*CPC Template Manaual

Harbin Institute of Technology

cycleke

July 28, 2020

Contents

1 Math

1.1 BGSG

```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 (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;
}
```

1.2   lucas

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

1.3   LinearProgramming

```
1   // CCPC Final 2017 F
2   // sum(P(s)) = 1, P(s) >= 0
3   // max and equal (sum(P(s)) | i in s)
4   #include <bits/stdc++.h>
5   using namespace std;
6
7   const int MAXN = int(3e3);
8   const int MAXM = int(3e3);
9   const double INF = 1e20, EPS = 1e-9;
10
11  int n, m;
12  double a[MAXM][MAXN], v;
13
14  void pivot(int l, int e) {
15    int i, j;
16    a[l][e] = 1 / a[l][e];
17    for (j = 0; j <= n; ++j)
18      if (j != e) a[l][j] *= a[l][e];
19    for (i = 1; i <= m; ++i)
20      if (i != l && fabs(a[i][e]) > EPS) {
21        for (j = 0; j <= n; ++j)
22          if (j != e) a[i][j] -= a[i][e] * a[l][j];
23        a[i][e] = -a[i][e] * a[l][e];
24      }
25    v += a[0][e] * a[l][0];
26    for (j = 1; j <= n; ++j)
27      if (j != e) a[0][j] -= a[0][e] * a[l][j];
28    a[0][e] = -a[0][e] * a[l][e];
29  }
30
31  double simplex() {
32    int e, l, i;
33    double mn;
34    v = 0;
35    while (true) {
36      for (e = 1; e <= n; ++e)
37        if (a[0][e] > EPS) break;
38      if (e > n) return v;
39      for (i = 1, mn = INF; i <= m; ++i)
40        if (a[i][e] > EPS && mn > a[i][0] / a[i][e])
41          mn = a[i][0] / a[i][e], l = i;
42      if (mn == INF) return INF;
```

```
43        pivot(l, e);
44    }
45  }
46
47  void solve() {
48    static int n, m, g[10];
49    static vector<int> con[10], able;
50
51    scanf("%d %d", &n, &m);
52    for (int i = 0; i < n; ++i) {
53      scanf("%d", g + i);
54      con[i].clear();
55    }
56
57    if (n == 1) {
58      printf("%.10f\n", m >= g[0] ? 1. : 0.);
59      return;
60    }
61
62    able.clear();
63    for (int s = 0, S = 1 << n; s < S; ++s) {
64      int sum = 0;
65      for (int i = 0; i < n; ++i)
66        if (s >> i & 1) sum += g[i];
67      if (sum > m) continue;
68      able.push_back(s);
69      for (int i = 0; i < n; ++i)
70        if (s >> i & 1) con[i].push_back(able.size());
71    }
72    ::n = able.size();
73    ::m = 0;
74    static random_device rd;
75    mt19937 gen(rd());
76    shuffle(able.begin(), able.end(), gen);
77    for (int step = 0; step < n; ++step) {
78      int f = ++::m;
79      for (int i = 0; i <= ::n; ++i) a[f][i] = 0;
80      for (int x : con[step]) ++a[f][x];
81      if (step + 1 < n) {
82        for (int x : con[step + 1]) --a[f][x];
83      } else {
84        for (int x : con[0]) --a[f][x];
85      }
86    }
87
88    ++::m;
89    a[::m][0] = 1;
90    for (int i = 1; i <= ::n; ++i) a[::m][i] = 1;
91
92    ++::m;
93    a[::m][0] = -1;
94    for (int i = 1; i <= ::n; ++i) a[::m][i] = -1;
95
96    for (int i = 0; i <= ::n; ++i) a[0][i] = 0;
97    for (int x : con[0]) ++a[0][x];
98    printf("%.10f\n", simplex());
99  }
100
101  int main() {
```

```
102    int o_o, case_number = 1;
103    for (scanf("%d", &o_o); case_number <= o_o; ++case_number) {
104      printf("Case #%d: ", case_number);
105      solve();
106    }
107    return 0;
108  }
109
110  // 备份
111  #include <bits/stdc++.h>
112  using namespace std;
113
114  typedef long double db;
115  const int MAXN = 3000;
116  const int MAXM = 3000;
117  const db EPS = 1e-9;
118  const db INF = 1e200;
119
120  namespace LP {
121  db a[MAXM][MAXN];
122  int idA[MAXN], idB[MAXN];
123  int m, n;
124
125  void put_out(int x) {
126    if (x == 0)
127      printf("Infeasible\n");
128    else
129      printf("Unbounded\n");
130    exit(0);
131  }
132  void pivot(int xA, int xB) {
133    swap(idA[xA], idB[xB]);
134    static int next[MAXN];
135    int i, j, last = MAXN - 1;
136    db tmp = -a[xB][xA];
137    a[xB][xA] = -1.0;
138    for (j = 0; j <= n; j++)
139      if (fabs(a[xB][j]) > EPS) a[xB][last = next[last] = j] /= tmp;
140    next[last] = -1;
141
142    for (i = 0; i <= m; i++)
143      if (i != xB && fabs(tmp = a[i][xA]) > EPS)
144        for (a[i][xA] = 0.0, j = next[MAXN - 1]; ~j; j = next[j])
145          a[i][j] += tmp * a[xB][j];
146  }
147  db calc() {
148    int xA, xB;
149    db Max, tmp;
150    while (1) {
151      xA = n + 1, idA[xA] = n + m + 1;
152      for (int i = 1; i <= n; i++)
153        if (a[0][i] > EPS && idA[i] < idA[xA]) xA = i;
154
155      if (xA == n + 1) return a[0][0];
156      xB = m + 1, idB[xB] = n + m + 1, Max = -INF;
157      for (int i = 1; i <= m; i++)
158        if (a[i][xA] < -EPS && ((tmp = a[i][0] / a[i][xA]) > Max + EPS ||
159                                (tmp > Max - EPS && idB[i] < idB[xB])))
160          Max = tmp, xB = i;
```

