cycleke（菜鸡）的 XCPC 模板

哈爾濱工業大學

cycleke

November 5, 2020

Contents

1 Math

1.1 BSGS

```cpp
// 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 (mpow(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] = {mpow(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 = mpow(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(mpow(g, cur, n));
  sort(ans.begin(), ans.end());
  return ans;
}
```

1.2 FFT

```cpp
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 (int k = 0; k < j; ++k, w *= wn) {
        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(nullptr), cout.tie(nullptr);

  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);
  }

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

1.3 Linear Recurrence

```cpp
struct LinearRecurrence {
```

```cpp
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;
}
```

```
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               extand(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];
```

```
            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 = is_prime ? BerlekampMassey(s, mod) : 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;
};
```

1.4 Linear Sieve

```
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];
// e 质因子最高次数, d 因数个数
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);
      }
    }
  }
}
```

1.5 Lucas

```
// 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] * mpow(f[kk] * f[nn - kk] % p, p - 2, p) % p;
    n /= p, k /= p;
  }
  return ret;
}
```

1.6 Miller Rabin

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

inline bool check(const ll &x, const ll &p) {
  if (!(x % p) || mpow(p % x, x - 1, x) ^ 1) return false;
  for (ll k = x - 1, t; ~k & 1;) {
    if (((t = mpow(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;
}
```

1.7 Pollard rho

```
mt19937_64 rnd(chrono::high_resolution_clock::now().time_since_epoch().count());
inline ll rand64(ll x) { return rnd() % x + 1; }

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

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));
  findfac(p), findfac(n / p);
}
```

1.8 burnside

```
// |X/G|={1 \over {|G|}} \sum_{g \in G}|X^g|
// Gym - 101873B:
// m边形,每边是n*n的矩形,用c种颜色染色,可进行水平旋转,问不同多边形个数。
#include <bits/stdc++.h>
using namespace std;

const int MOD = 1e9 + 7;

int mpow(int a, int b) {
  int r = 1;
  for (; b; b >>= 1, a = 1LL * a * a % MOD) (b & 1) && (r = 1LL * a * r % MOD);
  return r;
}

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

```

```
  int n, m, c, ans = 0;
  cin >> n >> m >> c;
  for (int i = 1; i <= m; ++i) ans = (ans + mpow(c, n * n * __gcd(i, m))) % MOD;
  ans = 1LL * ans * mpow(m, MOD - 2) % MOD;
  cout << ans << '\n';
  return 0;
}
```

1.9 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;
}
```

1.10 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;
}
```

1.11 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;
}
```

1.12 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 实数线性规划

```
// 求 max{cx | Ax <= b, x >= 0}
typedef vector<double> VD;
VD simplex(vector<VD> A, VD b, VD c) {
  int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
  vector<VD> D(n + 2, VD(m + 1, 0));
  vector<int> ix(n + m);
  for (int i = 0; i < n + m; ++i) ix[i] = i;
  for (int i = 0; i <= n; ++i) {
    for (int j = 0; j < m - 1; ++j) D[i][j] = -A[i][j];
    D[i][m - 1] = 1, D[i][m] = b[i];
    if (D[r][m] > D[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; ++j) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
  for (double d;;) {
    if (r < n) {
      swap(ix[s], ix[r + m]);
```

```
      D[r][s] = 1 / D[r][s];
      vector<int> speed_up;
      for (int j = 0; j <= m; ++j)
        if (j != s) {
          D[r][j] *= -D[r][s];
          if (D[r][j]) speed_up.push_back(j);
        }
      for (int i = 0; i <= n + 1; ++i)
        if (i != r) {
          for (int j : speed_up) D[i][j] += D[r][j] * D[i][s];
          D[i][s] *= D[r][s];
        }
    }
    r = -1, s = -1;
    for (int j = 0; j < m; ++j)
      if ((s < 0 || ix[s] > ix[j]) &&
          (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS && D[n][j] > EPS)))
        s = j;
    if (s < 0) break;
    for (int i = 0; i < n; ++i)
      if (D[i][s] < -EPS)
        if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS ||
            (d < EPS && ix[r + m] > ix[i + m]))
          r = i;
    if (r < 0) return VD(); //无边界
  }
  if (D[n + 1][m] < -EPS) return VD(); // 无解
  VD x(m - 1);
  for (int i = m; i < n + m; ++i)
    if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
  return x; // 最优值在D[n][m]
}
```

1.14 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;
}
```

1.15 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;
}
```

1.16 线性规划

```
/*
 * 给定 n 个约束条件, m 个未知数, 求 sum(a[0][i] * x[i]) 的最大值
 * 约束条件: sum(-a[i][j] * x[j]) <= a[i][0]
 * 若要求最小值,则进行对偶:即把目标函数的系数和约束条件右边的数交换,然后把矩阵转置
 */
const int MAXN = 3e3 + 3, MAXM = 3e3 + 3, INF = ~0U >> 2;
int n, m, a[MAXN][MAXM], nxt[MAXM];
void pivot(int l, int e) {
  a[l][e] = -1;
  int t = MAXM - 1;
  for (int i = 0; i <= m; ++i)
    if (a[l][i]) nxt[t] = i, t = i;
  nxt[t] = -1;
  for (int i = 0; i <= n; ++i)
    if (i != l && (t = a[i][e])) {
      a[i][e] = 0;
      for (int j = nxt[MAXM - 1]; ~j; j = nxt[j]) a[i][j] += a[l][j] * t;
    }
}
int simplex() {
  for (;;) {
    int mi = INF, l = 0, e = 0;
    for (int i = 1; i <= m; ++i)
      if (a[0][i] > 0) {
        e = i;
        break;
      }
    if (!e) return a[0][0];
    for (int i = 1; i <= n; ++i)
      if (a[i][e] < 0 && a[i][0] < mi) mi = a[i][0], l = i;
    pivot(l, e);
  }
}
```

1.17 线性规划例题 (CCPC Final 2017 F)

```
// 有N组人,每组人有ai个,可以进行若干次选择,每次选择一些至少有M个人的组,这些组的人都中奖。
// 现在要使每个人中奖概率相等,且中奖概率最大。N <= 10,M,ai <= 100

// 写法一
#include <bits/stdc++.h>
using namespace std;

const int MAXN = int(3e3);
const int MAXM = int(3e3);
const double INF = 1e20, EPS = 1e-9;

int n, m;
double a[MAXM][MAXN], v;
```

```
14
15  void pivot(int l, int e) {
16    int i, j;
17    a[l][e] = 1 / a[l][e];
18    for (j = 0; j <= n; ++j)
19      if (j != e) a[l][j] *= a[l][e];
20    for (i = 1; i <= m; ++i)
21      if (i != l && fabs(a[i][e]) > EPS) {
22        for (j = 0; j <= n; ++j)
23          if (j != e) a[i][j] -= a[i][e] * a[l][j];
24        a[i][e] = -a[i][e] * a[l][e];
25      }
26    v += a[0][e] * a[l][0];
27    for (j = 1; j <= n; ++j)
28      if (j != e) a[0][j] -= a[0][e] * a[l][j];
29    a[0][e] = -a[0][e] * a[l][e];
30  }
31
32  double simplex() {
33    int e, l, i;
34    double mn;
35    v = 0;
36    while (true) {
37      for (e = 1; e <= n; ++e)
38        if (a[0][e] > EPS) break;
39      if (e > n) return v;
40      for (i = 1, mn = INF; i <= m; ++i)
41        if (a[i][e] > EPS && mn > a[i][0] / a[i][e])
42          mn = a[i][0] / a[i][e], l = i;
43      if (mn == INF) return INF;
44      pivot(l, e);
45    }
46  }
47
48  void solve() {
49    static int n, m, g[10];
50    static vector<int> con[10], able;
51
52    scanf("%d %d", &n, &m);
53    for (int i = 0; i < n; ++i) {
54      scanf("%d", g + i);
55      con[i].clear();
56    }
57
58    if (n == 1) {
59      printf("%.10f\n", m >= g[0] ? 1. : 0.);
60      return;
61    }
62
63    able.clear();
64    for (int s = 0, S = 1 << n; s < S; ++s) {
65      int sum = 0;
66      for (int i = 0; i < n; ++i)
67        if (s >> i & 1) sum += g[i];
68      if (sum > m) continue;
69      able.push_back(s);
70      for (int i = 0; i < n; ++i)
71        if (s >> i & 1) con[i].push_back(able.size());
72    }
```

```
  ::n = able.size();
  ::m = 0;
  static random_device rd;
  mt19937 gen(rd());
  shuffle(able.begin(), able.end(), gen);
  for (int step = 0; step < n; ++step) {
    int f = ++::m;
    for (int i = 0; i <= ::n; ++i) a[f][i] = 0;
    for (int x : con[step]) ++a[f][x];
    if (step + 1 < n) {
      for (int x : con[step + 1]) --a[f][x];
    } else {
      for (int x : con[0]) --a[f][x];
    }
  }

  ++::m;
  a[::m][0] = 1;
  for (int i = 1; i <= ::n; ++i) a[::m][i] = 1;

  ++::m;
  a[::m][0] = -1;
  for (int i = 1; i <= ::n; ++i) a[::m][i] = -1;

  for (int i = 0; i <= ::n; ++i) a[0][i] = 0;
  for (int x : con[0]) ++a[0][x];
  printf("%.10f\n", simplex());
}

int main() {
  int o_o, case_number = 1;
  for (scanf("%d", &o_o); case_number <= o_o; ++case_number) {
    printf("Case #%d: ", case_number);
    solve();
  }
  return 0;
}

