Algos @bandiaoz 2022

for ACM

2022

BANDIAOZ

目录

0.1 DataStruct 1
0.1.1 Chtholly.cpp 1
0.1.2 DSU.cpp 4
0.1.3 Fenwick.cpp 5
0.1.4 Hash.cpp 6
0.1.5 LazySegmentTree.cpp 7
0.1.6 Matrix.cpp 10
0.1.7 Mo.cpp 13
0.1.8 NearestPointPair.cpp 14
0.1.9 PointDivideAndConquer1.cpp 17
0.1.10 PointDivideAndConquer2.cpp 20
0.1.11 Segtree.cpp 22
0.1.12 SegtreeNoneRecursive.cpp 24
0.1.13 SparseTable.cpp 28
0.1.14 TheKthFarPointPair.cpp 28
0.1.15 Trie01.cpp 31
0.1.16 dsu_on_tree.cpp 32
0.1.17 fhq-Treap(区间).cpp 34
0.1.18 fhq-Treap.cpp 37
0.2 Geometry 40
0.2.1 Circle.cpp 40
0.2.2 HalfPlane.cpp 46
0.2.3 Line.cpp 48
0.2.4 Point.cpp 51
0.2.5 PolygonAndConvex.cpp 53
0.2.6 Triangle.cpp 58
0.3 Graph 60
0.3.1 2sat.cpp 60
0.3.2 Cut_Point.cpp 61
0.3.3 Graph.cpp 63

0.3.4 HopcroftKarp.cpp . . . 67
0.3.5 MaxAssignment.cpp . . . 68
0.3.6 Mincost.cpp . . . 72
0.3.7 SCC.cpp . . . 75
0.3.8 Tree.cpp . . . 77
0.3.9 dijkstra.cpp . . . 79
0.3.10 dinic.cpp . . . 80
0.3.11 spfa.cpp . . . 82
0.3.12 treeHash.cpp . . . 84
0.3.13 二分图匹配.cpp . . . 85
0.3.14 二分图匹配 _ 匈牙利.cpp . . . 87
0.3.15 二分图带权匹配.cpp . . . 88
0.4 Math . . . 90
0.4.1 Comb.cpp . . . 90
0.4.2 Euler_sieve.cpp . . . 93
0.4.3 Euler_sieve_simple.cpp . . . 94
0.4.4 Frac.cpp . . . 95
0.4.5 Lucas.cpp . . . 96
0.4.6 Matrix.cpp . . . 99
0.4.7 Pollard_Rho.cpp . . . 104
0.4.8 exgcd.cpp . . . 107
0.4.9 xor_basis.cpp . . . 108
0.4.10 容斥.cpp . . . 110
0.4.11 除法分块.cpp . . . 112
0.5 Others . . . 112
0.5.1 BigNum2.cpp . . . 112
0.5.2 Simulated_annealing.cpp . . . 116
0.5.3 Z.cpp . . . 117
0.5.4 bignum.cpp . . . 119
0.5.5 gen.py . . . 124
0.5.6 pai.py . . . 124
0.5.7 威佐夫博弈.cpp . . . 124
0.5.8 杜教 BM.cpp . . . 125
0.6 String . . . 127
0.6.1 AhoCorasick.cpp . . . 127
0.6.2 kmp.cpp . . . 131
0.6.3 manacher.cpp . . . 132
0.6.4 trie.cpp . . . 133

0.7 poly 135
0.7.1 FFT.cpp 135
0.7.2 Lagrange1.cpp 137
0.7.3 Lagrange2.cpp 139

0.1 DataStruct

0.1.1 Chtholly.cpp

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

struct Chtholly {
    struct node {
        int l, r;
        mutable ll v;

        node(int l, int r, ll v) : l(l), r(r), v(v) {}
        int size() const { return r - l; }
        bool operator<(const node &A) const { return l < A.l; }
    };

    set<node> s;
    auto insert(int l, int r, ll v) { return s.insert(node(l, r, v)); }
    // 拆区间，将区间分为 [l,pos), [pos,r) 两段
    auto split(int pos) {
        auto it = s.lower_bound(node(pos, -1, 0));
        if (it != s.end() && it->l == pos) {
            return it;
        }
        --it;
        int L = it->l, R = it->r;
        ll V = it->v;
        s.erase(it);
        insert(L, pos, V);
        // 返回第二个区间的地址
        return insert(pos, R, V).first;
    }
    void add(int l, int r, ll x) {
        for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
            itl->v += x;
        }
    }
    // 区间推平，全部赋值 x
```

```
38      void assign_val(int l, int r, ll x) {
39          // 划分区间，注意顺序，否则会引起 itl 迭代器失效
40          auto itr = split(r), itl = split(l);
41          s.erase(itl, itr);
42          insert(l, r, x);
43      }
44      ll ranks(int l, int r, int k) {  // 区间第 k 小
45          vector<pair<ll, int>> vp;
46          for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
47              vp.push_back({itl->v, itl->size()});
48          }
49          sort(vp.begin(), vp.end());
50          for (auto it : vp) {
51              k -= it.second;
52              if (k <= 0) {
53                  return it.first;
54              }
55          }
56          assert(false);
57          return -1;
58      }
59      // 区间幂次和
60      ll sum(int l, int r, int ex, int mod) {
61          auto powmod = [](ll a, int b, int mod) {
62              ll ans = 1;
63              for (a %= mod; b; b >>= 1, a = a * a % mod) {
64                  if (b & 1) {
65                      ans = ans * a % mod;
66                  }
67              }
68              return ans;
69          };
70
71          ll res = 0;
72          for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
73              res = (res + itl->size() * powmod(itl->v, ex, mod)) % mod;
74          }
75          return res;
76      }
77  };
78
79  const int mod = 1e9 + 7;
80
81  int seed, vmax;
82  int rnd() {
83      int ret = seed;
84      seed = (seed * 7LL + 13) % mod;
85      return ret;
```

```
}

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

    int n, m;
    cin >> n >> m >> seed >> vmax;

    Chtholly cho;
    for (int i = 0; i < n; ++i) {
        int x = rnd() % vmax + 1;
        cho.insert(i, i + 1, x);
    }

    while (m--) {
        int op = rnd() % 4 + 1;

        int l = rnd() % n;
        int r = rnd() % n;
        if (l > r) {
            swap(l, r);
        }
        r++;

        ll x, y;
        if (op == 3) {
            x = rnd() % (r - l) + 1;
        } else {
            x = rnd() % vmax + 1;
        }

        if (op == 4) {
            y = rnd() % vmax + 1;
        }

        if (op == 1) {
            cho.add(l, r, x);
        } else if (op == 2) {
            cho.assign_val(l, r, x);
        } else if (op == 3) {
            cout << cho.ranks(l, r, x) << "\n";
        } else {
            cout << cho.sum(l, r, x, y) << "\n";
        }
    }

    return 0;
```

```
}
```

0.1.2 DSU.cpp

```cpp
#include <bits/stdc++.h>

struct DSU {
    std::vector<int> f, siz;

    DSU() {}
    DSU(int n) {
        init(n);
    }

    void init(int n) {
        f.resize(n);
        std::iota(f.begin(), f.end(), 0);
        siz.assign(n, 1);
    }

    int leader(int x) {
        while (x != f[x]) {
            x = f[x] = f[f[x]];
        }
        return x;
    }

    bool same(int x, int y) {
        return leader(x) == leader(y);
    }

    bool merge(int x, int y) {
        x = leader(x);
        y = leader(y);
        if (x == y) {
            return false;
        }
        siz[x] += siz[y];
        f[y] = x;
        return true;
    }

    int size(int x) {
        return siz[leader(x)];
    }
};
```

0.1.3 Fenwick.cpp

```cpp
#include <bits/stdc++.h>

template <typename T>
struct Fenwick {
    int n;
    std::vector<T> a;

    Fenwick(int n = 0) {
        init(n);
    }
    void init(int n) {
        this->n = n;
        a.assign(n, T());
    }
    void add(int x, T v) {
        for (int i = x + 1; i <= n; i += i & -i) {
            a[i - 1] += v;
        }
    }
    // return the sum of [0, x)
    T sum(int x) {
        auto ans = T();
        for (int i = x; i > 0; i -= i & -i) {
            ans += a[i - 1];
        }
        return ans;
    }
    // return the sum of [l, r)
    T rangeSum(int l, int r) {
        return sum(r) - sum(l);
    }
    int kth(T k) {
        int x = 0;
        for (int i = 1 << std::__lg(n); i; i /= 2) {
            if (x + i <= n && k >= a[x + i - 1]) {
                x += i;
                k -= a[x - 1];
            }
        }
        return x;
    }
};

constexpr int inf = 1E9;

struct Min {
```

```
47      int x = inf;
48      Min &operator+=(Min a) {
49          x = std::min(x, a.x);
50          return *this;
51      }
52  };
```

0.1.4 Hash.cpp

```
1   #include <bits/stdc++.h>
2   #include <bits/extc++.h>
3
4   using namespace std;
5   using namespace __gnu_pbds;
6   using ll = long long;
7
8   // https://codeforces.com/blog/entry/62393
9   struct custom_hash {
10      static uint64_t splitmix64(uint64_t x) {
11          // http://xorshift.di.unimi.it/splitmix64.c
12          x += 0x9e3779b97f4a7c15;
13          x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
14          x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
15          return x ^ (x >> 31);
16      }
17
18      size_t operator()(uint64_t x) const {
19          static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().count();
20          return splitmix64(x + FIXED_RANDOM);
21      }
22  };
23
24  // https://codeforces.com/blog/entry/62393?#comment-464874
25  struct custom_hash_pair {
26          static uint64_t splitmix64(uint64_t x) {
27                  // http://xorshift.di.unimi.it/splitmix64.c
28                  x += 0x9e3779b97f4a7c15;
29                  x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
30                  x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
31                  return x ^ (x >> 31);
32          }
33
34          size_t operator()(pair<uint64_t,uint64_t> x) const {
35                  static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().
                        count();
36                  return splitmix64(x.first + FIXED_RANDOM) ^ (splitmix64(x.second + FIXED_RANDOM) >>
                        1);
```

```
        }
};

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

    unordered_map<ll, int, custom_hash> safe_map;
    gp_hash_table<ll, int, custom_hash> safe_hash_table;

    unordered_map<pair<ll, ll>, int, custom_hash> safe_map_pair;
    gp_hash_table<pair<ll, ll>, int, custom_hash_pair> safe_hash_table_pair;

    return 0;
}
```

0.1.5 LazySegmentTree.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

struct Info {
    ll val;
    Info(ll val = 0) : val(val) {}
    friend Info operator+(const Info &A, const Info &B) {
        return Info(A.val + B.val);
    }
};

void apply(Info &a, ll b, int l, int r) {
    a.val += b * (r - l);
}

void apply(ll &a, ll b, int l, int r) {
    a += b;
}

template<class Info, class Tag, class Merge = plus<Info>>
class LazySegmentTree {
private:
    const int n;
    const Merge merge{};
    vector<Info> info;  // data of segment tree, 1-index
    vector<Tag> tag;  // lazy tag of segment tree

```

```
30      /* [x, y) and val: Add val to each element in range of [x, y)
31       * p: The id of subtree, which is an index of vector 'info'.
32       * [l, r): The range of p.
33       */
34      void innerPull(int p) {
35          info[p] = merge(info[p << 1], info[p << 1 | 1]);
36      }
37      void innerApply(int p, const Tag &v, int l, int r) {
38          ::apply(info[p], v, l, r);
39          ::apply(tag[p], v, l, r);
40      }
41      void push(int p, int l, int r) {
42          if (tag[p] != Tag()) {
43              int m = (l + r) / 2;
44              innerApply(p << 1, tag[p], l, m);
45              innerApply(p << 1 | 1, tag[p], m, r);
46              tag[p] = Tag();
47          }
48      }
49      void innerUpdate(int p, int x, int y, const Tag &v, int l, int r) {
50          if (x <= l && r <= y) {
51              innerApply(p, v, l, r);
52              return;
53          }
54          int m = (l + r) / 2;
55
56          push(p, l, r);
57          if (x < m) innerUpdate(p << 1, x, y, v, l, m);
58          if (y > m) innerUpdate(p << 1 | 1, x, y, v, m, r);
59          innerPull(p);
60      }
61      /* Query the sum-up value of range [x, y). */
62      Info innerQuery(int p, int x, int y, int l, int r) {
63          if (x <= l && r <= y) return info[p];
64          if (x >= r || y <= l) return Info();
65          int m = (l + r) / 2;
66
67          push(p, l, r);
68          return merge(innerQuery(p << 1, x, y, l, m), innerQuery(p << 1 | 1, x, y, m, r));
69      }
70
71  public:
72      LazySegmentTree(int n) : n(n), info(4 << (32 - __builtin_clz(n))), tag(4 << (32 - __builtin_clz(n
            ))) {}
73      LazySegmentTree(vector<Info> &init) : LazySegmentTree(init.size()) {
74          function<void(int, int, int)> innerBuild = [&](int p, int l, int r) {
75              if (r - l == 1) {
76                  info[p] = init[l];
```

```
77                  return;
78              }
79              int m = (l + r) / 2;
80              innerBuild(p << 1, l, m);
81              innerBuild(p << 1 | 1, m, r);
82              innerPull(p);
83          };
84          innerBuild(1, 0, n);
85      }
86      /* Add val to each element in range of [x, y) */
87      void update(int x, int y, Tag v) {
88          innerUpdate(1, x, y, v, 0, n);
89      }
90      /* Query the sum-up value of range [x, y) */
91      Info query(int x, int y) {
92          return innerQuery(1, x, y, 0, n);
93      }
94  };
95
96  int main() {
97      ios::sync_with_stdio(false);
98      cin.tie(nullptr);
99
100     int n, m;
101     cin >> n >> m;
102
103     vector<Info> a(n);
104     for (int i = 0; i < n; ++i) {
105         cin >> a[i].val;
106     }
107
108     LazySegmentTree<Info, ll> seg(a);
109     for (int i = 0; i < m; ++i) {
110         ll op, x, y, k;
111         cin >> op >> x >> y;
112         x--;
113         if (op == 1) {
114             cin >> k;
115             seg.update(x, y, k);
116         } else if (op == 2) {
117             cout << seg.query(x, y).val << "\n";
118         }
119     }
120
121     return 0;
122 }
123 // test problem: https://www.luogu.com.cn/problem/P3372
```

0.1.6 Matrix.cpp

```
/*program from Wolfycz*/
#include<vector>
#include<cstring>
#include<iostream>
using namespace std;
class Matrix {
private:
    int n, m;
    vector<vector<int>> mat;
public:
    Matrix(int n = 2, int m = 2):n(n), m(m) {  //n:row, m:column
        for (int i = 0; i < n; i++) {
            mat.push_back(vector<int>(m, 0));
        }
        vector<vector<int>>(mat).swap(mat); //Remove excess capacity
    }
    Matrix(Matrix&& ots):n(ots.n), m(ots.m), mat(ots.mat) {}    //move construction
    Matrix(const Matrix& ots):n(ots.n), m(ots.m), mat(ots.mat) {}   //copy construction
    Matrix(const vector<vector<int>>& mat):mat(mat) { //construc from vector(Binary)
        vector<vector<int>>(this->mat).swap(this->mat);
        n = this->mat.size(), m = !n ? 0 : this->mat.back().size();
    }
    Matrix& operator=(const Matrix& ots) {
        if (&ots == this) return *this;
        mat = ots.mat;
        n = ots.n, m = ots.m;
        return *this;
    }
    Matrix& operator=(Matrix&& ots) {
        if (&ots == this) return *this;
        mat = ots.mat;
        n = ots.n, m = ots.m;
        return *this;
    }
    Matrix& operator=(const vector<vector<int>>& mat) {   //Assign with vector(Binary)
        this->mat = mat;
        vector<vector<int>>(this->mat).swap(this->mat);
        n = this->mat.size(), m = !n ? 0 : this->mat.back().size();
        return *this;
    }
    void reshape(int n, int m) {    //surplus cut, deficiency fill zero
        vector<vector<int>>newValue;
        for (int i = 0; i < n; i++)
            newValue.push_back(vector<int>(m, 0));
        int Cnt = -1;
        for (int i = 0; i < this->n; i++) {
```

```
47              for (int j = 0; j < this->m; j++) {
48                  if (++Cnt == n * m)
49                      goto reshapeOut;
50                  newValue[Cnt / m][Cnt % m] = mat[i][j];
51              }
52
53          }
54      reshapeOut:
55          this->n = n, this->m = m;
56          mat = newValue;
57          vector<vector<int>>(mat).swap(mat);
58      }
59      void set(int v) {   //fill with v
60          for (int i = 0; i < n; i++)
61              for (int j = 0; j < m; j++)
62                  mat[i][j] = v;
63      }
64      void reset() { set(0); }
65      void clear() { vector<vector<int>>().swap(mat); } //Remove all capacity
66      void Identity() {  // Identity matrix
67          reset();
68          try {
69              if (n != m) {
70                  printf("error throw in I()\n");
71                  throw   string("Matrix is not square");
72              }
73              for (int i = 0; i < n; i++)
74                  mat[i][i] = 1;
75          } catch (string info) {
76              cout << info << endl;
77              exit(0);
78          }
79      }
80      int& at(int x, int y) { return mat[x][y]; }   //Another way to use subscripts
81      void print() {  //Commonly used in debug
82          for (int i = 0; i < n; i++) {
83              for (int j = 0; j < m; j++)
84                  printf("%d ", mat[i][j]);
85              putchar('\n');
86          }
87      }
88      int getn() const { return n; }
89      int getm() const { return m; }
90      friend Matrix operator+(const Matrix& A, const Matrix& B);
91      friend Matrix operator-(const Matrix& A, const Matrix& B);
92      friend Matrix operator*(const Matrix& A, const Matrix& B);  //multiplication cross
93  };
94  Matrix operator+(const Matrix& A, const Matrix& B) {
```

```
95      try {
96          if (A.n != B.n || A.m != B.m) {
97              printf("error throw in +(Matrix)\n");
98              throw string("Shape of the matrixes must be the same");
99          }
100         Matrix C = A;
101         for (int i = 0; i < C.n; i++)
102             for (int j = 0; j < C.m; j++)
103                 C.mat[i][j] += B.mat[i][j];
104         return C;
105     } catch (string info) {
106         cout << info << endl;
107         exit(0);
108     }
109     return Matrix();
110 }
111 Matrix operator-(const Matrix& A, const Matrix& B) {
112     try {
113         if (A.n != B.n || A.m != B.m) {
114             printf("error throw in -(Matrix)\n");
115             throw string("Shape of the matrixes must be the same");
116         }
117         Matrix C = A;
118         for (int i = 0; i < C.n; i++)
119             for (int j = 0; j < C.m; j++)
120                 C.mat[i][j] -= B.mat[i][j];
121         return C;
122     } catch (string info) {
123         cout << info << endl;
124         exit(0);
125     }
126     return Matrix();
127 }
128 Matrix operator*(const Matrix& A, const Matrix& B) {
129     try {
130         if (A.m != B.n) {
131             printf("error throw in *(Matrix)\n");
132             throw string("The columns of the first must be equal to the rows of the second");
133         }
134         Matrix C;
135         for (int i = 0; i < A.n; i++)
136             for (int k = 0; k < A.m; k++)
137                 for (int j = 0; j < B.m; j++)
138                     C.mat[i][j] += A.mat[i][k] * B.mat[k][j];
139         C.n = A.n, C.m = B.m;
140         return C;
141     } catch (string info) {
142         cout << info << endl;
```

```
        exit(0);
    }
    return Matrix();
}
int main() {
    return 0;
}
```

0.1.7 Mo.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

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

    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        a[i]--;
    }

    int q;
    cin >> q;
    vector<int> l(q), r(q);
    for (int i = 0; i < q; ++i) {
        cin >> l[i] >> r[i];
        l[i]--;
    }

    const int B = max(1.0, n / sqrt(q));
    vector<int> p(q);
    iota(p.begin(), p.end(), 0);
    sort(p.begin(), p.end(), [&](int i, int j) {
        if (l[i] / B == l[j] / B) return r[i] < r[j];
        else return l[i] < l[j];
    });

    vector<int> cnt(n);
    int L = 0, R = 0, res = 0;
    auto add = [&](int x, int f) {
        res -= cnt[x] / 2;
```

```
        cnt[x] += f;
        res += cnt[x] / 2;
    };

    vector<int> ans(q);
    for (auto i : p) {
        while (L > l[i]) add(a[--L], 1);
        while (R < r[i]) add(a[R++], 1);
        while (L < l[i]) add(a[L++], -1);
        while (R > r[i]) add(a[--R], -1);
        ans[i] = res;
    }

    for (int i = 0; i < q; ++i) {
        cout << ans[i] << "\n";
    }

    return 0;
}

// https://atcoder.jp/contests/abc242/tasks/abc242_g
```

0.1.8 NearestPointPair.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template<typename T, int K = 2>
struct KDTree {
    KDTree(int n) : n(n), lc(n, -1), rc(n, -1), boundary(n, vector<vector<T>>(K, vector<T>(2))){}
    KDTree(vector<array<T, K>> &st) : KDTree(st.size()) {
        a = st;
        function<int(int, int, int)> innerBuild = [&](int l, int r, int div) {
            if (l >= r) {
                return -1;
            }
            int mid = (l + r) >> 1;
            nth_element(a.begin() + l, a.begin() + mid, a.begin() + r, Cmp(div));
            lc[mid] = innerBuild(l, mid, (div + 1) % K);
            rc[mid] = innerBuild(mid + 1, r, (div + 1) % K);
            maintain(mid);
            return mid;
        };

        innerBuild(0, n, 0);
```

```
24      };
25      void query(int p, T &ans) {
26          innerQuery(0, n, p, ans);
27      }
28  private:
29      const int n;
30      vector<int> lc, rc;
31      vector<vector<vector<T>>> boundary;
32      vector<array<T, K>> a;
33
34      struct Cmp {
35          int div;
36          Cmp(const int &div) : div(div) {}
37          bool operator()(const array<T, K> &A, const array<T, K> &B) {
38              for (int i = 0; i < K; ++i) {
39                  if (A[(i + div) % K] != B[(i + div) % K]) {
40                      return A[(i + div) % K] < B[(i + div) % K];
41                  }
42              }
43              return false;
44          }
45      };
46      bool cmp(const array<T, K> &A, const array<T, K> &B, int div) {
47          Cmp cp(div);
48          return cp(A, B);
49      }
50      template<typename U> U sqr(U x) { return x * x; }
51      T dis(const array<T, K> &A, const array<T, K> &B) {
52          T ans = 0;
53          for (int i = 0; i < K; ++i) {
54              ans += sqr(A[i] - B[i]);
55          }
56          return ans;
57      }
58      void maintain(int i) {
59          for (int j = 0; j < K; ++j) {
60              boundary[i][j][0] = boundary[i][j][1] = a[i][j];
61              if (lc[i] != -1) {
62                  boundary[i][j][0] = min(boundary[i][j][0], boundary[lc[i]][j][0]);
63                  boundary[i][j][1] = max(boundary[i][j][1], boundary[lc[i]][j][1]);
64              }
65              if (rc[i] != -1) {
66                  boundary[i][j][0] = min(boundary[i][j][0], boundary[rc[i]][j][0]);
67                  boundary[i][j][1] = max(boundary[i][j][1], boundary[rc[i]][j][1]);
68              }
69          }
70      }
71      T fmin(int p, int i) { // the minimum distance to this area
```

```
72          // if i == -1, ignore this area when calculating the answer.
73          if (i == -1) {
74              return 1e18;
75          }
76          T ans = 0;
77          for (int j = 0; j < K; ++j) {
78              if (a[p][j] < boundary[i][j][0]) ans += sqr(boundary[i][j][0] - a[p][j]);
79              if (a[p][j] > boundary[i][j][1]) ans += sqr(a[p][j] - boundary[i][j][1]);
80          }
81          return ans;
82      }
83      void innerQuery(int l, int r, int p, T &ans) {
84          if (l >= r) return;
85          int mid = (l + r) >> 1;
86          if (p != mid) {
87              ans = min(ans, dis(a[p], a[mid]));
88          }
89          if (l + 1 == r) return;
90
91          T dl = fmin(p, lc[mid]), dr = fmin(p, rc[mid]);
92          if (dl < ans && dr < ans) {
93              if (dl < dr) {
94                  innerQuery(l, mid, p, ans);
95                  if (dr < ans) {
96                      innerQuery(mid + 1, r, p, ans);
97                  }
98              } else {
99                  innerQuery(mid + 1, r, p, ans);
100                 if (dl < ans) {
101                     innerQuery(l, mid, p, ans);
102                 }
103             }
104         } else if (dl < ans) {
105             innerQuery(l, mid, p, ans);
106         } else if (dr < ans) {
107             innerQuery(mid + 1, r, p, ans);
108         }
109     }
110 };
111
112 int main() {
113     ios::sync_with_stdio(false);
114     cin.tie(nullptr);
115
116     int n;
117     cin >> n;
118
119     vector<array<double, 2>> a(n);
```

```
120     for (int i = 0; i < n; ++i) {
121         cin >> a[i][0] >> a[i][1];
122     }
123 
124     KDTree<double> kdt(a);
125 
126     double ans = 2e18;
127     for (int i = 0; i < n; ++i) {
128         kdt.query(i, ans);
129     }
130 
131     cout << fixed << setprecision(4) << sqrt(ans) << "\n";
132 
133     return 0;
134 }
135 
136 // test problem: https://www.luogu.com.cn/problem/P1429
```