4

```
161
162      if (xB == m + 1) put_out(1);
163
164      pivot(xA, xB);
165    }
166    return a[0][0];
167  }
168  db solve() {
169    for (int i = 1; i <= n; i++) idA[i] = i;
170    for (int i = 1; i <= m; i++) idB[i] = n + i;
171    static db tmp[MAXN];
172    db Min = 0.0;
173    int l;
174    for (int i = 1; i <= m; i++)
175      if (a[i][0] < Min) Min = a[i][0], l = i;
176    if (Min > -EPS) return calc();
177
178    idA[++n] = 0;
179    for (int i = 1; i <= m; i++) a[i][n] = 1.0;
180    for (int i = 0; i <= n; i++) tmp[i] = a[0][i], a[0][i] = 0.0;
181    a[0][n] = -1.0;
182
183    pivot(n, l);
184
185    if (calc() < -EPS) put_out(0);
186    for (int i = 1; i <= m; i++)
187      if (!idB[i]) {
188        for (int j = 1; j <= n; j++)
189          if (fabs(a[0][j]) > EPS) {
190            pivot(j, i);
191            break;
192          }
193        break;
194      }
195
196    int xA;
197    for (xA = 1; xA <= n; xA++)
198      if (!idA[xA]) break;
199    for (int i = 0; i <= m; i++) a[i][xA] = a[i][n];
200    idA[xA] = idA[n], n--;
201
202    for (int i = 0; i <= n; i++) a[0][i] = 0.0;
203    for (int i = 1; i <= m; i++)
204      if (idB[i] <= n) {
205        for (int j = 0; j <= n; j++) a[0][j] += a[i][j] * tmp[idB[i]];
206      }
207
208    for (int i = 1; i <= n; i++)
209      if (idA[i] <= n) a[0][i] += tmp[idA[i]];
210    return calc();
211  }
212  db ans[MAXN];
213  void findAns() {
214    for (int i = 1; i <= n; i++) ans[i] = 0.0;
215    for (int i = 1; i <= m; i++)
216      if (idB[i] <= n) ans[idB[i]] = a[i][0];
217  }
218  void work() {
219    for (int i = 1; i <= m; ++i)
```

```
220        for (int j = 1; j <= n; ++j) a[i][j] *= -1;
221      printf("%.10f\n", -double(solve()));
222    }
223    } // namespace LP
224
225    void solve() {
226      static int n, m, g[10];
227      static vector<int> con[10], able;
228
229      scanf("%d %d", &n, &m);
230      for (int i = 0; i < n; ++i) {
231        scanf("%d", g + i);
232        con[i].clear();
233      }
234
235      if (n == 1) {
236        printf("%.10f\n", m >= g[0] ? 1.0 : 0.0);
237        return;
238      }
239
240      able.clear();
241      for (int s = 0; s < (1 << n); ++s) {
242        int sum = 0;
243        for (int i = 0; i < n; ++i)
244          if (s >> i & 1) sum += g[i];
245        if (sum > m) continue;
246
247        able.push_back(s);
248        for (int i = 0; i < n; ++i)
249          if (s >> i & 1) con[i].push_back(able.size());
250      }
251
252      LP::n = able.size();
253      LP::m = 0;
254
255      for (int step = 0; step < n; ++step) {
256        int &f = ++LP::m;
257        for (int i = 0; i <= LP::n; ++i) LP::a[f][i] = 0;
258        for (int x : con[step]) ++LP::a[f][x];
259        if (step + 1 < n) {
260          for (int x : con[step + 1]) --LP::a[f][x];
261        } else {
262          for (int x : con[0]) --LP::a[f][x];
263        }
264      }
265
266      ++LP::m;
267      LP::a[LP::m][0] = 1;
268      for (int i = 1; i <= LP::n; ++i) LP::a[LP::m][i] = 1;
269
270      ++LP::m;
271      LP::a[LP::m][0] = -1;
272      for (int i = 1; i <= LP::n; ++i) LP::a[LP::m][i] = -1;
273
274      for (int i = 0; i <= LP::n; ++i) LP::a[0][i] = 0;
275      for (int x : con[0]) ++LP::a[0][x];
276
277      static db a2[MAXM][MAXN];
278      for (int i = 1; i <= LP::m; ++i)
```

```
279       for (int j = 1; j <= LP::n; ++j) a2[i][j] = LP::a[i][j];
280     for (int i = 1; i <= LP::m; ++i)
281       for (int j = 1; j <= LP::n; ++j) LP::a[j][i] = a2[i][j];
282     swap(LP::n, LP::m);
283     for (int i = 1; i <= max(LP::n, LP::m); ++i) swap(LP::a[0][i], LP::a[i][0]);
284     LP::a[0][0] = 0;
285     for (int i = 1; i <= LP::m; ++i)
286       for (int j = 1; j <= LP::n; ++j) LP::a[i][j] *= -1;
287     for (int i = 1; i <= LP::m; ++i) LP::a[i][0] *= -1;
288     for (int i = 1; i <= LP::n; ++i) LP::a[0][i] *= -1;
289
290     LP::work();
291 }
292
293 int main() {
294     int o_o;
295     scanf("%d", &o_o);
296     for (int i = 1; i <= o_o; ++i) {
297       printf("Case #%d: ", i);
298       solve();
299     }
300     return 0;
301 }
```

1.4 exctr

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

1.5 杜教筛

```
 1 // e = mu x 1
 2 // d = 1 x 1
 3 // sigma = d x 1
 4 // phi = mu x id
 5 // id = phi x 1
 6 // id^2 = (id * phi) x id
 7
 8 // S = sum(f)
 9 // sum(fxg) = sum(g(i)S(n/i))
10 map<int, int> mp_mu;
11
12 int S_mu(int n) {
13     if (n < MAXN) return sum_mu[n];
14     if (mp_mu[n]) return mp_mu[n];
15     int ret = 1;
```

```
16    for (int i = 2, j; i <= n; i = j + 1) {
17      j = n / (n / i);
18      ret -= S_mu(n / i) * (j - i + 1);
19    }
20    return mp_mu[n] = ret;
21  }
22
23  ll S_phi(int n) {
24    ll res = 0;
25    for (int i = 1, j; i <= n; i = j + 1) {
26      j = n / (n / i);
27      res += 1LL * (S_mu(j) - S_mu(i - 1)) * (n / i) * (n / i);
28    }
29    return (res - 1) / 2 + 1;
30  }
```

1.6 Miller Rabin

```
1   inline ll mod_mul(const ll &a, const ll &b, const ll &mod) {
2     ll k = (ll)((1.0L * a * b) / (1.0L * mod)), t = a * b - k * mod;
3     t -= mod;
4     while (t < 0) t += mod;
5     return t;
6   }
7   inline ll mod_pow(ll a, ll b, const ll &mod) {
8     ll res = 1;
9     for (; b; b >>= 1, a = mod_mul(a, a, mod))
10      (b & 1) && (res = mod_mul(res, a, mod));
11    return res;
12  }
13
14  inline bool check(const ll &x, const ll &p) {
15    if (!(x % p) || mod_pow(p % x, x - 1, x) ^ 1) return false;
16    ll k = x - 1, t;
17    while (~k & 1) {
18      if (((t = mod_pow(p % x, k >>= 1, x)) ^ 1) && (t ^ (x - 1))) return false;
19      if (!(t ^ (x - 1))) return true;
20    }
21    return true;
22  }
23
24  inline bool Miller_Rabin(const ll &x) {
25    if (x < 2) return false;
26    static const int p[12] = {2, 3, 5, 7, 11, 13, 17, 19, 61, 2333, 4567, 24251};
27    for (int i = 0; i < 12; ++i) {
28      if (!(x ^ p[i])) return true;
29      if (!check(x, p[i])) return false;
30    }
31    return true;
32  }
```

1.7 burnside

```
1   // |X/G|={1 \over {|G|}} \sum_{g \in G}|X^g|
2   // Gym - 101873B
3   // 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。
4   #include <bits/stdc++.h>
```

```
5  using namespace std;
6
7  const int MOD = 1e9 + 7;
8
9  int mod_pow(int a, int b) {
10   int r = 1;
11   for (; b; b >>= 1, a = 1LL * a * a % MOD)
12     if (b & 1) r = 1LL * a * r % MOD;
13   return r;
14 }
15
16 int main() {
17   ios::sync_with_stdio(false);
18   cin.tie(nullptr);
19
20   int n, m, c;
21   cin >> n >> m >> c;
22
23   int ans = 0;
24   for (int i = 1; i <= m; ++i)
25     ans = (ans + mod_pow(c, n * n * __gcd(i, m))) % MOD;
26   ans = 1LL * ans * mod_pow(m, MOD - 2) % MOD;
27   cout << ans << '\n';
28   return 0;
29 }
```