// 写法二
#include <bits/stdc++.h>
using namespace std;

typedef long double db;
const int MAXN = 3000;
const int MAXM = 3000;
const db EPS = 1e-9;
const db INF = 1e200;

namespace LP {
db a[MAXM][MAXN];
int idA[MAXN], idB[MAXN];
int m, n;

void put_out(int x) {
  if (x == 0)
    printf("Infeasible\n");
  else
    printf("Unbounded\n");
  exit(0);
```

```
}
void pivot(int xA, int xB) {
  swap(idA[xA], idB[xB]);
  static int next[MAXN];
  int i, j, last = MAXN - 1;
  db tmp = -a[xB][xA];
  a[xB][xA] = -1.0;
  for (j = 0; j <= n; j++)
    if (fabs(a[xB][j]) > EPS) a[xB][last = next[last] = j] /= tmp;
  next[last] = -1;

  for (i = 0; i <= m; i++)
    if (i != xB && fabs(tmp = a[i][xA]) > EPS)
      for (a[i][xA] = 0.0, j = next[MAXN - 1]; ~j; j = next[j])
        a[i][j] += tmp * a[xB][j];
}
db calc() {
  int xA, xB;
  db Max, tmp;
  while (1) {
    xA = n + 1, idA[xA] = n + m + 1;
    for (int i = 1; i <= n; i++)
      if (a[0][i] > EPS && idA[i] < idA[xA]) xA = i;

    if (xA == n + 1) return a[0][0];
    xB = m + 1, idB[xB] = n + m + 1, Max = -INF;
    for (int i = 1; i <= m; i++)
      if (a[i][xA] < -EPS && ((tmp = a[i][0] / a[i][xA]) > Max + EPS ||
                              (tmp > Max - EPS && idB[i] < idB[xB])))
        Max = tmp, xB = i;

    if (xB == m + 1) put_out(1);

    pivot(xA, xB);
  }
  return a[0][0];
}
db solve() {
  for (int i = 1; i <= n; i++) idA[i] = i;
  for (int i = 1; i <= m; i++) idB[i] = n + i;
  static db tmp[MAXN];
  db Min = 0.0;
  int l;
  for (int i = 1; i <= m; i++)
    if (a[i][0] < Min) Min = a[i][0], l = i;
  if (Min > -EPS) return calc();

  idA[++n] = 0;
  for (int i = 1; i <= m; i++) a[i][n] = 1.0;
  for (int i = 0; i <= n; i++) tmp[i] = a[0][i], a[0][i] = 0.0;
  a[0][n] = -1.0;

  pivot(n, l);

  if (calc() < -EPS) put_out(0);
  for (int i = 1; i <= m; i++)
    if (!idB[i]) {
      for (int j = 1; j <= n; j++)
        if (fabs(a[0][j]) > EPS) {
```

```
191           pivot(j, i);
192           break;
193         }
194       break;
195     }
196
197   int xA;
198   for (xA = 1; xA <= n; xA++)
199     if (!idA[xA]) break;
200   for (int i = 0; i <= m; i++) a[i][xA] = a[i][n];
201   idA[xA] = idA[n], n--;
202
203   for (int i = 0; i <= n; i++) a[0][i] = 0.0;
204   for (int i = 1; i <= m; i++)
205     if (idB[i] <= n) {
206       for (int j = 0; j <= n; j++) a[0][j] += a[i][j] * tmp[idB[i]];
207     }
208
209   for (int i = 1; i <= n; i++)
210     if (idA[i] <= n) a[0][i] += tmp[idA[i]];
211   return calc();
212 }
213 db ans[MAXN];
214 void findAns() {
215   for (int i = 1; i <= n; i++) ans[i] = 0.0;
216   for (int i = 1; i <= m; i++)
217     if (idB[i] <= n) ans[idB[i]] = a[i][0];
218 }
219 void work() {
220   for (int i = 1; i <= m; ++i)
221     for (int j = 1; j <= n; ++j) a[i][j] *= -1;
222   printf("%.10f\n", -double(solve()));
223 }
224 } // namespace LP
225
226 void solve() {
227   static int n, m, g[10];
228   static vector<int> con[10], able;
229
230   scanf("%d %d", &n, &m);
231   for (int i = 0; i < n; ++i) {
232     scanf("%d", g + i);
233     con[i].clear();
234   }
235
236   if (n == 1) {
237     printf("%.10f\n", m >= g[0] ? 1.0 : 0.0);
238     return;
239   }
240
241   able.clear();
242   for (int s = 0; s < (1 << n); ++s) {
243     int sum = 0;
244     for (int i = 0; i < n; ++i)
245       if (s >> i & 1) sum += g[i];
246     if (sum > m) continue;
247
248     able.push_back(s);
249     for (int i = 0; i < n; ++i)
```

```
250       if (s >> i & 1) con[i].push_back(able.size());
251     }
252
253     LP::n = able.size();
254     LP::m = 0;
255
256     for (int step = 0; step < n; ++step) {
257       int &f = ++LP::m;
258       for (int i = 0; i <= LP::n; ++i) LP::a[f][i] = 0;
259       for (int x : con[step]) ++LP::a[f][x];
260       if (step + 1 < n) {
261         for (int x : con[step + 1]) --LP::a[f][x];
262       } else {
263         for (int x : con[0]) --LP::a[f][x];
264       }
265     }
266
267     ++LP::m;
268     LP::a[LP::m][0] = 1;
269     for (int i = 1; i <= LP::n; ++i) LP::a[LP::m][i] = 1;
270
271     ++LP::m;
272     LP::a[LP::m][0] = -1;
273     for (int i = 1; i <= LP::n; ++i) LP::a[LP::m][i] = -1;
274
275     for (int i = 0; i <= LP::n; ++i) LP::a[0][i] = 0;
276     for (int x : con[0]) ++LP::a[0][x];
277
278     static db a2[MAXM][MAXN];
279     for (int i = 1; i <= LP::m; ++i)
280       for (int j = 1; j <= LP::n; ++j) a2[i][j] = LP::a[i][j];
281     for (int i = 1; i <= LP::m; ++i)
282       for (int j = 1; j <= LP::n; ++j) LP::a[j][i] = a2[i][j];
283     swap(LP::n, LP::m);
284     for (int i = 1; i <= max(LP::n, LP::m); ++i) swap(LP::a[0][i], LP::a[i][0]);
285     LP::a[0][0] = 0;
286     for (int i = 1; i <= LP::m; ++i)
287       for (int j = 1; j <= LP::n; ++j) LP::a[i][j] *= -1;
288     for (int i = 1; i <= LP::m; ++i) LP::a[i][0] *= -1;
289     for (int i = 1; i <= LP::n; ++i) LP::a[0][i] *= -1;
290
291     LP::work();
292   }
293
294   int main() {
295     int o_o;
296     scanf("%d", &o_o);
297     for (int i = 1; i <= o_o; ++i) {
298       printf("Case #%d: ", i);
299       solve();
300     }
301     return 0;
302   }
```

1.18 线性规划例题 (UOJ 板题代码)

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

```

#ifdef __WIN32
#define LLFORMAT "I64"
#else
#define LLFORMAT "ll"
#endif

#define eps 1e-7

int simplex(vector<vector<double>> &a, vector<double> &b, vector<double> &c,
            vector<int> &basic) {
  int m = b.size(), n = c.size();
  while (true) {
    int k = -1;
    for (int j = 0; j < n; ++j)
      if (c[j] < -eps) {
        k = j;
        break;
      }
    if (k == -1) {
      double ans = 0;
      for (int i = 0; i < m; ++i) ans += c[basic[i]] * b[i];
      return 0;
    }
    int l = -1;
    for (int i = 0; i < m; ++i)
      if (a[i][k] > eps) {
        if (l == -1)
          l = i;
        else {
          double ti = b[i] / a[i][k], tl = b[l] / a[l][k];
          if (ti < tl - eps || (ti < tl + eps && basic[i] < basic[l])) l = i;
        }
      }
    if (l == -1) return -1;
    basic[l] = k;
    double tmp = 1 / a[l][k];
    for (int j = 0; j < n; ++j) a[l][j] *= tmp;
    b[l] *= tmp;
    for (int i = 0; i < m; ++i)
      if (i != l) {
        tmp = a[i][k];
        for (int j = 0; j < n; ++j) a[i][j] -= tmp * a[l][j];
        b[i] -= tmp * b[l];
      }
    tmp = c[k];
    for (int j = 0; j < n; ++j) c[j] -= tmp * a[l][j];
  }
}

int main() {
  ios::sync_with_stdio(false);
  int n, m, T;
  cin >> n >> m >> T;
  vector<double> c(n + m, 0);
  for (int i = 0; i < n; ++i) {
    cin >> c[i];
    c[i] *= -1;
  }
```

