];
      }
      fail[nc] = ch[id][f];
      dep[nc] = dep[fail[nc]] + 1;
      ch[id][lst] = nc;
      ++nc;
    }
    lst = ch[id][lst];
    return lst;
  }
} pt;

char S[MAXL];
int len[MAXL];
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  cin >> S;
  int n = strlen(S);
  pt.init();
  for (int i = 0; i < n; ++i) {
    len[i] = pt.l[pt.insert(S[i])];
  }
  pt.init();
  int ans = 0;
  for (int i = n - 1; i; --i) {
    ans = max(ans, len[i - 1] + pt.l[pt.insert(S[i])]);
  }
  cout << ans << "\n";

  return 0;
}
```

4.4 SAM

```
struct Node {
  int len;
  Node *link, *ch[ALPHABET_SIZE];
} node_pool[MAXS], *node_it, *root, *last;

Node *new_node(int len) {
  node_it->len = len;
  return node_it++;
}
void sam_init() {
  node_it = node_pool;
  last = root = new_node(0);
}
void sam_extend(int c, int val) {
  Node *p = last, *np = new_node(p->len + 1);
  for (last = np; p && !p->ch[c]; p = p->link) p->ch[c] = np;
  if (!p) {
    np->link = root;
  } else {
    Node *q = p->ch[c];
    if (q->len == p->len + 1) {
      np->link = q;
    } else {
      Node *nq = new_node(p->len + 1);
      memcpy(nq->ch, q->ch, sizeof(q->ch));
      nq->link = q->link;
      q->link = np->link = nq;
      for (; p && p->ch[c] == q; p = p->link) p->ch[c] = nq;
    }
  }
}
```

4.5 ACam

```
int ch[MAX_NODE][26], fail[MAX_NODE], dep[MAX_NODE], node_c;

int add_char(int u, int id) {
  if (ch[u][id] < 0) ch[u][id] = node_c++;
  return ch[u][id];
}
void build_acam() {
  queue<int> que;
  FOR(i, 0, 26)
    if (~ch[0][i]) {
      que.push(ch[0][i]);
      fail[ch[0][i]] = 0;
      dep[ch[0][i]] = 1;
    } else {
      ch[0][i] = 0;
    }
  while (!que.empty()) {
    int u = que.front();
    que.pop();
    FOR(i, 0, 26)
      if (~ch[u][i]) {
        que.push(ch[u][i]);
        fail[ch[u][i]] = ch[fail[u]][i];
        dep[ch[u][i]] = dep[u] + 1;
      } else {
        ch[u][i] = ch[fail[u]][i];
      }
  }
  FOR(i, 1, node_c) adj[fail[i]].push_back(i);
}
```

4.6 mancher

```
void mancher(char *s, int n) {
  str[0] = '~';
  str[1] = '!';
  for (int i = 1; i <= n; ++i) {
    str[i * 2] = s[i];
    str[i * 2 + 1] = '!';
  }
  for (int i = 1, j = 0; i <= n; ++i) {
    if (p[j] + j > i) {
      p[i] = min(p[2 * j - i], p[j] + j - i);
    } else {
      p[i] = 1;
    }
    while (str[i + p[i]] == str[i - p[i]]) {
      ++p[i];
    }
    if (i + p[i] > j + p[j]) {
      j = i;
    }
  }
}
```

4.7 kmp

```
void get_next(char *S, int *nxt, int n) {
  nxt[0] = -1;
  int j = -1;
  for (int i = 1; i < n; ++i) {
    while ((~j) && S[j + 1] != S[i]) {
      j = nxt[j];
    }
    nxt[i] = (S[j + 1] == S[i]) ? (++j) : j;
  }
}

int pattern(char *S, char *T, int *nxt, int n, int m) {
  int j = -1;
  for (int i = 0; i < m; ++i) {
    while ((~j) && S[j + 1] != T[i]) {
      j = nxt[j];
    }
    j += S[j + 1] == T[i];
    if (j == n - 1) {
      return i - n + 1;
    }
  }
  return -1;
}
```

4.8 hash