0.1.9 PointDivideAndConquer1.cpp

```
1  #include <bits/stdc++.h>
2  
3  using namespace std;
4  using ll = long long;
5  
6  template <typename T>
7  struct Fenwick {
8      const int n;
9      vector<T> a;
10     Fenwick(int n) : n(n), a(n) {}
11     void add(int x, T v) {
12         for (int i = x + 1; i <= n; i += i & -i) {
13             a[i - 1] += v;
14         }
15     }
16     // return the sum of [0, x)
17     T sum(int x) {
18         T ans = 0;
19         for (int i = x; i > 0; i -= i & -i) {
20             ans += a[i - 1];
21         }
22         return ans;
23     }
24     // return the sum of [l, r)
25     T rangeSum(int l, int r) {
26         return sum(r) - sum(l);
27     }
```

```
};

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

    int n;
    cin >> n;
    vector<vector<pair<int, int>>> g(n);
    vector<int> w(n - 1);
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v >> w[i];
        u--, v--;
        g[u].emplace_back(v, i);
        g[v].emplace_back(u, i);
    }

    int k;
    cin >> k;

    vector<int> sz(n);
    vector<bool> vis(n);
    Fenwick<int> fen(k + 1);
    function<void(int, int, int, int&)> dfs_rt = [&](int u, int f, int tot, int &rt) {
        int maxx = 0;
        sz[u] = 1;
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;
            dfs_rt(v, u, tot, rt);
            sz[u] += sz[v];
            maxx = max(maxx, sz[v]);
        }
        maxx = max(maxx, tot - sz[u]);
        if (maxx * 2 <= tot) {
            rt = u;
        }
    };

    function<void(int, int)> dfs_sz = [&](int u, int f) {
        sz[u] = 1;
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;
            dfs_sz(v, u);
            sz[u] += sz[v];
        }
    };

```

```
76      vector<int> d;
77      function<void(int, int, int)> dfs_dis = [&](int u, int f, int dis) {
78          d.push_back(dis);
79          for (auto [v, j] : g[u]) {
80              if (v == f || vis[v]) continue;
81              dfs_dis(v, u, dis + w[j]);
82          }
83      };
84
85      function<void(int, int, int)> dfs_clear = [&](int u, int f, int dis) {
86          if (dis) fen.add(dis, -1);
87          for (auto [v, j] : g[u]) {
88              if (v == f || vis[v]) continue;
89              dfs_clear(v, u, dis + w[j]);
90          }
91      };
92
93      function<int(int, int)> work = [&](int u, int tot) {
94          int rt = u;
95          dfs_rt(u, -1, tot, rt);
96          dfs_sz(rt, -1);
97          vis[rt] = true;
98
99          int ans = 0;
100         for (auto [v, j] : g[rt]) {
101             if (vis[v]) continue;
102             d.clear();
103             dfs_dis(v, rt, w[j]);
104             for (auto dd : d) {
105                 if (dd <= k) {
106                     ans += fen.sum(k - dd + 1) + 1;
107                 }
108             }
109             for (auto dd : d) {
110                 fen.add(dd, 1);
111             }
112         }
113         dfs_clear(rt, -1, 0);
114         for (auto [v, j] : g[rt]) {
115             if (vis[v]) continue;
116             ans += work(v, sz[v]);
117         }
118         return ans;
119     };
120
121     cout << work(0, n) << "\n";
122
123     return 0;
```

```
}

// test problem: https://www.luogu.com.cn/problem/P4178
```

0.1.10 PointDivideAndConquer2.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

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

    int n, m;
    cin >> n >> m;
    vector<vector<pair<int, int>>> g(n);
    vector<int> w(n);
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v >> w[i];
        u--, v--;
        g[u].emplace_back(v, i);
        g[v].emplace_back(u, i);
    }

    vector<int> ans(m), Q(m);
    for (int i = 0; i < m; ++i) {
        cin >> Q[i];
    }

    vector<int> sz(n);
    vector<bool> vis(n);
    function<void(int, int, int, int&)> dfs_rt = [&](int u, int f, int tot, int &rt) {
        int maxx = 0;
        sz[u] = 1;
        for (auto [v, j] : g[u]) {
            if (v == f || vis[v]) continue;
            dfs_rt(v, u, tot, rt);
            sz[u] += sz[v];
            maxx = max(maxx, sz[v]);
        }
        maxx = max(maxx, tot - sz[u]);
        if (maxx * 2 <= tot) {
            rt = u;
        }
```

```
42      };
43
44      function<void(int, int)> dfs_sz = [&](int u, int f) {
45          sz[u] = 1;
46          for (auto [v, j] : g[u]) {
47              if (v == f || vis[v]) continue;
48              dfs_sz(v, u);
49              sz[u] += sz[v];
50          }
51      };
52
53
54      vector<bool> mpd(10000001);
55      int cnt;
56      vector<int> d(n);
57
58      function<void(int, int, int)> dfs_ans = [&](int u, int f, int dis) {
59          ++cnt;
60          d[u] = dis;
61          for (int i = 0; i < m; ++i) {
62              if (d[u] == Q[i]) {
63                  ans[i] = true;
64              } else if (d[u] < Q[i]) {
65                  ans[i] |= mpd[Q[i] - d[u]];
66              }
67          }
68          for (auto [v, j] : g[u]) {
69              if (v == f || vis[v]) continue;
70              dfs_ans(v, u, dis + w[j]);
71          }
72      };
73
74      function<void(int, int, int)> dfs_dis = [&](int u, int f, int flag) {
75          for (int i = 0; i < m; ++i) {
76              if (d[u] <= Q[i]) {
77                  mpd[d[u]] = (flag == 1);
78              }
79          }
80          for (auto [v, j] : g[u]) {
81              if (v == f || vis[v]) continue;
82              dfs_dis(v, u, flag);
83          }
84      };
85
86
87      function<void(int, int)> work = [&](int u, int tot) {
88          int rt = u;
89          dfs_rt(u, -1, tot, rt);
```

```
90          dfs_sz(rt, -1);
91          vis[rt] = true;
92
93
94          for (auto [v, j] : g[rt]) {
95              if (vis[v]) continue;
96              dfs_ans(v, rt, w[j]);
97              dfs_dis(v, rt, 1);
98          }
99
100         dfs_dis(rt, -1, -1);
101
102         for (auto [v, j] : g[rt]) {
103             if (vis[v]) continue;
104             work(v, sz[v]);
105         }
106     };
107
108     work(0, n);
109
110     for (int i = 0; i < m; ++i) {
111         cout << (ans[i] ? "AYE" : "NAY") << "\n";
112     }
113
114     return 0;
115 }
```

0.1.11 Segtree.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template<class Info, class Merge = plus<Info>>
7  struct SegmentTree {
8      SegmentTree(int n) : n(n), merge(Merge()), info(4 << (32 - __builtin_clz(n))) {}
9      SegmentTree(vector<Info> init) : SegmentTree(init.size()) {
10         function<void(int, int, int)> build = [&](int p, int l, int r) {
11             if (r - l == 1) {
12                 info[p] = init[l];
13                 return;
14             }
15             int mid = (l + r) / 2;
16             build(p << 1, l, mid);
17             build(p << 1 | 1, mid, r);
18             innerPull(p);
```

```
19          };
20          build(1, 0, n);
21      }
22      void modify(int pos, const Info &x) {
23          innerModify(1, 0, n, pos, x);
24      }
25      Info rangeQuery(int l, int r) {
26          return innerRangeQuery(1, 0, n, l, r);
27      }
28
29  private:
30      const int n;
31      const Merge merge;
32      vector<Info> info;
33      void innerPull(int p) {
34          info[p] = merge(info[p << 1], info[p << 1 | 1]);
35      }
36      void innerModify(int p, int l, int r, int pos, const Info &x) {
37          if (r - l == 1) {
38              info[p] = info[p] + x;
39              return;
40          }
41          int mid = (l + r) / 2;
42          if (pos < mid) {
43              innerModify(p << 1, l, mid, pos, x);
44          } else {
45              innerModify(p << 1 | 1, mid, r, pos, x);
46          }
47          innerPull(p);
48      }
49      Info innerRangeQuery(int p, int l, int r, int x, int y) {
50          if (l >= y || r <= x) return Info();
51          if (l >= x && r <= y) return info[p];
52          int mid = (l + r) / 2;
53          return merge(innerRangeQuery(p << 1, l, mid, x, y), innerRangeQuery(p << 1 | 1, mid, r, x, y)
                ;
54      }
55  };
56
57  struct Info {
58      int val;
59      Info(int val = 0) : val(val) {}
60      friend Info operator+(const Info &A, const Info &B) {
61          return Info(A.val + B.val);
62      }
63  };
64
65  int main() {
```

```
66      ios::sync_with_stdio(false);
67      cin.tie(nullptr);
68
69      int n, m;
70      cin >> n >> m;
71      SegmentTree<Info> seg(n);
72      for (int i = 0; i < n; ++i) {
73          int x;
74          cin >> x;
75          seg.modify(i, x);
76      }
77
78      while (m--) {
79          int op, x, y;
80          cin >> op;
81          if (op == 1) {
82              cin >> x >> y;
83              x--;
84              seg.modify(x, y);
85          } else {
86              cin >> x >> y;
87              x--;
88              cout << seg.rangeQuery(x, y).val << "\n";
89          }
90      }
91
92      return 0;
93  }
94
95  // test problem: https://www.luogu.com.cn/problem/P3374
```

0.1.12 SegtreeNoneRecursive.cpp

```
1   // reference: https://atcoder.github.io/ac-library/master/document_en/segtree.html
2
3   #include <bits/stdc++.h>
4
5   using namespace std;
6   using ll = long long;
7
8   constexpr unsigned ceil_lg(int n) {
9       return n == 0 ? 0 : 32 - __builtin_clz(n - 1);
10  }
11  template <typename T> struct Segtree {
12    public:
13      Segtree() : Segtree(0) {}
14      explicit Segtree(int n) : Segtree(vector<typename T::S>(n, T::e())) {}
```

```
15      explicit Segtree(const vector<typename T::S>& a) : _n(int(a.size())) {
16          log = ceil_lg(_n);
17          size = 1 << log;
18          d = vector<typename T::S>(2 * size, T::e());
19          for (int i = 0; i < _n; i++) d[size + i] = a[i];
20          for (int i = size - 1; i >= 1; i--) {
21              update(i);
22          }
23      }
24      void set(int p, typename T::S x) {
25          assert(0 <= p && p < _n);
26          p += size;
27          d[p] = x;
28          for (int i = 1; i <= log; i++) update(p >> i);
29      }
30      typename T::S get(int p) const {
31          assert(0 <= p && p < _n);
32          return d[p + size];
33      }
34      typename T::S query(int l, int r) const {
35          assert(0 <= l && l <= r && r <= _n);
36          typename T::S sml = T::e(), smr = T::e();
37          l += size;
38          r += size;
39          while (l < r) {
40              if (l & 1) sml = T::op(sml, d[l++]);
41              if (r & 1) smr = T::op(d[--r], smr);
42              l >>= 1;
43              r >>= 1;
44          }
45          return T::op(sml, smr);
46      }
47      typename T::S queryAll() const { return d[1]; }
48      template <bool (*f)(typename T::S)> int max_right(int l) const {
49          return max_right(l, [](typename T::S x) { return f(x); });
50      }
51      // r = l or f(op(a[l], ..., a[r - 1])) = true
52      // r = n or f(op(a[l], ..., a[r])) = false
53      template <class F> int max_right(int l, F f) const {
54          assert(0 <= l && l <= _n);
55          assert(f(T::e()));
56          if (l == _n) return _n;
57          l += size;
58          typename T::S sm = T::e();
59          do {
60              while (l % 2 == 0) l >>= 1;
61              if (!f(T::op(sm, d[l]))) {
62                  while (l < size) {
```

```
                    l = (2 * l);
                    if (f(T::op(sm, d[l]))) {
                        sm = T::op(sm, d[l]);
                        l++;
                    }
                }
                return l - size;
            }
            sm = T::op(sm, d[l]);
            l++;
        } while ((l & -l) != l);
        return _n;
    }
    template <bool (*f)(typename T::S)> int min_left(int r) const {
        return min_left(r, [](typename T::S x) { return f(x); });
    }
    // r = l or f(op(a[l], ..., a[r - 1])) = true
    // r = n or f(op(a[l - 1], ..., a[r - 1])) = false
    template <class F> int min_left(int r, F f) const {
        assert(0 <= r && r <= _n);
        assert(f(T::e()));
        if (r == 0) return 0;
        r += size;
        typename T::S sm = T::e();
        do {
            r--;
            while (r > 1 && (r % 2)) r >>= 1;
            if (!f(T::op(d[r], sm))) {
                while (r < size) {
                    r = (2 * r + 1);
                    if (f(T::op(d[r], sm))) {
                        sm = T::op(d[r], sm);
                        r--;
                    }
                }
                return r + 1 - size;
            }
            sm = T::op(d[r], sm);
        } while ((r & -r) != r);
        return 0;
    }
  private:
    int _n, size, log;
    vector<typename T::S> d;
    void update(int k) { d[k] = T::op(d[2 * k], d[2 * k + 1]); }
};

struct SegtreeOP {
```

```
111     using S = int;
112     static S e() { return -1; }
113     static S op(const S &x, const S &y) {
114         return max(x, y);
115     }
116 };
117 
118 int main() {
119     ios::sync_with_stdio(false);
120     cin.tie(nullptr);
121 
122     int n, m;
123     cin >> n >> m;
124     vector<int> a(n);
125     for (int i = 0; i < n; ++i) {
126         cin >> a[i];
127     }
128 
129     Segtree<SegtreeOP> seg(a);
130     for (int i = 0; i < m; ++i) {
131         int op;
132         cin >> op;
133 
134         if (op == 1) {
135             int x, v;
136             cin >> x >> v;
137             x--;
138             seg.set(x, v);
139         } else if (op == 2) {
140             int l, r;
141             cin >> l >> r;
142             l--;
143             cout << seg.query(l, r) << "\n";
144         } else {
145             int x, v;
146             cin >> x >> v;
147             x--;
148             cout << seg.max_right(x, [&](int a) { return a < v; }) + 1 << "\n";
149         }
150     }
151 
152     return 0;
153 }
154 
155 // test problem: https://atcoder.jp/contests/practice2/tasks/practice2_j
```

0.1.13 SparseTable.cpp

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

// usage:
//   auto fun = [&](int i, int j) { return min(i, j); };
//   SparseTable<int, decltype(fun)> st(a, fun);
// or:
//   SparseTable<int> st(a, [&](int i, int j) { return min(i, j); });
// __builtin_clz() : Calculate the number of leading zeros

template <typename T, class F = function<T(const T&, const T&)>>
struct SparseTable {
    int n;
    vector<vector<T>> mat;
    F func;

    SparseTable(const vector<T>& a, const F& f) : func(f) {
        n = static_cast<int>(a.size());
        int max_log = 32 - __builtin_clz(n);
        mat.resize(max_log);
        mat[0] = a;
        for (int j = 1; j < max_log; j++) {
            mat[j].resize(n - (1 << j) + 1);
            for (int i = 0; i <= n - (1 << j); i++) {
                mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
            }
        }
    }

    // return the answer [from, to)
    T get(int from, int to) const {
        assert(0 <= from && from <= to && to <= n);
        int lg = 32 - __builtin_clz(to - from) - 1;
        return func(mat[lg][from], mat[lg][to - (1 << lg)]);
    }
};
```

0.1.14 TheKthFarPointPair.cpp

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;
```

```
5
6   template<typename T, int K = 2>
7   struct KDTree {
8       KDTree(int n) : n(n), lc(n, -1), rc(n, -1), boundary(n, vector<vector<T>>(K, vector<T>(2))){}
9       KDTree(vector<array<T, K>> &st) : KDTree(st.size()) {
10          a = st;
11          function<int(int, int, int)> innerBuild = [&](int l, int r, int div) {
12              if (l >= r) {
13                  return -1;
14              }
15              int mid = (l + r) >> 1;
16              nth_element(a.begin() + l, a.begin() + mid, a.begin() + r, Cmp(div));
17              lc[mid] = innerBuild(l, mid, (div + 1) % K);
18              rc[mid] = innerBuild(mid + 1, r, (div + 1) % K);
19              maintain(mid);
20              return mid;
21          };
22
23          innerBuild(0, n, 0);
24      };
25      T query(int k) {
26          priority_queue<T, vector<T>, greater<T>> q;
27          for (int i = 0; i < k; ++i) q.push(0);
28          for (int i = 0; i < n; ++i) {
29              innerQuery(0, n, i, q);
30          }
31          return q.top();
32      }
33  private:
34      const int n;
35      vector<int> lc, rc;
36      vector<vector<vector<T>>> boundary;
37      vector<array<T, K>> a;
38
39      struct Cmp {
40          int div;
41          Cmp(const int &div) : div(div) {}
42          bool operator()(const array<T, K> &A, const array<T, K> &B) {
43              for (int i = 0; i < K; ++i) {
44                  if (A[(i + div) % K] != B[(i + div) % K]) {
45                      return A[(i + div) % K] < B[(i + div) % K];
46                  }
47              }
48              return false;
49          }
50      };
51      bool cmp(const array<T, K> &A, const array<T, K> &B, int div) {
52          Cmp cp(div);
```

```
53          return cp(A, B);
54      }
55      template<typename U> U sqr(U x) { return x * x; }
56      T dis(const array<T, K> &A, const array<T, K> &B) {
57          T ans = 0;
58          for (int i = 0; i < K; ++i) {
59              ans += sqr(A[i] - B[i]);
60          }
61          return ans;
62      }
63      void maintain(int i) {
64          for (int j = 0; j < K; ++j) {
65              boundary[i][j][0] = boundary[i][j][1] = a[i][j];
66              if (lc[i] != -1) {
67                  boundary[i][j][0] = min(boundary[i][j][0], boundary[lc[i]][j][0]);
68                  boundary[i][j][1] = max(boundary[i][j][1], boundary[lc[i]][j][1]);
69              }
70              if (rc[i] != -1) {
71                  boundary[i][j][0] = min(boundary[i][j][0], boundary[rc[i]][j][0]);
72                  boundary[i][j][1] = max(boundary[i][j][1], boundary[rc[i]][j][1]);
73              }
74          }
75      }
76      T fmax(int p, int i) { // the maximum distance to this area
77          // if i == -1, ignore this area when calculating the answer.
78          if (i == -1) {
79              return 0;
80          }
81          T ans = 0;
82          for (int j = 0; j < K; ++j) {
83              ans += max(sqr(a[p][j] - boundary[i][j][0]), sqr(a[p][j] - boundary[i][j][1]));
84          }
85          return ans;
86      }
87      void innerQuery(int l, int r, int p, priority_queue<T, vector<T>, greater<T>> &q) {
88          if (l >= r) return;
89          int mid = (l + r) >> 1;
90          T tmp = dis(a[p], a[mid]);
91          if (tmp > q.top()) {
92              q.pop();
93              q.push(tmp);
94          }
95          T dl = fmax(p, lc[mid]), dr = fmax(p, rc[mid]);
96          if (dl > q.top() && dr > q.top()) {
97              if (dl > dr) {
98                  innerQuery(l, mid, p, q);
99                  if (dr > q.top()) {
100                     innerQuery(mid + 1, r, p, q);
```

```
                }
            } else {
                innerQuery(mid + 1, r, p, q);
                if (dl > q.top()) {
                    innerQuery(l, mid, p, q);
                }
            }
        } else if (dl > q.top()) {
            innerQuery(l, mid, p, q);
        } else if (dr > q.top()) {
            innerQuery(mid + 1, r, p, q);
        }
    }
};

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

    int n, k;
    cin >> n >> k;

    k *= 2;

    vector<array<ll, 2>> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i][0] >> a[i][1];
    }

    KDTree<ll> kdt(a);

    cout << kdt.query(k) << "\n";

    return 0;
}

// test problem: https://www.luogu.com.cn/problem/P4357
```

0.1.15 Trie01.cpp

```
// 01 Trie find maximal xor sum
template <typename T, int B = 30>
class Trie01 {
    vector<vector<int>> ch_;
    // vector<int> cnt;
    int emptyNode() {
        ch_.push_back(vector<int>(2, -1));
```

```
        // cnt.push_back(0);
        return ch_.size() - 1;
    }

  public:
    Trie01() : { emptyNode(); }
    void insert(T x) {
        for (int i = B, p = 0; i >= 0; --i) {
            int c = x >> i & 1;
            if (ch_[p][c] == -1) {
                ch_[p][c] = emptyNode();
            }
            p = ch_[p][c];
            // cnt[p]++;
        }
    }
    T getMax(T x) {
        T res = 0;
        for (int i = B, p = 0; i >= 0; --i) {
            int c = x >> i & 1;
            if (ch_[p][c ^ 1] != -1) {
                p = ch_[p][c ^ 1];
                res |= 1 << i;
            } else {
                p = ch_[p][c];
            }
        }
        return res;
    }
    T getMin(T x) {
        T res = 0;
        for (int i = B, p = 0; i >= 0; --i) {
            int c = x >> i & 1;
            if (ch_[p][c] != -1) {
                p = ch_[p][c];
            } else {
                p = ch_[p][c ^ 1];
                res |= 1 << i;
            }
        }
        return res;
    }
};
```

0.1.16 dsu_on_tree.cpp

```
#include <bits/stdc++.h>
```

```cpp

using namespace std;
using ll = long long;

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

    int n;
    cin >> n;
    vector<int> a(n);
    vector<vector<int>> g(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        g[u].push_back(v);
        g[v].push_back(u);
    }

    vector<int> fa(n, -1), sz(n, 1);
    function<void(int)> dfs_son = [&](int u) {
        if (u > 0) {
            g[u].erase(find(g[u].begin(), g[u].end(), fa[u]));
        }
        for (auto &v : g[u]) {
            fa[v] = u;
            dfs_son(v);
            sz[u] += sz[v];
            if (sz[v] > sz[g[u][0]]) {
                swap(v, g[u][0]);
            }
        }
    };

    dfs_son(0);

    int flag = -1, maxx = 0;
    vector<int> cnt(n + 1);
    vector<ll> ans(n);
    ll sum = 0;
    function<void(int, int)> count = [&](int u, int val) {
        cnt[a[u]] += val;
        if (cnt[a[u]] > maxx) {
            maxx = cnt[a[u]];
```

```
            sum = a[u];
        } else if (cnt[a[u]] == maxx) {
            sum += a[u];
        }
        for (auto v : g[u]) {
            if (v == flag) continue;
            count(v, val);
        }
    };

    function<void(int, bool)> dfs_dsu = [&](int u, bool keep) {
        // 搞轻儿子及其子树算答案删贡献
        for (auto v : g[u]) {
            if (v == g[u][0]) continue;
            dfs_dsu(v, 0);
        }
        // 搞重儿子及其子树算答案不删贡献
        if (g[u].size()) {
            dfs_dsu(g[u][0], true);
            flag = g[u][0];
        }
        // 暴力统计 u 及其所有轻儿子的贡献合并到刚算出的重儿子信息里
        count(u, 1);
        flag = -1;
        ans[u] = sum;
        // 把需要删除的贡献删一删
        if (!keep) {
            count(u, -1);
            sum = maxx = 0;
        }
    };

    dfs_dsu(0, false);

    for (int i = 0; i < n; ++i) {
        cout << ans[i] << " \n"[i == n - 1];
    }

    return 0;
}

// https://codeforces.com/problemset/problem/600/E
```

0.1.17 fhq-Treap(区间).cpp

```
#include <bits/stdc++.h>

```

```
3  using namespace std;
4  using ll = long long;
5
6  mt19937 rnd(random_device{}());
7  class fhqtreap {
8      struct node {
9          int val, siz, tag;
10         mt19937::result_type rnd;
11         node *ch[2];
12     } *root = nullptr;
13     int size(node *rt) {
14         return rt ? rt->siz : 0;
15     };
16     void maintain(node *rt) {
17         rt->siz = size(rt->ch[0]) + size(rt->ch[1]) + 1;
18     }
19     void spread(node *rt) {
20         if (!rt->tag)
21             return;
22
23         swap(rt->ch[0], rt->ch[1]);
24
25         if (rt->ch[0])
26             rt->ch[0]->tag ^= 1;
27
28         if (rt->ch[1])
29             rt->ch[1]->tag ^= 1;
30
31         rt->tag = 0;
32     }
33     pair<node *, node *> split(node *rt, int x) {
34         if (!rt)
35             return {};
36
37         spread(rt);
38
39         if (size(rt->ch[0]) >= x) {
40             auto [l, r] = split(rt->ch[0], x);
41             rt->ch[0] = r, maintain(rt);
42             return {l, rt};
43         } else {
44             auto [l, r] = split(rt->ch[1], x - size(rt->ch[0]) - 1);
45             rt->ch[1] = l, maintain(rt);
46             return {rt, r};
47         }
48     }
49     node *merge(node *lt, node *rt) {
50         if (!(lt && rt))
```

```
51              return lt ? lt : rt;
52
53          spread(lt), spread(rt);
54
55          if (lt->rnd < rt->rnd) {
56              lt->ch[1] = merge(lt->ch[1], rt), maintain(lt);
57              return lt;
58          } else {
59              rt->ch[0] = merge(lt, rt->ch[0]), maintain(rt);
60              return rt;
61          }
62      }
63      void output(node *rt, ostream &os) {
64          if (!rt)
65              return;
66
67          spread(rt);
68          output(rt->ch[0], os);
69          os << rt->val << ' ';
70          output(rt->ch[1], os);
71      }
72
73  public:
74      void insert(int x) {
75          // auto [p, r] = split(root, x);
76          // auto [l, m] = split(p, x - 1);
77          auto m = new node{x, 1, 0, rnd()};
78          root = merge(root, m);
79      }
80      void reverse(int x, int y) {
81          auto [p, r] = split(root, y);
82          auto [l, m] = split(p, x);
83          m->tag ^= 1;
84          root = merge(merge(l, m), r);
85      }
86      friend ostream &operator<<(ostream &os, fhqtreap &rhs) {
87          return rhs.output(rhs.root, os), os;
88      }
89  };
90
91  int main() {
92      ios::sync_with_stdio(false);
93      cin.tie(nullptr);
94
95      int n, m;
96      cin >> n >> m;
97
98      fhqtreap tree;
```

```
    for (int i = 1; i <= n; ++i) {
        tree.insert(i);
    }

    for (int i = 0; i < m; ++i) {
        int x, y;
        cin >> x >> y;
        x--;

        tree.reverse(x, y);
    }

    cout << tree;

    return 0;
}
```