```
62    auto C = c;
63    vector<vector<double>> a(m, vector<double>(n + m, 0));
64    vector<double> b(m);
65    vector<int> basic(m, -1), tmp;
66    for (int i = 0; i < m; ++i) {
67      for (int j = 0; j < n; ++j) cin >> a[i][j];
68      a[i][i + n] = 1;
69      cin >> b[i];
70      if (b[i] > -eps)
71        basic[i] = i + n;
72      else
73        tmp.push_back(i);
74    }
75    if (!tmp.empty()) {
76      sort(tmp.begin(), tmp.end(), [&](int i, int j) { return b[i] > b[j]; });
77      vector<vector<double>> A;
78      vector<double> B, C(n + m + 1, 0);
79      vector<int> Basic;
80      for (int i : tmp) {
81        vector<double> foo;
82        for (int j = 0; j < n + m; ++j) foo.push_back(-a[i][j]);
83        foo.push_back(1);
84        double bar = -b[i];
85        for (int i = 0; i < A.size(); ++i) {
86          double tmp = foo[Basic[i]];
87          for (int j = 0; j <= n + m; ++j) foo[j] -= tmp * A[i][j];
88          bar -= tmp * B[i];
89        }
90        for (int j = n + m; j >= 0; --j)
91          if (-eps < foo[j] - 1 && foo[j] - 1 < eps) {
92            Basic.push_back(j);
93            break;
94          }
95        for (int i = 0; i < A.size(); ++i) {
96          double tmp = A[i][Basic.back()];
97          for (int j = 0; j <= n + m; ++j) A[i][j] -= tmp * foo[j];
98          B[i] -= tmp * bar;
99        }
100       A.push_back(foo);
101       B.push_back(bar);
102     }
103     for (int i = 0; i < A.size(); ++i)
104       if (Basic[i] == n + m) {
105         for (int j = 0; j < n + m; ++j) C[j] = -A[i][j];
106       }
107     for (int i = 0; i < m; ++i)
108       if (b[i] > -eps) {
109         A.push_back(a[i]);
110         A[A.size() - 1].push_back(0);
111         B.push_back(b[i]);
112         Basic.push_back(basic[i]);
113       }
114     simplex(A, B, C, Basic);
115     bool flag = true;
116     for (int i = 0; i < m; ++i)
117       if (Basic[i] == n + m) {
118         if (B[i] > eps) {
119           cout << "Infeasible\n";
120           return 0;
```

```
121         }
122         int k = -1;
123         for (int j = 0; j < n + m; ++j)
124           if (A[i][j] > eps || A[i][j] < -eps) {
125             k = j;
126             break;
127           }
128         if (k != -1) {
129           double tmp = 1 / A[i][k];
130           Basic[i] = k;
131           for (int j = 0; j <= n + m; ++j) A[i][j] *= tmp;
132           B[i] *= tmp;
133           for (int l = 0; l < m; ++l)
134             if (l != i) {
135               tmp = A[l][k];
136               for (int j = 0; j <= n + m; ++j) A[l][j] -= tmp * A[i][j];
137               B[l] -= tmp * B[i];
138             }
139         } else
140           flag = false;
141         break;
142       }
143     if (flag) {
144       A.push_back(vector<double>(n + m, 0));
145       A[A.size() - 1].push_back(1);
146       B.push_back(0);
147       Basic.push_back(n + m);
148       for (int i = 0; i < A.size() - 1; ++i) {
149         double tmp = A[i].back();
150         for (int j = 0; j <= n + m; ++j) A[i][j] -= tmp * A[A.size() - 1][j];
151         B[i] -= tmp * B.back();
152       }
153     }
154     a = A;
155     b = B;
156     basic = Basic;
157     c.push_back(0);
158     for (int i = 0; i < a.size(); ++i) {
159       double tmp = c[basic[i]];
160       for (int j = 0; j <= n + m; ++j) c[j] -= tmp * a[i][j];
161     }
162   }
163   auto foo = simplex(a, b, c, basic);
164   if (foo == -1)
165     cout << "Unbounded" << endl;
166   else {
167     double res = 0;
168     vector<double> ans(n, 0);
169     for (int i = 0; i < basic.size(); ++i)
170       if (basic[i] < n) ans[basic[i]] = b[i];
171     for (int j = 0; j < n; ++j) res -= C[j] * ans[j];
172     cout << setprecision(8) << res << endl;
173     if (T == 1) {
174       for (int i = 0; i < n; ++i) cout << setprecision(8) << ans[i] << ' ';
175       cout << endl;
176     }
177   }
178   return 0;
179 }
```

1.19 自适应 Simpson

```
// 计算 \int_a^b f(x) dx
double simpson(double a, double b) {
  double c = a + (b - a) / 2;
  return (f(a) + 4 * f(c) + f(b)) * (b - a) / 6;
}
double integral(double a, double b, double eps, double A) {
  double c = a + (b - a) / 2;
  double L = simpson(a, c), R = simpson(c, b);
  if (fabs(L + R - A) <= 15 * eps) return L + R + (L + R - A) / 15;
  return integral(a, c, eps / 2, L) + integral(c, b, eps / 2, R);
}
double integral(double a, double b, double eps) {
  return integral(a, b, eps, simpson(a, b));
}
```

2 Dynamic Programming

2.1 斜率优化

```
// 树上斜率优化
// 定义dp_i 表示i节点传递到根节点的最短耗时,规定dp_root=-P。
// 有如下转移方程dp_u=dp_v+dist(u,v)^2+P,v 为 u 的祖先.
#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() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr), cout.tie(nullptr);

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

  return 0;
}
```

3 Data Structure

3.1 KD-tree

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

const int MAXN = 2e5 + 5;
typedef long long ll;

namespace KD_Tree {

const int DIM = 2;

inline ll sqr(int x) { return 1LL * x * x; }

struct Point {
  int x[DIM], id, c;

  ll dist2(const Point &b) const {
    return sqr(x[0] - b.x[0]) + sqr(x[1] - b.x[1]);
  }
};
```

```
21  struct QNode {
22    Point p;
23    ll dis2;
24
25    QNode() {}
26    QNode(Point _p, ll _dis2) : p(_p), dis2(_dis2) {}
27
28    bool operator<(const QNode &b) const {
29      return dis2 < b.dis2 || (dis2 == b.dis2 && p.id < b.p.id);
30    }
31  } ans;
32  struct cmpx {
33    int div;
34    cmpx(int _div) : div(_div) {}
35    bool operator()(const Point &a, const Point &b) {
36      for (int i = 0; i < DIM; ++i)
37        if (a.x[(i + div) % DIM] != b.x[(i + div) % DIM])
38          return a.x[(i + div) % DIM] < b.x[(i + div) % DIM];
39      return true;
40    }
41  };
42
43  bool cmp(const Point &a, const Point &b, int div) {
44    cmpx cp = cmpx(div);
45    return cp(a, b);
46  }
47
48  struct Node {
49    Point e;
50    Node *lc, *rc;
51    int div;
52  } node_pool[MAXN], *tail, *root;
53  void init() { tail = node_pool; }
54  Node *build(Point *a, int l, int r, int div) {
55    if (l >= r) return nullptr;
56    Node *p = tail++;
57    p->div = div;
58    int mid = (l + r) >> 1;
59    nth_element(a + l, a + mid, a + r, cmpx(div));
60    p->e = a[mid];
61    p->lc = build(a, l, mid, div ^ 1);
62    p->rc = build(a, mid + 1, r, div ^ 1);
63    return p;
64  }
65  void search(Point p, Node *x, int div) {
66    if (!x) return;
67    if (cmp(p, x->e, div)) {
68      search(p, x->lc, div ^ 1);
69      if (ans.dis2 == -1) {
70        if (x->e.c <= p.c) ans = QNode(x->e, p.dist2(x->e));
71        search(p, x->rc, div ^ 1);
72      } else {
73        QNode temp(x->e, p.dist2(x->e));
74        if (x->e.c <= p.c && temp < ans) ans = temp;
75        if (sqr(x->e.x[div] - p.x[div]) <= ans.dis2) search(p, x->rc, div ^ 1);
76      }
77    } else {
78      search(p, x->rc, div ^ 1);
79      if (ans.dis2 == -1) {
```

```
      if (x->e.c <= p.c) ans = QNode(x->e, p.dist2(x->e));
      search(p, x->lc, div ^ 1);
    } else {
      QNode temp(x->e, p.dist2(x->e));
      if (x->e.c <= p.c && temp < ans) ans = temp;
      if (sqr(x->e.x[div] - p.x[div]) <= ans.dis2) search(p, x->lc, div ^ 1);
    }
  }
}
void search(Point p) {
  ans.dis2 = -1;
  search(p, root, 0);
}
} // namespace KD_Tree

void solve() {
  static KD_Tree::Point p[MAXN];
  int n, m;
  cin >> n >> m;
  for (int i = 0; i < n; ++i) {
    p[i].id = i;
    cin >> p[i].x[0] >> p[i].x[1] >> p[i].c;
  }
  KD_Tree::init();
  KD_Tree::root = KD_Tree::build(p, 0, n, 0);

  for (KD_Tree::Point q; m; --m) {
    cin >> q.x[0] >> q.x[1] >> q.c;
    KD_Tree::search(q);
    cout << KD_Tree::ans.p.x[0] << ' ' << KD_Tree::ans.p.x[1] << ' '
         << KD_Tree::ans.p.c << '\n';
  }
}
int main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);

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

  return 0;
}

// 寻找远点
inline void cmin(int &a, int b) { b < a ? a = b : 1; }
inline void cmax(int &a, int b) { a < b ? a = b : 1; }
inline int ibs(int a) { return a < 0 ? -a : a; }
struct D {
  int d[2], mx0, mx1, mi0, mi1;
  D *l, *r;
} t[N], *rt;
int cpd, ans;
inline bool cmp(const D &a, const D &b) {
  return (a.d[cpd] ^ b.d[cpd]) ? a.d[cpd] < b.d[cpd]
                               : a.d[cpd ^ 1] < b.d[cpd ^ 1];
}
inline void kd_upd(D *u) {
  if (u->l) {
    cmax(u->mx0, u->l->mx0);
```

```
    cmax(u->mx1, u->l->mx1);
    cmin(u->mi0, u->l->mi0);
    cmin(u->mi1, u->l->mi1);
  }
  if (u->r) {
    cmax(u->mx0, u->r->mx0);
    cmax(u->mx1, u->r->mx1);
    cmin(u->mi0, u->r->mi0);
    cmin(u->mi1, u->r->mi1);
  }
}
D *kd_bld(int l, int r, int d) {
  int m = l + r >> 1;
  cpd = d;
  std::nth_element(t + l + 1, t + m + 1, t + r + 1, cmp);
  t[m].mx0 = t[m].mi0 = t[m].d[0];
  t[m].mx1 = t[m].mi1 = t[m].d[1];
  if (l ^ m) t[m].l = kd_bld(l, m - 1, d ^ 1);
  if (r ^ m) t[m].r = kd_bld(m + 1, r, d ^ 1);
  kd_upd(t + m);
  return t + m;
}
inline void kd_ins(D *ne) {
  int cd = 0;
  D *u = rt;
  while (true) {
    cmax(u->mx0, ne->mx0), cmin(u->mi0, ne->mi0);
    cmax(u->mx1, ne->mx1), cmin(u->mi1, ne->mi1);
    if (ne->d[cd] < u->d[cd]) {
      if (u->l)
        u = u->l;
      else {
        u->l = ne;
        return;
      }
    } else {
      if (u->r)
        u = u->r;
      else {
        u->r = ne;
        return;
      }
    }
    cd ^= 1;
  }
}
inline int dist(int x, int y, D *u) {
  int r = 0;
  if (x < u->mi0)
    r = u->mi0 - x;
  else if (x > u->mx0)
    r = x - u->mx0;
  if (y < u->mi1)
    r += u->mi1 - y;
  else if (y > u->mx1)
    r += y - u->mx1;
  return r;
}
inline void kd_quy(D *u, const int &x, const int &y) {
```

```
  int dl, dr, d0;
  d0 = ibs(u->d[0] - x) + ibs(u->d[1] - y);
  if (d0 < ans) ans = d0;
  dl = u->l ? dist(x, y, u->l) : inf;
  dr = u->r ? dist(x, y, u->r) : inf;
  if (dl < dr) {
    if (dl < ans) kd_quy(u->l, x, y);
    if (dr < ans) kd_quy(u->r, x, y);
  } else {
    if (dr < ans) kd_quy(u->r, x, y);
    if (dl < ans) kd_quy(u->l, x, y);
  }
}
```