1.8 类欧几里德算法

```
1  //求 f=sum((a*i+b)/c),g=sum((a*i+b)/c*i),h=sum(((a*i+b)/c)^2), for i in [0..n],
2  //整除向下
3  #include <bits/stdc++.h>
4  #define int long long
5  using namespace std;
6  const int P = 998244353;
7  int i2 = 499122177, i6 = 166374059;
8  struct data {
9    data() { f = g = h = 0; }
10   int f, g, h;
11 }; // 三个函数打包
12 data calc(int n, int a, int b, int c) {
13   int ac = a / c, bc = b / c, m = (a * n + b) / c, n1 = n + 1, n21 = n * 2 + 1;
14   data d;
15   if (a == 0) { // 迭代到最底层
16     d.f = bc * n1 % P;
17     d.g = bc * n % P * n1 % P * i2 % P;
18     d.h = bc * bc % P * n1 % P;
19     return d;
20   }
21   if (a >= c || b >= c) { // 取模
22     d.f = n * n1 % P * i2 % P * ac % P + bc * n1 % P;
23     d.g = ac * n % P * n1 % P * n21 % P * i6 % P + bc * n % P * n1 % P * i2 % P;
24     d.h = ac * ac % P * n % P * n1 % P * n21 % P * i6 % P +
25           bc * bc % P * n1 % P + ac * bc % P * n % P * n1 % P;
26     d.f %= P, d.g %= P, d.h %= P;
27
28     data e = calc(n, a % c, b % c, c); // 迭代
29
30     d.h += e.h + 2 * bc % P * e.f % P + 2 * ac % P * e.g % P;
```

```
31      d.g += e.g, d.f += e.f;
32      d.f %= P, d.g %= P, d.h %= P;
33      return d;
34    }
35    data e = calc(m - 1, c, c - b - 1, a);
36    d.f = n * m % P - e.f, d.f = (d.f % P + P) % P;
37    d.g = m * n % P * n1 % P - e.h - e.f, d.g = (d.g * i2 % P + P) % P;
38    d.h = n * m % P * (m + 1) % P - 2 * e.g - 2 * e.f - d.f;
39    d.h = (d.h % P + P) % P;
40    return d;
41  }
42
43  int T, n, a, b, c;
44  signed main() {
45    scanf("%lld", &T);
46    while (T--) {
47      scanf("%lld%lld%lld%lld", &n, &a, &b, &c);
48      data ans = calc(n, a, b, c);
49      printf("%lld %lld %lld\n", ans.f, ans.h, ans.g);
50    }
51    return 0;
52  }
```

1.9   Pollard rho

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

1.10   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(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;
```

```
60  }
```

1.11   gauss

```
1
2   const double EPS = 1e-9;
3   const int MAXN = MAX_NODE;
4   double a[MAXN][MAXN], x[MAXN];
5   int equ, var;
6
7   int gauss() {
8     int i, j, k, col, max_r;
9     for (k = 0, col = 0; k < equ && col < var; k++, col++) {
10      max_r = k;
11      for (i = k + 1; i < equ; i++)
12        if (fabs(a[i][col]) > fabs(a[max_r][col])) max_r = i;
13      if (fabs(a[max_r][col]) < EPS) return 0;
14
15      if (k != max_r) {
16        for (j = col; j < var; j++) swap(a[k][j], a[max_r][j]);
17        swap(x[k], x[max_r]);
18      }
19
20      x[k] /= a[k][col];
21      for (j = col + 1; j < var; j++) a[k][j] /= a[k][col];
22      a[k][col] = 1;
23
24      for (i = k + 1; i < equ; i++)
25        if (i != k) {
26          x[i] -= x[k] * a[i][col];
27          for (j = col + 1; j < var; j++) a[i][j] -= a[k][j] * a[i][col];
28          a[i][col] = 0;
29        }
30    }
31
32    for (col = equ - 1, k = var - 1; ~col; --col, --k) {
33      if (fabs(a[col][k]) > 0) {
34        for (i = 0; i < k; ++i) {
35          x[i] -= x[k] * a[i][col];
36          for (j = col + 1; j < var; j++) a[i][j] -= a[k][j] * a[i][col];
37          a[i][col] = 0;
38        }
39      }
40    }
41
42    return 1;
43  }
```

1.12   exgcd

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

1.13   china

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

1.14   LinearSieve

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

1.15   LinearRecurrence

```
1   struct LinearRecurrence {
2     using int64 = long long;
3     using vec = std::vector<int64>;
4
5     static void extand(vec &a, size_t d, int64 value = 0) {
```

13

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

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

```
124                   if (an[o][i + m] < 0) an[o][i + m] += mod;
125                 }
126               while (an[o].size() && an[o].back() == 0) an[o].pop_back();
127               for (size_t i = 0; i < bo[g].size(); ++i) {
128                 bn[o][i + m] -= coef * bo[g][i] % mod;
129                 if (bn[o][i + m] < 0) bn[o][i + m] -= mod;
130               }
131               while (bn[o].size() && bn[o].back() == 0) bn[o].pop_back();
132             }
133           }
134         }
135       }
136       return std::make_pair(an[0], bn[0]);
137     };
138
139     std::vector<std::tuple<int64, int64, int>> fac;
140     for (int64 i = 2; i * i <= mod; ++i)
141       if (mod % i == 0) {
142         int64 cnt = 0, pw = 1;
143         while (mod % i == 0) mod /= i, ++cnt, pw *= i;
144         fac.emplace_back(pw, i, cnt);
145       }
146     if (mod > 1) fac.emplace_back(mod, mod, 1);
147     std::vector<vec> as;
148     size_t n = 0;
149     for (auto &&x : fac) {
150       int64 mod, p, e;
151       vec a, b;
152       std::tie(mod, p, e) = x;
153       auto ss = s;
154       for (auto &&x : ss) x %= mod;
155       std::tie(a, b) = prime_power(ss, mod, p, e);
156       as.emplace_back(a);
157       n = std::max(n, a.size());
158     }
159     vec a(n), c(as.size()), m(as.size());
160     for (size_t i = 0; i < n; ++i) {
161       for (size_t j = 0; j < as.size(); ++j) {
162         m[j] = std::get<0>(fac[j]);
163         c[j] = i < as[j].size() ? as[j][i] : 0;
164       }
165       a[i] = crt(c, m);
166     }
167     return a;
168   }
169
170   LinearRecurrence(const vec &s, const vec &c, int64 mod)
171       : init(s), trans(c), mod(mod), m(s.size()) {}
172
173   LinearRecurrence(const vec &s, int64 mod, bool is_prime = true) : mod(mod) {
174     vec A = is_prime ? BerlekampMassey(s, mod) : ReedsSloane(s, mod);
175     if (A.empty()) A = {0};
176     m = A.size() - 1;
177     trans.resize(m);
178     for (int i = 0; i < m; ++i) { trans[i] = (mod - A[i + 1]) % mod; }
179     std::reverse(trans.begin(), trans.end());
180     init = {s.begin(), s.begin() + m};
181   }
182
```

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

2 Dynamic Programming

2.1 斜率优化

```
1  // 树上斜率优化
2  // 定义dpi 表示i节点传递到根节点的最短耗时，规定dproot=-P。
3  // 有如下转移方程dpu=dpv+dist(u,v)^2+P,v is an ancestor of u.
4
5  #include <bits/stdc++.h>
6  using namespace std;
7
8  typedef long long ll;
9  typedef pair<int, int> pii;
10 const int MAXN = 1e5 + 5;
11
12 vector<pii> adj[MAXN];
13 ll dp[MAXN], d[MAXN];
14 int n, p, q[MAXN], head, tail;
15
16 inline ll S(int a, int b) { return (d[b] - d[a]) << 1; }
17 inline ll G(int a, int b) { return dp[b] - dp[a] + d[b] * d[b] - d[a] * d[a]; }
18
```

