1 数据结构

1.1 并查集

```
struct DSU {
    vector<int> f, siz;
    DSU(int n) : f(n + 1), siz(n + 1, 1) {
        siz[0] = 0;
        iota(f.begin(), f.end(), 0);
    }
    bool same(int x, int y) {
        return find(x) == find(y);
    }
    int find(int x) {
        while (x != f[x]) x = f[x] = f[f[x]];
        return x;
    }
    bool merge(int x, int y) {
        x = find(x), y = find(y);
        if (x == y) return 0;
        if (siz[x] < siz[y]) swap(x, y);
        siz[x] += siz[y]; f[y] = x;
        return 1;
    }
    int size(int x) {
        return siz[find(x)];
    }
};
```

1.2 单调栈

```
    找出每个数左边离它最近的比它大/小的数
    stack<int> s;
    for (int i = 1; i <= n; i ++ )
    {
        while (sz(s) && check(s.top(), i)) s.pop();
        s.push(i);
    }
```

1.3 单调队列

```cpp
// 滑动窗口
int a[N], q[N]; // q为单调队列，需要注意，队列中存放的是数组下标。

int main()
{
    int n, k;
    //n个数，窗口长度为k
    cin >> n >> k;

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

    int hh = 0, tt = -1; // hh为队列头，tt为队列尾。tt < hh的原因是防止在刚进入循环时判断出错，所以要将tt小于hh，从而达到先入一个数的情况。
    for (int i = 0; i < n; i ++ ){ // 模拟队尾碰到新数a[i]的过程
        if (hh <= tt && i - k + 1 > q[hh]) hh ++ ; // 单调队列长度大于k，队头出队，加上hh <= tt的原因主要是防止初始状态时tt < hh。

        while (hh <= tt && a[q[tt]] >= a[i]) tt --; // 碰到新数a[i]后，若当前队尾大于a[i]，则放入后不满足单调递减
        q[ ++ tt] = i; // 新元素入队

        if (i >= k - 1) printf("%d ", a[q[hh]]); // 用单调队列维护一个单调递减的区间，故每次窗口内最大值一定是队头
    }

    printf("\n");

    // 以下同上，改为维护单调递增即可。
    hh = 0, tt = -1;
    for (int i = 0; i < n; i ++ ){
        if (hh <= tt && i - k + 1 > q[hh]) hh ++ ;

        while (hh <= tt && a[q[tt]] <= a[i]) tt --;
        q[ ++ tt] = i;

        if (i >= k - 1) printf("%d ", a[q[hh]]);
    }
    printf("\n");

    return 0;
}
```

1.4 树状数组

1.4.1 单点修改与区间查询

```cpp
template <typename T>
struct Fenwick {
    int n;
    vector<T> a;
    Fenwick (int n) : n (n), a (n + 10) {}
    void add (int x, T v) {
        for (int i = x; i <= n; i += i & -i) {
            a[i] += v;
        }
    }
    T sum (int x) {
```

```
        T ans = 0;
        for (int i = x; i; i -= i & -i) {
            ans += a[i];
        }
        return ans;
    }
    T rangeSum (int l, int r) {
        if (l <= 1) return sum(r);
        return sum (r) - sum (l - 1);
    }
    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]) {
                x += i;
                k -= a[x];
            }
        }
        return x;
    }
};
```

```
// sum sub
struct BIT {
  int tr[1000010], N;
  // BIT (int len) {setlen(len * 2); }
  void setlen(int len) {N = len << 1; for (int i = 0; i <= N; i ++ ) tr[i] = 0; }
  void add(int x, int k) {for (; x <= N; x += lowbit(x)) tr[x] += k; }
  int pre(int x) {int res = 0; for (; x; x -= lowbit(x)) res = res + tr[x]; return res; }
  int query(int l, int r) {if (l > r) swap(l, r); return pre(r) - pre(l - 1); }
};

//max min gcd
struct BIT {
  int tr[1000010], info[1000010], N;
  // BIT (int len) {setlen(len * 2); }
  void setlen(int len) {N = len << 1; for (int i = 0; i <= N; i ++ ) tr[i] = info[i] = -
inf; }
  void add(int x, int k) {
    info[x] = k;
    for (int i = x; i <= n; i += lowbit(i)) {
        tr[i] = info[i];
        for (int j = 1; j < lowbit(i); j <<= 1) {
            tr[i] = max(tr[i], tr[i - j]);
        }
    }
  }
  int query(int l, int r) {
    if (l > r) swap(l, r);
    int ans = -inf;
    while (r >= l) {
        ans = max(ans, info[r]);
        r -- ;
        for (; r - lowbit(r) >= l; r -= lowbit(r)) {
            ans = max(ans, tr[r]);
        }
    }
    return ans;
  }
};
```

```cpp


using i64 = long long;
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;
        }
    }

    T sum(int x) {
        auto ans = T();
        for (int i = x; i > 0; i -= i & -i) {
            ans += a[i - 1];
        }
        return ans;
    }

    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;
    }
};

struct Max {
    int v;
    Max(int x = -1E9) : v{x} {}

    Max &operator+=(Max a) {
        v = std::max(v, a.v);
        return *this;
    }
};
```

1.4.2 区间修改与单点查询

```
int t[maxn], q[maxn]; // 用t数组维护差分数组

int lowbit (int x) {
    return x & -x;
}

void add(int x, int k){
    for (; x <= n; x += x & -x) t[x] += k;
}

int query(int x){
    int ans = 0;
    for (; x; x -= x & -x) ans += t[x];
    return ans;
}

int range (int l, int r) {
    return query(r) - query(l - 1);
}

signed main()
{
    cin >> n >> p;
    for (int i = 1; i <= n; i ++ ){
        cin >> q[i];
        add(i, q[i] - q[i - 1]);
    }

    while (p -- ){
        int op, l, r, x;
        cin >> op;
        if (op == 1){
            cin >> l >> r >> x;
            add(l, x), add(r + 1, -x);
        }
        if (op == 2){
            cin >> x;
            cout << q[x] + ask(x) << endl; // ask(x)即为q[x]的增量
        }
    }

    return 0;
}

```

1.4.3 区间修改与区间查询

```
//区间修改与区间查询
#include <bits/stdc++.h>
using namespace std;
#define int long long
int n, p;
int q[1123456], t[1123456], s[1123456], sum[1123456];

inline int lowbit(int x)
{
    return x & -x;
}

```

```cpp
void add1(int x, int k)
{
    for (int i = x; i <= n; i += lowbit(i)) t[i] += k, s[i] += x * k;
}

int ask1(int x)
{
    int ans = 0;
    for (int i = x; i; i -= lowbit(i)) ans += (x + 1) * t[i] - s[i];
    return ans;
}

signed main()
{
    cin >> n >> p;
    for (int i = 1; i <= n; i ++ ){
        cin >> q[i];
        sum[i] = q[i] + sum[i - 1];
        add1(i, q[i] - q[i - 1]);
    }

    while (p -- ){
        int op, l, r, x;
        cin >> op;
        if (op == 1){
            cin >> l >> r >> x;
            add1(l, x), add1(r + 1, -x);
        }
        if (op == 2){
            cin >> l >> r;
            cout << ask1(r) - ask1(l - 1) << endl;
        }
    }

    return 0;
}
```

1.5 线段树

temp1

```cpp
constexpr int N = 2e5 + 10;
array<int, N> a;
struct node {
    ll sum;
    int tag;
}tr[N << 2];
#define ls u << 1
#define rs u << 1 | 1
node pushup (node l, node r) {
    node tmp;
    tmp.tag = 0;
    tmp.sum = l.sum + r.sum;
    return tmp;
}
void pushdown (int u, int l, int r) {
    if (tr[u].tag) {
        int mid = l + r >> 1;
        tr[ls].tag += tr[u].tag;
```

```
        tr[rs].tag += tr[u].tag;
        tr[ls].sum += (mid - l + 1) * tr[u].tag;
        tr[rs].sum += (r - mid) * tr[u].tag;
        tr[u].tag = 0;
    }
}
void build (int u, int l, int r) {
    if (l == r) {
        tr[u] = {a[l], 0};
    } else {
        int mid = l + r >> 1;
        build(ls, l, mid);
        build(rs, mid + 1, r);
        tr[u] = pushup(tr[ls], tr[rs]);
    }
}
void modify (int u, int L, int R, int l, int r, int v) {
    if (L >= l && R <= r) {
        tr[u].tag += v;
        tr[u].sum += (R - L + 1) * v;
    } else {
        pushdown(u, L, R);
        int mid = L + R >> 1;
        if (l <= mid) modify(ls, L, mid, l, r, v);
        if (r > mid) modify(rs, mid + 1, R, l, r, v);
        tr[u] = pushup(tr[ls], tr[rs]);
    }
}
node query(int u, int L, int R, int l, int r) {
    if (L >= l && R <= r) {
        return tr[u];
    } else {
        pushdown(u, L, R);
        int mid = L + R >> 1;
        if (r <= mid) {
            return query(ls, L, mid, l, r);
        } else if (l > mid) {
            return query(rs, mid + 1, R, l, r);
        } else {
            return pushup(query(ls, L, mid, l, r), query(rs, mid + 1, R, l, r));
        }
    }
}
int search (int u, int L, int R, int l, int r, int k) {
    if (L == l && R == r) {
        if (tr[u].mx < k) return -1;
        else {
            if (L == R) return L;
            int mid = L + R >> 1;
            if (tr[ls].mx >= k) return search(ls, L, mid, l, mid, k);
            else return search(rs, mid + 1, R, mid + 1, r, k);
        }
    }
    int mid = L + R >> 1;
    if (r <= mid) return search(ls, L, mid, l, r, k);
    else if (l > mid) return search(rs, mid + 1, R, l, r, k);
    else {
        int pos = search(ls, L, mid, l, mid, k);
        if (pos == -1) return search(rs, mid + 1, R, mid + 1, r, k);
        return pos;
    }
}
```

```
#undef ls
#undef rs
```

temp2

```
struct Info {
    i64 mx;
    Info() : mx(1ll << 60) {}
    Info(i64 mx) : mx(mx) {}
};
Info operator + (const Info &a, const Info &b) {
    return {max(a.mx, b.mx)};
}
template<class Info, class Merge = plus<Info>>
struct SegmentTree {
    const int n;
    const Merge merge;
    vector<Info> info;
    SegmentTree(int n) : n(n), merge(Merge()), info((n << 2) + 1) {}
    SegmentTree (vector<Info> init) : SegmentTree((int)init.size()) {
        function<void(int, int, int)> build = [&] (int p, int l, int r) {
            if (l == r) {
                info[p] = init[l - 1];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m); build(2 * p + 1, m + 1, r);
            pull(p);
        };
        build(1, 1, n);
    }
    void pull (int p) {
        info[p] = merge(info[2 * p], info[2 * p + 1]);
    }
    void modify (int p, int l, int r, int x, const Info &v) {
        if (l == r) {
            info[p] = v;
            return;
        }
        int m = (l + r) / 2;
        if (x <= m) {
            modify(2 * p, l, m, x, v);
        } else {
            modify(2 * p + 1, m + 1, r, x, v);
        }
        pull(p);
    }
    void modify (int p, const Info &v) {
        modify(1, 1, n, p, v);
    }
    Info query (int p, int l, int r, int x, int y) {
        if (l > y || r < x) {
            return Info();
        }
        if (l >= x && r <= y) {
            return info[p];
        }
        int m = (l + r) / 2;
        return merge(query(2 * p, l, m, x, y), query(2 * p + 1, m + 1, r, x, y));
    }
```

```
    Info query (int l, int r) {
        return query(1, 1, n, l, r);
    }
};
```

temp3

```
struct Info {
    int mx;
    Info() : mx(0) {}
    Info(int mx) : mx(mx) {}
};
using Tag = int;
Info operator + (const Info &a, const Info &b) {
    return {a.mx + b.mx};
}
void apply (Info &x, Tag &a, Tag f) {
    x.mx += f;
    a += f;
}


template<class Info, class Tag, class Merge = plus<Info>>
struct LazySegmentTree {
    const int n;
    const Merge merge;
    vector<Info> info;
    vector<Tag> tag;
    LazySegmentTree(int n) : n(n), merge(Merge()), info((n << 2) + 1), tag((n << 2) + 1)
{}
    LazySegmentTree (vector<Info> init) : LazySegmentTree((int)init.size()) {
        function<void(int, int, int)> build = [&] (int p, int l, int r) {
            if (l == r) {
                info[p] = init[l - 1];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m); build(2 * p + 1, m + 1, r);
            pull(p);
        };
        build(1, 1, n);
    }
    void pull (int p) {
        info[p] = merge(info[2 * p], info[2 * p + 1]);
    }
    void apply (int p, const Tag &v) {
        ::apply(info[p], tag[p], v);
    }
    void push(int p) {
        apply(2 * p, tag[p]);
        apply(2 * p + 1, tag[p]);
        tag[p] = Tag();
    }
    void modify (int p, int l, int r, int x, const Info &v) {
        if (l == r) {
            ::apply(info[p], tag[p], v.mx);
            return;
        }
        int m = (l + r) / 2;
        push(p);
```

```
        if (x <= m) {
            modify(2 * p, l, m, x, v);
        } else {
            modify(2 * p + 1, m + 1, r, x, v);
        }
        pull(p);
    }
    void modify (int p, const Info &v) {
        modify(1, 1, n, p, v);
    }
    void modify(int p, int l, int r, int x, int y, const Tag &v){
        if (l > y || r < x) {
            return;
        }
        if (l >= x && r <= y) {
            apply(p, v);
            return;
        }
        int m = (l + r) / 2;
        push(p);
        modify(2 * p, l, m, x, y, v);
        modify(2 * p + 1, m + 1, r, x, y, v);
        pull(p);
    }
    void modify(int l, int r, const Tag &v){
        return modify(1, 1, n, l, r, v);
    }
    Info query (int p, int l, int r, int x, int y) {
        if (l > y || r < x) {
            return Info();
        }
        if (l >= x && r <= y) {
            return info[p];
        }
        int m = (l + r) / 2;
        push(p);
        return merge(query(2 * p, l, m, x, y), query(2 * p + 1, m + 1, r, x, y));
    }
    Info query (int l, int r) {
        return query(1, 1, n, l, r);
    }
    int find_first (int p, int l, int r, int L, int R, const function<bool(const Info&)>
&f, Info &pre) {
        if (l < R || r < L) {
            return r + 1;
        }
        if (l <= L && r <= R) {
            if (!f(merge(pre, info[p]))) {
                pre = merge(pre, info[p]);
                return r + 1;
            }
            if (l == r) return r;
            int m = (l + r) / 2;
            push(p);
            int res;
            if (f(merge(pre, info[2 * p]))) {
                res = find_first(2 * p, l, m, L, R, f, pre);
            } else {
                pre = merge(pre, info[2 * p]);
                res = find_first(2 * p + 1, r, L, R, f, pre);
            }
            return res;
```

```
        }
        int m = (l + r) / 2;
        push(p);
        int res = m + 1;
        if (L <= m) {
            res = find_first(2 * p, l, m, L, R, f, pre);
        }
        if (R > m && res == m + 1) {
            res = find_first(2 * p + 1, m + 1, r, L, R, f, pre);
        }
        return res;
    }
    int find_first(int l, int r, const function<bool(const Info&)> &f){
        Info pre = Info();
        return find_first(1, 1, n, l, r, f, pre);
    }
    int find_last(int p, int l, int r, int L, int R, const function<bool(const Info&)>
&f, Info &suf){
        if (l > R || r < L){
            return l - 1;
        }
        if (l >= L && r <= R){
            if (!f(merge(info[p], suf))){
                suf = merge(info[p], suf);
                return l - 1;
            }
            if (l == r) return r;
            int m = (l + r) / 2;
            push(p);
            int res;
            if (f(merge(info[2 * p + 1], suf))){
                res = find_last(2 * p + 1, m + 1, r, L, R, f, suf);
            }
            else{
                suf = merge(info[2 * p + 1], suf);
                res = find_last(2 * p, l, m, L, R, f, suf);
            }
            return res;
        }
        int m = (l + r) / 2;
        push(p);
        int res = m;
        if (R > m){
            res = find_last(2 * p + 1, m + 1, r, L, R, f, suf);
        }
        if (L <= m && res == m){
            res = find_last(2 * p, l, m, L, R, f, suf);
        }
        return res;
    }
    int find_last(int l, int r, const function<bool(const Info&)> &f){
        Info suf = Info();
        return find_last(1, 1, n, l, r, f, suf);
    }
};

```

```
// pushup
void pushup1(int u) {
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
} // 父节点的区间和等于左右儿子的区间和。
void pushup2(int u) {
```

```
    tr[u].v = max(tr[u << 1].v, tr[u << 1 | 1].v);
} // 父节点的最大值等于左右儿子的最大值。


// pushdown
void pushdown(int u) // 例为维护区间加的lazytag
{
    auto &r = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    if (r.add){
        left.add += r.add, left.sum += (left.r - left.l + 1) * r.add;
        right.add += r.add, right.sum += (right.r - right.l + 1) * r.add;
        r.add = 0;
    }
}

// build
void build (int u, int l, int r) {
    if (l == r) tr[u] = {l, r, q[r]};
    else {
        tr[u] = {l, r, q[r], 0}; // 该节点的左端点，右端点，以及维护的值和lazytag（可以不为1）
        int mid = l + r >> 1;
        build(u << 1, l, mid); // 向左下递归建树
        build(u << 1 | 1, mid + 1, r); // 向右下递归建树
        pushup(u); // 最后自底向上更新节点所维护的值
    }
}


//modify

// 1.单点修改
void modify1(int u, int x, int v) // x为待修改的数的下标，v为修改后的值
{
    if (tr[u].l == x && tr[u].r == x) tr[u].v = v; // 修改
    else{ // 未达到就继续递归
        int mid = tr[u].l + tr[u].r >> 1;
        if (x <= mid) modify(u << 1, x, v);
        else modify(u << 1 | 1, x, v);
        pushup(u); // 修改后自底向上更新一下。
    }
}

// 2.区间修改（需要lazytag）
void modify2(int u, int l, int r, int d) {
    if (tr[u].l >= l && tr[u].r <= r) {
        tr[u].sum += (tr[u].r - tr[u].l + 1) * d;
        tr[u].add += d;
    }
    else {
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (l <= mid) modify(u << 1, l, r, d);
        if (r > mid) modify(u << 1 | 1, l, r, d);
        pushup(u);
    }
}


//query
int query(int u, int l, int r)
{
    if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;
```

```
68
69      pushdown(u); // 如果有则加
70      int mid = tr[u].l + tr[u].r >> 1;
71      int sum = 0;
72      if (l <= mid) sum += query(u << 1, l, r);
73      if (r > mid) sum += query(u << 1 | 1, l, r);
74      return sum;
75  }
```