3.2 LCT

```
struct LCT {
  struct node {
    int val, add;
    node *fa, *ch[2];
    void modify(const int &x) {
      val += x;
      add += x;
    }
  } node_mset[MaxS], *cnode, *null;
  LCT() {
    cnode = node_mset;
    null = cnode++;
    *null = (node){0, 0, null, {null, null}};
  }
  inline node *newnode() {
    *cnode = (node){0, 0, null, {null, null}};
    return cnode++;
  }
  inline bool isrt(node *u) const {
    return (u->fa->ch[0] != u) && (u->fa->ch[1] != u);
  }
  inline bool which(node *u) const { return u->fa->ch[1] == u; }
  void push_down(node *u) {
    if (!isrt(u)) push_down(u->fa);
    if (u->add) {
      u->ch[0]->modify(u->add);
      u->ch[1]->modify(u->add);
      u->add = 0;
    }
  }
  inline void rotate(node *u) {
    node *f = u->fa;
    int d = which(u);
    f->ch[d] = u->ch[d ^ 1];
    f->ch[d]->fa = f;
    u->ch[d ^ 1] = f;
    u->fa = f->fa;
    if (!isrt(f)) f->fa->ch[which(f)] = u;
    f->fa = u;
  }
  inline void splay(node *u) {
    push_down(u);
```

```
    for (node *f; !isrt(u); rotate(u))
      if (!isrt(f = u->fa)) rotate(which(u) == which(f) ? f : u);
  }
  inline void access(node *x) {
    for (node *y = null; x != null; x = x->fa) {
      splay(x);
      x->ch[1] = y;
      y = x;
    }
  }
  inline void cut(node *u) {
    access(u);
    splay(u);
    u->ch[0]->fa = null;
    u->ch[0] = null;
  }
  inline void link(node *u, node *v) {
    cut(u);
    u->fa = v;
  }
} tree;
```

3.3 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;
```

```
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     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);
```

```
  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;
}
```

3.4 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;
}
```

4 String

4.1 AC 自动机

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

int add_char(int u, int id) {
  if (tr[u][id] < 0) tr[u][id] = node_c++;
  return tr[u][id];
}
void build_acam() {
  queue<int> que;
```

```
  fail[0] = 0;
  for (int i = 0; i < 26; ++i)
    if (~tr[0][i]) {
      que.push(tr[0][i]);
      fail[tr[0][i]] = 0;
    } else {
      tr[0][i] = 0;
    }
  while (!que.empty()) {
    int u = que.front(), f = fail[u];
    que.pop();
    for (int i = 0; i < 26; ++i)
      if (~tr[u][i]) {
        que.push(tr[u][i]);
        fail[tr[u][i]][i] = tr[f][i];
      } else {
        tr[u][i] = tr[f][i];
      }
  }
  for (int i = 1; i < node_c; ++i) adj[fail[i]].push_back(i);
}
```

4.2 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;
}
```

4.3 SAM

```
// SPOJ Lexicographical Substring Search 求字典序第 k 大子串
#pragma GCC optimize(2)
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 90000 + 3;
const int ALPHABET = 26;

struct Node {
  int len, cnt;
  Node *link, *next[ALPHABET];
```

```
  void init(int len = 0) {
    link = nullptr;
    this->len = len, cnt = 0;
    memset(next, 0, sizeof(next));
  }
};

template <int MAX_LENGTH> class SAM {
public:
  Node *last, *root;

  void init() {
    pool_ptr = pool;
    last = root = new_node(0);
  }

  void extend(int chr) {
    Node *p = last, *np = new_node(p->len + 1);
    for (last = np; p && !p->next[chr]; p = p->link) p->next[chr] = np;
    if (!p) {
      np->link = root;
    } else {
      Node *q = p->next[chr];
      if (q->len == p->len + 1) {
        np->link = q;
      } else {
        Node *nq = new_node(p->len + 1);
        memcpy(nq->next, q->next, sizeof(q->next));
        nq->link = q->link, q->link = np->link = nq;
        for (; p && p->next[chr] == q; p = p->link) p->next[chr] = nq;
      }
    }
  }

  void toposort() {
    int size = pool_ptr - pool;
    memset(cnt, 0, size * sizeof(int));
    for (Node *it = pool; it < pool_ptr; ++it) ++cnt[it->len];
    for (int i = 1; i < size; ++i) cnt[i] += cnt[i - 1];
    for (Node *it = pool; it < pool_ptr; ++it) order[--cnt[it->len]] = it;
    for (int i = size - 1; ~i; --i) {
      Node *u = order[i];
      for (int j = 0; j < ALPHABET; ++j)
        u->cnt += u->next[j] ? u->next[j]->cnt + 1 : 0;
    }
  }

  void find_kth(int k, char *str) {
    char *ptr = str;
    Node *u = root;
    while (k) {
      for (int j = 0; j < ALPHABET; ++j) {
        if (!u->next[j]) continue;
        if (u->next[j]->cnt + 1 < k) {
          k -= u->next[j]->cnt + 1;
          continue;
        }
        --k, *ptr++ = j + 'a';
        u = u->next[j];
```

```
71          break;
72        }
73      }
74      *ptr = 0;
75    }
76
77  private:
78    int cnt[MAX_LENGTH * 2];
79    Node pool[MAX_LENGTH * 2], *pool_ptr, *order[MAX_LENGTH * 2];
80
81    Node *new_node(int len) {
82      pool_ptr->init(len);
83      return pool_ptr++;
84    }
85  };
86
87  SAM<MAXN> sam;
88  char str[MAXN];
89
90  int main(int argc, char *argv[]) {
91    ios::sync_with_stdio(false);
92    cin.tie(nullptr), cout.tie(nullptr);
93
94    cin >> str;
95    sam.init();
96    for (char *it = str; *it; ++it) sam.extend(*it - 'a');
97    sam.toposort();
98
99    int q, k;
100   for (cin >> q; q; --q) cin >> k, sam.find_kth(k, str), puts(str);
101
102   return 0;
103 }
```

4.4 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]]) { ++p[i]; }
15     if (i + p[i] > j + p[j]) { j = i; }
16   }
17 }
```

4.5 后缀数组（SAIS）