0.1.18 fhq-Treap.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template<typename key_t>
struct Treap {
    struct Node {
        key_t key;
        int pri;
        int l, r, sz;
        Node(key_t a, int b) : key(a), pri(b), l(-1), r(-1), sz(1) {}
    };

    int root = -1;
    vector<Node> tree;

    // split by key, the key of x treap less than y treap
    array<int, 2> split(int pos, key_t key) {
        if (pos == -1) return {-1, -1};

        if (tree[pos].key <= key) {
            array<int, 2> res = split(tree[pos].r, key);
            tree[pos].r = res[0];
            update(pos);
            return {pos, res[1]};
        } else {
            array<int, 2> res = split(tree[pos].l, key);
```

```
            tree[pos].l = res[1];
            update(pos);
            return {res[0], pos};
        }
    }
    // split by size, the size of x treap equal to sz
    array<int, 2> split_sz(int pos, int sz) {
        if (pos == -1) return {-1, -1};

        if (tree[tree[pos].l].sz + 1 <= sz) {
            array<int, 2> res = split_sz(tree[pos].r, sz - tree[tree[pos].l].sz - 1);
            tree[pos].r = res[0];
            update(pos);
            return {pos, res[1]};
        } else {
            array<int, 2> res = split_sz(tree[pos].l, sz);
            tree[pos].l = res[1];
            update(pos);
            return {res[0], pos};
        }
    }
    // small root heap, the key of x treap less than y treap
    int merge(int x, int y) {
        if (x == -1) return y;
        if (y == -1) return x;

        if (tree[x].pri > tree[y].pri) {
            swap(x, y);
        }

        array<int, 2> res = split(y, tree[x].key);
        tree[x].l = merge(tree[x].l, res[0]);
        tree[x].r = merge(tree[x].r, res[1]);
        update(x);
        return x;
    }
    void update(int pos) {
        tree[pos].sz = tree[tree[pos].l].sz + tree[tree[pos].r].sz + 1;
    }
    int create(key_t key) {
        mt19937 rng((unsigned int) chrono::steady_clock::now().time_since_epoch().count());
        int pri = (int)(rng() & ((1ll << 31) - 1));
        tree.emplace_back(key, pri);
        return (int)tree.size() - 1;
    }
    void insert(int &pos, key_t key) {
        int o = create(key);
        array<int, 2> res = split(pos, key);
```

```
77          pos = merge(merge(res[0], o), res[1]);
78      }
79      // Return rank with power is key
80      int rank(int &pos, key_t key) {
81          array<int, 2> res = split(pos, key - 1);
82          int rk = (res[0] == -1) ? 1 : tree[res[0]].sz + 1;
83          pos = merge(res[0], res[1]);
84          return rk;
85      }
86      // Return the key of the k largest
87      key_t kth(int &pos, int k) {
88          assert(k <= tree[pos].sz);
89          array<int, 2> res1 = split_sz(pos, k);
90          array<int, 2> res2 = split_sz(res1[0], k - 1);
91          key_t key = tree[res2[1]].key;
92          pos = merge(merge(res2[0], res2[1]), res1[1]);
93          return key;
94      }
95      // Delete one node that equal to key
96      void erase(int &pos, key_t key) {
97          array<int, 2> res1 = split(pos, key);
98          array<int, 2> res2 = split(res1[0], key - 1);
99
100         if (res2[1] != -1) {
101             res2[1] = merge(tree[res2[1]].l, tree[res2[1]].r);
102         }
103
104         pos = merge(merge(res2[0], res2[1]), res1[1]);
105     }
106     // Return the precursor of key
107     key_t pre(int &pos, key_t key) {
108         array<int, 2> res = split(pos, key - 1);
109         key_t ans = kth(res[0], tree[res[0]].sz);
110         pos = merge(res[0], res[1]);
111         return ans;
112     }
113     // Return the next of key
114     key_t nxt(int &pos, key_t key) {
115         array<int, 2> res = split(pos, key);
116         int ans = kth(res[1], 1);
117         pos = merge(res[0], res[1]);
118         return ans;
119     }
120
121     void insert(key_t x) { insert(root, x); }
122     void erase(int x) { erase(root, x); }
123     int rank(key_t x) { return rank(root, x); }
124     key_t kth(int x) { return kth(root, x); }
```

```
    key_t pre(key_t x) { return pre(root, x); }
    key_t nxt(key_t x) { return nxt(root, x); }
};

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

    int n;
    cin >> n;

    Treap<int> T;

    for (int i = 1; i <= n; i++) {
        int op, x;
        cin >> op >> x;

        if (op == 1) {
            T.insert(x);
        } else if (op == 2) {
            T.erase(x);
        } else if (op == 3) {
            cout << T.rank(x) << "\n";
        } else if (op == 4) {
            cout << T.kth(x) << "\n";
        } else if (op == 5) {
            cout << T.pre(x) << "\n";
        } else if (op == 6) {
            cout << T.nxt(x) << "\n";
        }
    }

    return 0;
}

// test problem: https://loj.ac/p/104
```

0.2 Geometry

0.2.1 Circle.cpp

```
#include "PolygonAndConvex.cpp"

double sqr(double x) { return x * x; }
double mysqrt(double n) {
    return sqrt(max(0.0, n));
}  // 防止出现 sqrt(-eps) 的情况
```

```
7
8   struct Circle {
9       Point o;
10      double r;
11      Circle(Point o = Point(), double r = 0) : o(o), r(r) {}
12      bool operator==(const Circle &c) { return o == c.o && !sgn(r - c.r); }
13      double area() { return PI * r * r; }
14      double perimeter() { return r * PI * 2; }
15      // 点在圆内，不包含边界
16      bool pointIn(const Point &p) { return sgn((p - o).norm() - r) < 0; }
17      // 判直线和圆相交，包括相切
18      friend int isLineCircleIntersection(Line L, Circle c) {
19          return L.disPointLine(c.o) < c.r + eps;
20      }
21      // 判线段和圆相交，包括端点和相切
22      friend int isSegCircleIntersection(Line L, Circle c) {
23          double t1 = dis(c.o, L.s) - c.r, t2 = dis(c.o, L.t) - c.r;
24          Point t = c.o;
25          if (t1 < eps || t2 < eps) return t1 > -eps || t2 > -eps;
26          t.x += L.s.y - L.t.y;
27          t.y += L.t.x - L.s.x;
28          return det(L.s - t, c.o - t) * det(L.t - t, c.o - t) < eps && L.disPointLine(c.o) < c.r + eps
                ;
29      }
30      // 判圆和圆相交，包括相切
31      friend int isCirCirIntersection(Circle c1, Circle c2) {
32          return dis(c1.o, c2.o) < c1.r + c2.r + eps &&
33                 dis(c1.o, c2.o) > fabs(c1.r - c2.r) - eps;
34      }
35      // 判圆和圆内含
36      friend int isCirCirContain(Circle c1, Circle c2) {
37          return sgn(dis(c1.o, c2.o) + min(c1.r, c2.r) - max(c1.r, c2.r)) <= 0;
38      }
39      // 计算圆上到点 p 最近点，如 p 与圆心重合，返回 p 本身
40      friend Point dotPointCircle(Point p, Circle C) {
41          Point u, v, c = C.o;
42          if (dis(p, c) < eps) return p;
43          u.x = c.x + C.r * fabs(c.x - p.x) / dis(c, p);
44          u.y = c.y + C.r * fabs(c.y - p.y) / dis(c, p) * ((c.x - p.x) * (c.y - p.y) < 0 ? -1 : 1);
45          v.x = c.x - C.r * fabs(c.x - p.x) / dis(c, p);
46          v.y = c.y - C.r * fabs(c.y - p.y) / dis(c, p) * ((c.x - p.x) * (c.y - p.y) < 0 ? -1 : 1);
47          return dis(u, p) < dis(v, p) ? u : v;
48      }
49      // 圆与线段交 用参数方程表示直线: P=A+t*(B-A)，带入圆的方程求解 t
50      friend vector<Point> segCircleIntersection(const Line &l, const Circle &c) {
51          double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
52          double A = dx * dx + dy * dy;
53          double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
```

```
54          double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
55          double delta = B * B - 4 * A * C;
56          vector<Point> res;
57          if (A < eps) return res;
58          if (sgn(delta) >= 0) {  // or delta > -eps ?
59              // 可能需要注意 delta 接近-eps 的情况，所以使用 mysqrt
60              double w1 = (-B - mysqrt(delta)) / (2 * A);
61              double w2 = (-B + mysqrt(delta)) / (2 * A);
62              if (sgn(w1 - 1) <= 0 && sgn(w1) >= 0) {
63                  res.push_back(l.s + w1 * (l.t - l.s));
64              }
65              if (sgn(w2 - 1) <= 0 && sgn(w2) >= 0 && fabs(w1 - w2) > eps) {
66                  res.push_back(l.s + w2 * (l.t - l.s));
67              }
68          }
69          return res;
70      }
71      // 圆与直线交
72      friend vector<Point> lineCircleIntersection(const Line &l, const Circle &c) {
73          double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
74          double A = dx * dx + dy * dy;
75          double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
76          double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
77          double delta = B * B - 4 * A * C;
78          vector<Point> res;
79          if (A < eps) return res;
80          if (sgn(delta) >= 0) {  // or delta > -eps ?
81              double w1 = (-B - mysqrt(delta)) / (2 * A);
82              double w2 = (-B + mysqrt(delta)) / (2 * A);
83              res.push_back(l.s + w1 * (l.t - l.s));
84              if (fabs(w1 - w2) > eps) res.push_back(l.s + w2 * (l.t - l.s));
85          }
86          return res;
87      }
88      // 计算圆与圆的交点 保证圆不重合
89      friend vector<Point> cirCirIntersection(Circle a, Circle b) {
90          Point c1 = a.o;
91          vector<Point> vec;
92          if (dis(a.o, b.o) + eps > a.r + b.r &&
93              dis(a.o, b.o) < fabs(a.r - b.r) + eps)
94              return vec;
95          Line L;
96          double t = (1.0 + (sqr(a.r) - sqr(b.r)) / sqr(dis(a.o, b.o))) / 2;
97          L.s = c1 + (b.o - a.o) * t;
98          L.t.x = L.s.x + a.o.y - b.o.y;
99          L.t.y = L.s.y - a.o.x + b.o.x;
100         return lineCircleIntersection(L, a);
101     }
```

```
102     // 将向量 p 逆时针旋转 angle 角度
103     // 求圆外一点对圆 (o,r) 的切点
104     friend vector<Point> tangentPointCircle(Point poi, Circle C) {
105         Point o = C.o;
106         double r = C.r;
107         vector<Point> vec;
108         double dist = (poi - o).norm();
109         if (dist < r - eps) return vec;
110         if (fabs(dist - r) < eps) {
111             vec.push_back(poi);
112             return vec;
113         }
114         Point res1, res2;
115         double line =
116             sqrt((poi.x - o.x) * (poi.x - o.x) + (poi.y - o.y) * (poi.y - o.y));
117         double angle = acos(r / line);
118         Point unitVector, lin;
119         lin.x = poi.x - o.x;
120         lin.y = poi.y - o.y;
121         unitVector.x = lin.x / sqrt(lin.x * lin.x + lin.y * lin.y) * r;
122         unitVector.y = lin.y / sqrt(lin.x * lin.x + lin.y * lin.y) * r;
123         res1 = rotate(unitVector, -angle) + o;
124         res2 = rotate(unitVector, angle) + o;
125         vec.push_back(res1);
126         vec.push_back(res2);
127         return vec;
128     }
129     // 扇形面积 a->b
130     double sectorArea(const Point &a, const Point &b) const {
131         double theta = atan2(a.y, a.x) - atan2(b.y, b.x);
132         while (theta < 0) theta += 2 * PI;
133         while (theta > 2.0 * PI) theta -= 2 * PI;
134         theta = min(theta, 2.0 * PI - theta);
135         return sgn(det(a, b)) * theta * r * r / 2.0;
136     }
137     // 与线段 AB 的交点计算面积 a->b
138     double areaSegCircle(const Line &L) const {
139         Point a = L.s, b = L.t;
140         vector<Point> p = segCircleIntersection(Line(a, b), *this);
141         bool ina = sgn((a - o).norm() - r) < 0;
142         bool inb = sgn((b - o).norm() - r) < 0;
143         if (ina) {
144             if (inb)
145                 return det(a - o, b - o) / 2;
146             else
147                 return det(a - o, p[0] - o) / 2 + sectorArea(p[0] - o, b - o);
148         } else {
149             if (inb)
```

```
150                 return det(p[0] - o, b - o) / 2 + sectorArea(a - o, p[0] - o);
151             else {
152                 if (p.size() == 2)
153                     return sectorArea(a - o, p[0] - o) +
154                            sectorArea(p[1] - o, b - o) +
155                            det(p[0] - o, p[1] - o) / 2;
156                 else
157                     return sectorArea(a - o, b - o);
158             }
159         }
160     }
161
162     // 圆与多边形交，结果可以尝试 +eps
163     friend double areaPolygonCircle(const Circle &c, const Polygon &a) {
164         int n = a.p.size();
165
166         double ans = 0;
167         for (int i = 0; i < n; ++i) {
168             if (sgn(det(a.p[i] - c.o, a.p[_next(i)] - c.o)) == 0) {
169                 continue;
170             }
171             ans += c.areaSegCircle((a.p[i], a.p[_next(i)]));
172         }
173         return ans;
174     }
175     // 两个圆的公共面积
176     friend double areaCircleCircle(const Circle &A, const Circle &B) {
177         double ans = 0.0;
178         Circle M = (A.r > B.r) ? A : B;
179         Circle N = (A.r > B.r) ? B : A;
180         double D = dis(M.o, N.o);
181         if ((D < M.r + N.r) && (D > M.r - N.r)) {
182             double alpha = 2.0 * acos((M.r * M.r + D * D - N.r * N.r) / (2.0 * M.r * D));
183             double beta = 2.0 * acos((N.r * N.r + D * D - M.r * M.r) / (2.0 * N.r * D));
184             ans = (alpha / (2 * PI)) * M.area() + (beta / (2 * PI)) * N.area() -
185                   0.5 * M.r * M.r * sin(alpha) - 0.5 * N.r * N.r * sin(beta);
186         } else if (D <= M.r - N.r) {
187             ans = N.area();
188         }
189         return ans;
190     }
191
192     // 三点求圆
193     Circle getCircle3(const Point &p0, const Point &p1, const Point &p2) {
194         double a1 = p1.x - p0.x, b1 = p1.y - p0.y, c1 = (a1 * a1 + b1 * b1) / 2;
195         double a2 = p2.x - p0.x, b2 = p2.y - p0.y, c2 = (a2 * a2 + b2 * b2) / 2;
196         double d = a1 * b2 - a2 * b1;
197         Point o(p0.x + (c1 * b2 - c2 * b1) / d, p0.y + (a1 * c2 - a2 * c1) / d);
```

```
198         return Circle(o, (o - p0).norm());
199     }
200     // 直径上两点求圆
201     Circle getCircle2(const Point &p0, const Point &p1) {
202         Point o((p0.x + p1.x) / 2, (p0.y + p1.y) / 2);
203         return Circle(o, (o - p0).norm());
204     }
205     // 最小圆覆盖 用之前可以随机化 random_shuffle
206     Circle minCirCover(vector<Point> &a) {
207         int n = a.size();
208         Circle c(a[0], 0);
209         for (int i = 1; i < n; ++i) {
210             if (!c.pointIn(a[i])) {
211                 c.o = a[i];
212                 c.r = 0;
213                 for (int j = 0; j < i; ++j) {
214                     if (!c.pointIn(a[j])) {
215                         c = getCircle2(a[i], a[j]);
216                         for (int k = 0; k < j; ++k) {
217                             if (!c.pointIn(a[k])) {
218                                 c = getCircle3(a[i], a[j], a[k]);
219                             }
220                         }
221                     }
222                 }
223             }
224         }
225         return c;
226     }
227     // 线段在圆内的长度
228     friend double lengthSegInCircle(Line a, Circle c) {
229         if (c.pointIn(a.s) && c.pointIn(a.t)) return a.norm();
230         vector<Point> vec = segCircleIntersection(a, c);
231         if (vec.size() == 0) return 0;
232         if (vec.size() == 1) {
233             if (c.pointIn(a.s)) return dis(vec[0], a.s);
234             if (c.pointIn(a.t)) return dis(vec[0], a.t);
235             return 0;
236         }
237         return dis(vec[0], vec[1]);
238     }
239     // 多边形在圆内的长度
240     friend double lengthPolygonInCircle(Polygon a, Circle c) {
241         double ans = 0;
242         for (int i = 0; i < a.n; ++i) {
243             Line li;
244             li.s = a.p[i];
245             li.t = a.p[(i + 1) % a.n];
```

```
246             ans += lengthSegInCircle(li, c);
247         }
248         return ans;
249     }
250     // 圆 b 在圆 a 内的长度
251     friend double lengthCircleInCircle(Circle a, Circle b) {
252         if (a.r > b.r && a.r - b.r + eps > dis(a.o, b.o)) return b.perimeter();
253         vector<Point> vec = cirCirIntersection(a, b);
254         if (vec.size() < 2) return 0;
255         // Line l1 = (vec[0], b.o), l2 = (vec[1], b.o);
256         double ans = b.r * arg_3(vec[0], b.o, vec[1]);
257         if (b.r >= a.r || !a.pointIn(b.o)) return b.r * ans;
258         return b.perimeter() - ans;
259     }
260 };
```

0.2.2 HalfPlane.cpp

```
1  #include "PolygonAndConvex.cpp"
2
3  const int inf = 1e9;
4
5  struct HalfPlane : public Line {  // 半平面
6      // ax + by + c <= 0
7      double a, b, c;
8      // s->t 的左侧表示半平面
9      HalfPlane(const Point &s = Point(), const Point &t = Point()) : Line(s, t) {
10         a = t.y - s.y;
11         b = s.x - t.x;
12         c = det(t, s);
13     }
14     HalfPlane(double a, double b, double c) : a(a), b(b), c(c) {}
15     // 求点p带入直线方程的值
16     double calc(const Point &p) const { return p.x * a + p.y * b + c; }
17     // 好像跟lineIntersection一样，那个是4个点计算。这个是用abc与两点进行计算
18     friend Point halfxLine(const HalfPlane &h, const Line &l) {
19         Point res;
20         double t1 = h.calc(l.s), t2 = h.calc(l.t);
21         res.x = (t2 * l.s.x - t1 * l.t.x) / (t2 - t1);
22         res.y = (t2 * l.s.y - t1 * l.t.y) / (t2 - t1);
23         return res;
24     }
25     // 用 abc 进行计算 尚未测试
26     friend Point halfxHalf(const HalfPlane &h1, const HalfPlane &h2) {
27         return Point(
28             (h1.b * h2.c - h1.c * h2.b) / (h1.a * h2.b - h2.a * h1.b) + eps,
29             (h1.a * h2.c - h2.a * h1.c) / (h1.b * h2.a - h1.a * h2.b) + eps);
```

```
30      }
31      // 凸多边形与半平面交(cut)
32      friend Convex halfxConvex(const HalfPlane &h, const Convex &c) {
33          Convex res;
34          for (int i = 0; i < c.n; ++i) {
35              if (h.calc(c.p[i]) < -eps)
36                  res.p.push_back(c.p[i]);
37              else {
38                  int j = i - 1;
39                  if (j < 0) j = c.n - 1;
40                  if (h.calc(c.p[j]) < -eps)
41                      res.p.push_back(halfxLine(h, Line(c.p[j], c.p[i])));
42                  j = i + 1;
43                  if (j == c.n) j = 0;
44                  if (h.calc(c.p[j]) < -eps) {
45                      res.p.push_back(halfxLine(h, Line(c.p[i], c.p[j])));
46                  }
47              }
48          }
49          res.n = res.p.size();
50          return res;
51      }
52      // 点在半平面内
53      friend int satisfy(const Point &p, const HalfPlane &h) {
54          return sgn(det(p - h.s, h.t - h.s)) <= 0;
55      }
56      friend bool operator<(const HalfPlane &h1, const HalfPlane &h2) {
57          int res = sgn(h1.vec().arg() - h2.vec().arg());
58          return res == 0 ? satisfy(h1.s, h2) : res < 0;
59      }
60      // 半平面交出的凸多边形
61      friend Convex halfx(vector<HalfPlane> &v) {
62          sort(v.begin(), v.end());
63          deque<HalfPlane> q;
64          deque<Point> ans;
65          q.push_back(v[0]);
66          for (int i = 1; i < v.size(); ++i) {
67              if (sgn(v[i].vec().arg() - v[i - 1].vec().arg()) == 0) continue;
68              while (ans.size() > 0 && !satisfy(ans.back(), v[i])) {
69                  ans.pop_back();
70                  q.pop_back();
71              }
72              while (ans.size() > 0 && !satisfy(ans.front(), v[i])) {
73                  ans.pop_front();
74                  q.pop_front();
75              }
76              ans.push_back(lineIntersection(q.back(), v[i]));
77              q.push_back(v[i]);
```

```
78          }
79          while (ans.size() > 0 && !satisfy(ans.back(), q.front())) {
80              ans.pop_back();
81              q.pop_back();
82          }
83          while (ans.size() > 0 && !satisfy(ans.front(), q.back())) {
84              ans.pop_front();
85              q.pop_front();
86          }
87          ans.push_back(lineIntersection(q.back(), q.front()));
88          Convex c(ans.size());
89          int i = 0;
90          for (deque<Point>::iterator it = ans.begin(); it != ans.end();
91               ++it, ++i) {
92              c.p[i] = *it;
93          }
94          return c;
95      }
96  };
97  // 多边形的核，逆时针
98  Convex core(const Polygon &a) {
99      Convex res;
100     res.p.push_back(Point(-inf, -inf));
101     res.p.push_back(Point(inf, -inf));
102     res.p.push_back(Point(inf, inf));
103     res.p.push_back(Point(-inf, inf));
104     res.n = 4;
105     for (int i = 0; i < a.n; i++) {
106         res = halfxConvex(HalfPlane(a.p[i], a.p[(i + 1) % a.n]), res);
107     }
108     return res;
109 }
110 // 凸多边形交出的凸多边形
111 Convex convexxConvex(Convex &c1, Convex &c2) {
112     vector<HalfPlane> h;
113     for (int i = 0; i < c1.p.size(); ++i)
114         h.push_back(HalfPlane(c1.p[i], c1.p[(i + 1) % c1.p.size()]));
115     for (int i = 0; i < c2.p.size(); i++)
116         h.push_back(HalfPlane(c2.p[i], c2.p[(i + 1) % c2.p.size()]));
117     return halfx(h);
118 }
```

0.2.3 Line.cpp

```
1  #include "Point.cpp"
2
3  const double PI = acos(-1);
```

```
4  struct Line {
5      int id;
6      Point s, t;
7      Line(const Point &s = Point(), const Point &t = Point()) : s(s), t(t) {}
8
9      Point vec() const { return t - s; }           // 化成矢量
10     double norm() const { return vec().norm(); }  // 线段长度
11     // 点是否在直线上
12     bool pointOnLine(const Point &p) {
13         return sgn(det(p - s, t - s)) == 0;
14     }
15     // 点是否在线段上，含线段端点
16     bool pointOnSeg(const Point &p) {
17         return pointOnLine(p) && sgn(dot(p - s, p - t)) <= 0;
18     }
19     // 点是否在线段上，不含线段端点
20     bool pointOnSegInterval(const Point &p) {
21         return pointOnLine(p) && sgn(dot(p - s, p - t) < 0);
22     }
23     // 点到直线的垂足
24     Point pedalPointLine(const Point &p) {
25         return s + vec() * ((dot(p - s, vec()) / norm()) / norm());
26     }
27     // 点到直线的距离
28     double disPointLine(const Point &p) {
29         return fabs(det(p - s, vec()) / norm());
30     }
31     // 点到线段的距离
32     double disPointSeg(const Point &p) {
33         if (sgn(dot(p - s, t - s)) < 0) return (p - s).norm();
34         if (sgn(dot(p - t, s - t)) < 0) return (p - t).norm();
35         return disPointLine(p);
36     }
37     // 计算点 p 与直线的关系， 返回ONLINE、LEFT、RIGHT 上0 左-1 右1
38     int relation(const Point &p) { return sgn(det(t - s, p - s)); }
39     // 判断 a, b 是否在直线的同侧或者同时在直线上
40     bool sameSide(const Point &a, const Point &b) {
41         return relation(a) == relation(b);
42     }
43     // 二维平面上点 p 关于直线的对称点
44     Point symPoint(const Point &p) {
45         return 2.0 * s - p + 2.0 * (t - s) * dot(p - s, t - s) / ((t.x - s.x) * (t.x - s.x) + (t.y -
               s.y) * (t.y - s.y));
46     }
47     // 判断两直线是否平行
48     friend bool isParallel(const Line &l1, const Line &l2) {
49         return sgn(det(l1.vec(), l2.vec())) == 0;
50     }
```