1.5.1 动态开点

```
1   int tot = 1;
2   struct node {
3       int ls, rs;
4       int val, tag;
5   }tr[maxn << 2];
6   int root = 1;
7   il void pushup (int u) {
8       tr[u].val = tr[tr[u].ls].val + tr[tr[u].rs].val;
9   }
10
11  il void pushtag (int &u, int l, int r, int k) {
12      if (!u) u = ++ tot;
13      tr[u].val += (r - l + 1) * k;
14      tr[u].tag += k;
15  }
16
17  il void build (int &u, int l, int r) {
18      if (!u) u = ++ tot;
19      if (l == r) {
20          tr[u].val = q[l];
21          return;
22      }
23      int mid = l + r >> 1;
24      build(tr[u].ls, l, mid);
25      build(tr[u].rs, mid + 1, r);
26      pushup(u);
27  }
28
29  il void pushdown (int u, int l, int r) {
30      if (tr[u].tag) {
31          int mid = l + r >> 1;
32          pushtag(tr[u].ls, l, mid, tr[u].tag);
33          pushtag(tr[u].rs, mid + 1, r, tr[u].tag);
34          tr[u].tag = 0;
35      }
36  }
37
38  il void update (int &u, int l, int r, int L, int R, int k) {
39      if (!u) u = ++ tot;
40      if (l >= L && r <= R) {
41          pushtag(u, l, r, k);
42          return;
43      }
44      pushdown(u, l, r);
45      int mid = l + r >> 1;
46      if (L <= mid) update(tr[u].ls, l, mid, L, R, k);
47      if (R > mid) update(tr[u].rs, mid + 1, r, L, R, k);
48      pushup(u);
49  }
```

```
il int ask (int u, int l, int r, int L, int R) {
    if (!u) return 0;
    if (L <= l && r <= R) return tr[u].val;
    pushdown(u, l, r);
    int sum = 0;
    int mid = l + r >> 1;
    if (L <= mid) sum += ask(tr[u].ls, l, mid, L, R);
    if (R > mid) sum += ask(tr[u].rs, mid + 1, r, L, R);
    return sum;
}

```

1.6 对顶堆

```
int q[i];

priority_queue<int> bg; // 大根堆
priority_queue<int, vector<int>, greater<int>> ss; // 小根堆

//bg.push(q[1]);

int mid = q[1];
cout << mid << endl;
for (int i = 2; i <= n; i ++ ) {
    if (q[i] > mid) ss.push(q[i]);
    else bg.push(q[i]);

    if (i & 1) { // 当i为奇数输出中位数
        while (bg.size() != ss.size()) {
            if (bg.size() > ss.size()) {
                ss.push(mid);v,
                mid = bg.top();
                bg.pop();
            }
            else {
                bg.push(mid);
                mid = ss.top();
                ss.pop();
            }
        }
        cout << mid << endl;
    }
}
```

1.7 主席树

```
constexpr int N = 1e5 + 10;
int q[N], rt[N], aft[N];
int cnt;
struct Chairman_Tree {
    struct Node {
        int l, r, val;
    } tree[N * 500];
    void init() {
        memset(rt, 0, sizeof rt);
        cnt = 0;
    }
    int buildTO(int l, int r) {
```

```
        int k = cnt++;
        tree[k].val = 0;
        if (l == r) return k;
        int mid = l + r >> 1;
        tree[k].l = buildTO(l, mid);
        tree[k].r = buildTO(mid + 1, r);
        return k;
    }
    void pushup(int u) {
        tree[u].val = tree[tree[u].l].val + tree[tree[u].r].val;
    }
    int update(int P, int l, int r, int ppos, int del) {
        int k = cnt++;
        if (l == r) {
            tree[k].val = tree[P].val + del;
        } else {
            int mid = l + r >> 1;
            if (ppos <= mid) {
                tree[k].l = update(tree[P].l, l, mid, ppos, del);
                tree[k].r = tree[P].r;
            } else {
                tree[k].l = tree[P].l;
                tree[k].r = update(tree[P].r, mid + 1, r, ppos, del);
            }
            pushup(k);
        }
        return k;
    }
    int query(int x, int y, int l, int r, int k) {
        if (l == r) {
            return l;
        }
        int mid = l + r >> 1;
        int res = tree[tree[y].l].val - tree[tree[x].l].val;
        if (res >= k) {
            return query(tree[x].l, tree[y].l, l, mid, k);
        } else {
            return query(tree[x].r, tree[y].r, mid + 1, r, k - res);
        }
    }
} T;

void solve() {
        scanf("%lld%lld", &n, &m);
        vector<int> a;
        for (int i = 1; i <= n; i ++ ) {
                scanf("%lld", &q[i]);
                a.pb(q[i]);
        }
        sort(all(a)); a.erase(unique(all(a)), a.end());
        for (int i = 1; i <= n; i ++ ) {
                q[i] = lower_bound(all(a), q[i]) - a.begin() + 1;
                // de(aft[i]);
        }
        tree.init();
        rt[0] = tree.buildTO(1, n);
        for (int i = 1; i <= n; i ++ ) {
                rt[i] = tree.update(rt[i - 1], 1, n, q[i], 1);
        }
        while (m -- ) {
                int l, r, k; scanf("%lld%lld%lld", &l, &r, &k);
                printf("%lld\n", a[tree.query(rt[l - 1], rt[r], 1, n, k) - 1]);
```

```
        }
}
```

1.8 Treap

1.8.1 旋转Treap

```
int tt;
int n, m, k;
int now, root;
int sz[maxn], key[maxn], cnt[maxn], rd[maxn], son[maxn][2];

inline void push_up(int x) {
    sz[x] = sz[son[x][0]] + sz[son[x][1]] + cnt[x];
}

inline void rotate (int &x, int y) {
    int ii = son[x][y ^ 1];
    son[x][y ^ 1] = son[ii][y];
    son[ii][y] = x;
    push_up(x), push_up(ii);
    x = ii;
}

void insert (int &u, int x) {
    if (!u) {
        u = ++ now;
        sz[u] = cnt[u] = 1;
        key[u] = x;
        rd[u] = rand();
        return;
    }
    if (key[u] == x) {
        cnt[u] ++ ;
        sz[u] ++ ;
        return;
    }
    int d = (x > key[u]);
    insert(son[u][d], x);
    if (rd[u] < rd[son[u][d]]) {
        rotate(u, d ^ 1);
    }
    push_up(u);
}

void del (int &u, int x) {
    if (!u) return;
    if (x != key[u]) {
        del (son[u][x > key[u]], x);
    }
    else {
        if (!son[u][0] && !son[u][1]) {
            cnt[u] -- ;
            sz[u] -- ;
            if (cnt[u] == 0) u = 0;
        }
        else if (son[u][0] && !son[u][1]) {
            rotate(u, 1);
            del (son[u][1], x);
        }
        else if (!son[u][0] && son[u][1]) {
```

```
            rotate(u, 0);
            del (son[u][0], x);
        }
        else {
            int d = rd[son[u][0]] > rd[son[u][1]];
            rotate(u, d);
            del(son[u][d], x);
        }
    }
    push_up(u);
}

int get_rank(int u, int x) { // 得到排名
    if (!u) return 0;
    if (key[u] == x) return sz[son[u][0]] + 1;
    if (key[u] < x) return sz[son[u][0]] + cnt[u] + get_rank(son[u][1], x);
    return get_rank(son[u][0], x);
}

int find (int u, int x) { // 查询排名
    if (!u) return 0;
    if(sz[son[u][0]] >= x) {
        return find(son[u][0], x);
    }
    else if (sz[son[u][0]] + cnt[u] < x) {
        return find(son[u][1], x - cnt[u] - sz[son[u][0]]);
    }
    else return key[u];
}

int pre (int u, int x) { // 查询前驱
    if (!u) return -inf;
    if (key[u] >= x) {
        return pre(son[u][0], x);
    }
    else return max(key[u], pre(son[u][1], x));
}

int suf (int u, int x) { // 查询后继
    if (!u) return inf;
    if (key[u] <= x) {
        return suf(son[u][1], x);
    }
    else return min(key[u], suf(son[u][0], x));
}

void solve() {
    int query;
    cin >> query;
    while (query -- ) {
        int op, x;
        cin >> op >> x;
        if (op == 1) insert(root, x);
        if (op == 2) del(root, x);
        if (op == 3) cout << get_rank(root, x) << endl;
        if (op == 4) cout << find(root, x) << endl;
        if (op == 5) cout << pre(root, x) << endl;
        if (op == 6) cout << suf(root, x) << endl;
    }
}
```

1.8.2 Fhq Treap

```
mt19937 rnd(233);
int root, idx;
int x, y, z;
struct fhq {
        int l, r;
        int key, val; // key权值，val堆值
        int size;
}tr[maxn];

inline int get_node (int key) {
        tr[ ++ idx].key = key;
        tr[idx].val = rnd();
        tr[idx].size = 1;
        return idx;
}

inline void pushup (int u) {
        tr[u].size = tr[tr[u].l].size + tr[tr[u].r].size + 1;
}

inline void split (int u, int k, int &x, int &y) {
        if (!u) x = y = 0;
        else {
                if (tr[u].key <= k) {
                        x = u;
                        split(tr[u].r, k, tr[u].r, y);
                }
                else {
                        y = u;
                        split(tr[u].l, k, x, tr[u].l);
                }
                pushup(u);
        }
}

inline int merge (int x, int y) {
        if (!x || !y) return x + y;
        if (tr[x].val > tr[y].val) {
                tr[x].r = merge(tr[x].r, y);
                pushup(x); return x;
        }
        else {
                tr[y].l = merge(x, tr[y].l);
                pushup(y); return y;
        }
}

inline void insert (int k) {
        split (root, k, x, y);
        root = merge(merge(x, get_node(k)), y);
}

inline void del (int k) {
        split (root, k, x, z);
        split (x, k - 1, x, y);
        y = merge(tr[y].l, tr[y].r);
        root = merge(merge(x, y), z);
}

inline int get_rank (int k) {
```

```
        split (root, k - 1, x, y);
        k = tr[x].size + 1;
        root = merge(x, y);
        return k;
}

inline int get_key (int k) {
        int p = root;
        while (p) {
                if (tr[tr[p].l].size + 1 == k) {
                        break;
                }
                else if (tr[tr[p].l].size >= k) {
                        p = tr[p].l;
                }
                else {
                        k -= tr[tr[p].l].size + 1;
                        p = tr[p].r;
                }
        }
        return tr[p].key;
}

inline int pre (int k) {
        split (root, k - 1, x, y);
        int p = x;
        while (tr[p].r) p = tr[p].r;
        k = tr[p].key;
        root = merge(x, y);
        return k;
}

inline int suf (int k) {
        split (root, k, x, y);
        int p = y;
        while (tr[p].l) p = tr[p].l;
        k = tr[p].key;
        root = merge(x, y);
        return k;
}

void solve() {
        read(n);
        for (int i = 1; i <= n; i ++ ) {
                int op, x; read(op); read(x);
                if (op == 1) insert(x);
                else if (op == 2) del(x);
                else if (op == 3) printf("%lld\n", get_rank(x));
                else if (op == 4) printf("%lld\n", get_key(x));
                else if (op == 5) printf("%lld\n", pre(x));
                else printf("%lld\n", suf(x));
        }
}


// 区间反转版本
mt19937 rnd(233);
int root, idx;
int x, y, z;
struct fhq {
        int l, r;
        int key, val; // key权值，val堆值
```

```
        int size;
        int tag;
}tr[maxn];

inline int get_node (int key) {
        tr[ ++ idx].key = key;
        tr[idx].val = rnd();
        tr[idx].size = 1;
        tr[idx].tag = 0;
        return idx;
}

inline void pushup (int u) {
        tr[u].size = tr[tr[u].l].size + tr[tr[u].r].size + 1;
}

inline void pushdown (int u) {
        if (tr[u].tag) {
                swap(tr[u].l, tr[u].r);
                tr[tr[u].l].tag ^= 1, tr[tr[u].r].tag ^= 1;
                tr[u].tag = 0;
        }
}

inline void split (int u, int k, int &x, int &y) {
        if (!u) x = y = 0;
        else {
                pushdown(u);
                if (tr[tr[u].l].size + 1 <= k) {
                        x = u;
                        split(tr[u].r, k - tr[tr[u].l].size - 1, tr[u].r, y);
                }
                else {
                        y = u;
                        split(tr[u].l, k, x, tr[u].l);
                }
                pushup(u);
        }
}

inline int merge (int x, int y) {
        if (!x || !y) return x + y;
        if (tr[x].val > tr[y].val) {
                pushdown(x);
                tr[x].r = merge(tr[x].r, y);
                pushup(x); return x;
        }
        else {
                pushdown(y);
                tr[y].l = merge(x, tr[y].l);
                pushup(y); return y;
        }
}

inline void insert (int k) {
        split (root, k, x, y);
        root = merge(merge(x, get_node(k)), y);
}

inline void del (int k) {
        split (root, k, x, z);
        split (x, k - 1, x, y);
```

```
        y = merge(tr[y].l, tr[y].r);
        root = merge(merge(x, y), z);
}

inline int get_rank (int k) {
        split (root, k - 1, x, y);
        k = tr[x].size + 1;
        root = merge(x, y);
        return k;
}

inline int get_key (int k) {
        int p = root;
        while (p) {
                if (tr[tr[p].l].size + 1 == k) {
                        break;
                }
                else if (tr[tr[p].l].size >= k) {
                        p = tr[p].l;
                }
                else {
                        k -= tr[tr[p].l].size + 1;
                        p = tr[p].r;
                }
        }
        return tr[p].key;
}

inline int pre (int k) {
        split (root, k - 1, x, y);
        int p = x;
        while (tr[p].r) p = tr[p].r;
        k = tr[p].key;
        root = merge(x, y);
        return k;
}

inline int suf (int k) {
        split (root, k, x, y);
        int p = y;
        while (tr[p].l) p = tr[p].l;
        k = tr[p].key;
        root = merge(x, y);
        return k;
}

inline void print (int u) {
        if (!u) return;
        pushdown(u);
        print(tr[u].l); printf("%lld ", tr[u].key); print(tr[u].r);
}

void solve() {
        read(n), read(m);
        // insert(-inf); insert(inf);
        for (int i = 1; i <= n; i ++ ) insert(i);

        while (m -- ) {
                int l, r; read(l); read(r);
                split (root, l - 1, x, y);
                split (y, r - l + 1, y, z);
                tr[y].tag ^= 1;
```

```
                root = merge(x, merge(y, z));
        }
        print(root);
}
```

1.9 轻重链剖分

```
int h[maxn], ne[maxn], e[maxn], idx;
int w[maxn], wt[maxn], id[maxn], top[maxn], fa[maxn], depth[maxn], sz[maxn], son[maxn];
int tt;
int n, m, k;
int root, mod, cnt;

struct node {
    int l, r;
    int sum, add;
}tr[maxn << 2];

inline void add (int a, int b) {
    e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;
}

void dfs1 (int u, int f) {
    depth[u] = depth[f] + 1;
    fa[u] = f;
    sz[u] = 1;
    int mxson = -1;
    for (int i = h[u]; ~i; i = ne[i]) {
        int j = e[i];
        if (j != f) {
            dfs1(j, u);
            sz[u] += sz[j];
            if (sz[j] > mxson) {
                mxson = sz[j];
                son[u] = j;
            }
        }
    }
}

void dfs2 (int u, int topf) {
    id[u] = ++ cnt;
    wt[cnt] = w[u];
    top[u] = topf;
    if (!son[u]) return;
    dfs2(son[u], topf);
    for (int i = h[u]; ~i; i = ne[i]) {
        int j = e[i];
        if (!id[j]) dfs2(j, j);
    }
}

inline void pushup (int u) {
    tr[u].sum = (tr[u << 1].sum + tr[u << 1 | 1].sum) % mod;
}

inline void pushdown (int u) {
    auto &r = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
```

```
    if (r.add) {
        right.add = (right.add + r.add) % mod;
        right.sum = (right.sum + (right.r - right.l + 1) * r.add) % mod;
        left.add = (left.add + r.add) % mod;
        left.sum = (left.sum + (left.r - left.l + 1) * r.add) % mod;
        r.add = 0;
    }
}

inline void build (int u, int l, int r) {
    if (l == r) {
        tr[u] = {l, r, wt[r], 0};
        return;
    }
    tr[u] = {l, r, 0, 0};
    int mid = (l + r) >> 1;
    build(u << 1, l, mid);
    build(u << 1 | 1, mid + 1, r);
    pushup(u);
}

inline void modify (int u, int l, int r, int d) {
    if (tr[u].l >= l && tr[u].r <= r) {
        tr[u].sum = (tr[u].sum + (tr[u].r - tr[u].l + 1) * d) % mod;
        tr[u].add += d;
    }
    else {
        pushdown(u);
        int mid = tr[u].r + tr[u].l >> 1;
        if (l <= mid) modify (u << 1, l, r, d);
        if (r > mid) modify (u << 1 | 1, l, r, d);
        pushup(u);
    }
}

inline int query (int u, int l, int r) {
    if (tr[u].l >= l && tr[u].r <= r) {
        return tr[u].sum;
    }
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    int sum = 0;
    if (l <= mid) sum = (sum + query(u << 1, l, r)) % mod;
    if (r > mid) sum = (sum + query(u << 1 | 1, l, r)) % mod;
    return sum;
}

inline void Uprange (int x, int y, int d) {
    while (top[x] != top[y]) {
        if (depth[top[x]] < depth[top[y]]) swap(x, y);
        modify(1, id[top[x]], id[x], d);
        x = fa[top[x]];
    }
    if (depth[x] > depth[y]) swap(x, y);
    modify(1, id[x], id[y], d);
}

inline int Qrange (int x, int y) {
    int sum = 0;
    while (top[x] != top[y]) {
        if (depth[top[x]] < depth[top[y]]) swap(x, y);
        sum = (sum + query(1, id[top[x]], id[x])) % mod;
```

```
        x = fa[top[x]];
    }
    if (depth[x] > depth[y]) swap(x, y);
    sum = (sum + query(1, id[x], id[y])) % mod;
    return sum;
}

inline int lca (int x, int y) {
    while (top[x] != top[y]) {
        depth[top[x]] > depth[top[y]] ? x = fa[top[x]] : y = fa[top[y]];
    }
    return depth[x] < depth[y] ? x : y;
}

void solve() {
    scanf("%d%d%d%d", &n, &m, &root, &mod);
    for (int i = 1; i <= n; i ++ ) scanf("%d", w + i);
    ms(h, -1);
    for (int i = 1; i < n; i ++ ) {
        int u, v; scanf("%d%d", &u, &v);
        add(u, v);add(v, u);
    }
    dfs1(root, 0);
    dfs2(root, root);
    build(1, 1, n);
    for (int i = 1; i <= m; i ++ ) {
        int op;
        scanf("%d", &op);
        if (op == 1) {
            int l, r, d;
            scanf("%d%d%d", &l, &r, &d);
            Uprange(l, r, d);
        }
        if (op == 2) {
            int l, r;
            scanf("%d%d", &l, &r);
            printf("%d\n", Qrange(l, r));
        }
        if (op == 3) {
            int x, d;
            scanf("%d%d", &x, &d);
            modify(1, id[x], id[x] + sz[x] - 1, d % mod);
        }
        if (op == 4) {
            int x;
            scanf("%d", &x);
            printf("%d\n", query(1, id[x], id[x] + sz[x] - 1));
        }
    }
}