```cpp
// UOJ 模板题, 最快算法
// 字符串必须为正数, BUFFER_SIZE 要随 MAX_LENGTH 同步变化, 1e6为25
#include <bits/stdc++.h>

const int BUFFER_SIZE = 1u << 23 | 1;
char buffer[BUFFER_SIZE], *buffer_ptr = buffer;
#define alloc(x, type, len)                                                    \
  type *x = (type *)buffer_ptr;                                                \
  buffer_ptr += (len) * sizeof(type);
#define clear_buffer()                                                         \
  memset(buffer, 0, buffer_ptr - buffer), buffer_ptr = buffer;

template <int MAX_LENGTH> class SuffixArray {
#define L_TYPE true
#define S_TYPE false
public:
  int sa[MAX_LENGTH], rank[MAX_LENGTH], height[MAX_LENGTH];
  void compute(int n, int m, int *s) {
    sais(n, m, s, sa);
    for (int i = 0; i < n; ++i) rank[sa[i]] = i;
    for (int i = 0, h = 0; i < n; ++i) {
      if (rank[i]) {
        int j = sa[rank[i] - 1];
        while (s[i + h] == s[j + h]) ++h;
        height[rank[i]] = h;
      } else {
        h = 0;
      }
      if (h) --h;
    }
  }

private:
  int l_bucket[MAX_LENGTH], s_bucket[MAX_LENGTH];

  void induce(int n, int m, int *s, bool *type, int *sa, int *bucket,
              int *l_bucket, int *s_bucket) {
    memcpy(l_bucket + 1, bucket, m * sizeof(int));
    memcpy(s_bucket + 1, bucket + 1, m * sizeof(int));
    sa[l_bucket[s[n - 1]]++] = n - 1;
    for (int i = 0; i < n; ++i) {
      int t = sa[i] - 1;
      if (t >= 0 && type[t] == L_TYPE) sa[l_bucket[s[t]]++] = t;
    }
    for (int i = n - 1; i >= 0; --i) {
      int t = sa[i] - 1;
      if (t >= 0 && type[t] == S_TYPE) sa[--s_bucket[s[t]]] = t;
    }
  }
  void sais(int n, int m, int *s, int *sa) {
    alloc(type, bool, n + 1);
    alloc(bucket, int, m + 1);
    type[n] = false;
    for (int i = n - 1; i >= 0; --i) {
      ++bucket[s[i]];
      type[i] = s[i] > s[i + 1] || (s[i] == s[i + 1] && type[i + 1] == L_TYPE);
    }
    for (int i = 1; i <= m; ++i) {
      bucket[i] += bucket[i - 1];
```

```
      s_bucket[i] = bucket[i];
    }
    memset(rank, -1, n * sizeof(int));

    alloc(lms, int, n + 1);
    int n1 = 0;
    for (int i = 0; i < n; ++i) {
      if (!type[i] && (i == 0 || type[i - 1])) lms[rank[i] = n1++] = i;
    }
    lms[n1] = n;
    memset(sa, -1, n * sizeof(int));
    for (int i = 0; i < n1; ++i) sa[--s_bucket[s[lms[i]]]] = lms[i];
    induce(n, m, s, type, sa, bucket, l_bucket, s_bucket);
    int m1 = 0;
    alloc(s1, int, n + 1);
    for (int i = 0, t = -1; i < n; ++i) {
      int r = rank[sa[i]];
      if (r != -1) {
        int len = lms[r + 1] - sa[i] + 1;
        m1 += t == -1 || len != lms[rank[t] + 1] - t + 1 ||
              memcmp(s + t, s + sa[i], len * sizeof(int)) != 0;
        s1[r] = m1;
        t = sa[i];
      }
    }
    alloc(sa1, int, n + 1);
    if (n1 == m1) {
      for (int i = 0; i < n1; ++i) sa1[s1[i] - 1] = i;
    } else {
      sais(n1, m1, s1, sa1);
    }
    memset(sa, -1, n * sizeof(int));
    memcpy(s_bucket + 1, bucket + 1, m * sizeof(int));
    for (int i = n1 - 1; i >= 0; --i) {
      int t = lms[sa1[i]];
      sa[--s_bucket[s[t]]] = t;
    }
    induce(n, m, s, type, sa, bucket, l_bucket, s_bucket);
  }
#undef S_TYPE
#undef L_TYPE
};

const int MAXN = 1e5 + 5;
SuffixArray<MAXN> sa;
char str[MAXN];
int s[MAXN];

int main() {
  int n = fread(str, 1, MAXN, stdin);
  while (str[n - 1] - 97u > 25) --n;
  for (int i = 0; i < n; ++i) s[i] = str[i] - 'a' + 1;
  sa.compute(n, 26, s);
  for (int i = 0; i < n; ++i) printf("%d%c", sa.sa[i] + 1, " \n"[i == n - 1]);
  for (int i = 1; i < n; ++i) printf("%d%c", sa.height[i], " \n"[i == n - 1]);
  return 0;
}
```

4.6 后缀数组（倍增）

```cpp
// UOJ 模板题
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e5 + 3;

template <int MAX_LENGTH> class SuffixArray {
public:
  int n, sa[MAX_LENGTH], rank[MAX_LENGTH], height[MAX_LENGTH];

  void compute(char *s, int n, int m) {
    int i, p, w, j, k;
    this->n = n;
    if (n == 1) {
      sa[0] = rank[0] = height[0] = 0;
      return;
    }
    memset(cnt, 0, m * sizeof(int));
    for (i = 0; i < n; ++i) ++cnt[rank[i] = s[i]];
    for (i = 1; i < m; ++i) cnt[i] += cnt[i - 1];
    for (i = n - 1; ~i; --i) sa[--cnt[rank[i]]] = i;
    for (w = 1; w < n; w <<= 1, m = p) {
      for (p = 0, i = n - 1; i >= n - w; --i) id[p++] = i;
      for (i = 0; i < n; ++i)
        if (sa[i] >= w) id[p++] = sa[i] - w;
      memset(cnt, 0, m * sizeof(int));
      for (i = 0; i < n; ++i) ++cnt[px[i] = rank[id[i]]];
      for (i = 1; i < m; ++i) cnt[i] += cnt[i - 1];
      for (i = n - 1; ~i; --i) sa[--cnt[px[i]]] = id[i];
      memcpy(old_rank, rank, n * sizeof(int));
      for (i = p = 1, rank[sa[0]] = 0; i < n; ++i)
        rank[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p - 1 : p++;
    }
    for (i = 0; i < n; ++i) rank[sa[i]] = i;
    for (i = k = height[rank[0]] = 0; i < n; height[rank[i++]] = k)
      if (rank[i])
        for (k > 0 ? --k : 0, j = sa[rank[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) table[0][i] = height[i];
    for (int i = 1; i <= lgn; ++i)
      for (int j = 0, l = 1 << (i - 1); j + l < n; ++j)
        table[i][j] = min(table[i - 1][j], table[i - 1][j + l]);
  }

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

private:
  int table[17][MAX_LENGTH], lg[MAX_LENGTH];
  int old_rank[MAX_LENGTH], id[MAX_LENGTH], px[MAX_LENGTH], cnt[MAX_LENGTH];
```

```
58
59    bool cmp(int x, int y, int w) {
60      return old_rank[x] == old_rank[y] && old_rank[x + w] == old_rank[y + w];
61    }
62  };
63
64  char s[MAXN];
65  SuffixArray<MAXN> sa;
66
67  int main(int argc, char *argv[]) {
68    int n = fread(s, 1, MAXN, stdin);
69    while (s[n - 1] - 97u > 25) --n;
70    for (int i = 0; i < n; ++i) s[i] -= 'a';
71    s[n] = '$';
72    sa.compute(s, n, 26);
73    for (int i = 0; i < n; ++i) printf("%d%c", sa.sa[i] + 1, " \n"[i == n - 1]);
74    for (int i = 1; i < n; ++i) printf("%d%c", sa.height[i], " \n"[i == n - 1]);
75    return 0;
76  }
```

4.7 哈希

```
1  /*
2   * 其他哈希:
3   * 集合哈希: 可以使用元素的哈希值映射为高进制的某一位, 也可以使用质数的积。
4   * 树哈希: 将子树当作集合哈希, 加入深度的影响
5   */
6
7  const unsigned int KEY = 6151;
8  const unsigned int MOD = 1610612741;
9  // 64 位哈希参数 KEY 随意 MOD 461168601842738784711
10 unsigned int hash[MAXN], p[MAXN];
11
12 inline unsigned int get_hash(int l, int r) {
13   return (hash[r] + MOD - 1ULL * hash[l - 1] * p[r - l + 1] % MOD) % MOD;
14 }
15
16 void init(char *s, int n) {
17   p[0] = 1;
18   for (int i = 1; i <= n; ++i) {
19     p[i] = p[i - 1] * KEY % MOD;
20     hash[i] = (1LL * hash[i - 1] * KEY + s[i]) % MOD;
21   }
22 }
```

4.8 哈希例题 (2020 秦皇岛 J)

```
1  // 两次哈希
2  #include <bits/stdc++.h>
3  using namespace std;
4
5  typedef long long ll;
6
7  const int MAXN = 3e5 + 3;
8  const int MAX_PRIME = 8960453 + 3;
9  const int MOD = 998244353;
```