```
19  void dfs(int u, int from) {
20    vector<int> dhead, dtail;
21    if (u ^ 1) {
22      while (head + 2 <= tail &&
23             S(q[head + 1], q[head]) * d[u] <= G(q[head + 1], q[head]))
24        dhead.push_back(q[head++]);
25      int v = q[head];
26      dp[u] = dp[v] + p + (d[u] - d[v]) * (d[u] - d[v]);
27    }
28    while (head + 2 <= tail &&
29           G(u, q[tail - 1]) * S(q[tail - 1], q[tail - 2]) <=
30               G(q[tail - 1], q[tail - 2]) * S(u, q[tail - 1]))
31      dtail.push_back(q[--tail]);
32    q[tail++] = u;
33    for (pii &e : adj[u]) {
34      if (e.first == from) continue;
35      d[e.first] = d[u] + e.second;
36      dfs(e.first, u);
37    }
38    --tail;
39    for (int i = dtail.size() - 1; ~i; --i) q[tail++] = dtail[i];
40    for (int i = dhead.size() - 1; ~i; --i) q[--head] = dhead[i];
41  }
42
43  void solve() {
44    cin >> n >> p;
45    for (int i = 1; i <= n; ++i) adj[i].clear();
46    for (int i = 1, u, v, w; i < n; ++i) {
47      cin >> u >> v >> w;
48      adj[u].emplace_back(v, w);
49      adj[v].emplace_back(u, w);
50    }
51    dp[1] = -p;
52    head = tail = 0;
53    dfs(1, 1);
54
55    ll ans = 0;
56    for (int i = 1; i <= n; ++i)
57      if (dp[i] > ans) ans = dp[i];
58    cout << ans << '\n';
59  }
60
61  int main() {
62    // freopen("in.txt", "r", stdin);
63    ios::sync_with_stdio(false);
64    cin.tie(0);
65
66    int o_o;
67    for (cin >> o_o; o_o; --o_o) solve();
68
69    return 0;
70  }
```

3 Data Structure

3.1 splay

```
1  #include <algorithm>
```

```cpp
 2  #include <cstdio>
 3  #include <cstring>
 4  #include <iostream>
 5  using namespace std;
 6
 7  const int MAXN = 2e5 + 10;
 8
 9  struct Node {
10    long long sum;
11    int id, val, lazy, size;
12    Node *fa, *ch[2];
13  } node_pool[MAXN], *pool_it, *root, *nil;
14
15  Node *newnode(int id, int val) {
16    pool_it->id = id;
17    pool_it->lazy = 0;
18    pool_it->size = 1;
19    pool_it->sum = pool_it->val = val;
20    pool_it->fa = pool_it->ch[0] = pool_it->ch[1] = nil;
21    return pool_it++;
22  }
23
24  void maintain(Node *u) {
25    if (u == nil) { return; }
26    u->size = u->ch[0]->size + u->ch[1]->size + 1;
27    u->sum = u->ch[0]->sum + u->ch[1]->sum + u->val;
28  }
29
30  void push_down(Node *u) {
31    if (u->lazy) {
32      if (u->ch[0] != nil) {
33        u->ch[0]->val += u->lazy;
34        u->ch[0]->sum += 1LL * u->ch[0]->size * u->lazy;
35        u->ch[0]->lazy += u->lazy;
36      }
37      if (u->ch[1] != nil) {
38        u->ch[1]->val += u->lazy;
39        u->ch[1]->sum += 1LL * u->ch[1]->size * u->lazy;
40        u->ch[1]->lazy += u->lazy;
41      }
42      u->lazy = 0;
43    }
44  }
45
46  inline void rot(Node *u) {
47    Node *f = u->fa, *ff = f->fa;
48    int d = u == f->ch[1];
49    push_down(f);
50    push_down(u);
51    if ((f->ch[d] = u->ch[d ^ 1]) != nil) f->ch[d]->fa = f;
52    if ((u->fa = ff) != nil) ff->ch[f == ff->ch[1]] = u;
53    f->fa = u;
54    u->ch[d ^ 1] = f;
55    maintain(f);
56    maintain(u);
57  }
58
59  void splay(Node *u, Node *target) {
60    for (Node *f; u->fa != target; rot(u))
```

19

```
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);
153      cout.tie(0);
154
155      int n;
156      cin >> n;
157
158      pool_it = node_pool;
159      nil = pool_it++;
160      nil->ch[0] = nil->ch[1] = nil->fa = nil;
161      nil->id = -1;
162      nil->val = 0;
163      root = nil;
164
165      insert(-0x3fffffff, 0);
166      insert(0x3fffffff, 0);
167
168      return 0;
169    }
```

3.2 lct

```
1    struct LCT {
2      struct node {
3        int val, add;
4        node *fa, *ch[2];
5        void modify(const int &x) {
```

```
 6        val += x;
 7        add += x;
 8      }
 9    } node_mset[MaxS], *cnode, *null;
10    LCT() {
11      cnode = node_mset;
12      null = cnode++;
13      *null = (node){0, 0, null, {null, null}};
14    }
15    inline node *newnode() {
16      *cnode = (node){0, 0, null, {null, null}};
17      return cnode++;
18    }
19    inline bool isrt(node *u) const {
20      return (u->fa->ch[0] != u) && (u->fa->ch[1] != u);
21    }
22    inline bool which(node *u) const { return u->fa->ch[1] == u; }
23    void push_down(node *u) {
24      if (!isrt(u)) push_down(u->fa);
25      if (u->add) {
26        u->ch[0]->modify(u->add);
27        u->ch[1]->modify(u->add);
28        u->add = 0;
29      }
30    }
31    inline void rotate(node *u) {
32      node *f = u->fa;
33      int d = which(u);
34      f->ch[d] = u->ch[d ^ 1];
35      f->ch[d]->fa = f;
36      u->ch[d ^ 1] = f;
37      u->fa = f->fa;
38      if (!isrt(f)) f->fa->ch[which(f)] = u;
39      f->fa = u;
40    }
41    inline void splay(node *u) {
42      push_down(u);
43      for (node *f; !isrt(u); rotate(u))
44        if (!isrt(f = u->fa)) rotate(which(u) == which(f) ? f : u);
45    }
46    inline void access(node *x) {
47      for (node *y = null; x != null; x = x->fa) {
48        splay(x);
49        x->ch[1] = y;
50        y = x;
51      }
52    }
53    inline void cut(node *u) {
54      access(u);
55      splay(u);
56      u->ch[0]->fa = null;
57      u->ch[0] = null;
58    }
59    inline void link(node *u, node *v) {
60      cut(u);
61      u->fa = v;
62    }
63  } tree;
```

3.3 kdtree

```
1   // 寻找近点
2   #include <bits/stdc++.h>
3   using namespace std;
4
5   const int MAXN = 2e5 + 5;
6   typedef long long ll;
7
8   namespace KD_Tree {
9
10  const int DIM = 2;
11
12  inline ll sqr(int x) { return 1LL * x * x; }
13
14  struct Point {
15    int x[DIM], id, c;
16
17    ll dist2(const Point &b) const {
18      return sqr(x[0] - b.x[0]) + sqr(x[1] - b.x[1]);
19    }
20  };
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) {
80          if (x->e.c <= p.c) ans = QNode(x->e, p.dist2(x->e));
81          search(p, x->lc, div ^ 1);
82        } else {
83          QNode temp(x->e, p.dist2(x->e));
84          if (x->e.c <= p.c && temp < ans) ans = temp;
85          if (sqr(x->e.x[div] - p.x[div]) <= ans.dis2) search(p, x->lc, div ^ 1);
86        }
87      }
88    }
89    void search(Point p) {
90      ans.dis2 = -1;
91      search(p, root, 0);
92    }
93    } // namespace KD_Tree
94
95    void solve() {
96      static KD_Tree::Point p[MAXN];
97      int n, m;
98      cin >> n >> m;
99      for (int i = 0; i < n; ++i) {
100       p[i].id = i;
101       cin >> p[i].x[0] >> p[i].x[1] >> p[i].c;
102     }
103     KD_Tree::init();
104     KD_Tree::root = KD_Tree::build(p, 0, n, 0);
105
106     for (KD_Tree::Point q; m; --m) {
107       cin >> q.x[0] >> q.x[1] >> q.c;
108       KD_Tree::search(q);
109       cout << KD_Tree::ans.p.x[0] << ' ' << KD_Tree::ans.p.x[1] << ' '
110            << KD_Tree::ans.p.c << '\n';
111     }
112   }
113   int main() {
114     ios::sync_with_stdio(false);
115     cin.tie(nullptr);
116
```