```

边权：

```
int n, m;
vector<pair<int, int>> G[N];
int w[N], wt[N], id[N], top[N], fa[N], depth[N], sz[N], son[N];
int root, cnt;
struct node {
  int l, r;
```

```
  int sum, add;
}tr[N << 2];
void dfs1 (int u, int f) {
  depth[u] = depth[f] + 1;
  fa[u] = f;
  sz[u] = 1;
  int mxson = -1;
  for (auto [j, w] : G[u]) {
    if (j != f) {
      dfs1(j, u);
      sz[u] += sz[j];
      if (sz[j] > mxson) {
        mxson = sz[j];
        son[u] = j;
      }
    }
  }
}
void dfs2 (int u, int topf) {
  id[u] = ++ cnt;
  wt[cnt] = w[u];
  top[u] = topf;
  if (!son[u]) return;
  dfs2(son[u], topf);
  for (auto [j, w] : G[u]) {
    if (!id[j]) dfs2(j, j);
  }
}
void pushup (int u) {
  tr[u].sum = (tr[u << 1].sum + tr[u << 1 | 1].sum);
}
void pushdown (int u) {
  auto &r = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
  if (r.add) {
    right.add = left.add = r.add;
    right.sum = left.sum = r.add;
    r.add = 0;
  }
}
void build (int u, int l, int r) {
  if (l == r) {
    tr[u] = {l, r, wt[r], 0};
    return;
  }
  tr[u] = {l, r, 0, 0};
  int mid = (l + r) >> 1;
  build(u << 1, l, mid);
  build(u << 1 | 1, mid + 1, r);
  pushup(u);
}
void modify (int u, int l, int r, int d) {
  if (tr[u].l >= l && tr[u].r <= r) {
    tr[u].sum = (tr[u].r - tr[u].l + 1) * d;
    tr[u].add = d;
  }
  else {
    pushdown(u);
    int mid = tr[u].r + tr[u].l >> 1;
    if (l <= mid) modify (u << 1, l, r, d);
    if (r > mid) modify (u << 1 | 1, l, r, d);
    pushup(u);
  }
```

```cpp
}
int query (int u, int l, int r) {
  if (tr[u].l >= l && tr[u].r <= r) {
    return tr[u].sum;
  }
  pushdown(u);
  int mid = tr[u].l + tr[u].r >> 1;
  int sum = 0;
  if (l <= mid) sum += query(u << 1, l, r);
  if (r > mid) sum += query(u << 1 | 1, l, r);
  return sum;
}
void Uprange (int x, int y, int d) {
  while (top[x] != top[y]) {
    if (depth[top[x]] < depth[top[y]]) swap(x, y);
    modify(1, id[top[x]], id[x], d);
    x = fa[top[x]];
  }
  if (x == y) return;
  if (depth[x] > depth[y]) swap(x, y);
  modify(1, id[x] + 1, id[y], d);
}

int Qrange (int x, int y) {
  int sum = 0;
  while (top[x] != top[y]) {
    if (depth[top[x]] < depth[top[y]]) swap(x, y);
    sum += query(1, id[top[x]], id[x]);
    x = fa[top[x]];
  }
  if (x == y) return sum;
  if (depth[x] > depth[y]) swap(x, y);
  sum += query(1, id[x] + 1, id[y]);
  return sum;
}
void Solve() {
  cin >> n;
  vector<pair<int, int>> edges(n);
  for (int i = 1; i < n; i ++ ) {
    int u, v, w; cin >> u >> v >> w;
    G[u].emplace_back(v, w); G[v].emplace_back(u, w);
    edges[i] = {u, v};
  }
  function<void(int, int)> dfs = [&] (int u, int fa) {
    for (auto [j, v] : G[u]) {
      if (j != fa) {
        w[j] = v;
        dfs(j, u);
      }
    }
  };
  dfs(1, 0); dfs1(1, 0); dfs2(1, 1);
  build(1, 1, n);
  cin >> m;
  while (m -- ) {
    int op; cin >> op;
    if (op == 1) {
      int l, r; cin >> l >> r;
      int sum = Qrange(l, r);
      cout << sum << '\n';
    } else {
      int id, d; cin >> id >> d;
```

```
      int l = edges[id].first, r = edges[id].second;
      Uprange(l, r, d);
    }
  }
}
```

1.10 Dsu on Tree

```
/*
dsu on tree的核心是对暴力(n^2)的优化
暴力(n^2)是以每个节点当根暴力统计，dsu on tree可以对每个节点，只暴力统计轻儿子，在统计时少算一次重儿子。
*/
// 处理重儿子
void dfs (int u, int fa) {
    sz[u] = 1;
    for (auto j : G[u]) {
        if (j != fa) {
            dfs(j, u);
            sz[u] += sz[j];
            if (sz[j] > sz[son[u]]) son[u] = j;
        }
    }
}

// 这一步需要结合题意
void count (int u, int fa, int v) {
    cnt[val[u]] += v; // 增加或删除贡献
    if (cnt[val[u]] > mx) {
        mx = cnt[val[u]];
        sum = val[u];
    }
    else if (cnt[val[u]] == mx) {
        sum += val[u];
    }

    // 统计除标记外的重儿子的所有子树的贡献
    for (auto j : G[u]) {
        if (j == fa || j == flag) continue;
        count(j, u, v);
    }
}

void dfs (int u, int fa, bool op) {
    // 1. 计算子树答案，先轻后重。
    for (auto j : G[u]) {
        if (j != fa && j != son[u]) {
            dfs(j, u, false);
        }
    }
    if (son[u]) {
        dfs(son[u], u, true);
        flag = son[u];
    }
    // 2. 计算当前节点答案，重儿子信息被保存，只算轻儿子。
    count(u, fa, 1);
    flag = 0;
    ans[u] = sum;

    // 把需要删除贡献的删一删
```

```
    if (!op) {
        count(u, fa, -1);
        sum = mx = 0;
    }
}


// HDU 6504
// calc max(子树中不同 + 其余不同)
const int maxn = 200010;
vector<int> G[maxn];
int sz[maxn], son[maxn];
int cnt1[maxn], cnt2[maxn], w[maxn];
int com, ans, flag;

il void dfs (int u, int fa) {
        sz[u] = 1;
        for (auto j : G[u]) {
                if (j != fa) {
                    dfs(j, u);
                    sz[u] += sz[j];
                    if (sz[j] > sz[son[u]]) son[u] = j;
        }
        }
}

il void count (int u, int fa) {
        cnt1[w[u]] -- ;
        if (!cnt2[w[u]] && cnt1[w[u]]) com ++ ;
        else if (!cnt1[w[u]] && cnt2[w[u]]) com -- ;
        cnt2[w[u]] ++ ;

        for (auto j : G[u]) {
                if (j != fa && j != flag) {
                        count(j, u);
                }
        }
}

il void del (int u, int fa) {
        cnt2[w[u]] -- ;
        if (!cnt1[w[u]] && cnt2[w[u]]) com ++ ;
        else if (!cnt2[w[u]] && cnt1[w[u]]) com -- ;
        cnt1[w[u]] ++ ;

        for (auto j : G[u]) {
                if (j != fa) {
                        del(j, u);
                }
        }
}

il void dfs (int u, int fa, int op) {
        for (auto j : G[u]) {
                if (j != fa && j != son[u]) {
                        dfs(j, u, false);
                }
        }
        if (son[u]) {
                dfs(son[u], u, true);
                flag = son[u];
        }
```

```
        count(u, fa);
        flag = 0;
        ans = max(ans, com);
        if (!op) del(u, fa);
}

void solve() {
        com = ans = flag = 0;
        ms(son, 0); ms(cnt1, 0); ms(cnt2, 0);
        for (int i = 1; i <= n; i ++ ) G[i].clear();
        for (int u = 2; u <= n; u ++ ) {
                int v; scanf("%d", &v);
                G[u].pb(v); G[v].pb(u);
        }
        for (int i = 1; i <= n; i ++ ) {
                scanf("%d", &w[i]);
                if (!cnt1[w[i]]) com ++ ;
                cnt1[w[i]] ++ ;
        }
        dfs(1, -1);
        dfs(1, -1, false);
        printf("%d\n", ans);
}
```

1.11 Splay

```
#define ls(x) tr[x].ch[0]
#define rs(x) tr[x].ch[1]
#define fa(x) tr[x].fa
#define root tr[0].ch[1]
struct node {
        int fa , ch[2], val, rec, sum, tag;
}tr[maxn];
int tot, pointnum;
void update (int x) {tr[x].sum = tr[ls(x)].sum + tr[rs(x)].sum + tr[x].rec;}
int ident (int x) {return tr[fa(x)].ch[0] == x ? 0 : 1;}
void connect (int x, int fa, int how) {tr[fa].ch[how] = x; tr[x].fa = fa;}


inline void pushdown (int u) {
        if (tr[u].tag) {
                tr[tr[u].ch[0]].tag ^= 1;
                tr[tr[u].ch[1]].tag ^= 1;
                swap(tr[u].ch[0], tr[u].ch[1]);
                tr[u].tag = 0;
        }
}

void rotate (int x) {
        int y = fa(x), r = fa(y);
        int yson = ident(x), rson = ident(y);
        connect(tr[x].ch[yson ^ 1], y, yson);
        connect(y, x, yson ^ 1);
        connect(x, r, rson);
        update(y), update(x);
}

void splay (int x, int to) {
        to = fa(to);
        while (fa(x) != to) {
```

```
                int y = fa(x);
                if (tr[y].fa == to) rotate(x);
                else if (ident(x) == ident(y)) rotate(y), rotate(x);
                else rotate(x), rotate(x);
        }
}

int newnode (int v, int f) {
        tr[ ++ tot].fa = f;
        tr[tot].rec = tr[tot].sum = 1;
        tr[tot].val = v;
        return tot;
}

void insert (int x) {
        int now = root;
        if (!root) {newnode(x, 0); root = tot;}
        else {
                while (1) {
                        tr[now].sum ++ ;
                        if (tr[now].val == x) {tr[now].rec ++ ; splay(now, root);
return;}

                        int ne = x < tr[now].val ? 0 : 1;
                        if (!tr[now].ch[ne]) {
                                int p = newnode(x, now);
                                tr[now].ch[ne] = p;
                                splay(p, root); return;
                        }
                        now = tr[now].ch[ne];
                }
        }
}

int find (int x) { // 找到值为x的某一结点
        int now = root;
        while (1) {
                if (!now) return 0;
                if (tr[now].val == x) {splay(now, root); return now;}
                int ne = x < tr[now].val ? 0 : 1;
                now = tr[now].ch[ne];
        }
}

void del (int x) {
        int p = find(x);
        if (!p) return;
        if (tr[p].rec > 1) {tr[p].rec -- , tr[p].sum -- ; return;}
        else {
                if (!tr[p].ch[0] && !tr[p].ch[1]) {root = 0; return;}
                else if (!tr[p].ch[0]) {
                        root = tr[p].ch[1]; tr[root].fa = 0; return;
                }
                else {
                        int left = tr[p].ch[0];
                        while (tr[left].ch[1]) left = tr[left].ch[1];
                        splay(left, tr[p].ch[0]);
                        connect (tr[p].ch[1], left, 1);
                        connect (left, 0, 1);
                        update(left);
                }
        }
```

```
}

int rk (int x) { // x的排名
        int now = root, ans = 0;
        while (1) {
                if (tr[now].val == x) return ans + tr[tr[now].ch[0]].sum + 1;
                int ne = x < tr[now].val ? 0 : 1;
                if (ne == 1) ans = ans + tr[tr[now].ch[0]].sum + tr[now].rec;
                now = tr[now].ch[ne];
        }
}

int kth (int x) { //  x排名的数
        int now = root;
        while (1) {
                int used = tr[now].sum - tr[tr[now].ch[1]].sum;
                if (tr[tr[now].ch[0]].sum < x && x <= used) {
                        splay(now, root); return tr[now].val;
                    // return now; 用于区间反转
                }
                if (x < used) now = tr[now].ch[0];
                else now = tr[now].ch[1], x -= used;
        }
}

int pre (int x) {
        int now = root, ans = -inf;
        while (now) {
                if (tr[now].val <= x) ans = max(ans, tr[now].val);
                int ne = x <= tr[now].val ? 0 : 1;
                now = tr[now].ch[ne];
        }
        return ans;
}

int suf (int x) {
        int now = root, ans = inf;
        while (now) {
                if (tr[now].val >= x) ans = min(ans, tr[now].val);
                int ne = x < tr[now].val ? 0 : 1;
                now = tr[now].ch[ne];
        }
        return ans;
}

inline void reverse (int x, int y) { // 区间反转
        int l = kth(x), r = kth(y + 2);
        splay(l, 0); splay(r, l);
        tr[tr[tr[root].ch[1]].ch[0]].tag ^= 1;
}

inline void print (int u) { // 中序遍历
        pushdown(u);
        if (tr[u].ch[0]) print(tr[u].ch[0]);
        if (tr[u].val > 1 && tr[u].val < n + 2) printf("%lld ", tr[u].val - 1);
        if (tr[u].ch[1]) print(tr[u].ch[1]);
}

```

1.12 莫队

```cpp
// 普通莫队，add和del可以修改
int sq;
struct query {
    int l, r, id;
    bool operator < (const query &it) const {
        if (l / sq != it.l / sq) return l < it.l;
        if (l / sq & 1) return r < it.r;
        return r > it.r;
    }
}Q[1000010];
int q[1000010], ans[1000010], cnt[2000010], cur, l = 1, r;

inline void add (int x) {
    if (!cnt[x]) cur ++ ;
    cnt[x] ++ ;
}

inline void del (int x) {
    cnt[x] -- ;
    if (!cnt[x]) cur -- ;
}

void solve() {
    read(n);
    sq = sqrt(n);
    for (int i = 1; i <= n; i ++ ) read(q[i]);
    int query; read(query);
    for (int i = 0; i < query; i ++ ) read(Q[i].l), read(Q[i].r), Q[i].id = i;
    sort(Q, Q + query);

    for (int i = 0; i < query; i ++ ) {
            while (l > Q[i].l) add(q[ -- l]);
            while (r < Q[i].r) add(q[ ++ r]);
            while (l < Q[i].l) del(q[l ++ ]);
            while (r > Q[i].r) del(q[r -- ]);
        ans[Q[i].id] = cur;
    }
    for (int i = 0; i < query; i ++ ) printf("%d\n", ans[i]);
}


//带修莫队，块长可以选择pow(n*t, 0.33333) 或者 pow(n, 0.66666)
const int maxn = 500010;
int sq;
struct query {
    int l, r, t, id;
    bool operator < (const query &it) const {
        if (l / sq != it.l / sq) return l < it.l;
        else if (r / sq != it.r / sq) return r < it.r;
        else return t < it.t;
    }
}Q[maxn];
struct change {
    int p, col;
}c[maxn];
int q[maxn], ans[maxn], cnt[maxn * 2], cur;
int l = 1, r = 0, qcnt, ccnt;

inline void add (int x) {
    if (!cnt[x]) cur ++ ;
    cnt[x] ++ ;
}
```

```
inline void del (int x) {
    cnt[x] -- ;
    if (!cnt[x]) cur -- ;
}
inline void work (int x, int ti) {
    if (c[ti].p >= Q[x].l && c[ti].p <= Q[x].r) {
        del(q[c[ti].p]);
        add(c[ti].col);
    }
    swap(q[c[ti].p], c[ti].col);
}

void solve() {
    read(n); read(m);
    sq = pow(n, 0.66666); // 块长
    for (int i = 1; i <= n; i ++ ) {
        read(q[i]);
    }
    char op[10];
    for (int i = 1; i <= m; i ++ ) {
        scanf("%s", op);
        if (op[0] == 'Q') {
            qcnt ++ ;
            read(Q[qcnt].l); read(Q[qcnt].r);
            Q[qcnt].t = ccnt;
            Q[qcnt].id = qcnt;
        }
        else {
            ccnt ++ ;
            read(c[ccnt].p); read(c[ccnt].col);
        }
    }
    sort(Q + 1, Q + qcnt + 1);

    int now = 0;
    for (int i = 1; i <= qcnt; i ++ ) {
        while (l > Q[i].l) add(q[ -- l]);
        while (r < Q[i].r) add(q[ ++ r]);
        while (l < Q[i].l) del(q[l ++ ]);
        while (r > Q[i].r) del(q[r -- ]);

        while (now < Q[i].t) work(i, ++ now);
        while (now > Q[i].t) work(i, now --);

        ans[Q[i].id] = cur;
    }

    for (int i = 1; i <= qcnt; i ++ ) printf("%d\n", ans[i]);
}
```