```
10  const ll BASE = 709;
11  const ll HASH_MOD = 461168601842738784711;
12
13  char s[MAXN];
14  int fac[MAXN], inv[MAXN], fac_inv[MAXN], prime[MAXN * 2];
15  ll ha[MAXN], p[MAXN], pref[MAXN], suff[MAXN], value[MAXN * 2];
16  pair<ll, int> bin[MAXN];
17
18  int cnt[MAXN], pidx[MAXN], sidx[MAXN];
19  bitset<MAX_PRIME> mark;
20
21  ll fmul(ll a, ll b) {
22    ll k = (ll)((1.l * a * b) / (1.l * HASH_MOD)), t = a * b - k * HASH_MOD;
23    for (t -= HASH_MOD; t < 0; t += HASH_MOD) {}
24    return t;
25  }
26  ll getRange(int l, int r) {
27    return (ha[r] - fmul(ha[l - 1], p[r - l + 1]) + HASH_MOD) % HASH_MOD;
28  }
29
30  int gao(int n, int d) {
31    int tot = 0;
32    for (int l = 1, r = d; r <= n; l += d, r += d) value[tot++] = getRange(l, r);
33    int bunch = n / d, rest = n % d;
34    if (rest) {
35      for (int r = n, l = n - d + 1; l >= 1; l -= d, r -= d)
36        value[tot++] = getRange(l, r);
37      sort(value, value + tot);
38      tot = unique(value, value + tot) - value;
39      pref[0] = pref[1] = suff[n] = suff[n + 1] = 1;
40      for (int i = d; i <= n; i += d) {
41        int idx = lower_bound(value, value + tot, getRange(i - d + 1, i)) - value;
42        pidx[i] = idx;
43        pref[i] = fmul(pref[i - d], prime[idx]);
44      }
45      for (int i = n - d + 1; i >= 1; i -= d) {
46        int idx = lower_bound(value, value + tot, getRange(i, i + d - 1)) - value;
47        sidx[i] = idx;
48        suff[i] = fmul(suff[i + d], prime[idx]);
49      }
50      int sc = 0, cur = fac[bunch];
51      memset(cnt, 0, tot * sizeof(int));
52      for (int l = 1, r = rest; r <= n; l += d, r += d) {
53        if (r + d <= n) {
54          ++cnt[sidx[r + 1]];
55          cur = 1ll * cur * inv[cnt[sidx[r + 1]]] % MOD;
56        }
57        bin[sc++] = {fmul(pref[l - 1], suff[r + 1]), l};
58      }
59      sort(bin, bin + sc);
60      mark[bin[0].second] = 1;
61      for (int i = 1; i < sc; ++i)
62        mark[bin[i].second] = bin[i].first != bin[i - 1].first;
63
64      int res = 0;
65      for (int l = 1, r = rest; r <= n; l += d, r += d) {
66        if (l - 1 >= 1) {
67          ++cnt[pidx[l - 1]];
68          cur = 1ll * cur * inv[cnt[pidx[l - 1]]] % MOD;
```

```
      }
      if (mark[l]) {
        res += cur;
        if (res >= MOD) res -= MOD;
      }
      if (r + 1 <= n) {
        cur = 1ll * cur * cnt[sidx[r + 1]] % MOD;
        --cnt[sidx[r + 1]];
      }
    }
    return res;
  } else {
    sort(value, value + tot);
    ll pre = value[0];
    int res = fac[bunch], cnt = 1;
    for (int i = 1; i < tot; ++i) {
      if (value[i] != pre) {
        if (cnt > 1) res = 1ll * res * fac_inv[cnt] % MOD;
        cnt = 1, pre = value[i];
      } else {
        ++cnt;
      }
    }
    if (cnt > 1) res = 1ll * res * fac_inv[cnt] % MOD;
    return res;
  }
}

void solve() {
  cin >> (s + 1);
  int n = strlen(s + 1);
  for (int i = 1; i <= n; ++i)
    ha[i] = (fmul(ha[i - 1], BASE) + s[i]) % HASH_MOD;

  int ans = 0;
  for (int d = 1; d <= n; ++d) {
    ans += gao(n, d);
    if (ans >= MOD) ans -= MOD;
    // cerr << "# " << ans << endl;
  }
  cout << ans << "\n";
}

void prework() {
  p[0] = 1;
  for (int i = 1; i < MAXN; ++i) p[i] = fmul(p[i - 1], BASE);
  fac[0] = fac[1] = 1;
  fac_inv[0] = fac_inv[1] = inv[1] = 1;
  for (int i = 2; i < MAXN; ++i) {
    fac[i] = 1ll * fac[i - 1] * i % MOD;
    inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
    fac_inv[i] = 1ll * fac_inv[i - 1] * inv[i] % MOD;
  }

  int pc = 0;
  for (int i = 2; i < MAX_PRIME; ++i) {
    if (!mark[i]) prime[pc++] = i;
    for (int j = 0; j < pc; ++j) {
      int t = i * prime[j];
```

```
      if (t >= MAX_PRIME) break;
      mark[t] = 1;
      if (i % prime[j] == 0) break;
    }
  }
}

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

  prework();

  int T;
  cin >> T;
  for (int step = 1; step <= T; ++step) {
    cout << "Case #" << step << ": ";
    solve();
  }

  return 0;
}
```

4.9 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;
}
```

4.10 扩展 KMP

```
// 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;
    }
  }
}
```

5 Graph Theory

5.1 KM

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

typedef long long ll;

const int MAXN = 400 + 3;
const int ALPHA = 1e9 + 10;
const ll NOT = 0;
const ll INF = 3ll * MAXN * ALPHA;
// 最大费用流时 NOT = 0, 最大费用流最大流时 NOT = -1ll * MAXN * ALPHA
// 点为 1 .. n (左为 1 .. nl, 右为 1 .. nr)
struct KM {
  int n, nl, nr, lk[MAXN], pre[MAXN];
  ll lx[MAXN], ly[MAXN], w[MAXN][MAXN], slack[MAXN];
  bitset<MAXN> vy;

  void init(int n) {
    this->n = n;
    memset(lk, 0, sizeof(int) * (n + 1));
    memset(pre, 0, sizeof(int) * (n + 1));
    memset(lx, 0, sizeof(ll) * (n + 1));
    memset(ly, 0, sizeof(ll) * (n + 1));
    memset(slack, 0, sizeof(ll) * (n + 1));
    for (int i = 0; i <= n; ++i) fill(w[i], w[i] + n + 1, NOT);
  }

  void add_edge(int x, int y, ll z) {
    if (w[y][x] < z) w[y][x] = z;
  }

  ll match() {
    for (int i = 1; i <= n; ++i)
      for (int j = 1; j <= n; ++j) lx[i] = max(lx[i], w[i][j]);
    for (int i = 1, py, p, x; i <= n; ++i) {
      for (int j = 1; j <= n; ++j) slack[j] = INF, vy[j] = 0;
      for (lk[py = 0] = i; lk[py]; py = p) {
        ll delta = INF;
        vy[py] = 1, x = lk[py];
        for (int y = 1; y <= n; ++y) {
          if (vy[y]) continue;
```

```
          if (lx[x] + ly[y] - w[x][y] < slack[y])
            slack[y] = lx[x] + ly[y] - w[x][y], pre[y] = py;
          if (slack[y] < delta) delta = slack[y], p = y;
        }
        for (int y = 0; y <= n; ++y)
          if (vy[y]) {
            lx[lk[y]] -= delta, ly[y] += delta;
          } else {
            slack[y] -= delta;
          }
      }
      for (; py; py = pre[py]) lk[py] = lk[pre[py]];
    }

    ll ans = 0;
    for (int i = 1; i <= n; ++i) {
      ans += lx[i] + ly[i];
      if (w[lk[i]][i] == NOT) ans -= NOT;
    }
    return ans;
  }
} km;

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

  int nl, nr, m;
  cin >> nl >> nr >> m;
  km.init(max(nl, nr));
  for (int x, y, z; m; --m) {
    cin >> x >> y >> z;
    km.add_edge(x, y, z);
  }
  cout << km.match() << "\n";
  for (int i = 1; i <= nl; ++i)
    cout << (km.w[km.lk[i]][i] == NOT ? 0 : km.lk[i]) << " \n"[i == nl];

  return 0;
}
```

5.2 SAP

```
struct MaxFlow {
  struct Edge {
    int to, rest;
  } edges[MAXM * 4];

  vector<int> adj[MAXN];
  int n, edges_c, dep[MAXN], depc[MAXN], s, t, last[MAXN];

  void init(int _n) {
    n = _n, edges_c = 0;
    for (int i = 1; i <= n; ++i) adj[i].clear();
  }

  void add_edge(int u, int v, int cap) {
```

```
    edges[edges_c] = {v, cap, 0};
    adj[u].push_back(edges_c++);
    edges[edges_c] = {u, 0, 0};
    adj[v].push_back(edges_c++);
  }

  int dfs(int u, int flow) {
    if (u == t || !flow) return flow;
    int v, e, temp, res = 0;
    for (int &i = last[u]; i < (int)adj[u].size(); ++i) {
      e = adj[u][i], v = edges[e].to;
      if (!edges[e].res || dep[v] != dep[u] - 1) continue;
      temp = dfs(v, min(flow, edges[e].cap - edges[e].flow));
      res += temp, flow -= temp;
      edges[e].rest -= temp, edges[e ^ 1].rest += temp;
      if (!flow || !dep[s]) return res;
    }
    last[u] = 0;
    if (!(--depc[dep[u]])) dep[s] = n + 1;
    ++depc[++dep[u]];
    return res;
  }
  int max_flow(int s, int t) {
    this->s = s, this->t = t;

    static queue<int> que;
    memset(dep + 1, 0, sizeof(int) * n);
    memset(depc + 1, 0, sizeof(int) * n);
    memset(last + 1, 0, sizeof(int) * n);
    while (!que.empty()) que.pop();
    dep[t] = 1, que.push(t);

    while (!que.empty()) {
      int u = que.front();
      que.pop();
      ++depc[dep[u]];
      for (int i = 0, v; i < (int)adj[u].size(); ++i) {
        v = edges[adj[u][i]].to;
        if (dep[v]) continue;
        dep[v] = dep[u] + 1;
        que.push(v);
      }
    }

    int res = 0;
    while (dep[s] <= n) res += dfs(s, INT_MAX);
    return res;
  }
};
```

5.3 dinic