```
117    int o_o;
118    for (cin >> o_o; o_o; --o_o) solve();
119
120    return 0;
121  }
122
123  // 寻找远点
124  inline void cmin(int &a, int b) { b < a ? a = b : 1; }
125  inline void cmax(int &a, int b) { a < b ? a = b : 1; }
126  inline int ibs(int a) { return a < 0 ? -a : a; }
127  struct D {
128    int d[2], mx0, mx1, mi0, mi1;
129    D *l, *r;
130  } t[N], *rt;
131  int cpd, ans;
132  inline bool cmp(const D &a, const D &b) {
133    return (a.d[cpd] ^ b.d[cpd]) ? a.d[cpd] < b.d[cpd]
134                                 : a.d[cpd ^ 1] < b.d[cpd ^ 1];
135  }
136  inline void kd_upd(D *u) {
137    if (u->l) {
138      cmax(u->mx0, u->l->mx0);
139      cmax(u->mx1, u->l->mx1);
140      cmin(u->mi0, u->l->mi0);
141      cmin(u->mi1, u->l->mi1);
142    }
143    if (u->r) {
144      cmax(u->mx0, u->r->mx0);
145      cmax(u->mx1, u->r->mx1);
146      cmin(u->mi0, u->r->mi0);
147      cmin(u->mi1, u->r->mi1);
148    }
149  }
150  D *kd_bld(int l, int r, int d) {
151    int m = l + r >> 1;
152    cpd = d;
153    std::nth_element(t + l + 1, t + m + 1, t + r + 1, cmp);
154    t[m].mx0 = t[m].mi0 = t[m].d[0];
155    t[m].mx1 = t[m].mi1 = t[m].d[1];
156    if (l ^ m) t[m].l = kd_bld(l, m - 1, d ^ 1);
157    if (r ^ m) t[m].r = kd_bld(m + 1, r, d ^ 1);
158    kd_upd(t + m);
159    return t + m;
160  }
161  inline void kd_ins(D *ne) {
162    int cd = 0;
163    D *u = rt;
164    while (true) {
165      cmax(u->mx0, ne->mx0), cmin(u->mi0, ne->mi0);
166      cmax(u->mx1, ne->mx1), cmin(u->mi1, ne->mi1);
167      if (ne->d[cd] < u->d[cd]) {
168        if (u->l)
169          u = u->l;
170        else {
171          u->l = ne;
172          return;
173        }
174      } else {
175        if (u->r)
```

```
176          u = u->r;
177        else {
178          u->r = ne;
179          return;
180        }
181      }
182      cd ^= 1;
183    }
184  }
185  inline int dist(int x, int y, D *u) {
186    int r = 0;
187    if (x < u->mi0)
188      r = u->mi0 - x;
189    else if (x > u->mx0)
190      r = x - u->mx0;
191    if (y < u->mi1)
192      r += u->mi1 - y;
193    else if (y > u->mx1)
194      r += y - u->mx1;
195    return r;
196  }
197  inline void kd_quy(D *u, const int &x, const int &y) {
198    int dl, dr, d0;
199    d0 = ibs(u->d[0] - x) + ibs(u->d[1] - y);
200    if (d0 < ans) ans = d0;
201    dl = u->l ? dist(x, y, u->l) : inf;
202    dr = u->r ? dist(x, y, u->r) : inf;
203    if (dl < dr) {
204      if (dl < ans) kd_quy(u->l, x, y);
205      if (dr < ans) kd_quy(u->r, x, y);
206    } else {
207      if (dr < ans) kd_quy(u->r, x, y);
208      if (dl < ans) kd_quy(u->l, x, y);
209    }
210  }
```

3.4   zkw

```
1  int tree[MAXN * 2], pre;
2
3  void init(int n, int *a) {
4    memset(tree, 0, sizeof(tree));
5    for (pre = 1; pre <= n; pre <<= 1) {}
6    for (int i = 1; i <= n; ++i) tree[i + pre] = a[i];
7    for (int i = pre; i; --i) tree[i] = max(tree[i << 1], tree[i << 1 | 1]);
8  }
9
10  void update(int pos, const int &val) {
11    tree[pos += pre] = val;
12    for (pos >>= 1; pos; pos >>= 1)
13      tree[pos] = max(tree[pos << 1], tree[pos << 1 | 1]);
14  }
15
16  int query(int s, int t) {
17    int res = 0;
18    for (s += pre - 1, t += pre + 1; s ^ t ^ 1; s >>= 1, t >>= 1) {
19      if (~s & 1) res = max(res, tree[s ^ 1]);
20      if (t & 1) res = max(res, tree[t ^ 1]);
```

```
21      }
22      return res;
23  }
```

4   String

4.1   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  }
```

4.2   后缀数组

```
 1  const int MAXN = 1e5 + 5;
 2  const int BUFFER_SIZE = MAXN * 50;
 3
 4  char buffer[BUFFER_SIZE], *buffer_ptr = buffer;
 5  #define alloc(x, type, len)                                          \
 6    type *x = (type *)buffer_ptr;                                      \
 7    buffer_ptr += (len) * sizeof(type)
 8
 9  template <int LENGTH> class SuffixArray {
10  public:
11    int sa[LENGTH], rk[LENGTH], h[LENGTH];
12
13    void get_sa(int *s, int n, int m) { sais(n, m, s, sa); }
14
15    void get_height(int *s, int n) {
16      for (int i = 1; i <= n; ++i) rk[sa[i]] = i;
17      for (int k = 0, i = 0, j; i < n; h[rk[i++]] = k)
18        for (k ? k-- : 0, j = sa[rk[i] - 1]; s[i + k] == s[j + k]; ++k) {}
19    }
20
21  private:
22    void induced_sort(int n, int m, int *s, bool *T, int *sa, int *cnt) {
23      static int lptr[LENGTH], rptr[LENGTH];
24      int tmp;
25      lptr[0] = rptr[0] = 0;
26      for (int i = 1; i <= m; ++i) lptr[i] = cnt[i - 1], rptr[i] = cnt[i] - 1;
27      for (int i = 0; i <= n; ++i)
28        sa[i] > 0 && !T[tmp = sa[i] - 1] ? sa[lptr[s[tmp]]++] = tmp : 0;
29      for (int i = n; i >= 0; --i)
```

```
30          sa[i] > 0 && T[tmp = sa[i] - 1] ? sa[rptr[s[tmp]]--] = tmp : 0;
31      }
32
33      bool compare(int *x, int *y, int n) {
34        while (n--)
35          if (*x++ != *y++) return 1;
36        return 0;
37      }
38
39      void sais(int n, int m, int *s, int *sa) {
40        --n;
41        alloc(T, bool, n + 3);
42        alloc(cnt, int, m + 3);
43        alloc(lms, int, n + 3);
44        alloc(tmp, int, n + 3);
45
46        T[n] = true, ++cnt[s[n]];
47        for (int i = n - 1; i >= 0; --i) {
48          ++cnt[s[i]];
49          T[i] = s[i] < s[i + 1] || (s[i] == s[i + 1] && T[i + 1]);
50        }
51        tmp[0] = 0;
52        for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1], tmp[i] = cnt[i] - 1;
53
54        int LMS = 0;
55        for (int i = 1; i <= n; ++i) T[i] && !T[i - 1] ? lms[LMS++] = i : 0;
56
57        memset(sa, -1, (n + 1) * sizeof(int));
58        for (int i = 0; i < LMS; ++i) sa[tmp[s[lms[i]]]--] = lms[i];
59        induced_sort(n, m, s, T, sa, cnt);
60
61        alloc(len, int, n + 10);
62        lms[LMS] = n + 1;
63        for (int i = 0; i < LMS; ++i) len[lms[i]] = lms[i + 1] - lms[i] + 1;
64
65        alloc(label, int, n + 10);
66        int m0 = 0;
67        memset(label, -1, (n + 1) * sizeof(int));
68        for (int i = 1, p = -1, q = -1; i <= n; ++i) {
69          if ((q = sa[i]) > 0 && T[q] && !T[q - 1]) {
70            if (p == -1 || len[p] != len[q] || compare(s + p, s + q, len[p])) ++m0;
71            label[q] = m0, p = q;
72          }
73        }
74        label[n] = 0;
75
76        alloc(s0, int, LMS + 3);
77        alloc(sa0, int, LMS + 3);
78        for (int i = 0, p = 0; i <= n; ++i) label[i] >= 0 ? s0[p++] = label[i] : 0;
79        if (m0 + 1 == LMS) {
80          for (int i = 0; i < LMS; ++i) sa0[s0[i]] = i;
81        } else {
82          sais(LMS, m0, s0, sa0);
83        }
84
85        tmp[0] = 0;
86        for (int i = 1; i <= m; ++i) tmp[i] = cnt[i] - 1;
87        memset(sa, -1, (n + 1) << 2);
88        for (int i = LMS - 1; i >= 0; --i) sa[tmp[s[lms[sa0[i]]]]--] = lms[sa0[i]];
```

```
89      induced_sort(n, m, s, T, sa, cnt);
90    }
91 };
```