1.13 二维数点

```
// 离线 + 扫描线 + 树状数组
// https://codeforces.com/gym/103687/problem/F
int n, m, k;
const int maxn = 2e5 + 10;
struct tp {
        static const int maxnum = 2e5 + 5;
        static const int maxn = 2e5 + 5;
        int tree[maxnum];
        void update (int idx, int x) {
```

```
                for (int pos = idx; pos < maxnum; pos += lowbit(pos)) tree[pos] += x;
        }
        int query (int n) {
                int ans = 0;
                for (int pos = n; pos; pos -= lowbit(pos)) ans += tree[pos];
                return ans;
        }
        int n = 0, m = 0; // n点数, m询问矩形数
        struct node1 {int x, y;} v[maxn]; // 点
        struct node2 {int x, y, id, type;} q[maxn << 2]; // 询问矩形
        static bool cmp1 (node1 &a, node1 &b) {return a.x < b.x;}
        static bool cmp2 (node2 &a, node2 &b) {return a.x < b.x;}
        int cnt = 0;
        void add (int x, int y) { v[ ++ n].x = x, v[n].y = y; } // 添加点
        void que (int x1, int y1, int x2, int y2, int i) { // 添加询问矩形, 左下角, 右上角, 询问id
                q[ ++ cnt] = {x2, y2, i, 1};
                q[ ++ cnt] = {x1 - 1, y2, i, -1};
                q[ ++ cnt] = {x2, y1 - 1, i, -1};
                q[ ++ cnt] = {x1 - 1, y1 - 1, i, 1};
                m ++ ;
        }
        void get (vector<int> &ans) { // 传入接受答案的数组
                sort(v + 1, v + n + 1, cmp1);
                sort(q + 1, q + cnt + 1, cmp2);
                int now = 1;
                for (int i = 1; i <= cnt; i ++ ) {
                        while (v[now].x <= q[i].x && now <= n) {
                                update(v[now].y, 1); now ++ ;
                        }
                        ans[q[i].id] += q[i].type * (query(q[i].y));
                }
                for (auto &x : ans) x = max(0ll, x);
        }
}t1, t2, t3, t4, t5, xx, yy;
int tr[maxn << 1];
il void add (int x, int k) {
        for (; x <= n; x += lowbit(x)) tr[x] += k;
}
il int query (int x) {
        int res = 0;
        for (; x; x -= lowbit(x)) res += tr[x];
        return res;
}

void solve() {
        scanf("%lld", &n);
        vector<int> p(n + 1), A(n + 1), B(n + 1);
        int sum = 0;
        for (int i = 1; i <= n; i ++ ) scanf("%lld", &p[i]);
        for (int i = 1; i <= n; i ++ ) {
                A[i] = query(p[i]); B[i] = p[i] - 1 - A[i];
                add(p[i], 1);
                sum += min(A[i], B[i]);
                xx.add(i, p[i]); yy.add(i, p[i]);
        }
        // de(sum);
        for (int i = 1; i <= n; i ++ ) {
                if (A[i] == B[i]) t1.add(i, p[i]);
                if (A[i] + 1 == B[i]) t2.add(i, p[i]);
                if (A[i] == B[i] + 1) t3.add(i, p[i]);
                if (A[i] + 2 <= B[i]) t4.add(i, p[i]);
```

```
                if (A[i] >= B[i] + 2) t5.add(i, p[i]);
        }
        scanf("%lld", &m);
        vector<int> x(m + 1), y(m + 1);
        vector<int> ans1(m + 1), ans2(m + 1), ans3(m + 1) , ans4(m + 1), ans5(m + 1);
        vector<int> ansxx(m + 1), ansyy(m + 1);
        for (int i = 1; i <= m; i ++ ) {
                scanf("%lld%lld", &x[i], &y[i]);
                if (x[i] > y[i]) swap(x[i], y[i]);
                int l = x[i] + 1, r = y[i] - 1;
                int mn = min(p[x[i]], p[y[i]]), mx = max(p[x[i]], p[y[i]]);
                t1.que(l, mn, r, mx, i);
                t2.que(l, mn, r, mx, i);
                t3.que(l, mn, r, mx, i);
                t4.que(l, mn, r, mx, i);
                t5.que(l, mn, r, mx, i);
                xx.que(x[i], 1, y[i], p[x[i]] - 1, i);
                yy.que(x[i], 1, y[i], p[y[i]] - 1, i);
        }
        t1.get(ans1); t2.get(ans2); t3.get(ans3); t4.get(ans4); t5.get(ans5);
        xx.get(ansxx); yy.get(ansyy);
        vector<int> ans(m + 1);
        for (int i = 1; i <= m; i ++ ) {
                if (p[x[i]] < p[y[i]]) {
                        ans[i] += sum - ans1[i] - ans2[i] - ans4[i] + ans5[i];
                }
                else ans[i] += sum - ans1[i] - ans3[i] + ans4[i] - ans5[i];
                ans[i] -= min(A[x[i]], B[x[i]]); ans[i] -= min(A[y[i]], B[y[i]]);
                ans[i] += min(A[x[i]] + ansxx[i], B[x[i]] - ansxx[i]);
                ans[i] += min(A[y[i]] - ansyy[i], B[y[i]] + ansyy[i]);
        }
        for (int i = 1; i <= m; i ++ ) printf("%lld\n", ans[i]);
}

```

1.14 珂朵莉树

```
//https://codeforces.com/contest/896/problem/C
struct Node {
        int l, r;
        mutable ll val;
        bool operator < (const Node &it) const { return l < it.l; }
        Node (int L, int R = -1, ll V = 0) : l(L), r(R), val(V) {}
};
struct Chtholly {
        #define IT set<Node>::iterator
        set<Node> s;
        IT split (int pos) {
                IT it = s.lower_bound(Node(pos));
                if (it != s.end() && it->l == pos) return it;
                it -- ;
                int l = it->l, r = it->r;
                ll val = it->val;
                s.erase(it); s.insert(Node(l, pos - 1, val));
                return s.insert(Node(pos, r, val)).fi;
        }
        void assign (int l, int r, int val) {
                IT it2 = split(r + 1), it1 = split(l);
                s.erase(it1, it2); s.insert(Node(l, r, val));
        }
        void add (int l, int r, int val) {

```

```
                IT it2 = split(r + 1), it1 = split(l);
                for (IT it = it1; it != it2; it ++ ) {
                        it->val += val;
                }
        }
        ll kth (int l, int r, int k) {
                IT it2 = split(r + 1), it1 = split(l);
                vector<pair<ll, int>> res; res.clear();
                for (IT it = it1; it != it2; it ++ ) {
                        res.pb(pair<ll, int>(it->val, it->r - it->l + 1));
                }
                sort(all(res));
                for (auto [v, cnt] : res) {
                        k -= cnt;
                        if (k <= 0) return v;
                }
        }
        ll qmi (int a, int x, ll y) {
                ll b = 1; a %= y;
                while (x) {
                        if (x & 1) b = (b * a) % y;
                        a = a * a % y;
                        x >>= 1;
                }
                return b;
        }
        ll query (int l, int r, int x, ll y) {
                IT it2 = split(r + 1), it1 = split(l);
                ll res = 0;
                for (IT it = it1; it != it2; it ++ ) {
                        res = (res + (it->r - it->l + 1) * qmi(it->val, x, y) % y) % y;
                }
                return res;
        }
};

```

1.15 李超树

```
constexpr int N = 2e5 + 10;
class LC_SegT {
    struct Func {
        ll k, b;
        Func(ll k = 0, ll b = 0) : k(k), b(b) {}
        ll operator()(const ll x) const {
            return k * x + b;
        }
    } tr[N << 2];
#define lc (u << 1)
#define rc (u << 1 | 1)
    void insert(int u, int L, int R, int l, int r, Func k) {
        int mid = L + R >> 1;
        if (L >= l && R <= r) {
            Func &f = tr[u];
            int res = (k(L) > f(L)) + (k(R) > f(R));
            if (res == 2) {
                f = k;
            } else if (res) {
                if (k(mid) > f(mid)) {
                    swap(f, k);
```

```
                }
                if (k(L) > f(L)) {
                    insert(lc, L, mid, l, r, k);
                } else {
                    insert(rc, mid + 1, R, l, r, k);
                }
            }
        } else {
            if (l <= mid) {
                insert(lc, L, mid, l, r, k);
            }
            if (r > mid) {
                insert(rc, mid + 1, R, l, r, k);
            }
        }
    }
#undef lc
#undef rc
public:
    int n;
    void init(const int Lim) {
        for (n = 1; n < Lim; n <<= 1) ;
    }
    void insert(Func k) {
        insert(1, 1, n, 1, n, k);
    }
    ll query(int x) {
        ll ret = 0;
        for (int p = x + n - 1; p; p >>= 1) {
            ret = max(ret, tr[p](x));
        }
        return ret;
    }
}T[4];
```

2. 图论

2.1 最短路

2.1.1 朴素Dijkstra O(n^2)

```
int g[N][N]; // 邻接矩阵存稠密图
int dist[N]; // 每个点距离点1的距离
bool st[N]; // 判断该点是否确定

int dijkstra()
{
    memset(dist, 0x3f, sizeof dist);

    dist[1] = 0;

    for (int i = 0; i < n; i ++ ){ // n个点循环n次

        int t = -1;

        for (int j = 1; j <= n; j ++ ){ // 找未确定的点中dist最小的点
            if (!st[j] && (t == -1 || dist[j] < dist[t]))
            t = j;
        }
```

```

        if (t == n && dist[t] != 0x3f3f3f3f) return dist[t];
        else if (t == n && dist[t] == 0x3f3f3f3f) return -1; // 提前结束循环的优化

        st[t] = true;

        for (int j = 1; j <= n; j ++ )// 用t更新其他未确定点的距离
            dist[j] = min(dist[j], dist[t] + g[t][j]);
    }

    if (dist[n] == 0x3f3f3f3f) return -1;
    else return dist[n];
}

int main()
{
    cin >> n >> m;

    memset(g, 0x3f, sizeof g);

    while (m -- ){
        int a, b, c;
        cin >> a >> b >> c;
        g[a][b] = min(g[a][b], c);
    }

    printf("%d\n", dijkstra());

    return 0;
}
```

2.1.2 堆优化Dijkstra O(mlogn)

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

typedef pair<int, int> PII;

const int N = 1e6 + 10;

int n, m;
int h[N], w[N], e[N], ne[N], idx;
int dist[N];
bool st[N];

int add(int a, int b, int c)
{
    e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++ ;
}

// 求1号点到n号点的最短距离，如果不存在，则返回-1
int dijkstra()
{
    memset(dist, 0x3f, sizeof dist);
    dist[1] = 0;
    priority_queue<PII, vector<PII>, greater<PII>> heap;
    heap.push({0, 1}); // first存储距离，second存储节点编号

    while (heap.size()){
        auto t = heap.top();
```

```
        heap.pop();

        int ver = t.second, distance = t.first;

        if (st[ver]) continue; // 防止产生冗余
        st[ver] = true;

        for (int i = h[ver]; i != -1; i = ne[i]){
            int j = e[i];
            if (dist[j] > dist[ver] + w[i]){
                dist[j] = dist[ver] + w[i];
                heap.push({dist[j], j});
            }
        }

    }

    if (dist[n] == 0x3f3f3f3f) return -1;
    else return dist[n];
}

int main()
{
    cin >> n >> m;
    memset(h, -1, sizeof h);
    while (m -- ){
        int a, b, c;
        cin >> a >> b >> c;
        add(a, b, c);
    }

    cout << dijkstra() << endl;

    return 0;
}
```

2.1.3 Bellman-ford O(nm)

```
// 有边数限制的最短路

struct Edge
{
    int a, b ,c;
}edges[M]; // 结构体存边

int n, m, k;
int dist[N];
int backup[N]; // 题目有特殊的边数限制，因此在更新时只能更新上次备份，否则会出现串联

void bellman_ford()
{
    memset(dist, 0x3f, sizeof dist);

    dist[1] = 0;
    for (int i = 0; i < k; i ++ ){
        memcpy(backup, dist, sizeof dist); // 每次都将上次的dist存到备份里
        for (int j = 0; j < m; j ++ ){
            auto e = edges[j];
            dist[e.b] = min(dist[e.b], backup[e.a] + e.c);
        }
```

```cpp
    }
}

int main()
{
    cin >> n >> m >> k;

    for (int i = 0; i < m ; i ++ ){
        int a, b, c;
        cin >> a >> b >> c;
        edges[i] = {a, b, c};
    }

    bellman_ford();

    if (dist[n] > 0x3f3f3f3f / 2) printf("impossible\n");
    else printf("%d\n", dist[n]);

    return 0;
}



// 无边数限制的最短路
#include <bits/stdc++.h>
using namespace std;

const int M = 1000010;

struct Edge
{
    int a, b ,c;
}edges[M]; // 结构体存边

int n, m, k;
int dist[M];


int bellman_ford()
{
    memset(dist, 0x3f, sizeof dist);

    dist[1] = 0;
    //// 如果第n次迭代仍然会松弛三角不等式，就说明存在一条长度是n+1的最短路径，由抽屉原理，路径中至少存在两个相同的点，说明图中存在负权回路。
    for (int i = 0; i < n; i ++ ){
        for (int j = 0; j < m; j ++ ){
            auto e = edges[j];
            dist[e.b] = min(dist[e.b], dist[e.a] + e.c);
        }
    }

    if (dist[n] == 0x3f3f3f3f) return -1;
    return dist[n];
}

int main()
{
    cin >> n >> m;

    for (int i = 0; i < m ; i ++ ){
        int a, b, c;
```

```
        cin >> a >> b >> c;
        edges[i] = {a, b, c};
    }

    if (bellman_ford() == -1) printf("impossible\n");
    else printf("%d\n", dist[n]);

    return 0;
}
```

2.1.4 SPFA O(nm)

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

const int N = 100010;

int n, m;
int h[N], w[N], e[N], ne[N], idx;
int dist[N];
bool st[N];

void add(int a, int b, int c)
{
    e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++ ;
}

int spfa()
{
    memset(dist, 0x3f, sizeof dist);
    dist[1] = 0;

    queue<int> q;
    q.push(1);
    st[1] = true;

    while (q.size()){
        int t = q.front();
        q.pop();

        st[t] = false;

        for (int i = h[t]; i != -1; i = ne[i]){
            int j = e[i];
            if (dist[j] > dist[t] + w[i]){
                dist[j] = dist[t] + w[i];
                if (!st[j]){
                    q.push(j);
                    st[j] = true;
                }
            }
        }
    }
    printf("%d\n", dist[n]);
    return dist[n];
}

int main()
{
    cin >> n >> m;
```

```
    memset(h, -1, sizeof h);

    while (m -- ){
        int a, b, c;
        cin >> a >> b >> c;
        add(a, b, c);
    }

    if (spfa() == 0x3f3f3f3f) printf("impossible");
    else printf("%d\n", spfa());

    return 0;
}


// 判断负权环
#include <bits/stdc++.h>
using namespace std;

const int N = 2010, M = 10010;

int n, m;
int h[N], w[M], e[M], ne[M], idx;
int dist[N], cnt[N];
bool st[N];

void add(int a, int b, int c)
{
    e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++ ;
}
// 优化：queue改stack，或入队总数大于2*n即有负环
bool spfa()
{
    queue<int> q;

    for (int i = 1; i <= n; i ++ ){
        st[i] = true;
        q.push(i);
    }

    while (q.size()){
        int t = q.front();
        q.pop();

        st[t] = false;

        for (int i = h[t]; i != -1; i = ne[i]){
            int j = e[i];
            if (dist[j] > dist[t] + w[i]){
                dist[j] = dist[t] + w[i];
                cnt[j] = cnt[t] + 1;

                if (cnt[j] >= n) return true;

                if (!st[j]){
                    q.push(j);
                    st[j] = true;
                }
            }
        }
    }
    return false;
```

```cpp
}

int main()
{
    cin >> n >> m;
    memset (h, -1, sizeof h);

    while (m -- ){
        int a, b, c;
        cin >> a >> b >> c;
        add(a, b, c);
    }

    if (spfa()) cout << "Yes" << endl;
    else cout << "No" << endl;

    return 0;
}

// SLF优化
int dist[maxn];
int cnt[maxn];
bool st[maxn];
vector<PII> G[maxn];

bool spfa(int s)
{
    deque<int> q;
    dist[s] = 0;
    q.push_back(s);
    cnt[s] ++ ;
    st[s] = 1;

    while (q.size()) {
        int t = q.front();
        q.pop_front();
        st[t] = 0;

        for (auto it : G[t]) {
            int j = it.first, w = it.second;
            if (dist[j] > dist[t] + w) {
                dist[j] = dist[t] + w;
                if (!st[j]) {
                    if (!q.empty() && dist[j] > dist[q.front()]) {
                        q.push_back(j);
                    }
                    else q.push_front(j);
                    cnt[j] ++ ;
                    if (cnt[j] > n) return true;
                    st[j] = 1;
                }
            }
        }
    }
    return false;
}
```

2.1.5 Floyd O(n^3)

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

const int N = 210, INF = 1e9;

int n, m, Q;
int d[N][N]; // i到j距离

void floyd()
{
    for (int k = 1; k <= n; k ++ ){
        for (int j = 1; j <= n; j ++ ){
            for (int i = 1; i <= n; i ++ ){
                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
            }
        }
    }
}

int main()
{
    cin >> n >> m >> Q;

    for (int i = 1; i <= n; i ++ ){
        for (int j = 1; j <= n; j ++ ){
            if (i == j) d[i][j] = 0;
            else d[i][j] = INF;
        }
    }

    while (m -- ){
        int a, b, c;
        cin >> a >> b >> c;
        d[a][b] = min(d[a][b], c);
    }

    floyd();

    while (Q -- ){
        int a, b;
        cin >> a >> b;
        int t =d[a][b];
        if (t > INF / 2) printf("impossible");
        else printf("%d\n", t);
    }

    return 0;
}

// 传递闭包
for (int k = 0; k < n; k ++ ) {
    for (int i = 0; i < n; i ++ ) {
        for (int j = 0; j < n; j ++ ) {
            d[i][j] |= d[i][k] && d[k][j];
        }
    }
}


// 求最小环
```

```cpp
int n, m;
int d[maxn][maxn], g[maxn][maxn];
int pos[maxn][maxn];
int path[maxn], cnt;

void get_path (int i, int j) {
        if (pos[i][j] == 0) return;

        int k = pos[i][j];
        get_path(i, k);
        path[cnt ++ ] = k;
        get_path(k, j);
}

int main () {
        cin >> n >> m;

        memset(g, 0x3f, sizeof g);
        for (int i = 1; i <= n; i ++ ) g[i][i] = 0;

        while (m -- ) {
                int a, b, c;
                cin >> a >> b >> c;
                g[a][b] = g[b][a] = min(g[a][b], c);
        }

        int res = inf;
        memcpy(d, g, sizeof d);
        for (int k = 1; k <= n; k ++ ) {
                for (int i = 1; i < k; i ++ ) {
                        for (int j = i + 1; j < k; j ++ ) {
                                if((long long)d[i][j] + g[j][k] + g[k][i] < res) {
                                        res = d[i][j] + g[j][k] + g[k][i];
                                        cnt = 0;
                                        path[cnt ++] = k;
                                        path[cnt ++] = i;
                                        get_path(i, j);
                                        path[cnt ++] = j;
                                }
                        }
                }
                for (int i = 1; i <= n; i ++ ) {
                        for (int j = 1; j <= n; j ++ ) {
                                if (d[i][j] > d[i][k] + d[k][j]) {
                                        d[i][j] = d[i][k] + d[k][j];
                                        pos[i][j] = k;
                                }
                        }
                }
        }

        if (res == inf) puts("No solution.");
        else {
                for (int i = 0; i < cnt; i ++ ) cout << path[i] << ' ';
                cout << endl;
        }
}


```

2.1.6 路径还原

```cpp
//以朴素dijkstra为例，记录一个path数组，当dist数组被更新时，就同步跟新path数组

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

const int maxn = 510;
const int inf = 0x3f3f3f3f;
int g[maxn][maxn];
int st[maxn];
int dist[500010];
int path[500010]; // 记录走来的路径
int n, m;

int dijkstra()
{
    memset(dist, 0x3f, sizeof dist);
    memset(path, -1 ,sizeof path);

    dist[1] = 0;

    for (int i = 1; i <= n; i ++ ){
        int t = -1;
        for (int j = 1; j <= n; j ++ ){
            if (!st[j] && (t == -1 || dist[j] < dist[t])) t = j;
        }

        st[t] = 1;

        for (int j = 1; j <= n; j ++ ){
            if (dist[j] > dist[t] + g[t][j]){
                dist[j] = dist[t] + g[t][j];
                path[j] = t; // 记录
            }
        }
    }
    return dist[n];
}

vector<int> get_path(int x){
    vector<int> p;
    for (; x != -1; x = path[x]) p.push_back(x);
    reverse(p.begin(), p.end()); //p中存下的是n到1的顺序，我们逆反一下顺序。
    return p;
}

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i ++ ){
        for (int j = 1; j <= n; j ++ ){
            g[i][j] = (i == j) ? 0 : inf;
        }
    }

    for (int i = 1; i <= m; i ++ ){
        int a, b, c;
        cin >> a >> b >> c;
        g[a][b] = min(g[a][b], c);
    }

```