```
51      // 利用相似三角形对应成比例求两直线的交点
52      friend Point lineIntersection(const Line &l1, const Line &l2) {
53          double s1 = det(l1.s - l2.s, l2.vec());
54          double s2 = det(l1.t - l2.s, l2.vec());
55          return (l1.t * s1 - l1.s * s2) / (s1 - s2);
56      }
57      // 求两直线交点的另一种方法
58      friend Point getLineIntersection(const Line &u, const Line &v) {
59          return u.s + (u.t - u.s) * det(u.s - v.s, v.s - v.t) /
60                           det(u.s - u.t, v.s - v.t);
61      }
62      // 判断直线l1和线段l2是否相交
63      friend bool isLineSegIntersection(Line l1, Line l2) {
64          return l1.relation(l2.s) * l1.relation(l2.t) <= 0;
65      }
66      // 判断线段交, 返回是否有交点
67      friend bool isSegIntersection(Line l1, Line l2) {
68          if (!sgn(det(l2.s - l1.s, l1.vec())) &&
69              !sgn(det(l2.t - l1.t, l1.vec()))) {
70              return l1.pointOnSeg(l2.s) || l1.pointOnSeg(l2.t) ||
71                     l2.pointOnSeg(l1.s) || l2.pointOnSeg(l1.t);
72          }
73          return !l1.sameSide(l2.s, l2.t) && !l2.sameSide(l1.s, l1.t);
74      }
75
76      // 规范相交, 两线段仅有一个非端点处的交点
77      // 判断线段相交, 并求线段交点, 1规范相交, 2相交, 0不交
78      friend int segSegIntersection(Line l1, Line l2, Point &p) {
79          Point a, b, c, d;
80          a = l1.s;
81          b = l1.t;
82          c = l2.s;
83          d = l2.t;
84          double s1, s2, s3, s4;
85          int d1, d2, d3, d4;
86          d1 = sgn(s1 = det(b - a, c - a)); // l1.relation(l2.s);
87          d2 = sgn(s2 = det(b - a, d - a)); // l1.relation(l2.t);
88          d3 = sgn(s3 = det(d - c, a - c)); // l2.relation(l1.s);
89          d4 = sgn(s4 = det(d - c, b - c)); // l2.relation(l1.t);
90
91          // 若规范相交则求交点的代码
92          if (d1 * d2 < 0 && d3 * d4 < 0) {
93              p.x = (c.x * s2 - d.x * s1) / (s2 - s1);
94              p.y = (c.y * s2 - d.y * s1) / (s2 - s1);
95              return 1;
96          }
97
98          // 判断非规范相交
```

```
99          // d1 == 0, 则证明a, b, c三点共线;
100         // 如果sgn(dot(a - c, b - c)) < 0, 则说明点c在点a, b的中间;
101         // 如果sgn(dot(a - c, b - c)) == 0,则说明点c与线段ab的端点a, 或者b重合。
102         // 如果sgn(dot(a - c, b - c)) > 0, 则说明点c在线段ab的外面。
103         if ((d1 == 0 && sgn(dot(a - c, b - c)) <= 0) ||
104             (d2 == 0 && sgn(dot(a - d, b - d)) <= 0) ||
105             (d3 == 0 && sgn(dot(c - a, d - a)) <= 0) ||
106             (d4 == 0 && sgn(dot(c - b, d - b)) <= 0)) {
107             return 2;
108         }
109         return 0;
110     }
111
112     // 直线沿法向量(指向直线逆时针方向, 若需要顺时针则移动 -d) 移动 d 距离
113     friend Line move(const Line &l, const double &d) {
114         Point t = l.vec();
115         t = t / t.norm();
116         t = rotate(t, PI / 2);
117         return Line(l.s + t * d, l.t + t * d);
118     }
119     // 计算线段 l1 到线段 l2 的最短距离
120     friend double disSegSeg(Line &l1, Line &l2) {
121         double d1, d2, d3, d4;
122         if (isSegIntersection(l1, l2))
123             return 0;
124         else {
125             d1 = l2.disPointSeg(l1.s);
126             d2 = l2.disPointSeg(l1.t);
127             d3 = l1.disPointSeg(l2.s);
128             d4 = l1.disPointSeg(l2.t);
129             return min(min(d1, d2), min(d3, d4));
130         }
131     }
132     // 两直线的夹角, 返回[0, PI] 弧度
133     friend double argLineLine(Line l1, Line l2) {
134         Point u = l1.vec();
135         Point v = l2.vec();
136         return acos(dot(u, v) / (u.norm() * v.norm()));
137     }
138 };
```

0.2.4 Point.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
```

```
5
6   const double eps = 1e-8;
7
8   int sgn(double x) { return abs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
9
10  struct Point {  // Point & Vector
11      double x, y;
12      Point(const double &x = 0, const double &y = 0) : x(x), y(y) {}
13
14      friend Point operator+(const Point &a, const Point &b) {
15          return Point(a.x + b.x, a.y + b.y);
16      }
17      friend Point operator-(const Point &a, const Point &b) {
18          return Point(a.x - b.x, a.y - b.y);
19      }
20      friend Point operator*(const double &c, const Point &a) {
21          return Point(c * a.x, c * a.y);
22      }
23      friend Point operator*(const Point &a, const double &c) {
24          return Point(c * a.x, c * a.y);
25      }
26      friend Point operator/(const Point &a, const double &c) {
27          return Point(a.x / c, a.y / c);
28      }
29      friend Point rotate(const Point &v, double theta) {  // 向量逆时针旋转 theta 弧度
30          return Point(v.x * cos(theta) - v.y * sin(theta),
31                       v.x * sin(theta) + v.y * cos(theta));
32      }
33      friend Point rotateAroundPoint(Point &v, Point &p, double theta) {
34          return rotate(v - p, theta) + p;
35      }
36      friend bool operator==(const Point &a, const Point &b) {
37          return !sgn(a.x - b.x) && !sgn(a.y - b.y);
38      }
39      friend bool operator<(const Point &a, const Point &b) {
40          return sgn(a.x - b.x) < 0 || (!sgn(a.x - b.x) && sgn(a.y - b.y) < 0);
41      }
42      // 向量模
43      double norm() { return sqrt(x * x + y * y); }
44      // 向量叉积
45      friend double det(const Point &a, const Point &b) {
46          return a.x * b.y - a.y * b.x;
47      }
48      // 向量点积
49      friend double dot(const Point &a, const Point &b) {
50          return a.x * b.x + a.y * b.y;
51      }
52      // 两点间距离
```

```
53      friend double dis(const Point &a, const Point &b) {
54          return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
55      }
56      friend Point intersection(Point u1, Point u2, Point v1, Point v2) {  // 线段交点，线段有交点才可
            用
57          return u1 + (u2 - u1) * det(u1 - v1, v1 - v2) / det(u1 - u2, v1 - v2);
58      }
59      double arg() { return atan2(y, x); }  // 返回弧度
60      friend double arg_2(Point u, Point v) {
61          return acos(dot(u, v) / (u.norm() * v.norm()));
62      }  // 两向量之间的夹角
63      friend double arg_3(const Point &a, const Point &b, const Point &c) {
64          return arg_2(a - b, c - b);
65      }  //  abc
66  };
```

0.2.5 PolygonAndConvex.cpp

```
1   #include "Line.cpp"
2
3   struct Polygon {
4   #define _next(i) ((i + 1) % n)
5       int n;
6       vector<Point> p;
7
8       Polygon(vector<Point> &v) : p(v) { n = p.size(); }
9       Polygon(int n = 0) : n(n) { p.resize(n); }
10
11      void addPoint(Point &a) {
12          p.push_back(a);
13          n++;
14      }
15      // 多边形周长
16      double perimeter() {
17          double sum = 0;
18          for (int i = 0; i < n; ++i) sum += (p[_next(i)] - p[i]).norm();
19          return sum;
20      }
21      // 多边形面积
22      double area() {
23          double sum = 0;
24          for (int i = 0; i < n; ++i) sum += det(p[i], p[_next(i)]);
25          return fabs(sum) / 2;
26      }  // eps
27      // 判断点与多边形的位置关系 0 外，1 内，2 边上
28      int pointIn(const Point &t) {
29          int num = 0;
```

```
30          for (int i = 0; i < n; i++) {
31              if (Line(p[i], p[_next(i)]).pointOnSeg(t)) return 2;
32              int k = sgn(det(p[_next(i)] - p[i], t - p[i]));
33              int d1 = sgn(p[i].y - t.y);
34              int d2 = sgn(p[_next(i)].y - t.y);
35              if (k > 0 && d1 <= 0 && d2 > 0) num++;
36              if (k < 0 && d2 <= 0 && d1 > 0) num--;
37          }
38          return num % 2;
39      }
40      // 多边形重心
41      Point baryCenter() {
42          Point ans;
43          if (sgn(area()) == 0) return ans;
44          for (int i = 0; i < n; ++i)
45              ans = ans + (p[i] + p[_next(i)]) * det(p[i], p[_next(i)]);
46          return ans / area() / 6 + eps;  // 要加 eps 吗?
47      }
48      // 判断多边形是否为凸多边形 (需要已经排好序)
49      bool isConvex() {  //不允许 3 点共线
50          int s[3] = {1, 1, 1};
51          for (int i = 0; i < n && (s[0] || s[2]) && s[1]; ++i) {
52              s[1 + sgn(det(p[_next(i)] - p[i], p[_next(_next(i))] - p[i]))] = 0;
53          }
54          return (s[0] || s[2]) && s[1];
55      }
56      bool isConvex_3() {  // 允许 3 点共线
57          int s[3] = {1, 1, 1};
58          for (int i = 0; i < n && (s[0] || s[2]); ++i) {
59              s[1 + sgn(det(p[_next(i)] - p[i], p[_next(_next(i))] - p[i]))] = 0;
60          }
61          return (s[0] || s[2]);
62      }
63      // 多边形边界上格点的数量
64      long long borderPointNum() {
65          long long num = 0;
66          for (int i = 0; i < n; ++i) {
67              num += gcd((long long)fabs(p[_next(i)].x - p[i].x),
68                         (long long)fabs(p[_next(i)].y - p[i].y));
69          }
70          return num;
71      }
72      // 多边形内格点数量
73      long long inSidePointNum() {
74          return (long long)(area()) + 1 - borderPointNum() / 2;
75      }
76      // 点 p 在以 l1l2 为对角线的矩形内边界上
77      inline int dotOnlineIn(Point p, Point l1, Point l2) {
```

```
78          return sgn(det(p - l2, l1 - l2)) && (l1.x - p.x) * (l2.x - p.x) < eps &&
79                 (l1.y - p.y) * (l2.y - p.y) < eps;
80      }
81      // 判线段在任意多边形内，顶点按顺时针或逆时针给出，与边界相交返回 1
82      int insidePolygon(Line l) {
83          vector<Point> t;
84          Point tt, l1 = l.s, l2 = l.t;
85          if (!pointIn(l.s) || !pointIn(l.t)) return 0;
86          for (int i = 0; i < n; ++i) {
87              if (l.sameSide(p[i], p[(i + 1) % n]) &&
88                  l.sameSide(p[i], p[(i + 1) % n]))
89                  return 0;
90              else if (dotOnlineIn(l1, p[i], p[(i + 1) % n]))
91                  t.push_back(l1);
92              else if (dotOnlineIn(l2, p[i], p[(i + 1) % n]))
93                  t.push_back(l2);
94              else if (dotOnlineIn(p[i], l1, l2))
95                  t.push_back(p[i]);
96          }
97          for (int i = 0; i < t.size(); ++i) {
98              for (int j = i + 1; j < t.size(); ++j) {
99                  if (!pointIn((t[i] + t[j]) / 2)) return 0;
100             }
101         }
102         return 1;
103     }
104 };
105
106 struct Convex : public Polygon {
107     Convex(int n = 0) : Polygon(n) {}
108     Convex(vector<Point> &a) { // 传入 n 个点构造凸包
109         Convex res(a.size() * 2 + 7);
110         sort(a.begin(), a.end());
111         a.erase(unique(a.begin(), a.end()), a.end());  // 去重点
112         int m = 0;
113         for (int i = 0; i < a.size(); ++i) {
114             // <0 则允许 3 点共线，<=0 则不允许
115             while (m > 1 && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])) <= 0)
116                 m--;
117             res.p[m++] = a[i];
118         }
119         int k = m;
120         for (int i = a.size() - 2; i >= 0; --i) {
121             while (m > k && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])) <= 0) {
122                 m--;
123             }
124             res.p[m++] = a[i];
125         }
```

```
126         if (m > 1) m--;
127         res.p.resize(m);
128         res.n = m;
129         *this = res;
130     }
131
132     // 需要先求凸包，若凸包每条边除端点外都有点，则可唯一确定凸包
133     bool isUnique(vector<Point> &v) {
134         if (sgn(area()) == 0) return 0;
135         for (int i = 0; i < n; ++i) {
136             Line l(p[i], p[_next(i)]);
137             bool flag = 0;
138             for (int j = 0; j < v.size(); ++j) {
139                 if (l.pointOnSegInterval(v[j])) {
140                     flag = 1;
141                     break;
142                 }
143             }
144             if (!flag) return 0;
145         }
146         return 1;
147     }
148     // O(n) 时间内判断点是否在凸包内 包含边
149     bool containon(const Point &a) {
150         for (int sign = 0, i = 0; i < n; ++i) {
151             int x = sgn(det(p[i] - a, p[_next(i)] - a));
152             if (x == 0) continue;  // return 0; // 改成不包含边
153             if (!sign)
154                 sign = x;
155             else if (sign != x)
156                 return 0;
157         }
158         return 1;
159     }
160     // O(logn) 时间内判断点是否在凸包内
161     bool containologn(const Point &a) {
162         Point g = (p[0] + p[n / 3] + p[2.0 * n / 3]) / 3.0;
163         int l = 0, r = n;
164         while (l + 1 < r) {
165             int m = (l + r) >> 1;
166             if (sgn(det(p[l] - g, p[m] - g)) > 0) {
167                 if (sgn(det(p[l] - g, a - g)) >= 0 &&
168                     sgn(det(p[m] - g, a - g)) < 0)
169                     r = m;
170                 else
171                     l = m;
172             } else {
173                 if (sgn(det(p[l] - g, a - g)) < 0 &&
```

```
174                 sgn(det(p[m] - g, a - g)) >= 0)
175                 l = m;
176             else
177                 r = m;
178         }
179     }
180     return sgn(det(p[r % n] - a, p[l] - a)) - 1;
181 }
182 // 最远点对（直径）
183 int fir, sec;  // 最远的两个点对应标号
184 double diameter() {
185     double mx = 0;
186     if (n == 1) {
187         fir = sec = 0;
188         return mx;
189     }
190     for (int i = 0, j = 1; i < n; ++i) {
191         while (sgn(det(p[_next(i)] - p[i], p[j] - p[i]) -
192                    det(p[_next(i)] - p[i], p[_next(j)] - p[i])) < 0) {
193             j = _next(j);
194         }
195         double d = dis(p[i], p[j]);
196         if (d > mx) {
197             mx = d;
198             fir = i;
199             sec = j;
200         }
201         d = dis(p[_next(i)], p[_next(j)]);
202         if (d > mx) {
203             mx = d;
204             fir = _next(i);
205             sec = _next(j);
206         }
207     }
208     return mx;
209 }
210
211 // 凸包是否与直线有交点 O(log(n))，需要 On 的预处理，适合判断与直线集是否有交点
212 vector<double> ang;  // 角度
213 bool isinitangle;
214 int finda(const double &x) {
215     return upper_bound(ang.begin(), ang.end(), x) - ang.begin();
216 }
217 double getAngle(const Point &p) {  // 获取向量角度 [0, 2PI]
218     double res = atan2(p.y, p.x);  // （-PI, PI】
219     //      if (res < 0) res += 2 * pi; //为何不可以
220     if (res < -PI / 2 + eps) res += 2 * PI;  // eps 修正精度
221     return res;
```

```
222     }
223     void initAngle() {
224         for (int i = 0; i < n; ++i) {
225             ang.push_back(getAngle(p[_next(i)] - p[i]));
226         }
227         isinitangle = 1;
228     }
229     bool isxLine(const Line &l) {
230         if (!isinitangle) initAngle();
231         int i = finda(getAngle(l.t - l.s));
232         int j = finda(getAngle(l.s - l.t));
233         if (sgn(det(l.t - l.s, p[i] - l.s) * det(l.t - l.s, p[j] - l.s) >= 0))
234             return 0;
235         return 1;
236     }
237 };
```

0.2.6 Triangle.cpp

```
1  #include "Line.cpp"
2
3  struct Triangle {
4      Triangle(const Point &a, const Point &b, const Point &c)
5          : a(a), b(b), c(c){};
6      Point a, b, c;
7      double getArea() { return det(b - a, c - a) * sin(arg_2(b - c, c - a)); }
8      // 外心
9      Point outCenter() {
10         Line u, v;
11         u.s = (a + b) / 2;
12         u.t.x = u.s.x - a.y + b.y;
13         u.t.y = u.s.y + a.x - b.x;
14         v.s = (a + c) / 2;
15         v.t.x = v.s.x - a.y + c.y;
16         v.t.y = v.s.y + a.x - c.x;
17         return lineIntersection(u, v);
18     }
19     // 內心
20     Point inCenter() {
21         Line u, v;
22         u.s = a;
23         double m = atan2(b.y - a.y, b.x - a.x);
24         double n = atan2(c.y - a.y, c.x - a.x);
25         u.t.x = u.s.x + cos((m + n) / 2);
26         u.t.y = u.s.y + sin((m + n) / 2);
27         v.s = b;
28         m = atan2(a.y - b.y, a.x - b.x);
```

```
29          n = atan2(c.y - b.y, c.x - b.x);
30          v.t.x = v.s.x + cos((m + n) / 2);
31          v.t.y = v.s.y + sin((m + n) / 2);
32          return lineIntersection(u, v);
33      }
34      // 垂心
35      Point perpenCenter() {
36          Line u, v;
37          u.s = c;
38          u.t.x = u.s.x - a.y + b.y;
39          u.t.y = u.s.y + a.x - b.x;
40          v.s = b;
41          v.t.x = v.s.x - a.y + c.y;
42          v.t.y = v.s.y + a.x - c.x;
43          return lineIntersection(u, v);
44      }
45
46      // 重心
47      // 到三角形三顶点距离的平方和最小的点
48      // 三角形内到三边距离之积最大的点
49      Point baryCenter() {
50          Line u((a + b) / 2, c), v((a + c) / 2, b);
51          return lineIntersection(u, v);
52      }
53
54      // 费马点 到三角形三顶点距离之和最小的点
55      Point fermentPoint() {
56          if (arg_3(a, b, c) >= 2 * PI / 3) return b;
57          if (arg_3(b, a, c) >= 2 * PI / 3) return a;
58          if (arg_3(a, c, b) >= 2 * PI / 3) return c;
59          Point ab = (a + b) / 2, ac = (a + c) / 2;
60          Point z1 = sqrt(3.0) * (a - ab), z2 = sqrt(3.0) * (a - ac);
61          z1 = rotate(z1, PI / 2);
62          z2 = rotate(z2, PI / 2);
63          if (arg_2(z1, c - ab) < PI / 2) {
64              z1.x = -z1.x;
65              z1.y = -z1.y;
66          }
67          if (arg_2(z2, b - ac) < PI / 2) {
68              z2.x = -z2.x;
69              z2.y = -z2.y;
70          }
71          return intersection(c, ab + z1, b, ac + z2);
72      }
73      // 模拟退火求费马点
74      Point FermatPoint() {
75          Point u, v;
76          double step = fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) + fabs(c.y);
```

```
        u = (a + b + c) / 3;
        while (step > 1e-10)
            for (int k = 0; k < 10; step /= 2, ++k)
                for (int i = -1; i <= 1; ++i) {
                    for (int j = -1; j <= 1; ++j) {
                        v.x = u.x + step * i;
                        v.y = u.y + step * j;
                        if (dis(u, a) + dis(u, b) + dis(u, c) > dis(v, a) + dis(v, b) + dis(v, c)) {
                            u = v;
                        }
                    }
                }
        return u;
    }
};
```

0.3 Graph

0.3.1 2sat.cpp

```
#include <bits/stdc++.h>

struct TwoSat {
    int n;
    std::vector<std::vector<int>> G;
    std::vector<bool> ans;
    TwoSat(int n) : n(n), G(2 * n), ans(n) {}
    void addClause(int u, bool f, int v, bool g) {
        G[2 * u + !f].push_back(2 * v + g);
        G[2 * v + !g].push_back(2 * u + f);
    }
    bool satisfiable() {
        std::vector<int> id(2 * n, -1), dfn(2 * n, -1), low(2 * n, -1);
        std::vector<int> stk;
        int now = 0, cnt = 0;
        std::function<void(int)> tarjan = [&](int u) {
            stk.push_back(u);
            dfn[u] = low[u] = now++;
            for (auto v : G[u]) {
                if (dfn[v] == -1) {
                    tarjan(v);
                    low[u] = std::min(low[u], low[v]);
                } else if (id[v] == -1) {
                    low[u] = std::min(low[u], dfn[v]);
                }
            }
            if (dfn[u] == low[u]) {
```

```
                int v;
                do {
                    v = stk.back();
                    stk.pop_back();
                    id[v] = cnt;
                } while (v != u);
                ++cnt;
            }
        };
        for (int i = 0; i < 2 * n; ++i) if (dfn[i] == -1) tarjan(i);
        for (int i = 0; i < n; ++i) {
            if (id[2 * i] == id[2 * i + 1]) return false;
            ans[i] = id[2 * i] > id[2 * i + 1];
        }
        return true;
    }
    std::vector<bool> answer() { return ans; }
};
```

0.3.2 Cut_Point.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

/** Modified from:
 * https://oi-wiki.org/graph/cut/
 * If the node is a root node and has at least 2 child nodes,
 * or the node have a child node that `low[v] >= dfn[u]`, can go to father node at most.
 * If you need to cut_edge, just change one thing: `low[v] > dfn[u]`, and you don't need to think
     about the root node.
 * if (low[v] > dfn[u]) {
        isbridge[v] = true;
        ++cnt_bridge;
    }
 * isbridge[x] is true -> (father[u], u) is a bridge.
*/
struct Cut_Point {
    int n;
    vector<bool> is_cut;

    Cut_Point(const vector<vector<int>>& g) : n(g.size()), is_cut(n) {
        int cur = 0;
        vector<int> low(n), dfn(n, -1);
        function<void(int, int)> tarjan = [&](int u, int f) {
            low[u] = dfn[u] = cur++;
```

```
            int child = 0;
            for (auto v : g[u]) {
                if (dfn[v] == -1) {
                    child++;
                    tarjan(v, u);
                    low[u] = min(low[u], low[v]);
                    if (u != f && low[v] >= dfn[u]) {
                        is_cut[u] = true;
                    }
                } else if (v != f) {
                    low[u] = min(low[u], dfn[v]);
                }
            }
            if (u == f && child >= 2) {
                is_cut[u] = true;
            }
        };
        for (int i = 0; i < n; i++) if (dfn[i] == -1) tarjan(i, i);
    }
};

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

    int n, m;
    cin >> n >> m;
    vector<vector<int>> g(n);
    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        g[u].push_back(v);
        g[v].push_back(u);
    }