4.3 SAM

```
1  struct Node {
2    int len;
3    Node *link, *ch[ALPHABET_SIZE];
4  } node_pool[MAXS], *node_it, *root, *last;
5
6  Node *new_node(int len) {
7    node_it->len = len;
8    return node_it++;
9  }
10 void sam_init() {
11   node_it = node_pool;
12   last = root = new_node(0);
13 }
14 void sam_extend(int c, int val) {
15   Node *p = last, *np = new_node(p->len + 1);
16   for (last = np; p && !p->ch[c]; p = p->link) p->ch[c] = np;
17   if (!p) {
18     np->link = root;
19   } else {
20     Node *q = p->ch[c];
21     if (q->len == p->len + 1) {
22       np->link = q;
23     } else {
24       Node *nq = new_node(p->len + 1);
25       memcpy(nq->ch, q->ch, sizeof(q->ch));
26       nq->link = q->link;
27       q->link = np->link = nq;
28       for (; p && p->ch[c] == q; p = p->link) p->ch[c] = nq;
29     }
30   }
31 }
```

4.4 ACam

```
1  int ch[MAX_NODE][26], fail[MAX_NODE], dep[MAX_NODE], node_c;
2
3  int add_char(int u, int id) {
4    if (ch[u][id] < 0) ch[u][id] = node_c++;
5    return ch[u][id];
6  }
7  void build_acam() {
8    queue<int> que;
9    for (int i = 0; i < 26; ++i)
10     if (~ch[0][i]) {
11       que.push(ch[0][i]);
12       fail[ch[0][i]] = 0;
13       dep[ch[0][i]] = 1;
14     } else {
15       ch[0][i] = 0;
16     }
17   while (!que.empty()) {
```

```
18      int u = que.front();
19      que.pop();
20      for (int i = 0; i < 26; ++i)
21        if (~ch[u][i]) {
22          que.push(ch[u][i]);
23          fail[ch[u][i]] = ch[fail[u]][i];
24          dep[ch[u][i]] = dep[u] + 1;
25        } else {
26          ch[u][i] = ch[fail[u]][i];
27        }
28    }
29    for (int i = 1; i < node_c; ++i) adj[fail[i]].push_back(i);
30  }
```

4.5   kmp

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

4.6   exkmp

```
1  // next[i]:x[i...m-1] 与 x[0...m-1] 的最长公共前缀
2  // extend[i]:y[i...n-1] 与 x[0...m-1] 的最长公共前缀
3  void prework(char x[], int m, int next[]) {
4    next[0] = m;
5    int j = 0;
6    while (j + 1 < m && x[j] == x[j + 1]) ++j;
7    next[1] = j;
8    int k = 1;
9    for (int i = 2; i < m; ++i) {
10     int p = next[k] + k - 1;
11     int L = next[i - k];
12     if (i + L < p + 1)
13       next[i] = L;
14     else {
15       j = max(0, p - i + 1);
16       while (i + j < m && x[i + j] == x[j]) j++;
17       next[i] = j;
18       k = i;
19     }
20   }
```

```
21  }
22  void exkmp(char x[], int m, char y[], int n, int next[], int extend[]) {
23    prework(x, m, next);
24    int j = 0;
25    while (j < n && j < m && x[j] == y[j]) ++j;
26    extend[0] = j;
27    int k = 0;
28    for (int i = 1; i < n; ++i) {
29      int p = extend[k] + k - 1;
30      int L = next[i - k];
31      if (i + L < p + 1)
32        extend[i] = L;
33      else {
34        j = max(0, p - i + 1);
35        while (i + j < n && j < m && y[i + j] == x[j]) j++;
36        extend[i] = j;
37        k = i;
38      }
39    }
40  }
```

4.7   hash

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

4.8   回文树

```
1   //最长双回文串
2   struct PT {
3     char s[MAXL];
4     int fail[MAXL], ch[26][MAXL], l[MAXL], dep[MAXL], lst, nc, n;
5     void init() {
6       l[0] = 0;
7       l[1] = -1;
8       fail[0] = fail[1] = 1;
9       for (int i = 0; i < 26; ++i) {
10        for (int j = 0; j < nc; ++j) { ch[i][j] = 0; }
11      }
12      for (int i = 2; i < nc; ++i) {
13        l[i] = 0;
14        fail[i] = 0;
```

```
15      }
16
17      lst = 0;
18      nc = 2;
19      n = 0;
20      s[0] = '#';
21    }
22
23    int insert(char c) {
24      int id = c - 'a';
25      s[++n] = c;
26      while (s[n - l[lst] - 1] != s[n]) { lst = fail[lst]; }
27      if (ch[id][lst] == 0) {
28        l[nc] = l[lst] + 2;
29        int f = fail[lst];
30        while (s[n - l[f] - 1] != s[n]) { f = fail[f]; }
31        fail[nc] = ch[id][f];
32        dep[nc] = dep[fail[nc]] + 1;
33        ch[id][lst] = nc;
34        ++nc;
35      }
36      lst = ch[id][lst];
37      return lst;
38    }
39 } pt;
40
41 char S[MAXL];
42 int len[MAXL];
43 int main() {
44    ios::sync_with_stdio(false);
45    cin.tie(0);
46    cout.tie(0);
47
48    cin >> S;
49    int n = strlen(S);
50    pt.init();
51    for (int i = 0; i < n; ++i) { len[i] = pt.l[pt.insert(S[i])]; }
52    pt.init();
53    int ans = 0;
54    for (int i = n - 1; i; --i) {
55      ans = max(ans, len[i - 1] + pt.l[pt.insert(S[i])]);
56    }
57    cout << ans << "\n";
58
59    return 0;
60 }
```

5   Graph Theory

5.1   上下界费用流

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 53;
5 const int MAX_NODE = 113;
6 const int MAX_EDGE = 1e5 + 5;
7 const int INF = 0x3f3f3f3f;
```