```
    printf("%d\n", dijkstra());
    vector<int> p = get_path(n);
    for (auto it : p){
        printf("%d ", it);
    }
    return 0;
}
```

2.1.7 最短路计数

```
int cnt[maxn]; // 长度为i的路径的数量。


while (q.size()) { // 以bfs为例
                auto t = q.front();
                q.pop();

                for (auto it : G[t]) {
                        if (dist[it] > dist[t] + 1) {
                                dist[it] = dist[t] + 1;
                                cnt[it] = cnt[t];
                                q.push(it);
                        }
                        else if (dist[it] == dist[t] + 1) cnt[it] = (cnt[it] + cnt[t]) %
mod;
                }
        }
```

2.2 最小生成树

2.2.1 Prim

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

const int N = 510, INF = 0x3f3f3f3f;

int n, m;
int g[N][N]; // 邻接矩阵存图
int dist[N]; // 1~i的距离
bool st[N];

int prim()
{
    memset(dist, 0x3f, sizeof dist);

    int res = 0;
    for (int i = 0; i < n; i ++ ){
        int t = -1;

        for (int j = 1; j <= n; j ++ ){
            if (!st[j] && (t == -1 || dist[t] > dist[j]))
            t = j;
        }

        if (i && dist[t] == INF) return INF;

        if (i) res += dist[t];
```

```
        st[t] = true;

        for (int j = 1; j <= n; j ++ ) dist[j] = min(dist[j], g[t][j]);
    }

    return res;
}

int main()
{
    cin >> n >> m;

    memset(g, 0x3f, sizeof g);

    while (m -- ){
        int a, b, c;
        cin >> a >> b >> c;
        g[a][b] = g[b][a] = min(g[a][b], c); // 无向图
    }

    int t = prim();

    if (t == INF) printf("impossible");
    else printf("%d\n", t);

    return 0;
}
```

2.2.2 Kruskal

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

const int N = 100010, M = 200010, INF = 0x3f3f3f3f;

int n, m;
int p[N]; // 并查集

struct Edge
{
    int a, b, w;
}edges[M];

bool cmp(Edge a, Edge b)
{
    return a.w < b.w;
}

int find (int x)
{
    if (p[x] != x) p[x] = find(p[x]);
    return p[x];
}

int kruskal()
{
    sort(edges, edges + m, cmp);

    for (int i = 0; i <= n; i ++ ) p[i] = i; // 并查集的初始操作

```

```cpp
    int res = 0, cnt = 0; // cnt表示连通的边数
    for (int i = 0; i < m; i ++ ){
        int a = edges[i].a, b = edges[i].b, w = edges[i].w;

        a = find(a), b = find(b);
        if (a != b){
            p[a] = b; // 将边加入集合
            res += w;
            cnt ++ ;
        }
    }
    if (cnt < n - 1) return INF;
    return res;
}

int main()
{
    cin >> n>> m;

    for (int i = 0; i < m; i ++ ){
        int a, b, w;
        cin >> a >> b >> w;
        edges[i] = {a, b, w};
    }

    int t = kruskal();

    if (t == INF) printf("impossible"); // 不连通
    else printf("%d", t);

    return 0;
}

// 次小生成树
#include <bits/stdc++.h>
using namespace std;

#define int long long
const int maxn = 510, maxm = 10010;
typedef pair<int, int> PII;
int n, m;
struct node {
        int a, b, w;
        bool f;
        bool operator < (const node &it) const {
                return w < it.w;
        }
}e[maxm];
int p[maxn];
int d1[maxn][maxn], d2[maxn][maxn];
vector<PII> G[maxn];

int find (int x) {
        if (p[x] != x) p[x] = find(p[x]);
        return p[x];
}

void dfs (int u, int fa, int maxd1, int maxd2, int d1[], int d2[]) {
        d1[u] = maxd1, d2[u] = maxd2;
        for (auto it : G[u]) {
                int j = it.first, w = it.second;
                if (j != fa) {
```

```
                        int td1 = maxd1, td2 = maxd2;
                        if (w > td1) td2 = td1, td1 = w;
                        else if (w < td1 && w > td2) td2 = w;
                        dfs(j, u, td1, td2, d1, d2);
                }
        }
}

signed main () {
        cin >> n >> m;
        for (int i = 1; i <= m; i ++ ) {
                int a, b, w;
                cin >> a >> b >> w;
                e[i] = {a, b, w};
        }

        sort(e + 1, e + m + 1);
        for (int i = 1; i <= n; i ++ ) p[i] = i;

        int sum = 0;
        for (int i = 1; i <= m; i ++ ) {
                int a = e[i].a, b = e[i].b, w = e[i].w;
                int pa = find(a), pb = find(b);
                if (pa != pb) {
                        p[pa] = pb;
                        sum += w;
                        G[a].push_back({b, w});
                        G[b].push_back({a, w});
                        e[i].f = 1;
                }
        }

        for (int i = 1; i <= n; i ++ ) {
                dfs(i, -1, -1e9, -1e9, d1[i], d2[i]);
        }

        int res = 1e18;
        for (int i = 1; i <= m; i ++ ) {
                if (!e[i].f) {
                        int a = e[i].a, b = e[i].b, w = e[i].w;
                        int t;
                        if (w > d1[a][b]) {
                                t = sum + w - d1[a][b];
                        }
                        else if (w > d2[a][b]) {
                                t = sum + w - d2[a][b];
                        }
                        res = min(res, t);
                }
        }
        cout << res << endl;
}
```

2.3 二分图匹配

2.3.1 染色法判断二分图

```cpp
/*
染色法的实现思路（DFS）：
1.用1，2代表两个颜色，0代表未染色，任选一个点染成1或2

2.遍历所有点，每次将未染色的点进行dfs

3.若染色失败即break/return
*/

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

const int N = 100010, M = 200010;

int n, m;
int h[N], e[M], ne[M], idx;
int color[N];

void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
}

bool dfs(int u, int c)
{
    color[u] = c; // 染色
    for (int i = h[u]; i != -1; i = ne[i]){
        int j = e[i];
        if (!color[j]){
            if (!dfs(j, 3 - c)) return false; // 如果在dfs递归的过程中出现染色失败，则整个图都不是二分图
        }
        else if (color[j] == c) return false; // 如果一条边的两端点同种颜色，则染色失败
    }

    return true; // 无染色错误则染色成功
}

int main()
{
    cin >> n >> m;

    memset(h, -1, sizeof h);

    while (m -- ){
        int a, b;
        cin >> a >> b;
        add(a, b), add(b, a);
    }

    bool flag = true;
    for (int i = 1; i <= n; i ++ ){ // 遍历所有点，因为二分图不一定是连通图
        if (!color[i]){
            if (!dfs(i, 1)){
                flag = false;
                break;
            }
        }
    }
    if (flag) printf("Yes\n");
```

```
    else printf("No");

    return 0;
}
```

2.3.2 匈牙利算法判断最大匹配

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

const int N = 510, M = 100010;

int n1, n2, m;
int h[N], e[M], ne[M], idx;
int match[N]; // match[a] = b, 表示点a目前匹配了b
bool st[N];  // st[a] = true, 表示点a目前已经有预定

void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;
}

bool find(int x) // 注意与并查集的find函数区别，为x找一个匹配，或x的匹配点被别人预定，x要重新找一个匹配
{
    // 匹配成功返回true
    for (int i = h[x]; i != -1; i = ne[i]){
        int j = e[i];
        if (!st[j]){ // 该点目前没有被匹配
            st[j] = true; // 预定该点
            if (match[j] == 0 || find(match[j])){ // j点没有匹配，或与j匹配的点可以更换匹配
                match[j] = x;
                return true;
            }
        }
    }
    return false;
}

int main()
{
    scanf("%d%d%d", &n1, &n2, &m);

    memset(h, -1, sizeof h);

    while (m -- ){
        int a, b;
        cin >> a >> b;
        add(a, b); // 从左集合找右集合，只存一条边也可以
    }

    int res = 0;
    for (int i = 0; i <= n1; i ++ ){ // 二分图不一定连通，因此要为所有点尝试匹配
        memset(st, false, sizeof st);
        if (find(i)) res ++;
    }
    cout << res << endl;

    return 0;
}
```

```

最小点覆盖 = 最大匹配
最大独立集 = n - 最大匹配
最小路径点覆盖= n - 最大匹配

```

2.4 拓扑排序

```cpp
const int N = 1e5 + 10;
int e[N],ne[N],h[N],idx;
int d[N];// d 代表每个元素的入度
int top[N];// top是拓扑排序的序列
int cnt = 1; // cnt代表top中有多少个元素
int n, m;
void add(int a,int b){
    e[idx] = b;
    ne[idx] = h[a];
    h[a] = idx ++;
}
bool topsort(){
    queue<int> q;
    int t;
    for(int i = 1;i <= n; ++i)// 将所有入度为0的点加入队列
        if(d[i] == 0) q.push(i);
    while(q.size()){
        t = q.front();//每次取出队列的首部
        top[cnt] = t;//加入到 拓扑序列中
        cnt ++; // 序列中的元素 ++
        q.pop();
        for(int i = h[t];i != -1; i = ne[i]){
            // 遍历 t 点的出边
            int j = e[i];
            d[j] --;// j 的入度 --
            if(d[j] == 0) q.push(j); //如果 j 入度为0，加入队列当中
        }
    }
    if(cnt < n) return 0;
    else return 1;

}
int main(){
    int a,b;
    cin >> n >> m;
    memset(h,-1,sizeof h);
    while(m--){
        cin >> a >> b;
        add(a,b);
        d[b] ++;// a -> b , b的入度++
    }
    if(topsort() == 0) cout << "-1"; // 序列不合法
    else {
        for(int i = 1;i <= n; ++i){
            cout << top[i] <<" ";
        }
    }
    return 0;
}
```

2.5 倍增LCA

```cpp
constexpr int MAXP = 30;
int pa[MAXP][N], dep[N];
void dfs (int u, int fa, int d) {
    pa[0][u] = fa;
    for (int i = 1; i < MAXP; i++) {
        pa[i][u] = pa[i - 1][pa[i - 1][u]];
    }
    dep[u] = d;
    for (auto v : g[u]) {
        if (v == fa) {
            continue;
        }
        dfs(v, u, d + 1);
    }
}
int lca (int x, int y) {
    if (dep[x] < dep[y]) swap(x, y);
    for (int i = MAXP - 1; i >= 0; i--) {
        if (dep[pa[i][x]] >= dep[y]) {
            x = pa[i][x];
        }
    }
    if (x == y) return x;
    for (int i = MAXP - 1; i >= 0; i--) {
        if (pa[i][x] != pa[i][y]) {
            x = pa[i][x], y = pa[i][y];
        }
    }
    return pa[0][x];
}
dfs(1, 0, 1);
```

2.6 Tarjan

2.6.1 SCC

```cpp
// 求scc的时候不要连无向边
vector<int> scc[N];
int cnt, idx;
int dfn[N], ins[N], low[N], bel[N];
stack<int> stk;
void tarjan (int u) {
    dfn[u] = low[u] = ++idx;
    ins[u] = true;
    stk.push(u);
    for (auto j : g[u]) {
        if (!dfn[j]) {
            tarjan(j);
            low[u] = min(low[u], low[j]);
        } else {
            if (ins[j]) low[u] = min(low[u], dfn[j]);
        }
    }
    if (dfn[u] == low[u]) {
        ++cnt;
        while (true) {
            int v = stk.top();
            ins[v] = false;
            bel[v] = cnt;
            scc[cnt].push_back(v);
            stk.pop();
```

```
            if (v == u) break;
        }
    }
}
```

2.6.2 BCC_V

```
//去掉一个点后，连通块数量增加，不一定联通
//luoguP3388
int dfn[maxn], low[maxn], idx, sz;
bool cut[maxn];
void dfs (int u, int fa, int rt) {
    dfn[u] = low[u] = ++ idx;
    int ch = 0;
    for (auto v : G[u]) {
        if (!dfn[v]) {
            dfs(v, u, rt);
            ch ++ ;
            low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u]) cut[u] = 1;
        } else if (v != fa) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (u == rt && ch <= 1) cut[u] = 0;
    sz += cut[u];
}
```

2.6.3 BCC_E

```
vector<pii> G[maxn];
int dfn[maxn], low[maxn], idx, sz;
vector<int> bridge;
// stack<int> stk;
// vector<int> bcc[maxn];
// int bel[maxn], cnt;
void dfs (int u, int id) {
    dfn[u] = low[u] = ++ idx;
    // stk.push(u);
    for (auto [v, id2] : G[u]) {
        if (!dfn[v]) {
            dfs(v, id2);
            low[u] = min(low[u], low[v]);
            if (low[v] == dfn[v]) bridge.pb(id2);
        } else if (id != id2) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    // if (dfn[u] == low[u]) {
    //  ++ cnt;
    //  while (true) {
    //      int v = stk.top();
    //      bcc[cnt].push_back(v);
    //      bel[v] = cnt;
    //      stk.pop();
    //      if (v == u) break;
    //  }
    // }
}
```

2.7 差分约束

```
/*
xa - xb <= c  ==>  add(a, b, c);
x >= c  ==>  x0 - x <= -c ==> add(0, x, -c);
输出 x - x0
跑最短路求得最大值 x <= c ==> x = c
如果要最小值，考虑使得x' = -x，求得x'的最大值，即为x的最小值
则-xa - (-xb) <= c  ==>  xb - xa <= c  ==>  add(b, a, c); 方向相反
最后输出x0 - x;

xa == xb  xa <= xb && xb <= xa  ==> add(a, b, 0), add(b, a, 0);
xa < xb  xa - xb <= -1  ==>  add(a, b, -1);
xa <= xb  xa - xb <= 0  ==>  add(a, b, 0);
*/
```

2.8 树哈希

```
//这棵树中最多能选出多少个互不同构的子树
vector<int> G[maxn];
mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());
ull h[maxn], bas = rnd();
ull H(ull x) {
    return x * x * x * 19890535 + 19260817;
}
ull F(ull x) {
    return H(x & ((1ll << 32) - 1)) + H(x >> 32);
}
void dfs (int u, int fa) {
    h[u] = bas;
    for (auto j : G[u]) {
        if (j != fa) {
            dfs(j, u);
            h[u] += F(h[j]);
        }
    }
}
void solve() {
    if (bas % 2 == 0) bas ++ ;
    cin >> n;
    for (int i = 1; i < n; i ++ ) {
        int u, v; cin >> u >> v;
        G[u].pb(v); G[v].pb(u);
    }
    dfs(1, -1);
    sort(h + 1, h + n + 1);
    int ans = unique(h + 1, h + n + 1) - h - 1;
    cout << ans << endl;
}
```

2.9 虚树

```
//https://www.luogu.com.cn/problem/P2495
vector<pii> G[maxn];
vector<int> E[maxn];
const int MAXP = 30;
int pa[maxn][MAXP], dep[maxn];
int stk[maxn], tp;
```

```cpp
vector<int> vec;
int minv[maxn], query[maxn], dfn[maxn], tot;
il void dfs (int u, int fa, int d) {
    pa[u][0] = fa;
    for (int i = 1; i < MAXP; i ++ ) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    dep[u] = d;
    dfn[u] = ++ tot;
    for (auto [j, w] : G[u]) {
        if (j != fa) {
            minv[j] = min(minv[u], w);
            dfs(j, u, d + 1);
        }
    }
}
il int lca (int x, int y) {
    if (dep[x] < dep[y]) swap(x, y);
    for (int i = MAXP - 1; i >= 0; i -- ) if (dep[pa[x][i]] >= dep[y]) x = pa[x][i];
    if (x == y) return x;
    for (int i = MAXP - 1; i >= 0; i -- ) if (pa[x][i] != pa[y][i]) x = pa[x][i], y = pa[y][i];
    return pa[x][0];
}
il int dp (int u) {
    int sum = 0, now = 0;
    for (auto j : E[u]) {
        sum += dp(j);
    }
    if (query[u]) now = minv[u];
    else now = min(minv[u], sum);
    query[u] = false;
    return now;
}
il int build_VT() {
    E[1].clear(); stk[tp = 1] = 1;
    for (auto j : vec) {
        E[j].clear();
        int a = lca(j, stk[tp]);
        if (a == stk[tp]) {
            stk[ ++ tp] = j;
            continue;
        }
        while (dep[stk[tp - 1]] > dep[a]) {
            E[stk[tp - 1]].pb(stk[tp]);
            tp -- ;
        }
        if (a == stk[tp - 1]) {
            E[a].pb(stk[tp]); tp -- ;
        } else {
            E[a].clear(); E[a].pb(stk[tp]);
            stk[tp] = a;
        }
        stk[ ++ tp] = j;
    }
    while (tp > 1) {
        E[stk[tp - 1]].pb(stk[tp]);
        tp -- ;
    }
    return dp(1);
}
void solve() {
    minv[1] = 1e18;
    cin >> n;
```

```
    for (int i = 1; i < n; i ++ ) {
        int u, v, w; cin >> u >> v >> w;
        G[u].pb({v, w}); G[v].pb({u, w});
    }
    dfs(1, 0, 1);
    cin >> m;
    while (m -- ) {
        int num; cin >> num;
        vec.clear();
        for (int i = 1; i <= num; i ++ ) {
            int x; cin >> x; vec.pb(x);
            query[x] = true;
        }
        sort(all(vec), [&] (int a, int b) {
            return dfn[a] < dfn[b];
        });
        cout << build_VT() << endl;
    }
}
```