    Cut_Point cut_point(g);
    cout << accumulate(cut_point.is_cut.begin(), cut_point.is_cut.end(), 0) << "\n";
    for (int i = 0; i < n; ++i) {
        if (cut_point.is_cut[i]) {
            cout << i + 1 << " ";
        }
    }

    return 0;
}
// test problem: https://www.luogu.com.cn/problem/P3388
```

0.3.3 Graph.cpp

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template <typename T>
class graph {
   public:
    struct edge {
        int from;
        int to;
        T cost;
    };

    vector<edge> edges;
    vector<vector<int>> g;
    int n;

    graph(int _n) : n(_n) { g.resize(n); }

    virtual int add(int from, int to, T cost) = 0;
};

template <typename T>
class forest : public graph<T> {
   public:
    using graph<T>::edges;
    using graph<T>::g;
    using graph<T>::n;

    forest(int _n) : graph<T>(_n) {}

    int add(int from, int to, T cost = 1) {
        assert(0 <= from && from < n && 0 <= to && to < n);
        int id = (int)edges.size();
        assert(id < n - 1);
        g[from].push_back(id);
        g[to].push_back(id);
        edges.push_back({from, to, cost});
        return id;
    }
};

template <typename T>
class dfs_forest : public forest<T> {
   public:
```

```
47      using forest<T>::edges;
48      using forest<T>::g;
49      using forest<T>::n;
50
51      vector<int> pv;
52      vector<int> pe;
53      vector<int> order;
54      vector<int> pos;
55      vector<int> end;
56      vector<int> sz;
57      vector<int> root;
58      vector<int> depth;
59      vector<T> dist;
60
61      dfs_forest(int _n) : forest<T>(_n) {}
62
63      void init() {
64          pv = vector<int>(n, -1);
65          pe = vector<int>(n, -1);
66          order.clear();
67          pos = vector<int>(n, -1);
68          end = vector<int>(n, -1);
69          sz = vector<int>(n, 0);
70          root = vector<int>(n, -1);
71          depth = vector<int>(n, -1);
72          dist = vector<T>(n);
73      }
74
75      void clear() {
76          pv.clear();
77          pe.clear();
78          order.clear();
79          pos.clear();
80          end.clear();
81          sz.clear();
82          root.clear();
83          depth.clear();
84          dist.clear();
85      }
86
87    private:
88      void do_dfs(int v) {
89          pos[v] = (int)order.size();
90          order.push_back(v);
91          sz[v] = 1;
92          for (int id : g[v]) {
93              if (id == pe[v]) {
94                  continue;
```

```
95              }
96              auto &e = edges[id];
97              int to = e.from ^ e.to ^ v;
98              depth[to] = depth[v] + 1;
99              dist[to] = dist[v] + e.cost;
100             pv[to] = v;
101             pe[to] = id;
102             root[to] = (root[v] != -1 ? root[v] : to);
103             do_dfs(to);
104             sz[v] += sz[to];
105         }
106         end[v] = (int)order.size() - 1;
107     }
108
109     void do_dfs_from(int v) {
110         depth[v] = 0;
111         dist[v] = T{};
112         root[v] = v;
113         pv[v] = pe[v] = -1;
114         do_dfs(v);
115     }
116
117    public:
118     void dfs(int v, bool clear_order = true) {
119         if (pv.empty()) {
120             init();
121         } else {
122             if (clear_order) {
123                 order.clear();
124             }
125         }
126         do_dfs_from(v);
127     }
128
129     void dfs_all() {
130         init();
131         for (int v = 0; v < n; v++) {
132             if (depth[v] == -1) {
133                 do_dfs_from(v);
134             }
135         }
136         assert((int)order.size() == n);
137     }
138 };
139
140 template <typename T>
141 class lca_forest : public dfs_forest<T> {
142    public:
```

```
    using dfs_forest<T>::edges;
    using dfs_forest<T>::g;
    using dfs_forest<T>::n;
    using dfs_forest<T>::pv;
    using dfs_forest<T>::pos;
    using dfs_forest<T>::end;
    using dfs_forest<T>::depth;

    int h;
    vector<vector<int>> pr;

    lca_forest(int _n) : dfs_forest<T>(_n) {}

    inline void build_lca() {
        assert(!pv.empty());
        int max_depth = 0;
        for (int i = 0; i < n; i++) {
            max_depth = max(max_depth, depth[i]);
        }
        h = 1;
        while ((1 << h) <= max_depth) {
            h++;
        }
        pr.resize(n);
        for (int i = 0; i < n; i++) {
            pr[i].resize(h);
            pr[i][0] = pv[i];
        }
        for (int j = 1; j < h; j++) {
            for (int i = 0; i < n; i++) {
                pr[i][j] = (pr[i][j - 1] == -1 ? -1 : pr[pr[i][j - 1]][j - 1]);
            }
        }
    }

    inline bool anc(int x, int y) {
        return (pos[x] <= pos[y] && end[y] <= end[x]);
    }

    inline int go_up(int x, int up) {
        assert(!pr.empty());
        up = min(up, (1 << h) - 1);
        for (int j = h - 1; j >= 0; j--) {
            if (up & (1 << j)) {
                x = pr[x][j];
                if (x == -1) {
                    break;
                }
```

```
            }
        }
        return x;
    }

    inline int lca(int x, int y) {
        assert(!pr.empty());
        if (anc(x, y)) {
            return x;
        }
        if (anc(y, x)) {
            return y;
        }
        for (int j = h - 1; j >= 0; j--) {
            if (pr[x][j] != -1 && !anc(pr[x][j], y)) {
                x = pr[x][j];
            }
        }
        return pr[x][0];
    }
};
```

0.3.4 HopcroftKarp.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

// O(sqrt(n)*m)
struct HopcroftKarp {
    vector<int> g, l, r;
    int ans;
    HopcroftKarp(int n, int m, const vector<pair<int, int>>& e)
        : g(e.size()), l(n, -1), r(m, -1), ans(0) {
        vector<int> deg(n + 1);
        for (auto& [x, y] : e) deg[x]++;
        for (int i = 1; i <= n; i++) deg[i] += deg[i - 1];
        for (auto& [x, y] : e) g[--deg[x]] = y;

        vector<int> a, p, q(n);
        for (;;) {
            a.assign(n, -1), p.assign(n, -1);
            int t = 0;
            for (int i = 0; i < n; i++)
                if (l[i] == -1) q[t++] = a[i] = p[i] = i;

```

```
            bool match = false;
            for (int i = 0; i < t; i++) {
                int x = q[i];
                if (~l[a[x]]) continue;
                for (int j = deg[x]; j < deg[x + 1]; j++) {
                    int y = g[j];
                    if (r[y] == -1) {
                        while (~y) r[y] = x, swap(l[x], y), x = p[x];
                        match = true, ans++;
                        break;
                    }

                    if (p[r[y]] == -1) q[t++] = y = r[y], p[y] = x, a[y] = a[x];
                }
            }

            if (!match) break;
        }
    }
};

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

    int l, r, m;
    cin >> l >> r >> m;
    vector<pair<int, int>> e(m);
    for (auto& [x, y] : e) {
        cin >> x >> y;
        x--, y--;
    }

    HopcroftKarp hk(l, r, e);
    cout << hk.ans << "\n";

    for (int i = 0; i < l; i++) {
        cout << hk.l[i] + 1 << " \n"[i == l - 1];
    }

    return 0;
}
// test problem: https://uoj.ac/problem/78
```

0.3.5 MaxAssignment.cpp

```
#include <bits/stdc++.h>
```

```
2
3  using namespace std;
4  using ll = long long;
5
6  template<class T>
7  struct MaxAssignment {
8  public:
9      T solve(int nx, int ny, vector<vector<T>> a) {
10         assert(0 <= nx && nx <= ny);
11         assert(int(a.size()) == nx);
12         for (int i = 0; i < nx; ++i) {
13             assert(int(a[i].size()) == ny);
14             for (auto x : a[i])
15                 assert(x >= 0);
16         }
17
18         auto update = [&](int x) {
19             for (int y = 0; y < ny; ++y) {
20                 if (lx[x] + ly[y] - a[x][y] < slack[y]) {
21                     slack[y] = lx[x] + ly[y] - a[x][y];
22                     slackx[y] = x;
23                 }
24             }
25         };
26
27         costs.resize(nx + 1);
28         costs[0] = 0;
29         lx.assign(nx, numeric_limits<T>::max());
30         ly.assign(ny, 0);
31         xy.assign(nx, -1);
32         yx.assign(ny, -1);
33         slackx.resize(ny);
34         for (int cur = 0; cur < nx; ++cur) {
35             queue<int> que;
36             visx.assign(nx, false);
37             visy.assign(ny, false);
38             slack.assign(ny, numeric_limits<T>::max());
39             p.assign(nx, -1);
40
41             for (int x = 0; x < nx; ++x) {
42                 if (xy[x] == -1) {
43                     que.push(x);
44                     visx[x] = true;
45                     update(x);
46                 }
47             }
48
49             int ex, ey;
```

```
            bool found = false;
            while (!found) {
                while (!que.empty() && !found) {
                    auto x = que.front();
                    que.pop();
                    for (int y = 0; y < ny; ++y) {
                        if (a[x][y] == lx[x] + ly[y] && !visy[y]) {
                            if (yx[y] == -1) {
                                ex = x;
                                ey = y;
                                found = true;
                                break;
                            }
                            que.push(yx[y]);
                            p[yx[y]] = x;
                            visy[y] = visx[yx[y]] = true;
                            update(yx[y]);
                        }
                    }
                }
                if (found)
                    break;

                T delta = numeric_limits<T>::max();
                for (int y = 0; y < ny; ++y)
                    if (!visy[y])
                        delta = min(delta, slack[y]);
                for (int x = 0; x < nx; ++x)
                    if (visx[x])
                        lx[x] -= delta;
                for (int y = 0; y < ny; ++y) {
                    if (visy[y]) {
                        ly[y] += delta;
                    } else {
                        slack[y] -= delta;
                    }
                }
                for (int y = 0; y < ny; ++y) {
                    if (!visy[y] && slack[y] == 0) {
                        if (yx[y] == -1) {
                            ex = slackx[y];
                            ey = y;
                            found = true;
                            break;
                        }
                        que.push(yx[y]);
                        p[yx[y]] = slackx[y];
                        visy[y] = visx[yx[y]] = true;
```

```
                        update(yx[y]);
                    }
                }
            }

            costs[cur + 1] = costs[cur];
            for (int x = ex, y = ey, ty; x != -1; x = p[x], y = ty) {
                costs[cur + 1] += a[x][y];
                if (xy[x] != -1)
                    costs[cur + 1] -= a[x][xy[x]];
                ty = xy[x];
                xy[x] = y;
                yx[y] = x;
            }
        }
        return costs[nx];
    }
    vector<int> assignment() {
        return xy;
    }
    pair<vector<T>, vector<T>> labels() {
        return make_pair(lx, ly);
    }
    vector<T> weights() {
        return costs;
    }
private:
    vector<T> lx, ly, slack, costs;
    vector<int> xy, yx, p, slackx;
    vector<bool> visx, visy;
};

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

    int nx, ny;
    cin >> nx >> ny;
    ny = max(nx, ny);

    MaxAssignment<ll> ma;
    vector a(nx, vector<ll>(ny));

    int m;
    cin >> m;
    while (m--) {
        int x, y, w;
        cin >> x >> y >> w;
```

```
146         x--, y--;
147         a[x][y] = w;
148     }
149
150     cout << ma.solve(nx, ny, a) << "\n";
151     auto ans = ma.assignment();
152     for (int i = 0; i < nx; ++i) {
153         cout << (a[i][ans[i]] == 0 ? 0 : ans[i] + 1) << " \n"[i == nx - 1];
154     }
155
156     return 0;
157 }
```

0.3.6 Mincost.cpp

```
1  #include <bits/stdc++.h>
2
3  template <typename cap_t, typename cost_t>
4  struct Mincost {
5      static constexpr cost_t INF = std::numeric_limits<cost_t>::max();
6      int n;
7      struct Edge {
8          int to;
9          cap_t cap;
10         cost_t cost;
11         Edge(int to, cap_t cap, cost_t cost) : to(to), cap(cap), cost(cost) {}
12     };
13     std::vector<Edge> e;
14     std::vector<std::vector<int>> g;
15     std::vector<int> cur, pre;
16     std::vector<bool> vis;
17     std::vector<cost_t> dis;
18     Mincost(int n) : n(n), g(n), vis(n) {}
19     void addEdge(int u, int v, cap_t c, cost_t w) {
20         g[u].push_back(e.size());
21         e.emplace_back(v, c, w);
22         g[v].push_back(e.size());
23         e.emplace_back(u, 0, -w);
24     }
25     bool spfa(int s, int t) {
26         pre.assign(n, -1);
27         dis.assign(n, INF);
28         std::queue<int> que;
29         que.push(s);
30         dis[s] = 0;
31         while (!que.empty()) {
32             int u = que.front();
```

```
33              que.pop();
34              vis[u] = false;
35              for (auto j : g[u]) {
36                  auto [v, c, w] = e[j];
37                  if (c > 0 && dis[v] > dis[u] + w) {
38                      dis[v] = dis[u] + w;
39                      pre[v] = j;
40                      if (!vis[v]) {
41                          que.push(v);
42                          vis[v] = true;
43                      }
44                  }
45              }
46          }
47          return dis[t] != INF;
48      }
49      std::pair<cap_t, cost_t> dfs(int u, int t, cap_t f) {
50          if (u == t) return {f, 0};
51          vis[u] = true;
52          cap_t r = f;
53          cost_t p = 0;
54          for (int &i = cur[u]; i < int(g[u].size()); ++ i) {
55              int j = g[u][i];
56              auto [v, c, w] = e[j];
57              if (!vis[v] && c > 0 && dis[v] == dis[u] + w) {
58                  auto a = dfs(v, t, std::min(c, r));
59                  e[j].cap -= a.first;
60                  e[j ^ 1].cap += a.first;
61                  r -= a.first;
62                  p += a.first * w + a.second;
63                  if (r == 0) break;
64              }
65          }
66          vis[u] = false;
67          return {f - r, p};
68      }
69      void augment(int s, int t, std::pair<cap_t, cost_t> &ans) {
70          int p = t;
71          cap_t _f = INF;
72          while (pre[p] != -1) {
73              _f = min(_f, e[pre[p]].cap);
74              p = e[pre[p] ^ 1].to;
75          }
76          ans.first += _f;
77          ans.second += _f * dis[t];
78          p = t;
79          while(pre[p] != -1) {
80              e[pre[p]].cap -= _f;
```

```
            e[pre[p] ^ 1].cap += _f;
            p = e[pre[p] ^ 1].to;
        }
    }
    // select dfs or augment
    // dfs() can multiple augment
    // augment() can augment a minimum cost flow
    std::pair<cap_t, cost_t> maxFlowMinCost(int s, int t) {
        std::pair<cap_t, cost_t> ans = {0, 0};
        while (spfa(s, t)) {
            cur.assign(n, 0);
            auto res = dfs(s, t, INF);
            ans.first += res.first;
            ans.second += res.second;

            // augment(s, t, ans);
        }
        return ans;
    }
};

using ll = long long;

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

    int n, m;
    std::cin >> n >> m;

    Mincost<ll, ll> flow(n);
    const int source = 0, sink = n - 1;

    for (int i = 0; i < m; ++ i) {
        int u, v;
        ll c, w;
        std::cin >> u >> v >> c >> w;
        u--, v--;
        flow.addEdge(u, v, c, w);
    }

    auto ans = flow.maxFlowMinCost(source, sink);
    std::cout << ans.first << " " << ans.second << "\n";

    return 0;
};

// test problem: https://loj.ac/p/102
```

0.3.7 SCC.cpp

```cpp
/**Note that strictly speaking this is not the tarjan's original algorithm
 * because we use a slightly different definition for lowlink. However this
 * algorithm is still correctly and easier to code.
 * In the tarjan's original algorithm, low means the smallest dfn that can be reached through *at most one reverse edge*,
 * but in this code, definition of low no longer limit at most one reverse edge.
 * before:
 * if (dfn[v] == -1) {
        tarjan(v);
        low[u] = min(low[u], low[v]);
    } else if (color[v] == -1) {
        low[u] = min(low[u], dfn[v]);
    }
  * update:
  * if (dfn[v] == -1) tarjan(v);
    if (color[v] == -1) low[u] = min(low[u], low[v]);
 * See: https://cs.stackexchange.com/questions/96635/tarjans-scc-example-showing-necessity-of-lowlink-definition-and-calculation-r?rq=1
*/
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

/** Modified from:
 * https://github.com/thallium/acm-algorithm-template/blob/master/src/Graph/tarjan_scc.hpp
 * Find strongly connected components of graph g. Components are numbered in *reverse topological order*,
 * starting from 0. It returns the number of components and an array which indicates whichcomponent
 * component each vertex belongs to.
 * We no longer need on_stk array, we can replace `if (on_stk[v])` with `color[v] == -1` when update low array,
 * because if a node have dfn but do not have color, it must on stack.
*/
struct Strongly_Connected_Components {
    int n;
    vector<int> color;
    vector<vector<int>> components;

    Strongly_Connected_Components(const vector<vector<int>>& g) : n(g.size()), color(n, -1) {
        int cur = 0;
        vector<int> low(n), dfn(n, -1), stk;
        function<void(int)> tarjan = [&](int u) {
            low[u] = dfn[u] = cur++;
            stk.push_back(u);
            for (auto v : g[u]) {
```

```
43                 if (dfn[v] == -1) tarjan(v);
44                 if (color[v] == -1) low[u] = min(low[u], low[v]);
45             }
46             if (low[u] == dfn[u]) {
47                 vector<int> component;
48                 int v;
49                 do {
50                     v = stk.back();
51                     stk.pop_back();
52                     color[v] = components.size();
53                     component.push_back(v);
54                 } while (u != v);
55                 components.push_back(component);
56             }
57         };
58         for (int i = 0; i < n; i++) if (dfn[i] == -1) tarjan(i);
59     }
60 };
61 
62 int main() {
63     ios::sync_with_stdio(false);
64     cin.tie(nullptr);
65 
66     int n, m;
67     cin >> n >> m;
68     vector<vector<int>> g(n);
69     for (int i = 0; i < m; ++i) {
70         int u, v;
71         cin >> u >> v;
72         g[u].push_back(v);
73     }
74 
75     Strongly_Connected_Components scc(g);
76     auto ans = scc.components;
77 
78     cout << ans.size() << "\n";
79     for (int i = ans.size() - 1; i >= 0; --i) {
80         cout << ans[i].size() << " ";
81         for (int j = 0; j < ans[i].size(); ++j) {
82             cout << ans[i][j] << " \n"[j == ans[i].size() - 1];
83         }
84     }
85 
86     return 0;
87 }
88 
89 // test problem: https://judge.yosupo.jp/problem/scc
```

0.3.8 Tree.cpp

```cpp
#include <bits/stdc++.h>

struct HLD {
    int n;
    std::vector<int> sz, top, dep, fa, in, out, seq;
    std::vector<std::vector<int>> g;
    int cur;

    HLD() {}
    HLD(int n) {
        init(n);
    }
    void init(int n) {
        this->n = n;
        sz.resize(n);
        top.resize(n);
        dep.resize(n);
        fa.resize(n);
        in.resize(n);
        out.resize(n);
        seq.resize(n);
        cur = 0;
        g.assign(n, {});
    }
    void addEdge(int u, int v) {
        g[u].push_back(v);
        g[v].push_back(u);
    }
    void work(int root = 0) {
        top[root] = root;
        dep[root] = 0;
        fa[root] = -1;
        dfs1(root);
        dfs2(root);
    }
    void dfs1(int u) {
        if (fa[u] != -1) {
            g[u].erase(std::find(g[u].begin(), g[u].end(), fa[u]));
        }

        sz[u] = 1;
        for (auto &v : g[u]) {
            fa[v] = u;
            dep[v] = dep[u] + 1;
            dfs1(v);
            sz[u] += sz[v];
```

```
47              if (sz[v] > sz[g[u][0]]) {
48                  std::swap(v, g[u][0]);
49              }
50          }
51      }
52      void dfs2(int u) {
53          in[u] = cur++;
54          seq[in[u]] = u;
55          for (auto v : g[u]) {
56              top[v] = v == g[u][0] ? top[u] : v;
57              dfs2(v);
58          }
59          out[u] = cur;
60      }
61      int lca(int u, int v) {
62          while (top[u] != top[v]) {
63              if (dep[top[u]] > dep[top[v]]) {
64                  u = fa[top[u]];
65              } else {
66                  v = fa[top[v]];
67              }
68          }
69          return dep[u] < dep[v] ? u : v;
70      }
71
72      int dist(int u, int v) {
73          return dep[u] + dep[v] - 2 * dep[lca(u, v)];
74      }
75
76      int jump(int u, int k) {
77          if (dep[u] < k) return -1;
78
79          int d = dep[u] - k;
80          while (dep[top[u]] > d) {
81              u = fa[top[u]];
82          }
83
84          return seq[in[u] - dep[u] + d];
85      }
86
87      bool isAncester(int u, int v) {
88          return in[u] <= in[v] && in[v] < out[u];
89      }
90
91      int rootedChild(int u, int v) {
92          if (u == v) {
93              return u;
94          }
```

```
95          if (!isAncester(u, v)) {
96              return fa[u];
97          }
98          auto it = std::upper_bound(g[u].begin(), g[u].end(), v, [&](int x, int y) {
99              return in[x] < in[y];
100         }) - 1;
101         return *it;
102     }
103
104     int rootedSize(int u, int v) {
105         if (u == v) {
106             return n;
107         }
108         if (!isAncester(v, u)) {
109             return sz[v];
110         }
111         return n - sz[rootedChild(v, u)];
112     }
113
114     int rootedLca(int a, int b, int c) {
115         return lca(a, b) ^ lca(b, c) ^ lca(c, a);
116     }
117 };
```

0.3.9 dijkstra.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  int main() {
7      ios::sync_with_stdio(false);
8      cin.tie(nullptr);
9
10     int n, m, s;
11     cin >> n >> m >> s; s--;
12     vector<vector<pair<int, int>>> g(n);
13     vector<int> w(m);
14     for (int i = 0; i < m; ++i) {
15         int u, v;
16         cin >> u >> v >> w[i];
17         u--, v--;
18         g[u].emplace_back(v, i);
19     }
20
21     auto dijkstra = [&]() {
```

```
        vector<int> dis(n, -1);
        priority_queue<pair<int, int>> h;
        h.emplace(0, s);
        while (!h.empty()) {
            auto [d, u] = h.top();
            h.pop();
            if (dis[u] != -1) continue;
            dis[u] = -d;
            for (auto [v, j] : g[u]) {
                h.emplace(d - w[j], v);
            }
        }
        return dis;
    };

    auto dis = dijkstra();
    for (int i = 0; i < n; ++i) {
        cout << dis[i] << " \n"[i == n - 1];
    }

    return 0;
}

// test problem: https://www.luogu.com.cn/problem/P4779
```

0.3.10 dinic.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

template<class cap_t>
struct Flow {
    static constexpr cap_t INF = numeric_limits<cap_t>::max();
    int n;
    struct Edge {
        int to;
        cap_t cap;
        Edge(int to, cap_t cap) : to(to), cap(cap) {}
    };
    vector<Edge> e;
    vector<vector<int>> g;
    vector<int> cur, h;
    Flow(int n) : n(n), g(n) {}
    bool bfs(int s, int t) {
        h.assign(n, -1);
```

```
        queue<int> que;
        h[s] = 0;
        que.push(s);
        while (!que.empty()) {
            int u = que.front();
            que.pop();
            for (int j : g[u]) {
                int v = e[j].to;
                cap_t c = e[j].cap;
                if (c > 0 && h[v] == -1) {
                    h[v] = h[u] + 1;
                    if (v == t) return true;
                    que.push(v);
                }
            }
        }
        return false;
    }
    cap_t dfs(int u, int t, cap_t f) {
        if (u == t) return f;
        cap_t r = f;
        for (int &i = cur[u]; i < int(g[u].size()); ++i) {
            int j = g[u][i];
            int v = e[j].to;
            cap_t c = e[j].cap;
            if (c > 0 && h[v] == h[u] + 1) {
                cap_t a = dfs(v, t, min(r, c));
                e[j].cap -= a;
                e[j ^ 1].cap += a;
                r -= a;
                if (r == 0) return f;
            }
        }
        return f - r;
    }
    void addEdge(int u, int v, cap_t c) {
        g[u].push_back(e.size());
        e.emplace_back(v, c);
        g[v].push_back(e.size());
        e.emplace_back(u, 0);
    }
    cap_t maxFlow(int s, int t) {
        cap_t ans = 0;
        while (bfs(s, t)) {
            cur.assign(n, 0);
            ans += dfs(s, t, INF);
        }
        return ans;
```

```
    }
};

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

    int n, m, source, sink;
    cin >> n >> m >> source >> sink;
    source--, sink--;
    Flow<ll> flow(n);
    for (int i = 0; i < m; ++i) {
        int u, v, c;
        cin >> u >> v >> c;
        u--, v--;
        flow.addEdge(u, v, c);
    }

    cout << flow.maxFlow(source, sink) << "\n";

    return 0;
}

// test problem: https://loj.ac/p/101
```

0.3.11 spfa.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

const int inf = 1e9;

void solve() {
    int n, m;
    cin >> n >> m;

    vector<vector<pair<int, int>>> g(n);
    vector<int> w(m);
    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v >> w[i];
        u--, v--;
        g[u].emplace_back(v, i);
        if (w[i] >= 0) {
            g[v].emplace_back(u, i);
```

```
21          }
22      }
23
24      auto spfa = [&](int s) { // true: no negative ring
25          vector<int> dis(n, inf), cnt(n);
26          vector<bool> vis(n);
27          dis[s] = 0;
28          vis[s] = true;
29          queue<int> q;
30          q.push(s);
31
32          while (!q.empty()) {
33              int u = q.front();
34              q.pop();
35              vis[u] = false;
36              for (auto [v, j] : g[u]) {
37                  if (dis[v] > dis[u] + w[j]) {
38                      dis[v] = dis[u] + w[j];
39                      cnt[v] = cnt[u] + 1;
40                      if (cnt[v] >= n) {
41                          return false;
42                      }
43                      if (vis[v] == false) {
44                          q.push(v);
45                          vis[v] = true;
46                      }
47                  }
48              }
49          }
50
51          return true;
52      };
53
54      cout << (spfa(0) ? "NO\n" : "YES\n");
55  }
56
57  int main() {
58      ios::sync_with_stdio(false);
59      cin.tie(nullptr);
60
61      int t;
62      cin >> t;
63
64      while (t--) {
65          solve();
66      }
67
68      return 0;
```

```
}

// test problem: https://www.luogu.com.cn/problem/P3385
```