```
struct MaxFlow {
  struct Edge {
    int to, rest;
  } edges[MAXM * 4];

  vector<int> adj[MAXN];
```

```
  int n, edges_c, dep[MAXN], s, t, last[MAXN];

  void init(int _n) {
    n = _n, edges_c = 0;
    for (int i = 1; i <= n; ++i) adj[i].clear();
  }

  void add_edge(int u, int v, int cap) {
    edges[edges_c] = {v, cap, 0};
    adj[u].push_back(edges_c++);
    edges[edges_c] = {u, 0, 0};
    adj[v].push_back(edges_c++);
  }

  bool bfs() {
    memset(dep + 1, -1, sizeof(int) * n);
    static queue<int> q;
    q.push(s);
    dep[s] = 0;
    while (!q.empty()) {
      int u = q.front();
      q.pop();
      for (int i = 0; i < adj[u].size(); ++i) {
        Edge &e = edges[adj[u][i]];
        if ((~dep[e.to]) || !e.rest) continue;
        dep[e.to] = dep[u] + 1;
        q.push(e.to);
      }
    }
    return ~dep[t];
  }

  int dfs(int u, int flow) {
    if (u == t || flow == 0) return flow;
    int res = 0, temp, e, v;
    for (int &i = last[u]; i < adj[u].size(); ++i) {
      e = adj[u][i], v = edges[e].to;
      if (dep[v] == dep[u] + 1 && edges[e].rest) {
        temp = dfs(v, min(edges[e].rest, flow));
        res += temp, flow -= temp;
        edges[e].rest -= temp, edges[e ^ 1].rest += temp;
        if (!flow) break;
      }
    }
    return flow;
  }

  int max_flow(int s, int t) {
    this->s = s, this->t = t;
    int res = 0;
    while (bfs()) {
      memset(last + 1, 0, sizeof(int) * n);
      res += dfs(s, INF);
    }
    return res;
  }
};
```

5.4 tarjan

```
vector<int> adj[MAXN];
int dfn[MAXN], low[MAXN], dfs_c;
int bel[MAXN], size[MAXN], scc, stk[MAXN], top, in_stack[MAXN];

void tarjan(int u) {
  dfn[u] = low[u] = ++dfs_c;
  stk[top++] = u;
  in_stack[u] = 1;
  for (size_t i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i];
    if (!dfn[v]) {
      tarjan(v);
      (low[v] < low[u]) && (low[u] = low[v]);
    } else if (in_stack[v] && dfn[v] < low[u]) {
      low[u] = dfn[v];
    }
  }
  if (low[u] == dfn[u]) {
    int v;
    size[++scc] = 0;
    do {
      v = stk[--top];
      in_stack[v] = 0;
      bel[v] = scc;
      ++size[scc];
    } while (u != v);
  }
}
```

5.5 一般图最大匹配

```
class GeneralMatch {
public:
  int n;
  vector<vector<int>> g;
  vector<int> match, aux, label, orig, parent;
  queue<int> q;
  int aux_time;

  GeneralMatch(int n)
      : match(n, -1), aux(n, -1), label(n), orig(n), parent(n, -1),
        aux_time(-1) {
    this->n = n;
    g.resize(n);
  }

  void add_edge(int u, int v) {
    g[u].push_back(v);
    g[v].push_back(u);
  }

  int find(int x) { return x == orig[x] ? x : orig[x] = find(orig[x]); }

  int lca(int u, int v) {
    ++aux_time;
    u = find(u), v = find(v);
```

```
    for (;; swap(u, v)) {
      if (~u) {
        if (aux[u] == aux_time) return u;
        aux[u] = aux_time;
        if (match[u] == -1) {
          u = -1;
        } else {
          u = find(parent[match[u]]);
        }
      }
    }
  }

  void blossom(int u, int v, int o) {
    while (find(u) != o) {
      parent[u] = v;
      v = match[u];
      q.push(v);
      label[v] = 0;
      orig[u] = orig[v] = o;
      u = parent[v];
    }
  }

  int bfs(int x) {
    iota(orig.begin(), orig.end(), 0);
    fill(label.begin(), label.end(), -1);
    while (!q.empty()) q.pop();
    q.push(x);
    label[x] = 0;
    while (!q.empty()) {
      int u = q.front();
      q.pop();
      for (int v : g[u]) {
        if (label[v] == -1) {
          parent[v] = u;
          label[v] = 1;
          if (match[v] == -1) {
            while (v != -1) {
              int pv = parent[v];
              int next_v = match[pv];
              match[v] = pv;
              match[pv] = v;
              v = next_v;
            }
            return 1;
          }
          q.push(match[v]);
          label[match[v]] = 0;
        } else if (label[v] == 0 && find(u) != find(v)) {
          int o = lca(u, v);
          blossom(u, v, o);
          blossom(v, u, o);
        }
      }
    }
    return 0;
  }

```

```
  int find_max_match() {
    int res = 0;
    for (int i = 0; i < n; ++i) {
      if (~match[i]) continue;
      res += bfs(i);
    }
    return res;
  }
};
```

5.6 上下界费用流

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

const int MAXN = 53;
const int MAX_NODE = 113;
const int MAX_EDGE = 1e5 + 5;
const int INF = 0x3f3f3f3f;

int n, s, t, ss, tt, tote;
int R[MAXN], C[MAXN], board[MAXN][MAXN];

struct Edge {
  int to, cap, flow, cost;
} edges[MAX_EDGE];
vector<int> adj[MAX_NODE];

int from[MAX_NODE], in[MAX_NODE];
void add_edge(int from, int to, int l, int r, int cost) {
  in[to] += l, in[from] -= l;
  edges[tote] = (Edge){to, r - l, 0, cost};
  adj[from].push_back(tote++);
  edges[tote] = (Edge){from, 0, 0, -cost};
  adj[to].push_back(tote++);
}

bool spfa(int s, int t) {
  static queue<int> q;
  static bool inq[MAX_NODE];
  static int dist[MAX_NODE];
  memset(inq + 1, 0, sizeof(bool) * tt);
  memset(dist + 1, 0x3f, sizeof(int) * tt);
  memset(from + 1, -1, sizeof(int) * tt);
  dist[0] = 0, from[0] = -1;
  q.push(0);
  while (!q.empty()) {
    int u = q.front();
    q.pop();
    inq[u] = false;
    for (int e : adj[u]) {
      if (edges[e].cap == edges[e].flow) continue;
      int v = edges[e].to, d = dist[u] + edges[e].cost;
      if (d >= dist[v]) continue;
      dist[v] = d;
      from[v] = e;
      if (!inq[v]) {
        q.push(v);
```

```
47          inq[v] = true;
48        }
49      }
50    }
51    return dist[t] < INF;
52  }
53
54  pair<int, int> min_cost_max_flow(int s, int t) {
55    int flow = 0, cost = 0;
56    while (spfa(s, t)) {
57      int mi = INF;
58      for (int it = from[t]; ~it; it = from[edges[it ^ 1].to])
59        mi = min(mi, edges[it].cap - edges[it].flow);
60      flow += mi;
61      for (int it = from[t]; ~it; it = from[edges[it ^ 1].to]) {
62        edges[it].flow += mi, edges[it ^ 1].flow -= mi;
63        cost += mi * edges[it].cost;
64      }
65    }
66    return make_pair(flow, cost);
67  }
68
69  void solve() {
70    tote = 0;
71    s = 2 * n + 1, t = 2 * n + 2, ss = 0, tt = 2 * n + 3;
72    for (int i = 0; i <= tt; ++i) adj[i].clear(), in[i] = 0;
73
74    memset(R + 1, 0, sizeof(int) * n);
75    memset(C + 1, 0, sizeof(int) * n);
76
77    for (int i = 1; i <= n; ++i)
78      for (int j = 1; j <= n; ++j) {
79        cin >> board[i][j];
80        R[i] += board[i][j];
81        C[j] += board[i][j];
82      }
83
84    for (int i = 1; i <= n; ++i) {
85      add_edge(s, i, R[i], R[i], 0);
86      add_edge(s, i + n, C[i], C[i], 0);
87    }
88
89    for (int i = 1, l, r; i <= n; ++i) {
90      cin >> l >> r;
91      add_edge(i, t, l, r, 0);
92    }
93    for (int i = 1, l, r; i <= n; ++i) {
94      cin >> l >> r;
95      add_edge(i + n, t, l, r, 0);
96    }
97
98    for (int step = n * n / 2, x1, y1, x2, y2; step; --step) {
99      cin >> x1 >> y1 >> x2 >> y2;
100     if (board[x1][y1] == board[x2][y2]) continue;
101     if (board[x2][y2]) swap(x1, x2), swap(y1, y2);
102     if (x1 == x2)
103       add_edge(y1 + n, y2 + n, 0, 1, 1);
104     else
105       add_edge(x1, x2, 0, 1, 1);
```

```
  }
  add_edge(t, s, 0, INF, 0);
  int sum = 0;
  for (int i = 1; i < tt; ++i) {
    if (in[i] > 0) {
      sum += in[i];
      add_edge(ss, i, 0, in[i], 0);
    } else if (in[i] < 0) {
      add_edge(i, tt, 0, -in[i], 0);
    }
  }

  pair<int, int> ans = min_cost_max_flow(ss, tt);
  if (sum != ans.first) {
    cout << "-1\n";
  } else {
    cout << ans.second << '\n';
  }
}

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

  while (cin >> n) solve();
  return 0;
}
```