2.10 2-SAT

```
//https://www.luogu.com.cn/problem/P4171
/*
存边时记得存下逆否命题的边，a -> ~b，b -> ~a都要连
ab至少选一：~a -> b，~b -> a
ab不能同时选-> a->~b，b->~a
ab都选：a -> b，~b -> ~a
a不能选：a -> ~a
*/
vector<int> G[maxn << 1];
int cnt, idx;
int dfn[maxn], ins[maxn], low[maxn], bel[maxn];
stack<int> stk;
il void tarjan (int u) {
    dfn[u] = low[u] = ++ idx;
    ins[u] = true;
    stk.push(u);
    for (auto j : G[u]) {
        if (!dfn[j]) {
            tarjan(j);
            low[u] = min(low[u], low[j]);
        } else {
            if (ins[j]) low[u] = min(low[u], dfn[j]);
        }
    }
    if (dfn[u] == low[u]) {
        ++ cnt;
        while (true) {
            int v = stk.top();
            ins[v] = false;
            bel[v] = cnt;
            stk.pop();
            if (v == u) break;
        }
    }
}
void solve() {
    cin >> n >> m;
    cnt = idx = 0;
    for (int i = 0; i <= 2 * (n + 1); i ++ ) {
```

```
        G[i].clear(); dfn[i] = 0;
    }
    for (int i = 1; i <= m; i ++ ) {
        string a, b; cin >> a >> b;
        int u = (stoi(a.substr(1)) - 1) * 2 + (a[0] == 'h'), v = (stoi(b.substr(1)) - 1) * 2 + (b[0] == 'h');
        G[u ^ 1].pb(v); G[v ^ 1].pb(u);
    }
    for (int i = 0; i < 2 * n; i ++ ) {
        if (!dfn[i]) tarjan(i);
    }
    for (int i = 0; i < n; i ++ ) {
        if (bel[2 * i] == bel[2 * i + 1]) {
            cout << "BAD" << endl; return;
        }
    }
    cout << "GOOD" << endl;
}
```

```
//https://darkbzoj.cc/problem/2199

const int maxn = 2e3 + 10;
vector<int> G[maxn << 1];
int cnt, idx;
int dfn[maxn], ins[maxn], low[maxn], bel[maxn];
stack<int> stk;
int vis[maxn];
il void tarjan (int u) {
    dfn[u] = low[u] = ++ idx;
    ins[u] = true;
    stk.push(u);
    for (auto j : G[u]) {
        if (!dfn[j]) {
            tarjan(j);
            low[u] = min(low[u], low[j]);
        } else {
            if (ins[j]) low[u] = min(low[u], dfn[j]);
        }
    }
    if (dfn[u] == low[u]) {
        ++ cnt;
        while (true) {
            int v = stk.top();
            ins[v] = false;
            bel[v] = cnt;
            stk.pop();
            if (v == u) break;
        }
    }
}
void dfs (int u) {
    vis[u] = 1;
    for (auto v : G[u]) {
        if (!vis[v]) dfs(v);
    }
}
bool check (int s, int t) {
    ms(vis, 0);
    dfs(s);
    return vis[t];
}
void solve() {
```

```
    cin >> n >> m;
    for (int i = 1; i <= m; i ++ ) {
        int u, v; string a, b;
        cin >> u >> a >> v >> b;
        u --, v -- ;
        u = 2 * u + (a == "Y"), v = 2 * v + (b == "Y");
        G[u ^ 1].pb(v); G[v ^ 1].pb(u);
    }
    for (int i = 0; i < 2 * n; i ++ ) {
        if (!dfn[i]) tarjan(i);
    }
    for (int i = 0; i < n; i ++ ) {
        if (bel[2 * i] == bel[2 * i + 1]) {
            cout << "IMPOSSIBLE" << endl; return;
        }
    }
    for (int i = 0; i < n; i ++ ) {
        if (check(i * 2, i * 2 + 1)) cout << "Y";
        else if (check(i * 2 + 1, i * 2)) cout << "N";
        else cout << "?";
    }
}
```

2.11 基环树

```
for (int i = 1; i <= n; i ++ ) {
  if (vis[i])
    continue;
  int u = i;
  //找环上一点u
  while (!vis[u]) {
    vis[u] = true;
    u = p[u];
  }
  int v = u;
  vector<int> cyc;
  //通过从u开始遍历找到环上所有点
  while (true) {
    cyc.pb(v);
    oncyc[v] = true;
    v = p[v];
    if (v == u) break;
  }
  maxd = 0;
  //对环上每个子树dfs
  for (auto v : cyc)
        dfs(v);
}


// dfs找环上边，e1代表正向边，e1^1代表反向边。
for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        G[u].pb(mkp(v, 2 * i));
        if (u != v) G[v].pb(mkp(u, 2 * i + 1));
    }
function<void(int, int)> dfs = [&] (int u, int id) {
  dfn[u] = ++idx;
  for (auto [v, id2] : G[u]) {
```

```
    if (!dfn[v]) {
      par[v] = u;
      pe[v] = id2;
      dfs(v, id2);
    } else if (id2 != (id ^ 1) && dfn[v] <= dfn[u]) {
      int w = u;
      while (w != v) {
        cyc.pb(pe[w]);
        w = par[w];
      }
      cyc.pb(id);
    }
  }
};
```

```
/*https://www.luogu.com.cn/problem/P2607
基环树最大独立集*/
const int maxn = 1e6 + 10;
vector<int> G[maxn];
void solve() {
    cin >> n;
    vector<int> val(n + 1), p(n + 1);
    for (int i = 1; i <= n; i ++ ) {
        cin >> val[i] >> p[i];
        G[p[i]].pb(i);
    }
    vector<int> vis(n + 1);
    vector<vector<int>> dp(2, vector<int>(n + 1)), dp2(2, vector<int>(n + 1));
    vector<int> oncyc(n + 1);
    function<void(int)> dfs = [&] (int u) {
        vis[u] = true;
        dp[1][u] = val[u];
        for (auto v : G[u]) {
            if (oncyc[v]) continue;
            dfs(v);
            dp[0][u] += max(dp[0][v], dp[1][v]);
            dp[1][u] += dp[0][v];
        }
    };
    int ans = 0;
    for (int i = 1; i <= n; i ++ ) {
        if (vis[i]) continue;
        int u = i;
        while (!vis[u]) {
            vis[u] = true;
            u = p[u];
        }
        int v = u;
        vector<int> cyc;
        while (true) {
            cyc.pb(v);
            oncyc[v] = true;
            v = p[v];
            if (u == v) break;
        }
        for (auto v : cyc)
            dfs(v);
        int cur = -1e18;
        for (int t = 0; t < 2; t ++ ) {
            for (int j = 0; j < 2; j ++ ) {
                if (t == j) dp2[j][0] = dp[j][cyc[0]];
                else dp2[j][0] = -1e18;
```

```
            }
            for (int i = 1; i < sz(cyc); i ++ ) {
                int u = cyc[i];
                dp2[0][i] = max(dp2[1][i - 1], dp2[0][i - 1]) + dp[0][u];
                dp2[1][i] = dp2[0][i - 1] + dp[1][u];
            }
            if (!t) cur = max({cur, dp2[0][sz(cyc) - 1], dp2[1][sz(cyc) - 1]});
            else cur = max(cur, dp2[0][sz(cyc) - 1]);
        }
        ans += cur;
    }
    cout << ans << endl;
}


/*https://www.luogu.com.cn/problem/P4381
基环树直径*/
const int maxn = 1e6 + 1e3;
vector<int> G[maxn];
void solve() {
  cin >> n;
  vector<int> L(n + 1), p(n + 1);
  for (int i = 1; i <= n; i ++ ) {
    cin >> p[i] >> L[i];
    G[p[i]].pb(i);
  }
  int maxd = 0;
  vector<int> vis(n + 1), dp(n + 1), oncyc(n + 1);
  function<void(int)> dfs = [&] (int u) {
    vis[u] = true;
    dp[u] = 0;
    for (auto v : G[u]) {
        if (oncyc[v]) continue;
        dfs(v);
        int d = dp[v] + L[v];
        maxd = max(maxd, d + dp[u]);
        dp[u] = max(dp[u], d);
    }
  };
  vector<int> s(n + 10), pre(n + 10), suf(n + 10);
  int ans = 0;
  for (int i = 1; i <= n; i ++ ) {
    if (vis[i])
        continue;
    int u = i;
    while (!vis[u]) {
        vis[u] = true;
        u = p[u];
    }
    int v = u;
    vector<int> cyc;
    while (true) {
        cyc.pb(v);
        oncyc[v] = true;
        v = p[v];
        if (v == u) break;
    }
    maxd = 0;
    for (auto v : cyc)
        dfs(v);
    for (int i = 2; i <= sz(cyc); i ++ ) {
        s[i] = s[i - 1] + L[cyc[i - 2]];
```

```
    }
    int totl = s[sz(cyc)] + L[cyc[sz(cyc) - 1]];
    pre[0] = -1e18;
    for (int i = 1; i <= sz(cyc); i ++ ) {
        pre[i] = max(pre[i - 1], dp[cyc[i - 1]] - s[i]);
    }
    suf[sz(cyc) + 1] = -1e18;
    for (int i = sz(cyc); i >= 1; i -- ) {
        suf[i] = max(suf[i + 1], dp[cyc[i - 1]] - s[i]);
    }
    for (int i = 1; i <= sz(cyc); i ++ ) {
        maxd = max(maxd, pre[i - 1] + s[i] + dp[cyc[i - 1]]);
        maxd = max(maxd, suf[i + 1] + s[i] + dp[cyc[i - 1]] + totl);
    }
    ans += maxd;
  }
  cout << ans << endl;
}
```

2.12 网络流

2.12.1 Dinic

O(n ^ 2m)

用之前一定要注意点数边数开够，开不够是会WA的，不会RE

```
constexpr int V = 400010;
constexpr int E = 4001000;
template<typename T>
struct FlowGraph {
    int s, t, vtot;
    int head[V], etot;
    int dis[V], cur[V];
    struct edge {
        int v, nxt;
        T f;
    }e[E << 1];
    void addedge (int u, int v, T f) {
        e[etot] = { v, head[u], f }; head[u] = etot++;
        e[etot] = { u, head[v], 0 }; head[v] = etot++;
    }

    bool bfs () {
        for (int i = 1; i <= vtot; i++) {
            dis[i] = 0;
            cur[i] = head[i];
        }
        queue<int> q;
        q.push (s); dis[s] = 1;
        while (!q.empty ()) {
            int u = q.front (); q.pop ();
            for (int i = head[u]; ~i; i = e[i].nxt) {
                if (e[i].f && !dis[e[i].v]) {
                    int v = e[i].v;
                    dis[v] = dis[u] + 1;
                    if (v == t) return true;
                    q.push (v);
                }
            }
```

```
        }
        return false;
    }

    T dfs (int u, T m) {
        if (u == t) return m;
        T flow = 0;
        for (int i = cur[u]; ~i; cur[u] = i = e[i].nxt) {
            if (e[i].f && dis[e[i].v] == dis[u] + 1) {
                T f = dfs (e[i].v, min (m, e[i].f));
                e[i].f -= f;
                e[i ^ 1].f += f;
                m -= f;
                flow += f;
                if (!m) break;
            }
        }
        if (!flow) dis[u] = -1;
        return flow;
    }
    T dinic () {
        T flow = 0;
        while (bfs ()) flow += dfs (s, numeric_limits<T>::max ());
        return flow;
    }
    void init (int s_, int t_, int vtot_) {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for (int i = 1; i <= vtot; i++) head[i] = -1;
    }
};

FlowGraph<ll> g;
void Solve () {
    int n, m, s, t;
    cin >> n >> m >> s >> t;
    g.init (s, t, n);
    for (int i = 1; i <= m; i++) {
        int u, v, f;
        cin >> u >> v >> f;
        g.addegde (u, v, f);
    }
    cout << g.dinic () << '\n';
}
```

2.12.2 MCMF

```
constexpr int V = 20100;
constexpr int E = 201000;
template<typename T>
struct MinCostGragh {
  int s, t, vtot;
  int head[V], etot;
  T dis[V], flow, cost;
  int pre[V];
  bool vis[V];

  struct edge {
    int v, nxt;
```

```
  T f, c;
}e[E * 2];
void addedge (int u, int v, T f, T c, T f2 = 0) {
  e[etot] = {v, head[u], f, c}; head[u] = etot++;
  e[etot] = {u, head[v], f2, -c}; head[v] = etot++;
}

bool spfa () {
  T inf = numeric_limits<T>::max() / 2;
  for (int i = 1; i <= vtot; i++) {
    dis[i] = inf;
    vis[i] = false;
    pre[i] = -1;
  }
  dis[s] = 0;
  vis[s] = true;
  queue<int> q;
  q.push(s);
  while (!q.empty()) {
    int u = q.front();
    for (int i = head[u]; ~i; i = e[i].nxt) {
      int v = e[i].v;
      if (e[i].f && dis[v] > dis[u] + e[i].c) {
        dis[v] = dis[u] + e[i].c;
        pre[v] = i;
        if (!vis[v]) {
          vis[v] = 1;
          q.push(v);
        }
      }
    }
    q.pop();
    vis[u] = false;
  }
  return dis[t] != inf;
}

void augment() {
  int u = t;
  T f = numeric_limits<T>::max();
  while (~pre[u]) {
    f = min(f, e[pre[u]].f);
    u = e[pre[u] ^ 1].v;
  }
  flow += f;
  cost += f * dis[t];
  u = t;
  while (~pre[u]) {
    e[pre[u]].f -= f;
    e[pre[u] ^ 1].f += f;
    u = e[pre[u] ^ 1].v;
  }
}

pair<T, T> solve() {
  flow = 0;
  cost = 0;
  while (spfa()) {
    augment();
  }
  return {flow, cost};
}
```

```
  void init(int s_, int t_, int vtot_) {
    s = s_;
    t = t_;
    vtot = vtot_;
    etot = 0;
    for (int i = 1; i <= vtot; i++) head[i] = -1;
  }
};

MinCostGragh<int> g;

void Solve() {
  int n, m, s, t;
  cin >> n >> m >> s >> t;
  g.init(s, t, n);
  for (int i = 1; i <= m; i++) {
    int u, v, f, c;
    cin >> u >> v >> f >> c;
    g.addedge(u, v, f, c);
  }
  auto [flow, cost] = g.solve();
  cout << flow << ' ' << cost << '\n';
}
```

2.13 三元环/四元环计数

```
//HDU 6184
void Solve() {
  int n, m;
  cin >> n >> m;
  vector<int> deg(n + 1);
  vector<vector<pair<int, int>>> adj(n + 1);
  vector<pair<int, int>> edgs;
  for (int i = 1; i <= m; i++) {
    int u, v;
    cin >> u >> v;
    ++deg[u], ++deg[v];
    edgs.emplace_back(u, v);
  }

  for (int i = 0; i < m; i++) {
    auto [a, b] = edgs[i];
    if (deg[a] > deg[b] || deg[a] == deg[b] && a > b) adj[a].emplace_back(b, i + 1);
    else adj[b].emplace_back(a, i + 1);
  }

  vector<pair<int, int>> vis(n + 1);
  vector<int> cnt(m + 1);
  i64 ans = 0, cur = 0;
  for (int u = 1; u <= n; u++) {
    ++cur;
    for (auto [v, id] : adj[u]) {
      vis[v] = make_pair(cur, id);
    }
    for (auto [v, id1] : adj[u]) {
      for (auto [k, id2] : adj[v]) {
        if (vis[k].first == cur) {
          cnt[vis[k].second]++;
          cnt[id1]++;
          cnt[id2]++;
```

```
        }
      }
    }
  }

  for (int i = 1; i <= m; i++) {
    ans += 1ll * cnt[i] * (cnt[i] - 1) / 2;
  }
  cout << ans << '\n';
}
```

```
//四元环计数 O((n + m)sqrt(m))
void Solve() {
  int n, m;
  cin >> n >> m;
  vector<vector<int>> adj(n + 1);
  for (int i = 1; i <= m; i++) {
    int u, v;
    cin >> u >> v;
    adj[u].emplace_back(v);
    adj[v].emplace_back(u);
  }
  vector<int> deg(n + 1), id(n + 1);
  for (int i = 1; i <= n; i++) {
    deg[id[i] = i] = adj[i].size();
  }
  sort(id.begin() + 1, id.end(), [&](auto &a, auto &b) {
    return deg[a] < deg[b];
  });
  vector<int> rk(n + 1);
  for (int i = 1; i <= n; i++) {
    rk[id[i]] = i;
  }

  vector<vector<int>> g(n + 1);
  for (int u = 1; u <= n; u++) {
    for (auto v : adj[u]) {
      if (rk[v] > rk[u]) g[u].emplace_back(v);
    }
  }

  vector<int> cnt(n + 1);
  i64 ans = 0;
  for (int u = 1; u <= n; u++) {
    for (auto v : adj[u]) {
      for (auto w : g[v]) {
        if (rk[w] > rk[u]) ans += cnt[w]++;
      }
    }
    for (auto v : adj[u]) {
      for (auto w : g[v]) {
        if (rk[w] > rk[u]) cnt[w] = 0;
      }
    }
  }
  cout << ans << '\n';
}
```

2.14 点分治

```cpp
//https://www.luogu.com.cn/problem/P3806
constexpr int N = 1e4 + 10;
int q[N], siz[N], mxsz[N], del[N];
vector<array<int, 2>> g[N];
int n, m;

int tot, rt;
void getp (int u, int fa) {
    siz[u] = 1, mxsz[u] = 0;
    for (auto [v, w] : g[u]) {
        if (v != fa && !del[v]) {
            getp(v, u);
            mxsz[u] = max(mxsz[u], siz[v]);
            siz[u] += siz[v];
        }
    }
    mxsz[u] = max(mxsz[u], tot - siz[u]);
    if (mxsz[u] < mxsz[rt]) {
        rt = u;
    }
}

int d[N], cnt;
int dist[N], has[10000010], ok[N];
void calcinfo (int u, int fa) {
    d[++cnt] = dist[u];
    for (auto [v, w] : g[u]) {
        if (v != fa && !del[v]) {
            dist[v] = dist[u] + w;
            calcinfo(v, u);
        }
    }
}

queue<int> que;

void dfz (int u, int fa) {
    has[0] = true;
    que.push(0);
    del[u] = true;

    for (auto [v, w] : g[u]) {
        if (v != fa && !del[v]) {
            dist[v] = w;
            calcinfo(v, u);
            for (int j = 1; j <= cnt; j++) {
                for (int i = 1; i <= m; i++) {
                    if (q[i] >= d[j]) {
                        ok[i] |= has[q[i] - d[j]];
                    }
                }
            }
            for (int j = 1; j <= cnt; j++) {
                if (d[j] < 10000010) {
                    que.push(d[j]);
                    has[d[j]] = true;
                }
            }
            cnt = 0;
        }
    }

```

```
    while (que.size()) {
        has[que.front()] = false;
        que.pop();
    }

    for (auto [v, w] : g[u]) {
        if (v != fa && !del[v]) {
            tot = siz[v];
            rt = 0;
            mxsz[rt] = 1e9;
            getp(v, u);
            getp(rt, -1);
            dfz(rt, u);
        }
    }
}

void Solve() {
    cin >> n >> m;

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

    for (int i = 1; i <= m; i++) {
        cin >> q[i];
    }

    rt = 0;
    mxsz[rt] = 1e9;
    tot = n;
    getp(1, -1);
    getp(rt, -1);
    dfz(rt, -1);

    for (int i = 1; i <= m; i++) {
        cout << (ok[i] ? "AYE" : "NAY") << '\n';
    }
}
```

3 数学

3.1 线性筛

```
constexpr int N = 1e5 + 10;
int primes[N], cnt;
bool st[N];
void LinearSieve(int N) {
    for (int i = 2; i <= N; i ++ ) {
        if (!st[i]) primes[cnt ++ ] = i;
        for (int j = 0; primes[j] <= N / i; j ++ ) {
            st[primes[j] * i] = true;
            if (i % primes[j] == 0) break;
        }
    }
}
```

3.2 快速乘 & 快速幂

```cpp
// 快速幂
int qmi (int a, int b) { // a ^ b
    int ans = 1;
    while (b) {
        if (b & 1) ans = ans * a % mod; // 或 ans = mul(ans, a);
        a = a * a % mod; // 或 ans = mul(a, a);
        b >>= 1;
    }
    return ans;
}


// 快速乘
int mul (int a, int b) { // a * b
    int ans = 0;
    while (b) {
        if (b & 1) ans = (ans + a) % mod;
        b >>= 1;
        a = (a * 2) % mod;
    }
    return ans;
}
//高精快速幂，mod过大时使用
typedef unsigned long long llong;
llong power (llong x,llong n,llong mod)
{
    __int128 a = (__int128) x;
    llong res = 1;
    while (n > 0) {
        if (n & 1) res = res * a %mod;
        a = a * a % mod;
        n >>= 1;
    }
    return res;
}
```