0.3.12 treeHash.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;
using ull = unsigned long long;

const ull mask = std::chrono::steady_clock::now().time_since_epoch().count();

ull shift(ull x) {
    x ^= mask;
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    x ^= mask;
    return x;
}

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

    int n;
    cin >> n;
    vector<vector<int>> g(n);
    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        g[u].push_back(v);
        g[v].push_back(u);
    }

    set<ull> trees;
    vector<ull> hash(n);
    function<int(int, int)> getHash = [&](int u, int f) {
        hash[u] = 1;
        for (int v : g[u]) {
            if (v == f) continue;
            getHash(v, u);
            hash[u] += shift(hash[v]);
        }
```

```
42          trees.insert(hash[u]);
43          return hash[u];
44      };
45
46      getHash(0, -1);
47      cout << trees.size() << "\n";
48
49      return 0;
50  }
51  // test problem: https://uoj.ac/problem/763
```

0.3.13 二分图匹配.cpp

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4   using ll = long long;
5
6   // HK O(sqrt(n)m)
7   class BipartiteMatching {
8   private:
9       int nl, nr;
10      vector<vector<int>> g;
11      int time;
12      vector<int> matchx, matchy, vis, levelx, levely;
13      bool find_match(int x) {
14          for (auto y : g[x]) {
15              if (levely[y] == levelx[x] + 1 && vis[y] != time) {
16                  vis[y] = time;
17                  if (matchy[y] == -1 || find_match(matchy[y])) {
18                      matchy[y] = x;
19                      matchx[x] = y;
20                      return true;
21                  }
22              }
23          }
24          return false;
25      }
26      bool find_path() {
27          vector<int> q;
28          for (int x = 0; x < nl; ++x) {
29              if (matchx[x] >= 0) {
30                  levelx[x] = 0;
31              } else {
32                  levelx[x] = 1;
33                  q.push_back(x);
34              }
```

```
        }
        bool found = false;
        fill(levely.begin(), levely.end(), 0);
        for (int i = 0; i < q.size(); ++i) {
            int x = q[i];
            for (auto y : g[x]) {
                if (levely[y] == 0) {
                    levely[y] = levelx[x] + 1;
                    if (int z = matchy[y]; z >= 0) {
                        levelx[z] = levely[y] + 1;
                        q.push_back(z);
                    } else {
                        found = true;
                    }
                }
            }
        }
        return found;
    }

public:
    BipartiteMatching(int nl, int nr) : nl(nl), nr(nr), g(nl), time(0) {}
    void addEdge(int x, int y) {
        assert(0 <= x && x < nl && 0 <= y && y < nr);
        g[x].push_back(y);
    }
    pair<int, vector<int>> solve() {
        matchx.resize(nl, -1);
        matchy.resize(nr, -1);
        vis.resize(nr, 0);
        levelx.resize(nl);
        levely.resize(nr);

        int ans = 0;
        time = 0;
        while (find_path()) {
            time++;
            for (int x = 0; x < nl; ++x) {
                if (matchx[x] == -1 && find_match(x)) ans += 1;
            }
        }
        return pair(ans, matchx);
    }
};

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

```
83
84      int nl, nr, m;
85      cin >> nl >> nr >> m;
86      BipartiteMatching graph(nl, nr);
87      for (int i = 0; i < m; ++i) {
88          int x, y;
89          cin >> x >> y;
90          x--, y--;
91          graph.addEdge(x, y);
92      }
93
94      auto [res, match] = graph.solve();
95      cout << res << "\n";
96      for (int i = 0; i < nl; ++i) {
97          cout << match[i] + 1 << " \n"[i == nl - 1];
98      }
99
100     return 0;
101 }
102 // test problem: https://uoj.ac/problem/78
```

0.3.14 二分图匹配 _ 匈牙利.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  // O(n_1 \cdot m + n_2)
7  // if n2 < n1: swap(n1, n2) 可以提高效率
8  int main() {
9      ios::sync_with_stdio(false);
10     cin.tie(nullptr);
11
12     int n1, n2, m;
13     cin >> n1 >> n2 >> m;
14
15     vector<vector<int>> g(n1);
16     while (m--) {
17         int u, v;
18         cin >> u >> v;
19         u--, v--;
20         g[u].push_back(v);
21     }
22
23     int ans = 0;
24     vector<int> matchu(n1, -1), matchv(n2, -1);
```

```
    for (int i = 0; i < n1; ++i) {
        vector<int> vis(n2);

        function<bool(int)> find = [&](int u) {
            for (auto v : g[u]) {
                if (vis[v]) continue;
                vis[v] = true;
                if (matchv[v] == -1 || find(matchv[v])) {
                    matchv[v] = u;
                    matchu[u] = v;
                    return true;
                }
            }
            return false;
        };

        if (find(i)) {
            ans++;
        }
    }

    cout << ans << "\n";
    for (int i = 0; i < n1; ++i) {
        cout << matchu[i] + 1 << " \n"[i == n1 - 1];
    }

    return 0;
}
// test problem: https://uoj.ac/problem/78
```

0.3.15 二分图带权匹配.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

constexpr ll inf = 1e12;

template<class T>
struct KuhnMunkres {
    int n;
    vector<vector<T>> w;
    vector<T> hl, hr;
    vector<int> fl, fr, pre;
    KuhnMunkres(int n) : n(n), w(n, vector<T>(n)),
        hl(n), hr(n), fl(n, -1), fr(n, -1), pre(n) {}
```

```
16
17      vector<int> km() {
18          for (int i = 0; i < n; i++) {
19              hl[i] = *max_element(w[i].begin(), w[i].end());
20          }
21          for (int s = 0; s < n; s++)
22              [&](int s) {
23                  vector<T> slack(n, inf);
24                  vector<int> vl(n), vr(n);
25                  queue<int> q;
26                  q.push(s);
27                  vr[s] = 1;
28                  auto check = [&](int u) {
29                      vl[u] = 1;
30                      if (fl[u] != -1) {
31                          q.push(fl[u]);
32                          vr[fl[u]] = 1;
33                          return 1;
34                      }
35                      while (u != -1) swap(u, fr[fl[u] = pre[u]]);
36                      return 0;
37                  };
38                  while (true) {
39                      while (not q.empty()) {
40                          int u = q.front();
41                          q.pop();
42                          for (int i = 0; i < n; i++) {
43                              T d = hl[i] + hr[u] - w[i][u];
44                              if (not vl[i] and slack[i] >= d) {
45                                  pre[i] = u;
46                                  if (d)
47                                      slack[i] = d;
48                                  else if (not check(i))
49                                      return;
50                              }
51                          }
52                      }
53                      T d = inf;
54                      for (int i = 0; i < n; i++)
55                          if (not vl[i]) d = min(d, slack[i]);
56                      for (int i = 0; i < n; i++) {
57                          if (vl[i]) {
58                              hl[i] += d;
59                          } else {
60                              slack[i] -= d;
61                          }
62                      }
63                      for (int i = 0; i < n; i++) {
```

```
                        if (vr[i]) hr[i] -= d;
                    }
                    for (int i = 0; i < n; i++) {
                        if (not vl[i] and not slack[i] and not check(i)) return;
                    }
                }
            }(s);
        return fl;
    }
};

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

    int n1, n2, m;
    cin >> n1 >> n2 >> m;
    int n = max(n1, n2);
    KuhnMunkres<ll> km(n);
    while (m--) {
        int u, v, c;
        cin >> u >> v >> c;
        u--, v--;
        km.w[u][v] = c;
    }

    auto res = km.km();
    ll ans = 0;
    for (int i = 0; i < n1; i++)
        if (res[i] != -1) ans += km.w[i][res[i]];
    cout << ans << "\n";
    for (int i = 0; i < n1; i++) {
        cout << (km.w[i][res[i]] ? res[i] + 1 : 0) << " \n"[i == n1 - 1];
    }

    return 0;
}
```

0.4 Math

0.4.1 Comb.cpp

```
#include <bits/stdc++.h>

template <typename T, T MOD>
struct ModInt {
    using prod_type = std::conditional_t<std::is_same_v<T, int>, long long, __int128>;
```

```
6     T val;
7     ModInt(const prod_type v = 0) : val(v % MOD) { if (val < 0) val += MOD; };
8     ModInt operator+() const { return ModInt(val); }
9     ModInt operator-() const { return ModInt(MOD - val); }
10    ModInt inv() const {
11        auto a = val, m = MOD, u = 0, v = 1;
12        while (a != 0) {
13            auto t = m / a;
14            m -= t * a;
15            std::swap(a, m);
16            u -= t * v;
17            std::swap(u, v);
18        }
19        assert(m == 1);
20        return u;
21    }
22    ModInt pow(prod_type n) const {
23        auto x = ModInt(1);
24        auto b = *this;
25        while (n > 0) {
26            if (n & 1)
27                x *= b;
28            n >>= 1;
29            b *= b;
30        }
31        return x;
32    }
33    friend ModInt operator+(ModInt lhs, const ModInt &rhs) { return lhs += rhs; }
34    friend ModInt operator-(ModInt lhs, const ModInt &rhs) { return lhs -= rhs; }
35    friend ModInt operator*(ModInt lhs, const ModInt &rhs) { return lhs *= rhs; }
36    friend ModInt operator/(ModInt lhs, const ModInt &rhs) { return lhs /= rhs; }
37    ModInt &operator+=(const ModInt &x) {
38        if ((val += x.val) >= MOD)
39            val -= MOD;
40        return *this;
41    }
42    ModInt &operator-=(const ModInt &x) {
43        if ((val -= x.val) < 0)
44            val += MOD;
45        return *this;
46    }
47    ModInt &operator*=(const ModInt &x) {
48        val = prod_type(val) * x.val % MOD;
49        return *this;
50    }
51    ModInt &operator/=(const ModInt &x) { return *this *= x.inv(); }
52    bool operator==(const ModInt &b) const { return val == b.val; }
53    bool operator!=(const ModInt &b) const { return val != b.val; }
```

```
54      friend std::istream &operator>>(std::istream &is, ModInt &x) noexcept {
55          return is >> x.val;
56      }
57      friend std::ostream &operator<<(std::ostream &os, const ModInt &x) noexcept {
58          return os << x.val;
59      }
60  };
61  using Z = ModInt<int, 1'000'000'007>;
62
63  struct Comb {
64      int n;
65      std::vector<Z> _fac, _invfac, _inv;
66
67      Comb() : n{0}, _fac{1}, _invfac{1}, _inv{0} {}
68      Comb(int n) : Comb() {
69          init(n);
70      }
71
72      void init(int m) {
73          if (m <= n) return;
74          _fac.resize(m + 1);
75          _invfac.resize(m + 1);
76          _inv.resize(m + 1);
77
78          for (int i = n + 1; i <= m; i++) {
79              _fac[i] = _fac[i - 1] * i;
80          }
81          _invfac[m] = _fac[m].inv();
82          for (int i = m; i > n; i--) {
83              _invfac[i - 1] = _invfac[i] * i;
84              _inv[i] = _invfac[i] * _fac[i - 1];
85          }
86          n = m;
87      }
88
89      Z fac(int m) {
90          if (m > n) init(2 * m);
91          return _fac[m];
92      }
93      Z invfac(int m) {
94          if (m > n) init(2 * m);
95          return _invfac[m];
96      }
97      Z inv(int m) {
98          if (m > n) init(2 * m);
99          return _inv[m];
100     }
101     Z binom(int n, int m) {
```

```
102         if (n < m || m < 0) return 0;
103         return fac(n) * invfac(m) * invfac(n - m);
104     }
105 } comb;
```

0.4.2 Euler_sieve.cpp

```
1   #include <bits/stdc++.h>
2
3   struct EluerSieve {
4       const int N;
5       std::vector<int> minp, num, d, phi, primes;
6
7       // minp[i] is the minimum prime factor of i
8       // d[i] is the number of factors of i
9       // num[i] is the number of minimun prime factors of i
10      EluerSieve(int n) : N(n), minp(n + 1), num(n + 1), d(n + 1), phi(n + 1) {
11          phi[1] = 1;
12          d[1] = 1;
13          for (int i = 2; i <= N; ++i) {
14              if (!minp[i]) {
15                  minp[i] = i;
16                  num[i] = 1;
17                  d[i] = 2;
18                  phi[i] = i - 1;
19                  primes.push_back(i);
20              }
21              for (auto p : primes) {
22                  if (i * p > n) break;
23
24                  minp[i * p] = p;
25                  if (i % p == 0) {
26                      num[i * p] = num[i] + 1;
27                      d[i * p] = d[i] / num[i * p] * (num[i * p] + 1);
28                      phi[i * p] = phi[i] * p;
29                      break;
30                  } else {
31                      num[i * p] = 1;
32                      d[i * p] = d[i] * 2;
33                      phi[i * p] = phi[i] * phi[p];
34                  }
35              }
36          }
37      }
38      int euler_phi(int n) {
39          int ans = n;
40          for (int i = 2; i * i <= n; i++)
```

```
            if (n % i == 0) {
                ans = ans / i * (i - 1);
                while (n % i == 0) n /= i;
            }
        if (n > 1) ans = ans / n * (n - 1);
        return ans;
    }
    std::vector<std::pair<int, int>> factor(int n) {
        std::vector<std::pair<int, int>> factors;
        while (n > 1) {
            int p = minp[n], cnt = 0;
            while (n % p == 0) {
                cnt++;
                n /= p;
            }
            factors.emplace_back(p, cnt);
        }
        return factors;
    };
};
```

0.4.3 Euler_sieve_simple.cpp

```
#include <bits/stdc++.h>

struct EluerSieveSimple {
    const int N;
    std::vector<int> minp, primes;

    EluerSieveSimple(int n) : N(n), minp(n + 1) {
        for (int i = 2; i <= N; ++i) {
            if (!minp[i]) {
                minp[i] = i;
                primes.push_back(i);
            }
            for (auto p : primes) {
                if (i * p > n) break;

                minp[i * p] = p;
                if (i % p == 0) break;
            }
        }
    }
    std::vector<std::pair<int, int>> factor(int n) {
        std::vector<std::pair<int, int>> factors;
        while (n > 1) {
            int p = minp[n], cnt = 0;
```

```
            while (n % p == 0) {
                cnt++;
                n /= p;
            }
            factors.emplace_back(p, cnt);
        }
        return factors;
    };
};
```

0.4.4 Frac.cpp

```
template<class T>
class Frac {
public:
    T num, den;
    Frac(T num, T den) : num(num), den(den) {
        if (den < 0) {
            den = -den;
            num = -num;
        }
    }
    Frac() : Frac(0, 1) {}
    Frac(T num) : Frac(num, 1) {}
    double toDouble() const {
        return 1.0 * num / den;
    }
    Frac &operator+=(const Frac &rhs) {
        num = num * rhs.den + rhs.num * den;
        den *= rhs.den;
        return *this;
    }
    Frac &operator-=(const Frac &rhs) {
        num = num * rhs.den - rhs.num * den;
        den *= rhs.den;
        return *this;
    }
    Frac &operator*=(const Frac &rhs) {
        num *= rhs.num;
        den *= rhs.den;
        return *this;
    }
    Frac &operator/=(const Frac &rhs) {
        num *= rhs.den;
        den *= rhs.num;
        if (den < 0) {
            num = -num;
```

```
            den = -den;
        }
        return *this;
    }
    friend Frac operator+(Frac lhs, const Frac &rhs) {
        return lhs += rhs;
    }
    friend Frac operator-(Frac lhs, const Frac &rhs) {
        return lhs -= rhs;
    }
    friend Frac operator*(Frac lhs, const Frac &rhs) {
        return lhs *= rhs;
    }
    friend Frac operator/(Frac lhs, const Frac &rhs) {
        return lhs /= rhs;
    }
    friend Frac operator-(const Frac &a) {
        return Frac(-a.num, a.den);
    }
    friend bool operator==(const Frac &lhs, const Frac &rhs) {
        return lhs.num * rhs.den == rhs.num * lhs.den;
    }
    friend bool operator!=(const Frac &lhs, const Frac &rhs) {
        return lhs.num * rhs.den != rhs.num * lhs.den;
    }
    friend bool operator<(const Frac &lhs, const Frac &rhs) {
        return lhs.num * rhs.den < rhs.num * lhs.den;
    }
    friend bool operator>(const Frac &lhs, const Frac &rhs) {
        return lhs.num * rhs.den > rhs.num * lhs.den;
    }
    friend bool operator<=(const Frac &lhs, const Frac &rhs) {
        return lhs.num * rhs.den <= rhs.num * lhs.den;
    }
    friend bool operator>=(const Frac &lhs, const Frac &rhs) {
        return lhs.num * rhs.den >= rhs.num * lhs.den;
    }
};
```

0.4.5 Lucas.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

int P = 1e9 + 7;
```

```
7  // assume -P <= x < P
8  int norm(int x) {
9      if (x < 0) x += P;
10     if (x >= P) x -= P;
11     return x;
12 }
13 template<class T>
14 T power(T a, ll b) {
15     T res = 1;
16     for (; b; b /= 2, a *= a) {
17         if (b % 2) res *= a;
18     }
19     return res;
20 }
21 struct Z {
22     int x;
23     Z(int x = 0) : x(norm(x)) {}
24     Z(int64_t x) : x(x % P) {}
25     int val() const {
26         return x;
27     }
28     Z operator-() const {
29         return Z(norm(P - x));
30     }
31     Z inv() const {
32         assert(x != 0);
33         return power(*this, P - 2);
34     }
35     Z &operator*=(const Z &rhs) {
36         x = int64_t(x) * rhs.x % P;
37         return *this;
38     }
39     Z &operator+=(const Z &rhs) {
40         x = norm(x + rhs.x);
41         return *this;
42     }
43     Z &operator-=(const Z &rhs) {
44         x = norm(x - rhs.x);
45         return *this;
46     }
47     Z &operator/=(const Z &rhs) {
48         return *this *= rhs.inv();
49     }
50     friend Z operator*(const Z &lhs, const Z &rhs) {
51         Z res = lhs;
52         res *= rhs;
53         return res;
54     }
```

```
55      friend Z operator+(const Z &lhs, const Z &rhs) {
56          Z res = lhs;
57          res += rhs;
58          return res;
59      }
60      friend Z operator-(const Z &lhs, const Z &rhs) {
61          Z res = lhs;
62          res -= rhs;
63          return res;
64      }
65      friend Z operator/(const Z &lhs, const Z &rhs) {
66          Z res = lhs;
67          res /= rhs;
68          return res;
69      }
70      friend istream &operator>>(istream &is, Z &a) {
71          int64_t v;
72          is >> v;
73          a = Z(v);
74          return is;
75      }
76      friend ostream &operator<<(ostream &os, const Z &a) {
77          return os << a.val();
78      }
79  };
80
81  struct Binom {
82      const int N;
83      vector<Z> fac, invfac;
84      Binom(int n) : N(n), fac(N + 1), invfac(N + 1) {
85          fac[0] = 1;
86          for (int i = 1; i <= N; i++) {
87              fac[i] = fac[i - 1] * i;
88          }
89          invfac[N] = fac[N].inv();
90          for (int i = N; i; i--) {
91              invfac[i - 1] = invfac[i] * i;
92          }
93      }
94
95      Z get(int n, int m) {
96          if (m < 0 || n < m) return Z(0);
97          return fac[n] * invfac[m] * invfac[n - m];
98      };
99  };
100
101 void solve() {
102     int n, m;
```

```
103     cin >> n >> m >> P;
104 
105     Binom binom(P - 1);
106 
107     function<ll(int, int, int)> Lucas = [&](int n, int m, int P) {
108         if (m == 0) return 1LL;
109         return 1LL * binom.get(n % P, m % P).val() * Lucas(n / P, m / P, P) % P;
110     };
111 
112     cout << Lucas(n + m, m, P) << "\n";
113 }
114 
115 int main() {
116     ios::sync_with_stdio(false);
117     cin.tie(nullptr);
118 
119     int t;
120     cin >> t;
121 
122     while (t--) {
123         solve();
124     }
125 
126     return 0;
127 }
128 // test problem: https://www.luogu.com.cn/problem/P3807
```

0.4.6 Matrix.cpp

```
1  #include <algorithm>
2  #include <cassert>
3  #include <iostream>
4  #include <vector>
5  using namespace std;
6 
7  // TODO: switch to `double` if `long double` is unnecessary and the time limit is tight.
8  // Using `long double` is more accurate, but it can be 50-60% slower than `double`.
9  using matrix_float = long double;
10 
11 // TODO: if using float_column_vector, we can write the float_matrix in the format matrix[x] = a row
       of coefficients
12 // used to build the x-th element of the float_column_vector. So matrix[0][2] is the coefficient that
        element 2
13 // contributes to the next element 0.
14 // The other option is to take a single-row 1 * n float_matrix and multiply it by the n * n
       float_matrix. Then
15 // matrix[0][2] is the coefficient that 0 contributes to the next element 2.
```

```
16 struct float_column_vector {
17     int rows;
18     vector<matrix_float> values;
19 
20     float_column_vector(int _rows = 0) {
21         init(_rows);
22     }
23 
24     template<typename T>
25     float_column_vector(const vector<T> &v) {
26         init(v);
27     }
28 
29     void init(int _rows) {
30         rows = _rows;
31         values.assign(rows, 0);
32     }
33 
34     template<typename T>
35     void init(const vector<T> &v) {
36         rows = int(v.size());
37         values = vector<matrix_float>(v.begin(), v.end());
38     }
39 
40     matrix_float& operator[](int index) { return values[index]; }
41     const matrix_float& operator[](int index) const { return values[index]; }
42 };
43 
44 // Warning: very inefficient for many small matrices of fixed size. For that, use
       float_matrix_fixed_size.cc instead.
45 struct float_matrix {
46     static float_matrix IDENTITY(int n) {
47         float_matrix identity(n);
48 
49         for (int i = 0; i < n; i++) {
50             identity[i][i] = 1;
51         }
52 
53         return identity;
54     }
55 
56     int rows, cols;
57     vector<vector<matrix_float>> values;
58 
59     float_matrix(int _rows = 0, int _cols = -1) {
60         init(_rows, _cols);
61     }
62 
```

```
63      template<typename T>
64      float_matrix(const vector<vector<T>> &v) {
65          init(v);
66      }
67
68      void init(int _rows, int _cols = -1) {
69          rows = _rows;
70          cols = _cols < 0 ? rows : _cols;
71          values.assign(rows, vector<matrix_float>(cols, 0));
72      }
73
74      template<typename T>
75      void init(const vector<vector<T>> &v) {
76          rows = int(v.size());
77          cols = v.empty() ? 0 : int(v[0].size());
78          values.assign(rows, vector<matrix_float>(cols, 0));
79
80          for (int i = 0; i < rows; i++) {
81              assert(int(v[i].size()) == cols);
82              copy(v[i].begin(), v[i].end(), values[i].begin());
83          }
84      }
85
86      vector<matrix_float>& operator[](int index) { return values[index]; }
87      const vector<matrix_float>& operator[](int index) const { return values[index]; }
88
89      bool is_square() const {
90          return rows == cols;
91      }
92
93      float_matrix operator*(const float_matrix &other) const {
94          assert(cols == other.rows);
95          float_matrix product(rows, other.cols);
96
97          for (int i = 0; i < rows; i++)
98              for (int j = 0; j < cols; j++)
99                  if (values[i][j] != 0)
100                     for (int k = 0; k < other.cols; k++)
101                         product[i][k] += values[i][j] * other[j][k];
102
103         return product;
104     }
105
106     float_matrix& operator*=(const float_matrix &other) {
107         return *this = *this * other;
108     }
109
110     float_column_vector operator*(const float_column_vector &column) const {
```

```
111         assert(cols == column.rows);
112         float_column_vector product(rows);
113 
114         for (int i = 0; i < rows; i++)
115             for (int j = 0; j < cols; j++)
116                 product[i] += values[i][j] * column[j];
117 
118         return product;
119     }
120 
121     float_matrix& operator*=(matrix_float mult) {
122         for (int i = 0; i < rows; i++)
123             for (int j = 0; j < cols; j++)
124                 values[i][j] *= mult;
125 
126         return *this;
127     }
128 
129     float_matrix operator*(matrix_float mult) const {
130         return float_matrix(*this) *= mult;
131     }
132 
133     float_matrix& operator+=(const float_matrix &other) {
134         assert(rows == other.rows && cols == other.cols);
135 
136         for (int i = 0; i < rows; i++)
137             for (int j = 0; j < cols; j++)
138                 values[i][j] += other[i][j];
139 
140         return *this;
141     }
142 
143     float_matrix operator+(const float_matrix &other) const {
144         return float_matrix(*this) += other;
145     }
146 
147     float_matrix& operator-=(const float_matrix &other) {
148         assert(rows == other.rows && cols == other.cols);
149 
150         for (int i = 0; i < rows; i++)
151             for (int j = 0; j < cols; j++)
152                 values[i][j] -= other[i][j];
153 
154         return *this;
155     }
156 
157     float_matrix operator-(const float_matrix &other) const {
158         return float_matrix(*this) -= other;
```

```
159     }
160
161     float_matrix pow(int64_t p) const {
162         assert(p >= 0);
163         assert(is_square());
164         float_matrix m = *this, result = IDENTITY(rows);
165
166         while (p > 0) {
167             if (p & 1) {
168                 result *= m;
169             }
170             p >>= 1;
171             if (p > 0) {
172                 m *= m;
173             }
174         }
175
176         return result;
177     }
178
179     void print(ostream &os) const {
180         for (int i = 0; i < rows; i++)
181             for (int j = 0; j < cols; j++)
182                 os << values[i][j] << (j < cols - 1 ? ' ' : '\n');
183
184         os << '\n';
185     }
186 };
187
188
189 #include <iomanip>
190
191 void read_matrix(float_matrix &m) {
192     int r, c;
193     cin >> r >> c;
194     m = float_matrix(r, c);
195
196     for (int i = 0; i < r; i++) {
197         for (int j = 0; j < c; j++) {
198             double x;
199             cin >> x;
200             m[i][j] = x;
201         }
202     }
203 }
204
205 int main() {
206     cout << setprecision(16);
```

```
207
208     float_matrix m1, m2;
209     read_matrix(m1);
210     read_matrix(m2);
211     (m1 + m1).print(cout);
212     (m2 - m2).print(cout);
213     (m1 * m2).print(cout);
214
215     read_matrix(m1);
216     int64_t p;
217     cin >> p;
218     (m1 * p).print(cout);
219     m1.pow(p).print(cout);
220 }
```