```
 8
 9  int n, s, t, ss, tt, tote;
10  int R[MAXN], C[MAXN], board[MAXN][MAXN];
11
12  struct Edge {
13    int to, cap, flow, cost;
14  } edges[MAX_EDGE];
15  vector<int> adj[MAX_NODE];
16
17  int from[MAX_NODE], in[MAX_NODE];
18  void add_edge(int from, int to, int l, int r, int cost) {
19    in[to] += l, in[from] -= l;
20    edges[tote] = (Edge){to, r - l, 0, cost};
21    adj[from].push_back(tote++);
22    edges[tote] = (Edge){from, 0, 0, -cost};
23    adj[to].push_back(tote++);
24  }
25
26  bool spfa(int s, int t) {
27    static queue<int> q;
28    static bool inq[MAX_NODE];
29    static int dist[MAX_NODE];
30    memset(inq + 1, 0, sizeof(bool) * tt);
31    memset(dist + 1, 0x3f, sizeof(int) * tt);
32    memset(from + 1, -1, sizeof(int) * tt);
33    dist[0] = 0, from[0] = -1;
34    q.push(0);
35    while (!q.empty()) {
36      int u = q.front();
37      q.pop();
38      inq[u] = false;
39      for (int e : adj[u]) {
40        if (edges[e].cap == edges[e].flow) continue;
41        int v = edges[e].to, d = dist[u] + edges[e].cost;
42        if (d >= dist[v]) continue;
43        dist[v] = d;
44        from[v] = e;
45        if (!inq[v]) {
46          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);
106    }
107    add_edge(t, s, 0, INF, 0);
108    int sum = 0;
109    for (int i = 1; i < tt; ++i) {
110      if (in[i] > 0) {
111        sum += in[i];
112        add_edge(ss, i, 0, in[i], 0);
113      } else if (in[i] < 0) {
114        add_edge(i, tt, 0, -in[i], 0);
115      }
116    }
117
118    pair<int, int> ans = min_cost_max_flow(ss, tt);
119    if (sum != ans.first) {
120      cout << "-1\n";
121    } else {
122      cout << ans.second << '\n';
123    }
124  }
125
```

```
126  int main() {
127    ios::sync_with_stdio(false);
128    cin.tie(nullptr);
129
130    while (cin >> n) solve();
131    return 0;
132  }
```

5.2 tarjan

```
 1  vector<int> adj[MAXN];
 2  int dfn[MAXN], low[MAXN], dfs_c;
 3  int bel[MAXN], size[MAXN], scc, stk[MAXN], top, in_stack[MAXN];
 4
 5  void tarjan(int u) {
 6    dfn[u] = low[u] = ++dfs_c;
 7    stk[top++] = u;
 8    in_stack[u] = 1;
 9    for (size_t i = 0; i < adj[u].size(); ++i) {
10      int v = adj[u][i];
11      if (!dfn[v]) {
12        tarjan(v);
13        (low[v] < low[u]) && (low[u] = low[v]);
14      } else if (in_stack[v] && dfn[v] < low[u]) {
15        low[u] = dfn[v];
16      }
17    }
18    if (low[u] == dfn[u]) {
19      int v;
20      size[++scc] = 0;
21      do {
22        v = stk[--top];
23        in_stack[v] = 0;
24        bel[v] = scc;
25        ++size[scc];
26      } while (u != v);
27    }
28  }
```

5.3 一般图最大匹配

```
 1  class GeneralMatch {
 2  public:
 3    int n;
 4    vector<vector<int>> g;
 5    vector<int> match, aux, label, orig, parent;
 6    queue<int> q;
 7    int aux_time;
 8
 9    GeneralMatch(int n)
10        : match(n, -1), aux(n, -1), label(n), orig(n), parent(n, -1),
11          aux_time(-1) {
12      this->n = n;
13      g.resize(n);
14    }
15
16    void add_edge(int u, int v) {
```

```
17      g[u].push_back(v);
18      g[v].push_back(u);
19    }
20
21    int find(int x) { return x == orig[x] ? x : orig[x] = find(orig[x]); }
22
23    int lca(int u, int v) {
24      ++aux_time;
25      u = find(u), v = find(v);
26      for (;; swap(u, v)) {
27        if (~u) {
28          if (aux[u] == aux_time) return u;
29          aux[u] = aux_time;
30          if (match[u] == -1) {
31            u = -1;
32          } else {
33            u = find(parent[match[u]]);
34          }
35        }
36      }
37    }
38
39    void blossom(int u, int v, int o) {
40      while (find(u) != o) {
41        parent[u] = v;
42        v = match[u];
43        q.push(v);
44        label[v] = 0;
45        orig[u] = orig[v] = o;
46        u = parent[v];
47      }
48    }
49
50    int bfs(int x) {
51      iota(orig.begin(), orig.end(), 0);
52      fill(label.begin(), label.end(), -1);
53      while (!q.empty()) q.pop();
54      q.push(x);
55      label[x] = 0;
56      while (!q.empty()) {
57        int u = q.front();
58        q.pop();
59        for (int v : g[u]) {
60          if (label[v] == -1) {
61            parent[v] = u;
62            label[v] = 1;
63            if (match[v] == -1) {
64              while (v != -1) {
65                int pv = parent[v];
66                int next_v = match[pv];
67                match[v] = pv;
68                match[pv] = v;
69                v = next_v;
70              }
71              return 1;
72            }
73            q.push(match[v]);
74            label[match[v]] = 0;
75          } else if (label[v] == 0 && find(u) != find(v)) {
```

```
76              int o = lca(u, v);
77              blossom(u, v, o);
78              blossom(v, u, o);
79            }
80          }
81        }
82        return 0;
83      }
84
85      int find_max_match() {
86        int res = 0;
87        for (int i = 0; i < n; ++i) {
88          if (~match[i]) continue;
89          res += bfs(i);
90        }
91        return res;
92      }
93    };
```

5.4   sap

```
1   struct MF {
2     struct Edge {
3       int to, cap, flow;
4     } edges[MAXM * 4];
5
6     vector<int> adj[MAXN];
7     int n, edges_c, dep[MAXN], depc[MAXN], s, t, last[MAXN];
8
9     void init(int _n) {
10      n = _n;
11      for (int i = 1; i <= n; ++i) adj[i].clear();
12      edges_c = 0;
13    }
14
15    void add_edge(int v, int u, int cap) {
16      edges[edges_c] = {v, cap, 0};
17      adj[u].push_back(edges_c++);
18      edges[edges_c] = {u, 0, 0};
19      adj[v].push_back(edges_c++);
20    }
21
22    int dfs(int u, int flow) {
23      if (u == t || !flow) return flow;
24      int v, e, temp, res = 0;
25      for (int &i = last[u]; i < (int)adj[u].size(); ++i) {
26        e = adj[u][i];
27        v = edges[e].to;
28        if (edges[e].cap == edges[e].flow) continue;
29        if (dep[v] != dep[u] - 1) continue;
30        temp = dfs(v, min(flow, edges[e].cap - edges[e].flow));
31        edges[e].flow += temp, edges[e ^ 1].flow -= temp;
32        res += temp, flow -= temp;
33        if (!flow) return res;
34        if (!dep[s]) return res;
35      }
36      last[u] = 0;
37      if (!(--depc[dep[u]])) dep[s] = n + 1;
```

```
38      ++depc[++dep[u]];
39      return res;
40    }
41    int max_flow(int s, int t) {
42      this->s = s, this->t = t;
43
44      static queue<int> que;
45      memset(dep + 1, 0, sizeof(int) * n);
46      memset(depc + 1, 0, sizeof(int) * n);
47      memset(last + 1, 0, sizeof(int) * n);
48      while (!que.empty()) que.pop();
49      dep[t] = 1, que.push(t);
50
51      while (!que.empty()) {
52        int u = que.front();
53        que.pop();
54        ++depc[dep[u]];
55        for (int i = 0, v; i < (int)adj[u].size(); ++i) {
56          v = edges[adj[u][i]].to;
57          if (dep[v]) continue;
58          dep[v] = dep[u] + 1;
59          que.push(v);
60        }
61      }
62
63      int res = 0;
64      while (dep[s] <= n) res += dfs(s, INT_MAX);
65      return res;
66    }
67  };
```

5.5 最小费用流

```
1  class MinCostFlow {
2  public:
3    struct Result {
4      int flow, cost;
5    };
6    struct Edge {
7      int to, next, rest, cost;
8    };
9
10   vector<bool> inq;
11   vector<int> head, dist, from, flow;
12   vector<Edge> edges;
13
14   MinCostFlow(int n, int m) : inq(n), head(n, -1), dist(n), from(n), flow(n) {
15     edges.reserve(2 * m);
16   }
17
18   void add_edge(int u, int v, int capacity, int cost) {
19     internal_add_edge(u, v, capacity, cost);
20     internal_add_edge(v, u, 0, -cost);
21   }
22
23   void internal_add_edge(int u, int v, int capacity, int cost) {
24     edges.push_back((Edge){v, head[u], capacity, cost});
25     head[u] = edges.size() - 1;
```