3.3 拓展欧几里得算法 exgcd

```cpp
// 求x, y，使得ax + by = n
int exgcd(int a, int b, int &x, int &y) {
    if (!b) {
        x = 1; y = 0;
        return a;
    }
    int d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}
//调整
x *= n / g, y *= n / g;
int t = x / (b / g);
x -= b / g * t, y += a / g * t;
if (x < 0) x += b / g, y -= a / g;
```

```cpp
int a, b, c;
cin >> a >> b >> c;
int x, y;
int g = exgcd(a, b, x, y);
if (c % g != 0) {
    cout << -1 << '\n';
    return;
}
x *= c / g, y *= c / g;
int tx = (-x) / (b / g);
if (x + tx * (b / g) <= 0) tx++;
int ty = y / (a / g);
if (y - ty * (a / g) <= 0) ty--;
//没有正整数解
if (tx > ty) {
    // x的最小整数解，y的最小整数解
    cout << x + tx * (b / g) << ' ' << y - ty * (a / g) << '\n';
} else {
    //正整数解个数
    cout << ty - tx + 1 << ' ';
    //x的最小正整数解，y最小正整数解
    cout << x + tx * (b / g) << ' ' << y - ty * (a / g) << ' ';
    //x的最大正整数解，y的最大正整数解
    cout << x + ty * (b / g) << ' ' << y - tx * (a / g) << '\n';
}
```

3.4 欧拉函数

```cpp
// 求1~N中与N互质的个数
int phi(int x)
{
    int res = x;
    for (int i = 2; i <= x / i; i ++ )
        if (x % i == 0)
        {
            res = res / i * (i - 1);
            while (x % i == 0) x /= i;
        }
    if (x > 1) res = res / x * (x - 1);

    return res;
}

// 筛法求欧拉函数
int primes[N], cnt;     // primes[]存储所有素数
int euler[N];           // 存储每个数的欧拉函数
bool st[N];         // st[x]存储x是否被筛掉


void get_eulers(int n)
{
    euler[1] = 1;
    for (int i = 2; i <= n; i ++ )
    {
        if (!st[i])
        {
            primes[cnt ++ ] = i;
            euler[i] = i - 1;
        }
        for (int j = 0; primes[j] <= n / i; j ++ )
```

```
        {
            int t = primes[j] * i;
            st[t] = true;
            if (i % primes[j] == 0)
            {
                euler[t] = euler[i] * primes[j];
                break;
            }
            euler[t] = euler[i] * (primes[j] - 1);
        }
    }
}
```

3.5 约数 个数/和 定理

```
// 约数个数
//n=p1^a1*p2^a2*p3^a3*…*pk^ak
//f(n) = (a1+1)(a2+1)(a3+1)…(ak+1)
int main()
{
    int n;
    cin >> n;

    unordered_map<int, int> primes;

    while (n -- )
    {
        int x;
        cin >> x;

        for (int i = 2; i <= x / i; i ++ )
            while (x % i == 0)
            {
                x /= i;
                primes[i] ++ ;
            }

        if (x > 1) primes[x] ++ ;
    }

    LL res = 1;
    for (auto p : primes) res = res * (p.second + 1) % mod;

    cout << res << endl;

    return 0;
}

// 约数之和
//n=p1^a1*p2^a2*p3^a3*…*pk^ak
//f(n)=(p1^0+p1^1+p1^2+…p1^a1)(p2^0+p2^1+p2^2+…p2^a2)…(pk^0+pk^1+pk^2+…pk^ak）
unordered_map<int, int> mp;
    while(t -- )
    {
        int x; scanf("%d", &x); // 即n
        for(int i = 2; i <= x / i; i ++)
        {
            while(x % i == 0)
            {
                x /= i;
```

```
                mp[i]++;
            }
        }
        if(x > 1) mp[x] ++;
    }
    long long res = 1;
    for(auto p : mp)
    {
        long long a = p.first, b = p.second;
        long long t = 1;
        while(b -- )
        {
            t = (t * a + 1) % mod; // 秦九韶算法
        }
        res = res * t % mod;
    }
    cout << res << endl;
```

3.6 组合数

结论

1.有n个**完全相同**的元素，将其分为k组。

考虑$k-1$块板子插入$n-1$个空白。

$$\binom{n-1}{k-1} \tag{1}$$

2.有n个**完全相同**的元素，将其分为k组，允许每组为空。

考虑求一组新解$x_i' = x_i - 1$，即$\sum_{i=1}^{k} x_i\prime = n$，把1全部移至等式右边。

$$\binom{n+k-1}{k-1} \tag{2}$$

3.有n个**完全相同**的元素，将其分为k组，每组有下界a_i。

考虑求一组新解$x_i' = x_i - (a_i - 1)$，同上。

$$\binom{n+k-1-\sum a_i}{k-1} \tag{3}$$

(1)中的情况可看为下界是1的特殊情况。

4.$1 \sim n$的排列中选出k个两两不相邻的方案。

$$\binom{n-k+1}{k} \tag{4}$$

5.多重集组合数，$S = \{n_1 \cdot a_1, n_2 \cdot a_2, n_3 \cdot a_3, \ldots, n_k \cdot a_k\}$，则$S$的全排列个数:

$$\frac{n!}{\prod_{i=1}^{k} n_i!} \tag{5}$$

6.多重集组合数，$S = \{n_1 \cdot a_1, n_2 \cdot a_2, n_3 \cdot a_3, \ldots, n_k \cdot a_k\}$，从$S$中任选出$r$个元素的方案数，其中$(r < n_i, \forall i \in [i,k])$

用隔板法解决，等价于$x_1 + x_2 + \ldots + x_k = r$的非负整数解的数目。

$$\binom{r+k-1}{k-1} \tag{6}$$

7.多重集组合数，$S = \{n_1 \cdot a_1, n_2 \cdot a_2, n_3 \cdot a_3, \ldots, n_k \cdot a_k\}$，从$S$中任选出$r$个元素的方案数

答案有:

$$\left|\bigcap_{i=1}^{k} S_i\right| = |U| - \left|\bigcup_{i=1}^{k} \overline{S_i}\right| \tag{7}$$

根据容斥，得:

$$\begin{aligned}\left|\bigcup_{i=1}^{k} \overline{S_i}\right| &= \sum_i \left|\overline{S_i}\right| - \sum_{i,j} \left|\overline{S_i} \cap \overline{S_j}\right| + \sum_{i,j,k} \left|\overline{S_i} \cap \overline{S_j} \cap \overline{S_k}\right| - \cdots \\ &\quad + (-1)^{k-1} \left|\bigcap_{i=1}^{k} \overline{S_i}\right| \\ &= \sum_i \binom{k+r-n_i-2}{k-1} - \sum_{i,j} \binom{k+r-n_i-n_j-3}{k-1} + \sum_{i,j,k} \binom{k+r-n_i-n_j-n_k-4}{k-1} - \cdots \\ &\quad + (-1)^{k-1} \binom{k+r-\sum_{i=1}^{k} n_i - k - 1}{k-1}\end{aligned} \tag{8}$$

最后可以转化成:

$$Ans = \sum_{p=0}^{k} (-1)^p \sum_A \binom{k+r-1-\sum_A n_{A_i} - p}{k-1} \tag{9}$$

范德蒙德卷积

$$\sum_{i=0}^{k} \binom{n}{i} \binom{m}{k-i} = \binom{n+m}{k} \tag{10}$$

推论1:

$$\sum_{i=1}^{n} \binom{n}{i} \binom{n}{i-1} = \binom{2n}{n+1} \tag{11}$$

推论2

$$\sum_{i=0}^{n} \binom{n}{i}^2 = \binom{2n}{n} \tag{12}$$

推论3

$$\sum_{i=0}^{m} \binom{n}{i} \binom{m}{i} = \binom{n+m}{m} \tag{13}$$

板子

```
vector<mint>fac(n + 1), ifac(n + 1);
fac[0] = 1;
for (int i = 1; i <= n; i++) {
    fac[i] = fac[i - 1] * i;
}
ifac[n] = fac[n].inv();
for (int i = n; i >= 1; i--) {
    ifac[i - 1] = ifac[i] * i;
}

auto C = [&] (int n, int m) {
    if (n < m || n < 0 || m < 0) {
        return mint(0);
    }
    return fac[n] * ifac[m] * ifac[n - m];
```

```
};
```

```
constexpr int N = 2e5 + 10;
mint fac[N], ifac[N];
void Init(int n) {
    fac[0] = 1;
    for (int i = 1; i <= n; i++) {
        fac[i] = fac[i - 1] * i;
    }
    ifac[n] = fac[n].inv();
    for (int i = n; i >= 1; i--) {
        ifac[i - 1] = ifac[i] * i;
    }
}
mint C (int n, int m) {
    if (n < m || n < 0 || m < 0) {
        return mint(0);
    }
    return fac[n] * ifac[m] * ifac[n - m];
}
```

```
const int N = 2010;
int C[N][N];

void Init() {
    for (int i = 0; i < N; ++i) C[i][0] = 1;
    for (int i = 0; i < N; ++i)
        for (int j = 1; j <= i; ++j)
            C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % Mod;
}
```

```
// lucas定理
// C(a, b) = C(a % p, b % p) * C(a / p, b / p); p为质数
int a, b, p; // 求C(a, b) % p;
inline int qmi (int a, int b) { int res = 1; while (b) { if (b & 1) res = res * a % p; a = a * a % p; b >>= 1; } return res; }
inline int inv (int x) {return qmi(x, p - 2)};
int fac[maxn]; inline void getfac () {for (int i = 1; i <= maxn - 10; i ++ ) fac[i] = fac[i - 1] * i % p;}
inline int C (int n, int m) {
    if (n < m) return 0;
    if (n < p) return fac[n] * inv(fac[n - m]) % p * inv(fac[m]) % p;
    return C(n / p, m / p) * C(n % p, m % p) % p;
}

// k次前缀和的组合数
C（k-1+i, k-1）
```

3.7 数论分块

$$\sum_{i=1}^{m}\lfloor\frac{n}{i}\rfloor \tag{14}$$

```
// 下取整
```

```
inline int getdown (int n) {
    int ans = 0;
    for(int l = 1, r, len; l <= m; l = r + 1) {
        if (n / i == 0) break;
        r = min(m, n / (n / l)), len = r - l + 1;
        ans += len * (n / l);
    }
    return ans;
}

// 上取整
for (int l = 1, r; l <= n; l = r + 1) {
    if (l == n) {
      ans++;
      break;
    } else {
      int v = (n + l - 1) / l;
      r = (n + v - 2) / (v - 1) - 1;
      ans += (r - l + 1) * v;
    }
  }
```

3.8 卡特兰数

$$f(n) = C_{2n}^{n} - C_{2n}^{n-1} \tag{15}$$

```
1.n 个元素进栈序列为：1，2，3，4，...，n，则有多少种出栈序列。
2.n 对括号，则有多少种 “括号匹配” 的括号序列
3.n + 1 个叶子节点能够构成多少种形状不同的（国际）满二叉树
4.电影票一张 50 coin，且售票厅没有 coin，m 个人各自持有 50 coin，n 个人各自持有 100 coin。则有多少种排队方式，可以让每个人都买到电影票。
5.8 个高矮不同的人需要排成两队，每队 4 个人。其中，每排都是从低到高排列，且第二排的第 i 个人比第一排中第 i 个人高，则有多少种排队方式?
6.在一个凸多边形中，通过若干条互不相交的对角线，把这个多边形划分成了若干个三角形。任务是键盘上输入凸多边形的边数n，求不同划分的方案数f（n）。

前几项：
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...
```

3.9 线性求逆

```
ll inv[maxn] = {0,1};
for (int i = 2; i < maxn; i ++ )
    inv[i] = (mod - mod / i) * inv[mod % i] % mod;
```

3.10 博弈论

```
经典nim
n堆，每次拿任意个，不可不拿，先拿完赢
// 先手是否必胜？
cin >> n;
int res = 0;
for (int i = 0; i < n; i ++ ){
    int a;
    cin >> a;
    res ^= a;
}

```

```
if (res) printf("Yes\n");
else printf("No\n");


anti-nim游戏
n堆，每次拿任意个，不可不拿，先拿完输
先手胜当且仅当 ①所有堆石子数都为1且游戏的SG值为0（即有偶数个孤单堆-每堆只有1个石子数）；②存在某堆石子数大于1且游戏的SG值不为0.
 bool ok = 0;
int ans = 0;
for (int i = 0; i < sz(res); i ++ ) {
    ans ^= res[i];
    if (res[i] > 1) ok = 1;
}

if ((!ok && ans == 0) || (ok && ans != 0)) puts("Alice win");
else puts("Bob win");


对称博弈

```

树上博弈

设T为一个森林，其中有n颗有根树，且树根都在地面上。$Alice$、Bob每次选择某一棵树的一条边，删除这条边以及这条边所连接的地上部分，最后无法操作的人输掉博弈。

克朗原理

$$SG(x)=\begin{cases}0 & t=0\\ (SG(x_1)+1)\oplus(SG(x_2)+1)\oplus\cdots\oplus(SG(x_n)+1) & t>0\end{cases}$$

3.11 最小二乘法求线性回归方程

$$k=\frac{\sum_{i=1}^{n}x_iy_i-n\bar{x}\bar{y}}{\sum_{i=1}^{n}x_i^2-n\bar{x}^2} \tag{16}$$

$$d=\bar{y}-k\bar{x} \tag{17}$$

```
for (int i = 1; i <= n; i ++ ) q[i] = read();
    double sumx = 0, sumy = 0, iq = 0, ii = 0;
    for (int i = 1; i <= n; i ++ ) {
        sumx += i, sumy += q[i];
        iq += i * q[i];
        ii += i * i;
    }
    double k = (iq - sumx * sumy / n) / (ii - sumx * sumx / n);
    double d = sumy / n - k * sumx / n;

    double ans = 0;
    for (int i = 1; i <= n; i ++ ) {
        ans += (k * i + d - q[i]) * (k * i + d - q[i]);
    }
    printf("%.6lf\n", ans);
```

3.12 线性基

temp1

```
const int B = 60;
struct LinearBasis {
    array<ll, 61> a, tmp;
    bool flag;
    LinearBasis () {
        fill(a.begin(), a.end(), 0);
        fill(tmp.begin(), tmp.end(), 0);
        flag = false;
    }
    bool ins (ll x) {
        for (int i = B; i >= 0; i--) {
            if (x >> i & 1) {
                if (!a[i]) {
                    a[i] = x;
                    return true;
                } else x ^= a[i];
            }
        }
        flag = true;
        return false;
    }
    bool check (ll x) {
        for (int i = B; i >= 0; i--) {
            if (x & (1ll << i)) {
                if (!a[i]) return false;
                else x ^= a[i];
            }
        }
        return true;
    }
    ll qmax (ll ret = 0) {
        for (int i = B; i >= 0; i--) {
            ret = max(ret, ret ^ a[i]);
        }
        return ret;
    }
    ll qmin () {
        if (flag) return 0;
        for (int i = 0; i <= B; i++) {
            if (a[i]) return a[i];
        }
    }
    ll query (ll k) {
        ll ret = 0, cnt = 0;
        k -= flag;
        if (!k) return 0;
        for (int i = 0; i <= B; i++) {
            for (int j = i - 1; j >= 0; j--) {
                if (a[i] >> j & 1) a[i] ^= a[j];
            }
            if (a[i]) tmp[cnt++] = a[i];
        }
        if (k >= (1ll << cnt)) return -1;
        for (int i = 0; i < cnt; i++) {
            if (k >> i & 1) ret ^= tmp[i];
        }
        return ret;
    }
    void debug() {
        for (int i = B; i >= 0; i--) {
```

```
            dbg(i, a[i]);
        }
    }
};
```

temp2

```
const int MAXL = 60;
struct LinearBasis {
    int a[MAXL + 1];
    int cnt = 0;

    LinearBasis () {
        fill(a, a + MAXL + 1, 0);
    }

    LinearBasis (int *x, int n) { // 快速构造线性基
        build (x, n);
    }

    inline bool insert (int t) { // 线性基动态插入数
        for (int j = MAXL; j >= 0; j -- ) {
            if (!t) return false;
            if (!(t & (1ll << j))) continue;

            if (a[j]) t ^= a[j];
            else {
                cnt ++ ;
                // for (int k = 0; k < j; k ++ ) {
                //         if (t & (1ll << k)) t ^= a[k];
                // }
                // for (int k = j + 1; k <= MAXL; k ++ ) {
                //         if (a[k] & (1ll << j)) a[k] ^= t;
                // }
                a[j] = t;
                return false;
            }
        }
        return true;
    }

    void build (int *x, int n) {
        fill(a, a + MAXL + 1, 0);
        for (int i = 1; i <= n; i ++ ) insert(x[i]);
    }
    inline void rebuild () { // 重构成易于求kth的形式
        for (int i = 0; i <= MAXL; i ++ ) {
            for (int j = i - 1; j >= 0; j -- ) {
                if (a[i] & (1ll << j)) a[i] ^= a[j];
            }
        }
    }

    inline void mergefrom (const LinearBasis &b) { // 合并两个线性基
        for (int i = 0; i <= MAXL; i ++ ) insert(b.a[i]);
    }

    static LinearBasis merge (const LinearBasis &a, const LinearBasis &b) {
        LinearBasis res = a;
        for (int i = 0; i <= MAXL; i ++ ) res.insert(b.a[i]);
        return res;
```

```
    }

    inline int querymax () { // 查询子集最大异或和
        int res = 0;
        for (int i = 0; i <= MAXL; i ++ ) res ^= a[i];
        return res;
    }
    inline int querymin () { // 查询子集最小异或和
        for (int i = 0; i <= MAXL; i ++ ) if (a[i]) return a[i];
    }

    inline int kth (int k) { // k大异或和
        if (cnt < n) k -- ;
        if (k >= (1ll << cnt)) return -1;

        int ans = 0;
        for (int i = 0, j = 0; i <= MAXL; i ++ ) {
            if (a[i]) {
                if (k & (1ll << j)) ans ^= a[i];
                j ++ ;
            }
        }
        return ans;
    }

    inline int rank (int x) { // 查询异或和排名，不去重需要 k % mod * qmi(2, n - a.cnt) % mod + 1) % mod;
        int ans = 0;
        for (int i = 0, j = 0; i <= MAXL; i ++ ) {
            if (a[i]) {
                if (x & (1ll << i)) ans |= (1ll << j);
                j ++ ;
            }
        }
        return ans;
    }
};
```