0.4.7 Pollard_Rho.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  struct Pollard_Rho {
7  private:
8      uint64_t mod_mul64(uint64_t a, uint64_t b, uint64_t mod) {
9          assert(a < mod && b < mod);
10         if (mod <= 1LLU << 32) return a * b % mod;
11         if (mod <= 1LLU << 63) {
12             uint64_t q = uint64_t((long double) a * b / mod);
13             uint64_t result = a * b - q * mod;
14             if (result > 1LLU << 63) {
15                 result += mod;
16             } else if (result >= mod) {
17                 result -= mod;
18             }
19             return result;
20         }
21     #ifdef __SIZEOF_INT128__
22         return uint64_t(__uint128_t(a) * b % mod);
23     #endif
24         assert(false);
25     }
26     uint64_t mod_pow64(uint64_t a, uint64_t b, uint64_t mod) {
27         uint64_t result = 1;
28         for (; b; b >>= 1, a = mod_mul64(a, a, mod)) {
29             if (b & 1) result = mod_mul64(result, a, mod);
30         }
```

```
31          return result;
32      }
33      bool miller_rabin(uint64_t n) {
34          if (n < 2) return false;
35          // Check small primes.
36          for (uint64_t p : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}) {
37              if (n % p == 0) return n == p;
38          }
39          // https://miller-rabin.appspot.com/
40          auto get_miller_rabin_bases = [&]() -> vector<uint64_t> {
41              if (n < 341531) return {9345883071009581737LLU};
42              if (n < 1050535501) return {336781006125, 9639812373923155};
43              if (n < 350269456337) return {4230279247111683200, 14696767155120705706LLU,
                    16641139526367750375LLU};
44              if (n < 55245642489451) return {2, 141889084524735, 1199124725622454117,
                    11096072698276303650LLU};
45              if (n < 7999252175582851) return {2, 4130806001517, 149795463772692060,
                    186635894390467037, 3967304179347715805};
46              if (n < 585226005592931977) return {2, 123635709730000, 9233062284813009,
                    43835965440333360, 761179012939631437, 1263739024124850375};
47              return {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
48          };
49          int r = __builtin_ctzll(n - 1);
50          uint64_t d = (n - 1) >> r;
51          for (uint64_t a : get_miller_rabin_bases()) {
52              if (a % n == 0) continue;
53              uint64_t x = mod_pow64(a % n, d, n);
54              if (x == 1 || x == n - 1) continue;
55              for (int i = 0; i < r - 1 && x != n - 1; i++) {
56                  x = mod_mul64(x, x, n);
57              }
58              if (x != n - 1) return false;
59          }
60          return true;
61      }
62      int64_t solve(int64_t x) {
63          int64_t s = 0, t = 0;
64          int64_t c = (int64_t)rand() % (x - 1) + 1;
65          int step = 0, goal = 1;
66          int64_t val = 1;
67          for (goal = 1;; goal *= 2, s = t, val = 1) {  // 倍增优化
68              for (step = 1; step <= goal; ++step) {
69                  t = ((__int128)t * t + c) % x;
70                  val = (__int128)val * abs(t - s) % x;
71                  if ((step % 127) == 0) {
72                      int64_t d = gcd(val, x);
73                      if (d > 1) return d;
74                  }
```

```
            }
            int64_t d = gcd(val, x);
            if (d > 1) return d;
        }
    }
    void fac(int64_t x, vector<int64_t> &ans) {
        if (x == 1) return;
        if (miller_rabin(x)) { // 如果 x 为质数
            ans.push_back(x);
            return;
        }
        int64_t p = x;
        while (p >= x) p = solve(x);  // 使用该算法
        while ((x % p) == 0) x /= p;
        fac(x, ans), fac(p, ans);  // 继续向下分解 x 和 p
    }

public:
    Pollard_Rho() { srand((unsigned)time(NULL)); }
    vector<int64_t> pollard_Rho(int64_t x) {
        vector<int64_t> ans;
        fac(x, ans);
        return ans;
    }
    bool isPrime(int64_t x) {
        return miller_rabin(x);
    }
};

void solve() {
    ll n;
    cin >> n;
    Pollard_Rho poll;
    auto ans = poll.pollard_Rho(n);
    ll maxx = *max_element(ans.begin(), ans.end());

    if (maxx == n) {
        cout << "Prime\n";
    } else {
        cout << maxx << "\n";
    }
}

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

    int t;
```

```
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}
// test problem: https://www.luogu.com.cn/problem/P4718
```

0.4.8 exgcd.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

void solve() {
    ll a, b, c;
    cin >> a >> b >> c;

    // ax + by = gcd(a, b) = d
    // return tuple(d, x, y)
    function<tuple<int64_t, int64_t, int64_t>(int64_t, int64_t)> exgcd = [&](int64_t a, int64_t b) {
        if (b == 0) {
            return tuple(a, 1LL, 0LL);
        }
        auto [d, x, y] = exgcd(b, a % b);
        return tuple(d, y, x - a / b * y);
    };

    auto [d, x, y] = exgcd(a, b);

    if (c % d != 0) {
        cout << "-1\n";
    } else {
        x *= c / d;
        y *= c / d;

        ll dx = b / d;
        ll dy = a / d;

        ll l = ceil(1.0 * (-x + 1) / dx);
        ll r = floor(1.0 * (y - 1) / dy);

        if (l > r) {
            cout << x + l * dx << " " << y - r * dy << "\n";
```

```
        } else {
            ll minx = x + l * dx, maxx = x + r * dx;
            ll miny = y - r * dy, maxy = y - l * dy;
            cout << r - l + 1 << " " << minx << " " << miny << " " << maxx << " " << maxy << "\n";
        }
    }
}

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

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

// test problem: https://www.luogu.com.cn/problem/P5656
```

0.4.9 xor_basis.cpp

```
template<typename T, int BITS = 30>
struct xor_basis {
    // A list of basis values sorted in decreasing order, where each value has a unique highest bit.
    vector<T> basis(BITS);
    int n = 0;

    T min_value(T start) const {
        if (n == BITS) {
            return 0;
        }
        for (int i = 0; i < n; i++) {
            start = min(start, start ^ basis[i]);
        }
        return start;
    }

    T max_value(T start = 0) const {
        if (n == BITS) {
            return (T(1) << BITS) - 1;
        }
        for (int i = 0; i < n; i++) {
```

```
            start = max(start, start ^ basis[i]);
        }
        return start;
    }

    bool add(T x) {
        x = min_value(x);
        if (x == 0) {
            return false;
        }

        basis[n++] = x;
        int k = n - 1;

        // Insertion sort.
        while (k > 0 && basis[k] > basis[k - 1]) {
            swap(basis[k], basis[k - 1]);
            k--;
        }

        // Remove the highest bit of x from other basis elements.
        // TODO: this can be removed for speed if desired.
        for (int i = k - 1; i >= 0; i--) {
            basis[i] = min(basis[i], basis[i] ^ x);
        }

        return true;
    }

    void merge(const xor_basis<T> &other) {
        for (int i = 0; i < other.n && n < BITS; i++) {
            add(other.basis[i]);
        }
    }

    void merge(const xor_basis<T> &a, const xor_basis<T> &b) {
        if (a.n > b.n) {
            *this = a;
            merge(b);
        } else {
            *this = b;
            merge(a);
        }
    }
};
```

0.4.10 容斥.cpp

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

constexpr int mod = 998244353;
// assume -mod <= x < 2mod
int norm(int x) {
    if (x < 0) x += mod;
    if (x >= mod) x -= mod;
    return x;
}
template<class T>
T power(T a, int b) {
    T res = 1;
    for (; b; b /= 2, a *= a)
        if (b % 2) res *= a;
    return res;
}
struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(ll x) : x(x % mod) {}

    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(mod - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, mod - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = ll(x) * rhs.x % mod;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
```

```
47      Z &operator/=(const Z &rhs) {
48          return *this *= rhs.inv();
49      }
50      friend Z operator*(const Z &lhs, const Z &rhs) {
51          Z res = lhs;
52          res *= rhs;
53          return res;
54      }
55      friend Z operator+(const Z &lhs, const Z &rhs) {
56          Z res = lhs;
57          res += rhs;
58          return res;
59      }
60      friend Z operator-(const Z &lhs, const Z &rhs) {
61          Z res = lhs;
62          res -= rhs;
63          return res;
64      }
65      friend Z operator/(const Z &lhs, const Z &rhs) {
66          Z res = lhs;
67          res /= rhs;
68          return res;
69      }
70  };
71
72  int main() {
73      ios::sync_with_stdio(false);
74      cin.tie(nullptr);
75
76      int n, L;
77      cin >> n >> L;
78      vector<int> s(n);
79      for (int i = 0; i < n; ++i) {
80          string t;
81          cin >> t;
82          for (auto c : t) {
83              s[i] |= 1 << (c - 'a');
84          }
85      }
86
87      Z ans = 0;
88      vector<Z> f(1 << n);
89      for (int mask = 1; mask < (1 << n); ++mask) {
90          int cur = (1 << 26) - 1;
91          for (int i = 0; i < n; ++i) {
92              if (mask >> i & 1) {
93                  cur &= s[i];
94              }
```

```
        }
        f[mask] = power(Z(__builtin_popcount(cur)), L);
        ans += (__builtin_popcount(mask) & 1 ? 1 : -1) * f[mask];
    }

    cout << ans.val() << "\n";

    return 0;
}

// test problem: https://atcoder.jp/contests/abc246/tasks/abc246_f
```

0.4.11 除法分块.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

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

    // n / l = n / (l + 1) = ... = n / r,1 <= l <= r <= k
    auto block = [&](int n, int k) {
        vector<array<int, 2>> ans;
        for (int l = 1, r; l <= k; l = r + 1) {
            r = (n / l ? min(k, n / (n / l)) : k);
            ans.push_back({l, r});
        }
        for (auto [l, r] : ans) {
            cout << l << " " << r << " " << n / l << "\n";
        }
    };

    block(24, 24);

    return 0;
}
```

0.5 Others

0.5.1 BigNum2.cpp

```
// #include <bits/stdc++.h>
#include <iostream>
```

```
3  #include <vector>
4  using namespace std;
5  struct BigNum : vector<int>  //用标准库vector做基类，完美解决位数问题，同时更易于实现
6  {
7      //将低精度转高精度的初始化，可以自动被编译器调用
8      //因此无需单独写高精度数和低精度数的运算函数，十分方便
9      BigNum(int n = 0)  //默认初始化为0，但0的保存形式为空
10     {
11         push_back(n);
12         check();
13     }
14     BigNum &check()  //在各类运算中经常用到的进位小函数，不妨内置
15     {
16         while (!empty() && !back())
17             pop_back();  //去除最高位可能存在的0
18         if (empty())
19             return *this;
20         for (int i = 1; i < size(); ++i)  //处理进位
21         {
22             (*this)[i] += (*this)[i - 1] / 10;
23             (*this)[i - 1] %= 10;
24         }
25         while (back() >= 10) {
26             push_back(back() / 10);
27             (*this)[size() - 2] %= 10;
28         }
29         return *this;  //为使用方便，将进位后的自身返回引用
30     }
31 };
32 //输入输出
33 istream &operator>>(istream &is, BigNum &n) {
34     string s;
35     is >> s;
36     n.clear();
37     for (int i = s.size() - 1; i >= 0; --i)
38         n.push_back(s[i] - '0');
39     return is;
40 }
41 ostream &operator<<(ostream &os, const BigNum &n) {
42     if (n.empty())
43         os << 0;
44     for (int i = n.size() - 1; i >= 0; --i)
45         os << n[i];
46     return os;
47 }
48 //比较，只需要写两个，其他的直接代入即可
49 //常量引用当参数，避免拷贝更高效
50 bool operator!=(const BigNum &a, const BigNum &b) {
```

```
51      if (a.size() != b.size())
52          return 1;
53      for (int i = a.size() - 1; i >= 0; --i)
54          if (a[i] != b[i])
55              return 1;
56      return 0;
57  }
58  bool operator==(const BigNum &a, const BigNum &b) {
59      return !(a != b);
60  }
61  bool operator<(const BigNum &a, const BigNum &b) {
62      if (a.size() != b.size())
63          return a.size() < b.size();
64      for (int i = a.size() - 1; i >= 0; --i)
65          if (a[i] != b[i])
66              return a[i] < b[i];
67      return 0;
68  }
69  bool operator>(const BigNum &a, const BigNum &b) {
70      return b < a;
71  }
72  bool operator<=(const BigNum &a, const BigNum &b) {
73      return !(a > b);
74  }
75  bool operator>=(const BigNum &a, const BigNum &b) {
76      return !(a < b);
77  }
78  //加法，先实现+=，这样更简洁高效
79  BigNum &operator+=(BigNum &a, const BigNum &b) {
80      if (a.size() < b.size())
81          a.resize(b.size());
82      for (int i = 0; i != b.size(); ++i)
83          a[i] += b[i];
84      return a.check();
85  }
86  BigNum operator+(BigNum a, const BigNum &b) {
87      return a += b;
88  }
89  //减法，返回差的绝对值，由于后面有交换，故参数不用引用
90  BigNum &operator-=(BigNum &a, BigNum b) {
91      if (a < b)
92          swap(a, b);
93      for (int i = 0; i != b.size(); a[i] -= b[i], ++i)
94          if (a[i] < b[i])  //需要借位
95          {
96              int j = i + 1;
97              while (!a[j])
98                  ++j;
```

```
            while (j > i) {
                --a[j];
                a[--j] += 10;
            }
        }
    return a.check();
}
BigNum operator-(BigNum a, const BigNum &b) {
    return a -= b;
}
//乘法不能先实现*=, 原因自己想
BigNum operator*(const BigNum &a, const BigNum &b) {
    BigNum n;
    n.assign(a.size() + b.size() - 1, 0);
    for (int i = 0; i != a.size(); ++i)
        for (int j = 0; j != b.size(); ++j)
            n[i + j] += a[i] * b[j];
    return n.check();
}
BigNum &operator*=(BigNum &a, const BigNum &b) {
    return a = a * b;
}
//除法和取模先实现一个带余除法函数
BigNum divmod(BigNum &a, const BigNum &b) {
    BigNum ans;
    for (int t = a.size() - b.size(); a >= b; --t) {
        BigNum d;
        d.assign(t + 1, 0);
        d.back() = 1;
        BigNum c = b * d;
        while (a >= c) {
            a -= c;
            ans += d;
        }
    }
    return ans;
}
BigNum operator/(BigNum a, const BigNum &b) {
    return divmod(a, b);
}
BigNum &operator/=(BigNum &a, const BigNum &b) {
    return a = a / b;
}
BigNum &operator%=(BigNum &a, const BigNum &b) {
    divmod(a, b);
    return a;
}
BigNum operator%(BigNum a, const BigNum &b) {
```

```
147     return a %= b;
148 }
149 //顺手实现一个快速幂，可以看到和普通快速幂几乎无异
150 BigNum pow(const BigNum &n, const BigNum &k) {
151     if (k.empty())
152         return 1;
153     if (k == 2)
154         return n * n;
155     if (k.back() % 2)
156         return n * pow(n, k - 1);
157     return pow(pow(n, k / 2), 2);
158 }
159 
160 int main() {
161 }
```

0.5.2 Simulated_annealing.cpp

```
1  #include <bits/stdc++.h>
2  
3  using namespace std;
4  using ll = long long;
5  
6  const double eps = 1e-8;
7  
8  int main() {
9      ios::sync_with_stdio(false);
10     cin.tie(nullptr);
11 
12     int n;
13     cin >> n;
14 
15     vector<tuple<int, int, int>> a(n);
16     for (int i = 0; i < n; ++i) {
17         int x, y, z;
18         cin >> x >> y >> z;
19         a[i] = tuple(x, y, z);
20     }
21 
22     auto solve = [&]() {
23         double step = 10000, ans = 1e30;
24         tuple<double, double, double> tp;
25         int pos = 0;
26 
27         auto dis = [&](auto A, auto B) {
28             auto [x1, y1, z1] = A;
29             auto [x2, y2, z2] = B;
```

```
30              return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) + (z2 - z1) * (z2 - z1));
31          };
32
33          while (step > eps) {
34              for (int i = 0; i < n; ++i) { //找一个最远的点
35                  if (dis(tp, a[pos]) < dis(tp, a[i])) {
36                      pos = i;
37                  }
38              }
39              double mt = dis(tp, a[pos]);
40              ans = min(ans, mt);
41              auto [x, y, z] = tp;
42              auto [px, py, pz] = a[pos];
43              x += (px - x) / mt * step;
44              y += (py - y) / mt * step;
45              z += (pz - z) / mt * step;
46              tp = tuple(x, y, z);
47
48              step *= 0.98;
49          }
50          return ans;
51      };
52
53      cout << fixed << setprecision(8) << solve() << "\n";
54
55      return 0;
56  }
57
58  // test problem: https://vjudge.net/problem/Gym-101981D
```

0.5.3 Z.cpp

```
1   template <typename T, T MOD>
2   struct ModInt {
3       using prod_type = std::conditional_t<std::is_same_v<T, int>, long long, __int128>;
4       T val;
5       ModInt(const prod_type v = 0) : val(v % MOD) { if (val < 0) val += MOD; };
6       ModInt operator+() const { return ModInt(val); }
7       ModInt operator-() const { return ModInt(MOD - val); }
8       ModInt inv() const {
9           auto a = val, m = MOD, u = 0, v = 1;
10          while (a != 0) {
11              auto t = m / a;
12              m -= t * a;
13              std::swap(a, m);
14              u -= t * v;
15              std::swap(u, v);
```

```
        }
        assert(m == 1);
        return u;
    }
    ModInt pow(prod_type n) const {
        auto x = ModInt(1);
        auto b = *this;
        while (n > 0) {
            if (n & 1)
                x *= b;
            n >>= 1;
            b *= b;
        }
        return x;
    }
    friend ModInt operator+(ModInt lhs, const ModInt &rhs) { return lhs += rhs; }
    friend ModInt operator-(ModInt lhs, const ModInt &rhs) { return lhs -= rhs; }
    friend ModInt operator*(ModInt lhs, const ModInt &rhs) { return lhs *= rhs; }
    friend ModInt operator/(ModInt lhs, const ModInt &rhs) { return lhs /= rhs; }
    ModInt &operator+=(const ModInt &x) {
        if ((val += x.val) >= MOD)
            val -= MOD;
        return *this;
    }
    ModInt &operator-=(const ModInt &x) {
        if ((val -= x.val) < 0)
            val += MOD;
        return *this;
    }
    ModInt &operator*=(const ModInt &x) {
        val = prod_type(val) * x.val % MOD;
        return *this;
    }
    ModInt &operator/=(const ModInt &x) { return *this *= x.inv(); }
    bool operator==(const ModInt &b) const { return val == b.val; }
    bool operator!=(const ModInt &b) const { return val != b.val; }
    friend std::istream &operator>>(std::istream &is, ModInt &x) noexcept {
        return is >> x.val;
    }
    friend std::ostream &operator<<(std::ostream &os, const ModInt &x) noexcept {
        return os << x.val;
    }
};
using ModInt1000000007 = ModInt<int, 1'000'000'007>;
using ModInt998244353 = ModInt<int, 998244353>;
```

0.5.4 bignum.cpp

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

class BigNum {
   private:
    int a[1000];
    int len;

   public:
    BigNum() {
        len = 1;
        memset(a, 0, sizeof(a));
    }
    BigNum(const int b);
    BigNum(char *s);
    BigNum(const BigNum &T);
    BigNum &operator=(const BigNum &n);

    friend istream &operator>>(istream &, BigNum &);
    friend ostream &operator<<(ostream &, BigNum &);

    BigNum operator+(const BigNum &T) const;
    BigNum operator-(const BigNum &T) const;
    BigNum operator*(const BigNum &T) const;
    BigNum operator/(const int &b) const;
    BigNum operator|(const BigNum &T) const;
    BigNum operator%(const BigNum &T) const;

    bool operator>(const BigNum &T) const;
    bool operator>(const int &t) const;
};

BigNum::BigNum(const int b) {
    len = 0;
    memset(a, 0, sizeof(a));
    int t = b;
    while (t) {
        int x = t % 10;
        a[len++] = x;
        t /= 10;
    }
}
BigNum::BigNum(char *s) {
    memset(a, 0, sizeof(a));
    int l = strlen(s);
```

```
47      len = l;
48      int cnt = 0;
49      for (int i = l - 1; i >= 0; --i)
50          a[cnt++] = s[i] - '0';
51  }
52  BigNum::BigNum(const BigNum &T) : len(T.len) {
53      memset(a, 0, sizeof(a));
54      for (int i = 0; i < len; ++i)
55          a[i] = T.a[i];
56  }
57  BigNum &BigNum::operator=(const BigNum &n) {
58      len = n.len;
59      memset(a, 0, sizeof(a));
60      for (int i = 0; i < len; ++i)
61          a[i] = n.a[i];
62      return *this;
63  }
64  istream &operator>>(istream &in, BigNum &b) {
65      char ch[1000];
66      in >> ch;
67      int l = strlen(ch);
68      int count = 0;
69      for (int i = l - 1; i > 0; --i) {
70          b.a[count++] = ch[i] - '0';
71      }
72      if (ch[0] == '-')
73          b.a[count - 1] = 0 - b.a[count - 1];
74      else
75          b.a[count++] = ch[0] - '0';
76      b.len = count;
77      return in;
78  }
79  ostream &operator<<(ostream &out, BigNum &b) {
80      for (int i = b.len - 1; i >= 0; --i)
81          cout << b.a[i];
82      return out;
83  }
84  BigNum BigNum::operator+(const BigNum &T) const {
85      BigNum t(*this);
86      int big;
87      big = T.len > len ? T.len : len;
88      for (int i = 0; i < big; ++i) {
89          t.a[i] += T.a[i];
90          if (t.a[i] >= 10) {
91              t.a[i + 1]++;
92              t.a[i] -= 10;
93          }
94      }
```

```
95      if (t.a[big] != 0)
96          t.len = big + 1;
97      else
98          t.len = big;
99      return t;
100 }
101 BigNum BigNum::operator-(const BigNum &T) const {
102     int big;
103     bool flag;
104     BigNum t1, t2;
105     if (*this > T) {
106         t1 = *this;
107         t2 = T;
108         flag = 0;
109     } else {
110         t1 = T;
111         t2 = *this;
112         flag = 1;
113     }
114     big = t1.len;
115     for (int i = 0; i < big; ++i) {
116         if (t1.a[i] < t2.a[i]) {
117             int j = i + 1;
118             while (t1.a[j] == 0)
119                 j++;
120             t1.a[j--]--;
121             while (j > i)
122                 t1.a[j--] += 9;
123             t1.a[i] += 10 - t2.a[i];
124         } else
125             t1.a[i] -= t2.a[i];
126     }
127     t1.len = big;
128     while (t1.a[t1.len - 1] == 0 && t1.len > 1) {
129         t1.len--;
130         big--;
131     }
132     if (flag)
133         t1.a[big - 1] = 0 - t1.a[big - 1];
134     return t1;
135 }
136 BigNum BigNum::operator*(const BigNum &T) const {
137     BigNum ret;
138     int up;
139     int temp, temp1;
140     int i, j;
141     for (i = 0; i < len; ++i) {
142         up = 0;
```

```
143         for (j = 0; j < T.len; ++j) {
144             temp = a[i] * T.a[j] + ret.a[i + j] + up;
145             if (temp >= 10) {
146                 temp1 = temp % 10;
147                 up = temp / 10;
148                 ret.a[i + j] = temp1;
149             } else {
150                 up = 0;
151                 ret.a[i + j] = temp;
152             }
153         }
154         if (up != 0)
155             ret.a[i + j] = up;
156     }
157     ret.len = i + j;
158     while (ret.a[ret.len - 1] == 0 && ret.len > 1)
159         ret.len--;
160     return ret;
161 }
162 BigNum BigNum::operator/(const int &b) const {
163     BigNum ret;
164     int down = 0;
165     for (int i = len - 1; i >= 0; --i) {
166         ret.a[i] = (a[i] + down * 10) / b;
167         down = a[i] + down * 10 - ret.a[i] * b;
168     }
169     ret.len = len;
170     while (ret.a[ret.len - 1] == 0 && ret.len > 1)
171         ret.len--;
172     return ret;
173 }
174 BigNum BigNum::operator|(const BigNum &T) const {
175     BigNum ans;
176     BigNum a = *this, b = T;
177     int len1 = len, len2 = T.len;
178     int t = len1 - len2;
179     BigNum x = 1;
180     BigNum ten = 10;
181     for (int i = 0; i < t; ++i) {
182         b = b * ten;
183         x = x * ten;
184     }
185     while (a > T || (!(a > T) && !(T > a))) {
186         while (a > b || (!(a > b) && !(b > a))) {
187             a = a - b;
188             ans = ans + x;
189         }
190         b = b / 10;
```