```

const unsigned int KEY = 6151;
const unsigned int MOD = 1610612741;

unsigned int hash[MAXN], p[MAXN];

inline unsigned int get_hash(int l, int r) {
  return (hash[r] + MOD - 1ULL * hash[l - 1] * p[r - l + 1] % MOD) % MOD;
}

void init(char *s, int n) {
  p[0] = 1;
  for (int i = 1; i <= n; ++i) {
    p[i] = p[i - 1] * KEY % MOD;
    hash[i] = (1LL * hash[i - 1] * KEY + s[i]) % MOD;
  }
}
```

5 Graph Theory

5.1 KM

```
int n, m, match[MAXN];
int adj[MAXN][MAXN], lx[MAXN], ly[MAXN], slack[MAXN];
int visx[MAXN], visx_c, visy[MAXN], visy_c;

bool dfs(int x) {
  visx[x] = visx_c;
  for (int y = 0; y < m; ++y)
    if (visy[y] ^ visy_c) {
      int t = lx[x] + ly[y] - adj[x][y];
      if (!t) {
        visy[y] = visy_c;
        if (match[y] < 0 || dfs(match[y])) return match[y] = x, true;
      } else
        (slack[y] > t) && (slack[y] = t);
    }
  return false;
}

int KM() {
  memset(match, -1, sizeof(int) * m);
  memset(ly, 0, sizeof(int) * m);
  for (int i = 0; i < n; ++i) {
    lx[i] = -INF;
    for (int j = 0; j < m; ++j) (adj[i][j] > lx[i]) && (lx[i] = adj[i][j]);
  }
  for (int x = 0; x < n; ++x) {
    fill(slack, slack + m, INF);
    for (;;) {
      ++visx_c, ++visy_c;
      if (dfs(x)) break;
      int d = INF;
      for (int i = 0; i < m; ++i)
        (visy[i] ^ visy_c) && (d > slack[i]) && (d = slack[i]);
      for (int i = 0; i < n; ++i) (visx[i] == visx_c) && (lx[i] -= d);
      for (int i = 0; i < m; ++i)
        (visy[i] ^ visy_c) ? slack[i] -= d : ly[i] += d;
    }
  }
  int res = 0;
  for (int i = 0; i < m; ++i) (~match[i]) && (res += adj[match[i]][i]);
  return res;
}
```

6 Computational Geometry

7 Java

7.1 进制转换

```java
import java.io.*;
import java.util.*;
import java.math.*;

/**
 * Built using CHelper plug-in
 * Actual solution is at the top
 */
public class Main {
    public static void main(String[] args) {
        InputStream inputStream = System.in;
        OutputStream outputStream = System.out;
        Scanner in = new Scanner(inputStream);
        PrintWriter out = new PrintWriter(outputStream);
        Solver solver = new Solver();
        int testCount = Integer.parseInt(in.next());
        for (int i = 1; i <= testCount; i++)
            solver.solve(i, in, out);
        out.close();
    }

    static class Solver {
        public void solve(int testNumber, Scanner in, PrintWriter out) {
            int a = in.nextInt();
            int b = in.nextInt();
            String num = in.next();

            BigInteger value = BigInteger.ZERO;
            for (int i = 0; i < num.length(); ++i) {
                value = value.multiply(BigInteger.valueOf(a));
                value = BigInteger.valueOf(getValue(num.charAt(i))).add(value);
            }
            out.println(a + " " + num);

            if (value.equals(BigInteger.ZERO)) {
                out.println(b + " 0");
                out.println();
                return;
            }

            out.print(b + " ");

            char[] ans = new char[1000];
            int length = 0;
            while (!value.equals(BigInteger.ZERO)) {
                int digit = value.mod(BigInteger.valueOf(b)).intValue();
                value = value.divide(BigInteger.valueOf(b));
                ans[length] = getChar(digit);
                ++length;
            }

            for (int i = length - 1; i >= 0; --i) {
                out.print(ans[i]);
            }
            out.println("\n");
        }

        private int getValue(char ch) {
            if (ch >= 'A' && ch <= 'Z') {
                return ch - 'A' + 10;
            }
            if (ch >= 'a' && ch <= 'z') {
                return ch - 'a' + 36;
            }
            return ch - '0';
        }

        private char getChar(int x) {
            if (x < 10) {
                return (char) ('0' + x);
            } else if (x < 36) {
                return (char) ('A' + x - 10);
            } else {
                return (char) ('a' + x - 36);
            }
        }