5.7 最小费用流

```
class MinCostFlow {
public:
  struct Result {
    int flow, cost;
  };
  struct Edge {
    int to, next, rest, cost;
  };

  vector<bool> inq;
  vector<int> head, dist, from, flow;
  vector<Edge> edges;

  MinCostFlow(int n, int m) : inq(n), head(n, -1), dist(n), from(n), flow(n) {
    edges.reserve(2 * m);
  }

  void add_edge(int u, int v, int capacity, int cost) {
    internal_add_edge(u, v, capacity, cost);
    internal_add_edge(v, u, 0, -cost);
  }

  void internal_add_edge(int u, int v, int capacity, int cost) {
    edges.push_back((Edge){v, head[u], capacity, cost});
    head[u] = edges.size() - 1;
  }

  Result augment(int source, int sink) {
```

```
    fill(dist.begin(), dist.end(), INT_MAX);
    dist[source] = 0;
    flow[source] = INT_MAX;
    queue<int> q;
    q.push(source);
    while (!q.empty()) {
      int u = q.front();
      q.pop();
      inq[u] = false;
      for (int it = head[u]; ~it; it = edges[it].next) {
        auto &e = edges[it];
        int v = e.to;
        if (e.rest > 0 && dist[u] + e.cost < dist[v]) {
          from[v] = it;
          dist[v] = dist[u] + e.cost;
          flow[v] = min(e.rest, flow[u]);
          if (!inq[v]) {
            q.push(v);
            inq[v] = true;
          }
        }
      }
    }

    if (dist[sink] == INT_MAX) return {0, 0};
    int min_flow = flow[sink];
    for (int u = sink; u != source; u = edges[from[u] ^ 1].to) {
      edges[from[u]].rest -= min_flow;
      edges[from[u] ^ 1].rest += min_flow;
    }
    return {min_flow, dist[sink]};
  }

  Result min_cost_flow(int source, int sink) {
    int flow = 0, cost = 0;
    for (;;) {
      auto result = augment(source, sink);
      if (!result.flow) break;
      flow += result.flow, cost += result.cost;
    }
    return {flow, cost};
  }
};
```

5.8 高标预流推进

```
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;
const int N = 1e4 + 4, M = 2e5 + 5, INF = 0x3f3f3f3f;
int n, m, s, t;

struct qxx {
  int nex, t, v;
};
qxx e[M * 2];
int h[N], cnt = 1;
```

```
13 void add_path(int f, int t, int v) { e[++cnt] = (qxx){h[f], t, v}, h[f] = cnt; }
14 void add_flow(int f, int t, int v) {
15   add_path(f, t, v);
16   add_path(t, f, 0);
17 }
18
19 int ht[N], ex[N], gap[N]; // 高度;超额流;gap 优化
20 bool bfs_init() {
21   memset(ht, 0x3f, sizeof(ht));
22   queue<int> q;
23   q.push(t), ht[t] = 0;
24   while (q.size()) { // 反向 BFS, 遇到没有访问过的结点就入队
25     int u = q.front();
26     q.pop();
27     for (int i = h[u]; i; i = e[i].nex) {
28       const int &v = e[i].t;
29       if (e[i ^ 1].v && ht[v] > ht[u] + 1) ht[v] = ht[u] + 1, q.push(v);
30     }
31   }
32   return ht[s] != INF; // 如果图不连通,返回 0
33 }
34 struct cmp {
35   bool operator()(int a, int b) const { return ht[a] < ht[b]; }
36 };                                          // 伪装排序函数
37 priority_queue<int, vector<int>, cmp> pq; // 将需要推送的结点以高度高的优先
38 bool vis[N];                                // 是否在优先队列中
39 int push(int u) { // 尽可能通过能够推送的边推送超额流
40   for (int i = h[u]; i; i = e[i].nex) {
41     const int &v = e[i].t, &w = e[i].v;
42     if (!w || ht[u] != ht[v] + 1) continue;
43     int k = min(w, ex[u]); // 取到剩余容量和超额流的最小值
44     ex[u] -= k, ex[v] += k, e[i].v -= k, e[i ^ 1].v += k; // push
45     if (v != s && v != t && !vis[v])
46       pq.push(v), vis[v] = 1; // 推送之后,v 必然溢出,则入堆,等待被推送
47     if (!ex[u]) return 0; // 如果已经推送完就返回
48   }
49   return 1;
50 }
51 void relabel(int u) { // 重贴标签(高度)
52   ht[u] = INF;
53   for (int i = h[u]; i; i = e[i].nex)
54     if (e[i].v) ht[u] = min(ht[u], ht[e[i].t]);
55   ++ht[u];
56 }
57 int hlpp() {                  // 返回最大流
58   if (!bfs_init()) return 0; // 图不连通
59   ht[s] = n;
60   memset(gap, 0, sizeof(gap));
61   for (int i = 1; i <= n; i++)
62     if (ht[i] != INF) gap[ht[i]]++; // 初始化 gap
63   for (int i = h[s]; i; i = e[i].nex) {
64     const int v = e[i].t, w = e[i].v; // 队列初始化
65     if (!w) continue;
66     ex[s] -= w, ex[v] += w, e[i].v -= w, e[i ^ 1].v += w; // 注意取消 w 的引用
67     if (v != s && v != t && !vis[v]) pq.push(v), vis[v] = 1; // 入队
68   }
69   while (pq.size()) {
70     int u = pq.top();
71     pq.pop(), vis[u] = 0;
```

```
    while (push(u)) { // 仍然溢出
      // 如果 u 结点原来所在的高度没有结点了,相当于出现断层
      if (!--gap[ht[u]])
        for (int i = 1; i <= n; i++)
          if (i != s && i != t && ht[i] > ht[u] && ht[i] < n + 1) ht[i] = n + 1;
      relabel(u);
      ++gap[ht[u]]; // 新的高度,更新 gap
    }
  }
  return ex[t];
}
int main() {
  scanf("%d%d%d%d", &n, &m, &s, &t);
  for (int i = 1, u, v, w; i <= m; i++) {
    scanf("%d%d%d", &u, &v, &w);
    add_flow(u, v, w);
  }
  printf("%d", hlpp());
  return 0;
}
```

6 Java

6.1 进制转换

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

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");
```

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

7 Others

7.1 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 {
```

```
  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; }
```

```
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
```

7.2 duipai

```
1  #/usr/bin/bash
2
3  while true; do
4    python gen.py > in.txt
5    time ./my < in.txt > out.txt
6    time ./std < in.txt > ans.txt
7    if diff out.txt ans.txt; then
8      echo AC
9    else
10     echo WA
11     exit 0
12   fi
13 done
```

7.3 emacs

```
1  (defun myc++ ()
2    (c-set-style "stroustrup")
3    (setq tab-width 2)
4    (setq indent-tabs-mode nil)
5    (setq c-basic-offset 2)
6    (c-toggle-hungry-state)
7    (defun compile-and-run()
8      (interactive)
9      (setq file-name (file-name-sans-extension (file-name-nondirectory buffer-file-name)))
10     (compile
11      (format "g++ %s.cpp -o %s -Wall -Wextra -Wshadow -O2 && ./%s < in.txt"
12              file-name file-name file-name)))
13   (local-set-key (kbd "C-c C-c") 'compile-and-run)
14   (local-set-key (kbd "C-c C-k") 'kill-compilation))
15 (add-hook 'c++-mode-hook 'myc++)
16
17 (global-set-key [(meta ?o)] 'other-window)
```

```
(global-set-key [(meta ?/)] 'hippie-expand)
(global-set-key [(control tab)] ' senator-completion-menu-popup)
(setq hippie-expand-try-functions-list
      '(try-expand-dabbrev
        try-expand-dabbrev-visible
        try-expand-dabbrev-all-buffers
        try-expand-dabbrev-from-kill
        try-complete-file-name-partially
        try-complete-file-name
        try-expand-all-abbrevs
        try-expand-list
        try-expand-line))

(setq auto-save-mode nil)
(setq make-backup-files nil)

(ido-mode t)
(show-paren-mode 1)
(delete-selection-mode t)
(global-linum-mode t)
(global-auto-revert-mode t)
```

7.4 myalloc

```
// 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) {}
};
```

7.5 vimrc

```
syntax enable
set syntax=on
set nobackup
set noswapfile
set noundofile
set nu
set smartindent
set cindent
set noeb
set tabstop=2
set softtabstop=2
set shiftwidth=2
set expandtab
```

```
14
15 :imap jk <Esc>
16
17 map <F5> : call Complie() <CR>
18 func Complie()
19   exec "w"
20   exec "!g++ % -o %< -g -Wall -std=gnu++14 -static"
21 endfunc
22
23 map <F6> : call Run() <CR>
24 func Run()
25   exec "!./%<"
26 endfunc
27
28 map <F9> : call DeBug() <CR>
29 func DeBug()
30   exec "!gdb %<"
31 endfunc
```