```
191         x = x / 10;
192     }
193     return ans;
194 }
195 BigNum BigNum::operator%(const BigNum &T) const {
196     BigNum ans;
197     BigNum a = *this, b = T;
198     int len1 = len, len2 = T.len;
199     int t = len1 - len2;
200     BigNum x = 1;
201     BigNum ten = 10;
202     for (int i = 0; i < t; ++i) {
203         b = b * ten;
204         x = x * ten;
205     }
206     while (a > T || (!(a > T) && !(T > a))) {
207         while (a > b || (!(a > b) && !(b > a))) {
208             a = a - b;
209             ans = ans + x;
210         }
211         b = b / 10;
212         x = x / 10;
213     }
214     return a;
215 }
216 bool BigNum::operator>(const BigNum &T) const {
217     int ln;
218     if (len > T.len)
219         return true;
220     else if (len < T.len)
221         return false;
222 
223     ln = len - 1;
224     while (a[ln] == T.a[ln] && ln >= 0)
225         ln--;
226     if (ln >= 0 && a[ln] > T.a[ln])
227         return true;
228     else
229         return false;
230 }
231 bool BigNum::operator>(const int &t) const {
232     BigNum b(t);
233     return *this > b;
234 }
235 
236 int main() {
237 
238 }
```

0.5.5 gen.py

```
from random import *

# make data randint(l, r)

n = randint(1, 100000)

print(n)
```

0.5.6 pai.py

```
import os

stdName = "A"
bfName = "B"
dirName = "pai"

os.system("g++ -std=c++20 -Wall {0:}.cpp -o std".format(stdName))
os.system("g++ -std=c++20 -Wall {0:}.cpp -o bf".format(bfName))

os.system("mkdir {0:}".format(dirName))
os.system("mv std {0:}".format(dirName))
os.system("mv bf {0:}".format(dirName))

tc = 0
while True:
    os.system("python gen.py > ./{0:}/in.in".format(dirName))
    os.system("time ./{0:}/std < ./{0:}/in.in > ./{0:}/std.out".format(dirName))
    os.system("./{0:}/bf < ./{0:}/in.in > ./{0:}/bf.out".format(dirName))
    if os.system("diff ./{0:}/bf.out ./{0:}/std.out".format(dirName)):
        print("WA")
        exit(0)
    else:
        tc += 1
        print("AC #", tc)
```

0.5.7 威佐夫博弈.cpp

```
#include <algorithm>
#include <cmath>
#include <iostream>
#define gold (sqrt(5.0) + 1) / 2
using namespace std;
typedef long long ll;

int main() {
```

```
    ios::sync_with_stdio(false);
    int a, b;
    while (cin >> a >> b) {
        int big = max(a, b);
        int small = min(a, b);
        double now = double(big - small) * gold;
        if ((int)now == small)
            cout << 0 << endl;  //后手必胜
        else
            cout << 1 << endl;  //先手必胜
    }
}
```

0.5.8 杜教 BM.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i, a, n) for (long long i = a; i < n; i++)
#define per(i, a, n) for (long long i = n - 1; i >= a; i--)
#define pb push_back
#define all(x) (x).begin(), (x).end()
#define SZ(x) ((long long)(x).size())
typedef vector<long long> VI;
typedef long long ll;
typedef pair<long long, long long> PII;
const ll mod = 1e9 + 7;
ll powmod(ll a, ll b) {
    ll res = 1;
    a %= mod;
    assert(b >= 0);
    for (; b; b >>= 1) {
        if (b & 1)
            res = res * a % mod;
        a = a * a % mod;
    }
    return res;
}
// head

namespace linear_seq {
const long long N = 10010;
ll res[N], base[N], _c[N], _md[N];

vector<long long> Md;
void mul(ll *a, ll *b, long long k) {
    rep(i, 0, k + k) _c[i] = 0;
    rep(i, 0, k) if (a[i]) rep(j, 0, k)
```

```
        _c[i + j] = (_c[i + j] + a[i] * b[j]) % mod;
    for (long long i = k + k - 1; i >= k; i--)
        if (_c[i])
            rep(j, 0, SZ(Md)) _c[i - k + Md[j]] = (_c[i - k + Md[j]] - _c[i] * _md[Md[j]]) % mod;
    rep(i, 0, k) a[i] = _c[i];
}
long long solve(ll n, VI a, VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
    // printf("%d\n",SZ(b));
    ll ans = 0, pnt = 0;
    long long k = SZ(a);
    assert(SZ(a) == SZ(b));
    rep(i, 0, k) _md[k - 1 - i] = -a[i];
    _md[k] = 1;
    Md.clear();
    rep(i, 0, k) if (_md[i] != 0) Md.push_back(i);
    rep(i, 0, k) res[i] = base[i] = 0;
    res[0] = 1;
    while ((1ll << pnt) <= n) pnt++;
    for (long long p = pnt; p >= 0; p--) {
        mul(res, res, k);
        if ((n >> p) & 1) {
            for (long long i = k - 1; i >= 0; i--)
                res[i + 1] = res[i];
            res[0] = 0;
            rep(j, 0, SZ(Md)) res[Md[j]] = (res[Md[j]] - res[k] * _md[Md[j]]) % mod;
        }
    }
    rep(i, 0, k) ans = (ans + res[i] * b[i]) % mod;
    if (ans < 0) ans += mod;
    return ans;
}
VI BM(VI s) {
    VI C(1, 1), B(1, 1);
    long long L = 0, m = 1, b = 1;
    rep(n, 0, SZ(s)) {
        ll d = 0;
        rep(i, 0, L + 1) d = (d + (ll)C[i] * s[n - i]) % mod;
        if (d == 0)
            ++m;
        else if (2 * L <= n) {
            VI T = C;
            ll c = mod - d * powmod(b, mod - 2) % mod;
            while (SZ(C) < SZ(B) + m)
                C.pb(0);
            rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
            L = n + 1 - L;
            B = T;
            b = d;
```

```
            m = 1;
        } else {
            ll c = mod - d * powmod(b, mod - 2) % mod;
            while (SZ(C) < SZ(B) + m) C.pb(0);
            rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
            ++m;
        }
    }
    return C;
}
long long gao(VI a, ll n) {
    VI c = BM(a);
    c.erase(c.begin());
    rep(i, 0, SZ(c)) c[i] = (mod - c[i]) % mod;
    return solve(n, c, VI(a.begin(), a.begin() + SZ(c)));
}
}; // namespace linear_seq

int main() {
    int n;
    cin >> n;
    cout << linear_seq::gao(VI{0, 1, 5, 18, 58, 177, 522, 1503, 4252, 11869}, n - 1) << "\n";
}
```

0.6 String

0.6.1 AhoCorasick.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

/** Modified from:
 * https://github.com/kth-competitive-programming/kactl/blob/master/content/strings/AhoCorasick.h
 * Try to handdle duplicated patterns beforehand, otherwise change 'end' to
 * vector; empty patterns are not allowed. Time: construction takes $O(26N)$,
 * where $N =$ sum of length of patterns. find(x) is $O(N)$, where N = length of x.
 * findAll is $O(N+M)$ where M is number of occurrence of all pattern (up to N*sqrt(N)) */
struct AhoCorasick {
    enum { alpha = 26, first = 'a' }; // change this!
    struct Node {
        // back: failure link, points to longest suffix that is in the trie.
        // end: longest pattern that ends here, is -1 if no patten ends here.
        // nmatches: number of (patterns that is a suffix of current node)/(duplicated patterns),
            depends on needs.
        // output: output link, points to the longest pattern that is a suffix of current node
```

```
19          int back, end = -1, nmatches = 0, output = -1;
20          array<int, alpha> ch;
21          Node(int v = -1) { fill(ch.begin(), ch.end(), v); }
22      };
23      vector<Node> N;
24      int n;
25      AhoCorasick() : N(1), n(0) {}
26      void insert(string &s) {
27          assert(!s.empty());
28          int p = 0;
29          for (char c : s) {
30              if (N[p].ch[c - first] == -1) {
31                  N[p].ch[c - first] = N.size();
32                  N.emplace_back();
33              }
34              p = N[p].ch[c - first];
35          }
36          N[p].end = n++;
37          N[p].nmatches++;
38      }
39      void build() {
40          N[0].back = (int)N.size();
41          N.emplace_back(0);
42          queue<int> q;
43          q.push(0);
44          while (!q.empty()) {
45              int p = q.front();
46              q.pop();
47              for (int i = 0; i < alpha; i++) {
48                  int pnx = N[N[p].back].ch[i];
49                  auto &nxt = N[N[p].ch[i]];
50                  if (N[p].ch[i] == -1) N[p].ch[i] = pnx;
51                  else {
52                      nxt.back = pnx;
53                      // if prev is an end node, then set output to prev node,
54                      // otherwise set to output link of prev node
55                      nxt.output = N[pnx].end == -1 ? N[pnx].output : pnx;
56                      // if we don't want to distinguish info of patterns that is
57                      // a suffix of current node, we can add info to the ch
58                      // node like this: nxt.nmatches+=N[pnx].nmatches;
59                      q.push(N[p].ch[i]);
60                  }
61              }
62          }
63      }
64      // for each position, finds the longest pattern that ends here
65      vector<int> find(const string &text) {
66          int len = text.length();
```

```
67          vector<int> res(len);
68          int p = 0;
69          for (int i = 0; i < len; i++) {
70              p = N[p].ch[text[i] - first];
71              res[i] = N[p].end;
72          }
73          return res;
74      }
75      // for each position, finds the all that ends here
76      vector<vector<int>> find_all(const string &text) {
77          int len = text.length();
78          vector<vector<int>> res(len);
79          int p = 0;
80          for (int i = 0; i < len; i++) {
81              p = N[p].ch[text[i] - first];
82              res[i].push_back(N[p].end);
83              for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
84                  assert(N[ind].end != -1);
85                  res[i].push_back(N[ind].end);
86              }
87          }
88          return res;
89      }
90      int find_cnt(const string &text) {
91          int len = text.length();
92          vector<int> num(n + 1, 0);
93          int p = 0, ans = 0;
94          for (int i = 0; i < len; i++) {
95              p = N[p].ch[text[i] - first];
96              if (N[p].end != -1) {
97                  if (!num[N[p].end]) {
98                      num[N[p].end]++;
99                      ans += N[p].nmatches;
100                 }
101             }
102             for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
103                 if (!num[N[ind].end]) {
104                     num[N[ind].end]++;
105                     ans += N[ind].nmatches;
106                 }
107             }
108         }
109         return ans;
110     }
111     pair<int, vector<int>> find_maxcnt(const string &text) {
112         int len = text.length();
113         vector<int> num(n + 1, 0);
114         int p = 0, ans = 0;
```

```
        for (int i = 0; i < len; i++) {
            p = N[p].ch[text[i] - first];
            if (N[p].end != -1) {
                if (!num[N[p].end]) {
                    num[N[p].end]++;
                    ans = max(ans, N[p].nmatches);
                }
            }
            for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
                if (!num[N[ind].end]) {
                    num[N[ind].end]++;
                    ans += N[ind].nmatches;
                }
            }
        }
        vector<int> idx;
        for (int i = 0; i < n; i++) {
            if (num[i] == ans) {
                idx.push_back(i);
            }
        }
        return pair(ans, idx);
    }
};

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

    int n;
    cin >> n;

    AhoCorasick ac;
    for (int i = 0; i < n; ++i) {
        string s;
        cin >> s;
        ac.insert(s);
    }

    ac.build();

    string t;
    cin >> t;

    cout << ac.find_cnt(t) << "\n";

    return 0;
}
```

```

// test problem: https://www.luogu.com.cn/problem/P3808
```

0.6.2 kmp.cpp

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

vector<int> prefixFunction(string s) {
    int n = (int)s.size();
    vector<int> p(n);
    for (int i = 1; i < n; ++i) {
        int j = p[i - 1];
        while (j > 0 && s[i] != s[j]) j = p[j - 1];
        if (s[i] == s[j]) ++j;
        p[i] = j;
    }
    return p;
}

// KMP based on prefixFunction. return all match postion in t
// also can create string st = s + '#' + t, and call prefixFunction(st),
// if p[i] == s.length(), it's a successful match: s in t
vector<int> kmp(string s, string t) {
    vector<int> ans;
    int n = (int)s.size(), m = (int)t.size();
    if (n > m) return ans;
    auto p = prefixFunction(s);
    for (int i = 0, j = 0; i < m; ++i) {
        while (j > 0 && s[j] != t[i]) j = p[j - 1];
        if (s[j] == t[i] && ++j == n) ans.emplace_back(i - n + 1);
    }
    return ans;
}

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

    string t, s;
    cin >> t >> s;

    string st = s + '#' + t;
    auto ans = prefixFunction(st);
    for (int i = s.length() + 1; i < st.length(); ++i) {
```

```
43          if (ans[i] == s.length()) {
44              cout << i - 2 * s.length() + 1 << "\n";
45          }
46      }
47
48      for (int i = 0; i < s.length(); ++i) {
49          cout << ans[i] << " \n"[i == s.length() - 1];
50      }
51
52      return 0;
53  }
54
55  // test problem: https://www.luogu.com.cn/problem/P3375
```

0.6.3 manacher.cpp

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4   using ll = long long;
5
6   template <typename T>
7   vector<int> manacher(int n, const T &s) {
8       if (n == 0) {
9           return vector<int>();
10      }
11      vector<int> res(2 * n - 1, 0);
12      int l = -1, r = -1;
13      for (int z = 0; z < 2 * n - 1; z++) {
14          int i = (z + 1) >> 1;
15          int j = z >> 1;
16          int p = (i >= r ? 0 : min(r - i, res[2 * (l + r) - z]));
17          while (j + p + 1 < n && i - p - 1 >= 0) {
18              if (!(s[j + p + 1] == s[i - p - 1])) {
19                  break;
20              }
21              p++;
22          }
23          if (j + p > r) {
24              l = i - p;
25              r = j + p;
26          }
27          res[z] = p;
28      }
29      return res;
30      // res[2 * i] = odd radius in position i
31      // res[2 * i + 1] = even radius between positions i and i + 1
```

```
    // s = "abaa" -> res = {0, 0, 1, 0, 0, 1, 0}
    // s = "aaa" -> res = {0, 1, 1, 1, 0}
    // in other words, for every z from 0 to 2 * n - 2:
    // calculate i = (z + 1) >> 1 and j = z >> 1
    // now there is a palindrome from i - res[z] to j + res[z]
    // (watch out for i > j and res[z] = 0)
}
template <typename T>
vector<int> manacher(const T &s) {
    return manacher((int)s.size(), s);
}

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

    string s;
    cin >> s;
    int n = s.length();

    auto ans = manacher(s);

    int len = 0, id = -1;
    for (int z = 0; z < 2 * n - 1; ++z) {
        if (z % 2 == 0 && 1 + 2 * ans[z] > len) { // odd length of palindrome
            len = 1 + 2 * ans[z];
            id = z / 2 - ans[z];
        } else if (z % 2 == 1 && 2 * ans[z] > len) { // even length of palindrome
            len = 2 * ans[z];
            id = z / 2 - ans[z] + 1;
        }
    }

    cout << s.substr(id, len) << "\n";

    return 0;
}
```

0.6.4 trie.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

class Trie {
    vector<vector<int>> ch_;
```

```
8       vector<int> cnt;
9       int getnum(char ch) {
10          if (ch <= '9') return ch - '0';
11          if (ch <= 'Z') return ch - 'A' + 10;
12          return ch - 'a' + 36;
13      }
14      int emptyNode() {
15          ch_.push_back(vector<int>(62, -1));
16          cnt.push_back(0);
17          return ch_.size() - 1;
18      }
19
20     public:
21      Trie() { emptyNode(); }
22      void insert(string s) {
23          for (int i = 0, p = 0; i < s.size(); ++i) {
24              int c = getnum(s[i]);
25              if (ch_[p][c] == -1) {
26                  ch_[p][c] = emptyNode();
27              }
28              p = ch_[p][c];
29              cnt[p]++;
30          }
31      }
32      int query(string s) {
33          int p = 0;
34          for (int i = 0; i < s.size(); ++i) {
35              int c = getnum(s[i]);
36              if (ch_[p][c] == -1) return 0;
37              p = ch_[p][c];
38          }
39          return cnt[p];
40      }
41  };
42
43  void solve() {
44      int n, q;
45      cin >> n >> q;
46
47      Trie trie;
48      for (int i = 0; i < n; ++i) {
49          string s;
50          cin >> s;
51          trie.insert(s);
52      }
53
54      while (q--) {
55          string s;
```

```
56          cin >> s;
57          cout << trie.query(s) << "\n";
58      }
59  }
60
61  int main() {
62      ios::sync_with_stdio(false);
63      cin.tie(nullptr);
64
65      int t;
66      cin >> t;
67
68      while (t--) {
69          solve();
70      }
71
72      return 0;
73  }
74  // test problem: https://www.luogu.com.cn/problem/P8306
```

0.7 poly

0.7.1 FFT.cpp

```
1   #include <bits/stdc++.h>
2
3   using cd = std::complex<double>;
4   constexpr double PI = M_PI;
5
6   static void _fft(std::vector<cd> &a, bool invert) {
7       int n = a.size();
8       // permute the array to do in-place calculation
9       for (int i = 1, j = 0; i < n; i++) {
10          int bit = n >> 1;
11          for (; j & bit; bit >>= 1) {
12              j ^= bit;
13          }
14          j ^= bit;
15          if (i < j) swap(a[i], a[j]);
16      }
17      for (int len = 2; len <= n; len <<= 1) {
18          double ang = 2 * PI / len * (invert ? -1 : 1);
19          cd wlen(cos(ang), sin(ang));
20          for (int i = 0; i < n; i += len) {
21              cd w(1);
22              for (int j = 0; j < len / 2; j++) {
23                  cd u = a[i + j], v = a[i + j + len / 2] * w;
```

```
                a[i + j] = u + v;
                a[i + j + len / 2] = u - v;
                w *= wlen;
            }
        }
    }
    if (invert) {
        for (auto &x : a) x /= n;
    }
}

// calculates the convolution of a and b
// represent the coefficients of the polynomial *from low to high*
static std::vector<int> convolve(const std::vector<int> &a, const std::vector<int> &b) {
    std::vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end());
    int n = 1 << (std::__lg(size(a) + size(b) - 1) + 1);
    fa.resize(n);
    fb.resize(n);

    _fft(fa, false);
    _fft(fb, false);
    for (int i = 0; i < n; i++) {
        fa[i] *= fb[i];
    }
    _fft(fa, true);

    std::vector<int> result(n);
    for (int i = 0; i < n; i++) {
        result[i] = round(fa[i].real());
    }
    return result;
}

using namespace std;
using ll = long long;

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

    int n, m;
    cin >> n >> m;
    vector<int> a(n + 1), b(m + 1);
    for (int i = 0; i <= n; ++i) {
        cin >> a[i];
    }
    for (int i = 0; i <= m; ++i) {
        cin >> b[i];
```

```
72      }
73
74      auto c = convolve(a, b);
75
76      for (int i = 0; i <= n + m; ++i) {
77          cout << c[i] << " \n"[i == n + m];
78      }
79
80      return 0;
81  }
82
83  // test problem: https://www.luogu.com.cn/problem/P3803
```

0.7.2 Lagrange1.cpp

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4   using ll = long long;
5
6   constexpr int mod = 998244353;
7   // assume -mod <= x < 2mod
8   int norm(int x) {
9       if (x < 0) x += mod;
10      if (x >= mod) x -= mod;
11      return x;
12  }
13  template<class T>
14  T power(T a, int64_t b) {
15      T res = 1;
16      for (; b; b /= 2, a *= a) {
17          if (b % 2) res *= a;
18      }
19      return res;
20  }
21  struct Z {
22      int x;
23      Z(int x = 0) : x(norm(x)) {}
24      Z(int64_t x) : x(x % mod) {}
25      int val() const {
26          return x;
27      }
28      Z operator-() const {
29          return Z(norm(mod - x));
30      }
31      Z inv() const {
32          assert(x != 0);
```

```
33          return power(*this, mod - 2);
34      }
35      Z &operator*=(const Z &rhs) {
36          x = int64_t(x) * rhs.x % mod;
37          return *this;
38      }
39      Z &operator+=(const Z &rhs) {
40          x = norm(x + rhs.x);
41          return *this;
42      }
43      Z &operator-=(const Z &rhs) {
44          x = norm(x - rhs.x);
45          return *this;
46      }
47      Z &operator/=(const Z &rhs) {
48          return *this *= rhs.inv();
49      }
50      friend Z operator*(const Z &lhs, const Z &rhs) {
51          Z res = lhs;
52          res *= rhs;
53          return res;
54      }
55      friend Z operator+(const Z &lhs, const Z &rhs) {
56          Z res = lhs;
57          res += rhs;
58          return res;
59      }
60      friend Z operator-(const Z &lhs, const Z &rhs) {
61          Z res = lhs;
62          res -= rhs;
63          return res;
64      }
65      friend Z operator/(const Z &lhs, const Z &rhs) {
66          Z res = lhs;
67          res /= rhs;
68          return res;
69      }
70      friend istream &operator>>(istream &is, Z &a) {
71          int64_t v;
72          is >> v;
73          a = Z(v);
74          return is;
75      }
76      friend ostream &operator<<(ostream &os, const Z &a) {
77          return os << a.val();
78      }
79  };
80
```

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

    int n, k;
    cin >> n >> k;

    vector<int> x(n), y(n);
    for (int i = 0; i < n; ++i) {
        cin >> x[i] >> y[i];
    }

    Z ans = 0;
    for (int i = 0; i < n; ++i) {
        Z s1 = y[i], s2 = 1;
        for (int j = 0; j < n; ++j) {
            if (i != j) {
                s1 *= k - x[j];
                s2 *= x[i] - x[j];
            }
        }
        ans += s1 * s2.inv();
    }

    cout << ans << "\n";

    return 0;
}

// f(k) = \sum_{i = 0}^{n} y_{i} \prod_{i \neq j} \frac{k - x[j]}{x[i] - x[j]}

// test problem: https://www.luogu.com.cn/problem/P4781
```

0.7.3 Lagrange2.cpp

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

constexpr int mod = 998244353;
// assume -mod <= x < 2mod
int norm(int x) {
    if (x < 0) x += mod;
    if (x >= mod) x -= mod;
    return x;
}
```

```
template<class T>
T power(T a, int64_t b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) res *= a;
    }
    return res;
}
struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(int64_t x) : x(x % mod) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(mod - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, mod - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = int64_t(x) * rhs.x % mod;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator/=(const Z &rhs) {
        return *this *= rhs.inv();
    }
    friend Z operator*(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res *= rhs;
        return res;
    }
    friend Z operator+(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res += rhs;
        return res;
    }
    friend Z operator-(const Z &lhs, const Z &rhs) {
```

```
61          Z res = lhs;
62          res -= rhs;
63          return res;
64      }
65      friend Z operator/(const Z &lhs, const Z &rhs) {
66          Z res = lhs;
67          res /= rhs;
68          return res;
69      }
70      friend istream &operator>>(istream &is, Z &a) {
71          int64_t v;
72          is >> v;
73          a = Z(v);
74          return is;
75      }
76      friend ostream &operator<<(ostream &os, const Z &a) {
77          return os << a.val();
78      }
79  };
80
81  int main() {
82      ios::sync_with_stdio(false);
83      cin.tie(nullptr);
84
85      int n;
86      cin >> n;
87
88      vector<int> a;
89      vector<Z> g;
90      auto insert = [&](int x, int y) {
91          a.push_back(x);
92          int n = a.size();
93          vector<Z> pre{1};
94          for (int i = 0; i < n - 1; ++i) {
95              pre.push_back(pre.back() * (a[i] - x));
96          }
97          Z inv = pre.back().inv();
98          g.push_back((n % 2 == 0 ? -1 : 1) * y * inv);
99
100         for (int i = n - 2; i >= 0; --i) {
101             g[i] *= inv * pre[i];
102             inv *= a[i] - x;
103         }
104     };
105
106     auto query = [&](int k) {
107         int n = a.size();
108         vector<Z> suf(n);
```

```
109         suf.back() = 1;
110         for (int i = n - 1; i >= 1; --i) {
111             suf[i - 1] = suf[i] * (k - a[i]);
112         }
113 
114         Z ans = 0, j = 1;
115         for (int i = 0; i < n; ++i) {
116             ans += j * suf[i] * g[i];
117             j *= k - a[i];
118         }
119 
120         return ans;
121     };
122 
123     for (int i = 0; i < n; ++i) {
124         int op;
125         cin >> op;
126         if (op == 1) {
127             int x, y;
128             cin >> x >> y;
129             insert(x, y);
130         } else {
131             int k;
132             cin >> k;
133             cout << query(k) << "\n";
134         }
135     }
136 
137     return 0;
138 }
139 // test problem: https://loj.ac/p/166
```