```
26    }
27
28    Result augment(int source, int sink) {
29      fill(dist.begin(), dist.end(), INT_MAX);
30      dist[source] = 0;
31      flow[source] = INT_MAX;
32      queue<int> q;
33      q.push(source);
34      while (!q.empty()) {
35        int u = q.front();
36        q.pop();
37        inq[u] = false;
38        for (int it = head[u]; ~it; it = edges[it].next) {
39          auto &e = edges[it];
40          int v = e.to;
41          if (e.rest > 0 && dist[u] + e.cost < dist[v]) {
42            from[v] = it;
43            dist[v] = dist[u] + e.cost;
44            flow[v] = min(e.rest, flow[u]);
45            if (!inq[v]) {
46              q.push(v);
47              inq[v] = true;
48            }
49          }
50        }
51      }
52
53      if (dist[sink] == INT_MAX) return {0, 0};
54      int min_flow = flow[sink];
55      for (int u = sink; u != source; u = edges[from[u] ^ 1].to) {
56        edges[from[u]].rest -= min_flow;
57        edges[from[u] ^ 1].rest += min_flow;
58      }
59      return {min_flow, dist[sink]};
60    }
61
62    Result min_cost_flow(int source, int sink) {
63      int flow = 0, cost = 0;
64      for (;;) {
65        auto result = augment(source, sink);
66        if (!result.flow) break;
67        flow += result.flow, cost += result.cost;
68      }
69      return {flow, cost};
70    }
71 };
```

6   Java

6.1   进制转换

```
1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4
5 /**
6  * Built using CHelper plug-in
7  * Actual solution is at the top
```

```
 8    */
 9    public class Main {
10        public static void main(String[] args) {
11            InputStream inputStream = System.in;
12            OutputStream outputStream = System.out;
13            Scanner in = new Scanner(inputStream);
14            PrintWriter out = new PrintWriter(outputStream);
15            Solver solver = new Solver();
16            int testCount = Integer.parseInt(in.next());
17            for (int i = 1; i <= testCount; i++)
18                solver.solve(i, in, out);
19            out.close();
20        }
21
22        static class Solver {
23            public void solve(int testNumber, Scanner in, PrintWriter out) {
24                int a = in.nextInt();
25                int b = in.nextInt();
26                String num = in.next();
27
28                BigInteger value = BigInteger.ZERO;
29                for (int i = 0; i < num.length(); ++i) {
30                    value = value.multiply(BigInteger.valueOf(a));
31                    value = BigInteger.valueOf(getValue(num.charAt(i))).add(value);
32                }
33                out.println(a + " " + num);
34
35                if (value.equals(BigInteger.ZERO)) {
36                    out.println(b + " 0");
37                    out.println();
38                    return;
39                }
40
41                out.print(b + " ");
42
43                char[] ans = new char[1000];
44                int length = 0;
45                while (!value.equals(BigInteger.ZERO)) {
46                    int digit = value.mod(BigInteger.valueOf(b)).intValue();
47                    value = value.divide(BigInteger.valueOf(b));
48                    ans[length] = getChar(digit);
49                    ++length;
50                }
51
52                for (int i = length - 1; i >= 0; --i) {
53                    out.print(ans[i]);
54                }
55                out.println("\n");
56            }
57
58            private int getValue(char ch) {
59                if (ch >= 'A' && ch <= 'Z') {
60                    return ch - 'A' + 10;
61                }
62                if (ch >= 'a' && ch <= 'z') {
63                    return ch - 'a' + 36;
64                }
65                return ch - '0';
66            }
```

```
67
68          private char getChar(int x) {
69              if (x < 10) {
70                  return (char) ('0' + x);
71              } else if (x < 36) {
72                  return (char) ('A' + x - 10);
73              } else {
74                  return (char) ('a' + x - 36);
75              }
76          }
77
78      }
79 }
```

7 Others

7.1 myalloc

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

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 vimrc

```
1  syntax enable
2  set syntax=on
3  set nobackup
```

```
 4  set noswapfile
 5  set noundofile
 6  set nu
 7  set smartindent
 8  set cindent
 9  set noeb
10  set tabstop=2
11  set softtabstop=2
12  set shiftwidth=2
13  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++17 -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
```

7.4 emacs

```
 1  (defun comp ()
 2    (interactive)
 3    (save-some-buffers t)
 4    (setq filename (file-name-nondirectory buffer-file-name))
 5    (setq progname (file-name-sans-extension filename))
 6    (setq suffix (file-name-extension filename))
 7    (compile (concat "g++ " filename " -o " progname " -O2 -Wall -Werror")))
 8  (add-hook 'c++-mode
 9           '(lambda ()
10               (c-set-style "K&R")
11               (setq tab-width 2)
12               (setq indent-tabs-mode nil)
13               (setq c-basic-offset 2)))
14  (global-set-key [f5] 'comp)
15
16  (ido-mode t)
17  (delete-selection-mode t)
18  (global-auto-revert-mode t)
```

7.5 FastIO

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

```
 6  } _endl(0), _prs(1), _setprecision(2);
 7
 8  const int IO_SIZE = 1 << 16 | 127;
 9
10  struct FastIO {
11    char in[IO_SIZE], *p, *pp, out[IO_SIZE], *q, *qq, ch[20], *t, b, K, prs;
12    FastIO() : p(in), pp(in), q(out), qq(out + IO_SIZE), t(ch), b(1), K(6) {}
13    ~FastIO() { fwrite(out, 1, q - out, stdout); }
14    inline char getc() {
15      return p == pp && (pp = (p = in) + fread(in, 1, IO_SIZE, stdin), p == pp)
16                 ? (b = 0, EOF)
17                 : *p++;
18    }
19    inline void putc(char x) {
20      q == qq && (fwrite(out, 1, q - out, stdout), q = out), *q++ = x;
21    }
22    inline void puts(const char str[]) {
23      fwrite(out, 1, q - out, stdout), fwrite(str, 1, strlen(str), stdout),
24          q = out;
25    }
26    inline void getline(string &s) {
27      s = "";
28      for (char ch; (ch = getc()) != '\n' && b;) s += ch;
29    }
30  #define indef(T)                                                          \
31    inline FastIO &operator>>(T &x) {                                       \
32      x = 0;                                                                \
33      char f = 0, ch;                                                       \
34      while (!isdigit(ch = getc()) && b) f |= ch == '-';                    \
35      while (isdigit(ch)) x = (x << 1) + (x << 3) + (ch ^ 48), ch = getc(); \
36      return x = f ? -x : x, *this;                                         \
37    }
38    indef(int);
39    indef(long long);
40
41    inline FastIO &operator>>(string &s) {
42      s = "";
43      char ch;
44      while (isspace(ch = getc()) && b) {}
45      while (!isspace(ch) && b) s += ch, ch = getc();
46      return *this;
47    }
48    inline FastIO &operator>>(double &x) {
49      x = 0;
50      char f = 0, ch;
51      double d = 0.1;
52      while (!isdigit(ch = getc()) && b) f |= (ch == '-');
53      while (isdigit(ch)) x = x * 10 + (ch ^ 48), ch = getc();
54      if (ch == '.')
55        while (isdigit(ch = getc())) x += d * (ch ^ 48), d *= 0.1;
56      return x = f ? -x : x, *this;
57    }
58  #define outdef(_T)                                                        \
59    inline FastIO &operator<<(_T x) {                                       \
60      !x && (putc('0'), 0), x < 0 && (putc('-'), x = -x);                   \
61      while (x) *t++ = x % 10 + 48, x /= 10;                                \
62      while (t != ch) *q++ = *--t;                                          \
63      return *this;                                                         \
64    }
```

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