    }
}
```

8 Others

8.1 vimrc

```
syntax enable
set syntax=on
set nobackup
set noswapfile
set noundofile
set nu
set smartindent
set cindent
set foldmethod=marker
set foldlevel=3
set foldenable
set autowrite
set noeb
set tabstop=2
set softtabstop=2
set shiftwidth=2
set expandtab
set anti enc=utf-8
set guifont=GoMono\ Nerd\ Font\ 13

:imap jk <Esc>

map <F5> : call Complie() <CR>

func Complie()
  exec "w"
  exec "!g++ % -o %< -g -Wall -std=c++11"
endfunc

map <F6> : call Run() <CR>

func Run()
  exec "!./%<"
endfunc

map <F9> : call DeBug() <CR>

func DeBug()
  exec "!gdb %<"
endfunc

set guioptions-=T
set guioptions-=m
set guioptions-=r
set guioptions-=egrL
set cursorline
:nn <M-1> 1gt
:nn <M-2> 2gt
:nn <M-3> 3gt
:nn <M-4> 4gt
:nn <M-5> 5gt
:nn <M-6> 6gt
:nn <M-7> 7gt
:nn <M-8> 8gt
:nn <M-9> 9gt
:nn <M-t> :tabnew<CR>
:nn <M-w> :close<CR>
:nn <C-Tab> :tabnext<CR>
```

8.2 FastIO

```
namespace FastIO {
struct Control {
  int ct, val;
  Control(int Ct, int Val = -1) : ct(Ct), val(Val) {}
  inline Control operator()(int Val) { return Control(ct, Val); }
} _endl(0), _prs(1), _setprecision(2);

const int IO_SIZE = 1 << 16 | 127;

struct FastIO {
  char in[IO_SIZE], *p, *pp, out[IO_SIZE], *q, *qq, ch[20], *t, b, K, prs;
  FastIO() : p(in), pp(in), q(out), qq(out + IO_SIZE), t(ch), b(1), K(6) {}
  ~FastIO() { fwrite(out, 1, q - out, stdout); }
  inline char getc() {
    return p == pp && (pp = (p = in) + fread(in, 1, IO_SIZE, stdin), p == pp)
               ? (b = 0, EOF)
               : *p++;
  }
  inline void putc(char x) {
    q == qq && (fwrite(out, 1, q - out, stdout), q = out), *q++ = x;
  }
  inline void puts(const char str[]) {
    fwrite(out, 1, q - out, stdout), fwrite(str, 1, strlen(str), stdout),
      q = out;
  }
  inline void getline(string &s) {
    s = "";
    for (char ch; (ch = getc()) != '\n' && b;) s += ch;
  }
#define indef(T)                                                       \
  inline FastIO &operator>>(T &x) {                                    \
    x = 0;                                                             \
    char f = 0, ch;                                                    \
    while (!isdigit(ch = getc()) && b) f |= ch == '-';                 \
    while (isdigit(ch)) x = (x << 1) + (x << 3) + (ch ^ 48), ch = getc();      \
    return x = f ? -x : x, *this;                                      \
  }
  indef(int);
  indef(long long);

  inline FastIO &operator>>(string &s) {
    s = "";
    char ch;
    while (isspace(ch = getc()) && b) {}
    while (!isspace(ch) && b) s += ch, ch = getc();
    return *this;
  }
  inline FastIO &operator>>(double &x) {
    x = 0;
    char f = 0, ch;
    double d = 0.1;
    while (!isdigit(ch = getc()) && b) f |= (ch == '-');
    while (isdigit(ch)) x = x * 10 + (ch ^ 48), ch = getc();
    if (ch == '.')
      while (isdigit(ch = getc())) x += d * (ch ^ 48), d *= 0.1;
    return x = f ? -x : x, *this;
  }
#define outdef(_T)                                                     \
  inline FastIO &operator<<(_T x) {                                    \
    !x && (putc('0'), 0), x < 0 && (putc('-'), x = -x);                \
    while (x) *t++ = x % 10 + 48, x /= 10;                             \
    while (t != ch) *q++ = *--t;                                       \
    return *this;                                                      \
  }
  outdef(int);
  outdef(long long);
  inline FastIO &operator<<(char ch) { return putc(ch), *this; }
  inline FastIO &operator<<(const char str[]) { return puts(str), *this; }
  inline FastIO &operator<<(const string &s) { return puts(s.c_str()), *this; }
  inline FastIO &operator<<(double x) {
    int k = 0;
    this->operator<<(int(x));
    putc('.');
    x -= int(x);
    prs && (x += 5 * pow(10, -K - 1));
    while (k < K) putc(int(x *= 10) ^ 48), x -= int(x), ++k;
    return *this;
  }
  inline FastIO &operator<<(const Control &cl) {
    switch (cl.ct) {
    case 0: putc('\n'); break;
    case 1: prs = cl.val; break;
    case 2: K = cl.val; break;
    }
    return *this;
  }
  inline operator bool() { return b; }
};
} // namespace FastIO
```

8.3 head

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;
typedef pair<int, int> pii;
typedef tuple<int, int, int> tiii;
typedef vector<int> vi;
typedef vector<vi> vvi;
typedef vector<long long> vll;
typedef vector<pii> vpii;

#define SZ(a) int((a).size())
#define ALL(a) (a).begin(), (a).end()
#define EXIST(s, x) ((s).find(x) != (s).end())
#define A_EXIST(a, x) (find(ALL(a), x) != (a).end())
#define ZERO(a) memset((a), 0, sizeof(a))

#define FOR(i, a, b) for (int i = int(a); i < int(b); ++i)
#define REP(i, a, b) for (int i = int(b) - 1; i >= a; --i)
#define FOR2(i, a, b, j, c, d) FOR(i, a, b) FOR(j, c, d)
#define REP2(i, a, b, j, c, d) REP(i, a, b) REP(j, c, d)
#define EACH(i, s) for (auto i = (s).begin(); i != (s).end(); ++i)
#define debug(...) fprintf(stderr, __VA_ARGS__)
#define dbg(x)                                                          \
  cerr << "debug: " << __FUNCTION__ << "() @ " << __TIMESTAMP__ << "\n"        \
       << __FILE__ << " L" << __LINE__ << "\n"                          \
       << #x " = " << (x) << endl

const int INF = 0x3ffffff;
const ll LL_INF = 0x3fffffffffffffffll;
const int MOD = 1e9 + 7;
const ll HASH_KEY = 6151;
const ll HASH_MOD = 1610612741;

// mt19937 rdm(chrono::steady_clock::now().time_since_epoch().count());
// uniform_int_distribution<int> u_int(begin, end);
// uniform_real_distribution<double> u_read(begin, end);
/* -- HEAD END -- */

void solve() {}

int main(int argc, char *argv[]) {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);
  cout << fixed << setprecision(10);

```

```cpp
  int o_o;
  for (o_o = 1; o_o; --o_o) solve();

  return 0;
}
```

8.4 myalloc

```cpp
// useage: vector<int, myalloc<int>> L;
static char space[10000000], *sp = space;
template <typename T> struct myalloc : allocator<T> {
  myalloc() {}
  template <typename T2> myalloc(const myalloc<T2> &a) {}
  template <typename T2> myalloc<T> &operator=(const myalloc<T2> &a) {
    return *this;
  }
  template <typename T2> struct rebind { typedef myalloc<T2> other; };
  inline T *allocate(size_t n) {
    T *result = (T *)sp;
    sp += n * sizeof(T);
    return result;
  }
  inline void deallocate(T *p, size_t n) {}
};
```

8.5 duipai

```bash
#/usr/bin/bash

while true; do
  python gen_data.py
  ./E < input.txt > output.txt
 ./E_r <input.txt > r.txt
  if diff output.txt r.txt; then
    printf AC
  else
    echo WA
    exit 0
  fi
done
```