3.13 Pollard-Rho

```
namespace prime_fac {

    const int S = 8; // 随机算法判定次数，8~10 就够了

// 龟速乘
    long long mult_mod(long long a, long long b, long long c) {
        a %= c, b %= c;
        long long ret = 0;
        long long tmp = a;
        while (b) {
            if (b & 1) {
                ret += tmp;
                if (ret > c) ret -= c;
            }
            tmp <<= 1;
            if (tmp > c) tmp -= c;
            b >>= 1;
        }
        return ret;
    }

```

```cpp
    // 快速幂
    long long qow_mod(long long a, long long n, long long _mod) {
        long long ret = 1;
        long long temp = a % _mod;
        while (n) {
            if (n & 1) ret = mult_mod(ret, temp, _mod);
            temp = mult_mod(temp, temp, _mod);
            n >>= 1;
        }
        return ret;
    }

    // 是合数返回true，不一定是合数返回false
    bool check(long long a, long long n, long long x, long long t) {
        long long ret = qow_mod(a, x, n);
        long long last = ret;
        for (int i = 1; i <= t; i++) {
            ret = mult_mod(ret, ret, n);
            if (ret == 1 && last != 1 && last != n - 1) return true;
            last = ret;
        }
        if (ret != 1) return true;
        return false;
    }

    // 是素数返回true，不是返回false
    mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
    bool Miller_Rabin(long long n) {
        if (n < 2) return false;
        if (n == 2) return true;
        if ((n & 1) == 0) return false;
        long long x = n - 1;
        long long t = 0;
        while ((x & 1) == 0) { x >>= 1; t++; }

        for (int i = 0; i < S; i++) {
            long long a = rng() % (n - 1) + 1;
            if (check(a, n, x, t))
                return false;
        }

        return true;
    }

    long long factor[100];// 存质因数
    int tol; // 质因数的个数，0~tol-1

    long long gcd(long long a, long long b) {
        long long t;
        while (b) {
            t = a;
            a = b;
            b = t % b;
        }
        if (a >= 0) return a;
        return -a;
    }

    long long pollard_rho(long long x, long long c) {
        long long i = 1, k = 2;
        long long x0 = rng() % (x - 1) + 1;
        long long y = x0;
```

```
        while (1) {
            i++;
            x0 = (mult_mod(x0, x0, x) + c) % x;
            long long d = gcd(y - x0, x);
            if (d != 1 && d != x) return d;
            if (y == x0) return x;
            if (i == k) { y = x0; k += k; }
        }
    }
    // 对n质因数分解，存入factor，k一般设置为107左右
    void findfac(long long n, int k) {
        if (n == 1) return;
        if (Miller_Rabin(n)) {
            factor[tol++] = n;
            return;
        }
        long long p = n;
        int c = k;
        while (p >= n) p = pollard_rho(p, c--);
        findfac(p, k);
        findfac(n / p, k);
    }
    vector<int> fac(long long n) {
        tol = 0;
        vector<int>ret;
        findfac(n, 107);
        for (int i = 0; i < tol; i++)ret.push_back(factor[i]);
        return ret;
    }
}

//vector<int> fac = prime_fac::fac(n);
```

板子2

```
using i64 = long long;
using i128 = __int128;

i64 power (i64 a, i64 b, i64 m) {
    i64 res = 1;
    for (; b; b >>= 1, a = i128 (a) * a % m) {
        if (b & 1) {
            res = i128 (res) * a % m;
        }
    }
    return res;
}

bool isprime (i64 p) {
    if (p < 2) {
        return 0;
    }
    i64 d = p - 1, r = 0;
    while (!(d & 1)) {
        r++;
        d >>= 1;
    }
    int prime[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23 };
    for (auto a : prime) {
        if (p == a) {
```

```
            return true;
        }
        i64 x = power (a, d, p);
        if (x == 1 || x == p - 1) {
            continue;
        }
        for (int i = 0; i < r - 1; i++) {
            x = i128 (x) * x % p;
            if (x == p - 1) {
                break;
            }
        }
        if (x != p - 1) {
            return false;
        }
    }
    return true;
}

mt19937 rng ((unsigned int)chrono::steady_clock::now ().time_since_epoch ().count ());

i64 pollard_rho (i64 x) {
    i64 s = 0, t = 0;
    i64 c = i64 (rng ()) % (x - 1) + 1;
    i64 val = 1;
    for (int goal = 1; ; goal <<= 1, s = t, val = 1) {
        for (int step = 1; step <= goal; step++) {
            t = (i128 (t) * t + c) % x;
            val = i128 (val) * abs (t - s) % x;
            if (step % 127 == 0) {
                i64 g = gcd (val, x);
                if (g > 1) {
                    return g;
                }
            }
        }
        i64 g = gcd (val, x);
        if (g > 1) {
            return g;
        }
    }
}

unordered_map<i64, int> getprimes (i64 x) {
    unordered_map<i64, int> p;
    function<void (i64)> get = [&](i64 x) {
        if (x < 2) {
            return;
        }
        if (isprime (x)) {
            p[x]++;
            return;
        }
        i64 mx = pollard_rho (x);
        get (x / mx);
        get (mx);
    };
    get (x);
    return p;
}
```

3.14 矩阵类

```cpp
template<class Type>
class Matrix {
private:
    vector<vector<Type>> data;
public:
    int width, height;
    Matrix (int height = 0, int width = 0, Type value = 0);
    Matrix<Type> (const Matrix<Type> &other);
    Matrix<Type> operator + (const Matrix<Type> &other);
    Matrix<Type> operator - (const Matrix<Type> &other);
    Matrix<Type> operator * (const Matrix<Type> &other);
    Matrix<Type> operator ~();
    Matrix<Type> pow(int x);
    vector<Type> operator [](int row) const;
    vector<Type>& operator [](int row);
    void print();
    static Matrix<Type> eye(int n);
};
typedef Matrix<double> Mat;
template<class Type>
Matrix<Type>::Matrix(const Matrix<Type> &other) {
    height = other.height;
    width = other.width;
    data = other.data;
}
template<class Type>
Matrix<Type>::Matrix(int height_, int width_, Type value_) {
    height = height_;
    width = width_;
    data.resize(height);
    for (int i = 0; i < height; i ++ ) {
        data[i].resize(width, value_);
    }
}
template<class Type>
void Matrix<Type>::print() {
    for (int i = 0; i < height; i ++ ) {
        for (int j = 0; j < width; j ++ ) {
            cout << data[i][j] << ' ';
        }
        cout << endl;
    }
}
template<class Type>
Matrix<Type> Matrix<Type> :: operator + (const Matrix <Type> &other) {
    if (other.height != height || other.width != width) {
        throw -1;
    }
    Matrix<Type> res(height, width);
    for (int i = 0; i < height; i ++ ) {
        for (int j = 0; j < width; j ++ ) {
            res.data[i][j] = data[i][j] + other.data[i][j];
        }
    }
    return res;
}
template<class Type>
Matrix<Type> Matrix<Type> :: operator - (const Matrix <Type> &other) {
    if (other.height != height || other.width != width) {
        throw -1;
```

```
    }
    Matrix<Type> res(height, width);
    for (int i = 0; i < height; i ++ ) {
        for (int j = 0; j < width; j ++ ) {
            res.data[i][j] = data[i][j] - other.data[i][j];
        }
    }
    return res;
}
template<class Type>
Matrix<Type> Matrix<Type> :: operator * (const Matrix <Type> &other) {
    if (other.height != width) {
        throw -2;
    }
    Matrix<Type> res(height, other.width);
    for (int i = 0; i < height; i ++ ) {
        for (int j = 0; j < other.width; j ++ ) {
            for (int k = 0; k < width; k ++ ) {
                res.data[i][j] += data[i][k] * other.data[k][j];
            }
        }
    }
    return res;
}
template<class Type>
Matrix<Type> Matrix<Type> :: operator ~() {
    Matrix<Type> res(width, height);
    for (int i = 0; i < width; i ++ ) {
        for (int j = 0; j < height; j ++ ) {
            res[i][j] = data[j][i];
        }
    }
    return res;
}
template<class Type>
vector<Type> Matrix<Type> :: operator [] (int row) const {
    if (row > height) throw -5;
    return data[row];
}
template<class Type>
vector<Type>& Matrix<Type> :: operator [] (int row) {
    if (row > height) throw -5;
    return data[row];
}
template<class Type>
Matrix<Type> Matrix<Type> :: eye(int n){
    Matrix<Type> res(n, n);
    for (int i = 0; i < n; i ++ ){
        res[i][i] = 1;
    }
    return res;
}
template<class Type>
Matrix<Type> pow (Matrix<Type> mat, int x) {
    Matrix<Type> res = mat.eye;
    while (x) {
        if (x & 1) res = res * mat;
        mat = mat * mat;
        x >>= 1;
    }
    return res;
}
```

```
void solve() {
    Mat a(3, 3);
    a[0] = {1, 1, 1};
    Mat b(3, 4);
    (a * b).print();
}
```

3.15 叉积求极角排序

```
//https://codeforces.com/contest/598/problem/C
//求两两点之间最小夹角
struct Point {
    int x, y;
    int id;
    il Point (int x = 0, int y = 0, int id = 0) : x(x), y(y), id(id) {}
};
il int dot (const Point &a, const Point &b) {return a.x * b.x + a.y * b.y;}
il int det (const Point &a, const Point &b) {return a.x * b.y - a.y * b.x;}
il void sortPoint (vector<Point> &vec) {
    vector<Point> p, q;
    for (auto cur : vec) {
        if (cur.y > 0 || (cur.y == 0 && cur.x > 0)) p.pb(cur);
        else q.pb(cur);
    }
    sort(all(p),[&](const Point &a, const Point &b) {
        return det(a, b) > 0;
    });
    sort(all(q),[&](const Point &a, const Point &b) {
        return det(a, b) > 0;
    });
    vec.clear();
    for (auto cur : p) vec.pb(cur);
    for (auto cur : q) vec.pb(cur);
}
void solve() {
    cin >> n;
    vector<Point> a(n);
    for (int i = 0; i < n; i ++ ) {
        cin >> a[i].x >> a[i].y;
        a[i].id = i;
    }
    sortPoint(a);
    int p = 0, q = n - 1;
    for (int i = 0; i < n - 1; i ++ ) {
        Point v1(dot(a[i], a[i + 1]), abs(det(a[i], a[i + 1])));
        Point v2(dot(a[p], a[q]), abs(det(a[p], a[q])));
        if (det(v1, v2) > 0) p = i, q = i + 1;
    }
    cout << a[p].id + 1 << ' ' << a[q].id + 1 << endl;
}

using T = double;
struct Point {
    T x;
    T y;
    Point(T x = 0, T y = 0) : x(x), y(y) {}

    Point &operator+=(const Point &p) {
        x += p.x, y += p.y;
        return *this;
    }
```

```
    Point &operator-=(const Point &p) {
        x -= p.x, y -= p.y;
        return *this;
    }
    Point &operator*=(const T &v) {
        x *= v, y *= v;
        return *this;
    }
    friend Point operator-(const Point &p) {
        return Point(-p.x, -p.y);
    }
    friend Point operator+(Point lhs, const Point &rhs) {
        return lhs += rhs;
    }
    friend Point operator-(Point lhs, const Point &rhs) {
        return lhs -= rhs;
    }
    friend Point operator*(Point lhs, const T &rhs) {
        return lhs *= rhs;
    }
};

T dot(const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y;
}

T cross(const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
}
```

3.16 两圆交面积

```
struct Circle {
    long double x, y;
    long double r;
    long double area (const Circle &b) {
        long double d = sqrt((x - b.x) * (x - b.x) + (y - b.y) * (y - b.y));
        if (d >= r + b.r) return 0;
        long double r1 = r, r2 = b.r;
        if (r1 > r2) swap(r1, r2);
        if (r2 - r1 >= d) return pi * r1 * r1;
        long double ang1 = acos((r1 * r1 + d * d - r2 * r2) / (2 * r1 * d));
        long double ang2 = acos((r2 * r2 + d * d - r1 * r1) / (2 * r2 * d));
        return ang1 * r1 * r1 - 0.5 * r1 * r1 * sin(ang1 * 2) + ang2 * r2 * r2 - 0.5 * r2 * r2 * sin(ang2 * 2);
    }
};
```

3.17 全概率公式/贝叶斯公式

全概率公式：

(1) $\forall i \neq j, A_i A_j = \varnothing$

(2)$A_1 \cup A_2 \cup \ldots A_n = \Omega.$

则：

$P(B) = \sum_{i=1}^{n} P(A_i) P(B|A_i)$

贝叶斯公式：

设$A_1, A_2, \ldots, A_n$是完备事件组，且$P(A_i) > 0 (i = 1, 2, \ldots, n)$, B为任意事件，$P(B) > 0$,则

$$P(A_k|B) = \frac{P(A_k)P(B|A_k)}{P(B)} = \frac{P(A_k)P(B|A_k)}{\sum\limits_{i=1}^{n} P(A_i)P(B|A_i)}$$

3.18 多项式卷积

3.18.1 FFT

$$c[x] = \sum a[i] * b[x - i] \tag{18}$$

Ver线下

```
typedef complex<double> C;
typedef vector<double> vd;
void fft(vector<C>& a) {
  int n = a.size(), L = 31 - __builtin_clz(n);
  static vector<complex<long double>> R(2, 1);
  static vector<C> rt(2, 1); // (^ 10% fas te r i f double)
  for (static int k = 2; k < n; k *= 2) {
    R.resize(n); rt.resize(n);
    auto x = polar(1.0L, acos(-1.0L) / k);
    for (int i = k; i < 2 * k; i++) rt[i] = R[i] = i&1 ? R[i/2] * x : R[i/2];
  }
  vector<int> rev(n);
  for (int i = 0; i < n; i++)
    rev[i] = (rev[i / 2] | (i & 1) << L) / 2;
  for (int i = 0; i < n; i++)
    if (i < rev[i]) swap(a[i], a[rev[i]]);
  for (int k = 1; k < n; k *= 2)
    for (int i = 0; i < n; i += 2 * k) {
      for (int j = 0; j < k; j++) {
        C z = rt[j + k] * a[i + j + k]; // (25% fas te r i f hand-ro l led )
        a[i + j + k] = a[i + j] - z;
        a[i + j] += z;
      }
    }
}
vd convo (const vd& a, const vd& b) {
  if (a.empty() || b.empty()) return {};
  vd res(a.size() + b.size() - 1);
  int L = 32 - __builtin_clz(res.size()), n = 1 << L;
  vector<C> in(n), out(n);
  copy(a.begin(), a.end(), begin(in));
  for (int i = 0; i < b.size(); i++) in[i].imag(b[i]);
  fft(in);
  for (C& x : in) x *= x;
  for (int i = 0; i < n; i++) out[i] = in[-i & (n - 1)] - conj(in[i]);
  fft(out);
  for (int i = 0; i < res.size(); i++) res[i] = imag(out[i]) / (4 * n);
  return res;
}

void Solve() {
  int n, m;
  cin >> n >> m;
  n++, m++;
```

```
  vd a(n), b(m);
  for (auto &x : a) cin >> x;
  for (auto &x : b) cin >> x;
  auto c = conv(a, b);
  for (auto x : c) cout << (int)(x + 0.0001) << ' ';
}
```

Ver线上

```
// FFT_MAXN = 2^k
// fft_init() to precalc FFT_MAXN-th roots
#define rep(i, a, b) for (int i = a; i < (int)b; i++)
typedef long double db;
const int FFT_MAXN=262144;
const db pi=acosl(-1.);
struct cp{
    db a,b;
    cp operator+(const cp&y)const{return (cp){a+y.a,b+y.b};}
    cp operator-(const cp&y)const{return (cp){a-y.a,b-y.b};}
    cp operator*(const cp&y)const{return (cp){a*y.a-b*y.b,a*y.b+b*y.a};}
    cp operator!()const{return (cp){a,-b};};
}nw[FFT_MAXN+1];int bitrev[FFT_MAXN];
void dft(cp*a,int n,int flag=1){
    int d=0;while((1<<d)*n!=FFT_MAXN)d++;
    rep(i,0,n)if(i<(bitrev[i]>>d))swap(a[i],a[bitrev[i]>>d]);
    for (int l=2;l<=n;l<<=1){
        int del=FFT_MAXN/l*flag;
        for (int i=0;i<n;i+=l){
            cp *le=a+i,*ri=a+i+(l>>1),*w=flag==1?nw:nw+FFT_MAXN;
            rep(k,0,l>>1){
                cp ne=*ri**w;
                *ri=*le-ne,*le=*le+ne;
                le++,ri++,w+=del;
            }
        }
    }
    if(flag!=1)rep(i,0,n)a[i].a/=n,a[i].b/=n;
}
void fft_init(){
    int L=0;while((1<<L)!=FFT_MAXN)L++;
    bitrev[0]=0;rep(i,1,FFT_MAXN)bitrev[i]=bitrev[i>>1]>>1|((i&1)<<(L-1));
    nw[0]=nw[FFT_MAXN]=(cp){1,0};
    rep(i,0,FFT_MAXN+1)nw[i]=(cp){cosl(2*pi/FFT_MAXN*i),sinl(2*pi/FFT_MAXN*i)}; //very
slow
}

void convo(db*a,int n,db*b,int m,db*c){
    static cp f[FFT_MAXN>>1],g[FFT_MAXN>>1],t[FFT_MAXN>>1];
    int N=2;while(N<=n+m)N<<=1;
    rep(i,0,N)
        if(i&1){
            f[i>>1].b=(i<=n)?a[i]:0.0;
            g[i>>1].b=(i<=m)?b[i]:0.0;
        }else{
            f[i>>1].a=(i<=n)?a[i]:0.0;
            g[i>>1].a=(i<=m)?b[i]:0.0;
        }
    dft(f,N>>1);dft(g,N>>1);
    int del=FFT_MAXN/(N>>1);
    cp qua=(cp){0,0.25},one=(cp){1,0},four=(cp){4,0},*w=nw;
```

```
    rep(i,0,N>>1){
        int j=i?(N>>1)-i:0;
        t[i]=(four*!(f[j]*g[j])-(!f[j]-f[i])*(!g[j]-g[i])*(one+*w))*qua;
        w+=del;
    }
    dft(t,N>>1,-1);
    rep(i,0,n+m+1)c[i]=(i&1)?t[i>>1].a:t[i>>1].b;
}

void mul(int *a,int *b,int n){// n<=N, 0<=a[i],b[i]<mo
    static cp f[N],g[N],t[N],r[N];
    int nn=2;while(nn<=n+n)nn<<=1;
    rep(i,0,nn){
        f[i]=(i<=n)?(cp){(db)(a[i]>>15),(db)(a[i]&32767)}:(cp){0,0};
        g[i]=(i<=n)?(cp){(db)(b[i]>>15),(db)(b[i]&32767)}:(cp){0,0};
    }
    swap(n,nn);
    dft(f,n,1);dft(g,n,1);
    rep(i,0,n){
        int j=i?n-i:0;
        t[i]=( (f[i]+!f[j])*(!g[j]-g[i]) + (!f[j]-f[i])*(g[i]+!g[j]) )*(cp){0,0.25};
        r[i]=(!f[j]-f[i])*(!g[j]-g[i])*(cp){-0.25,0} + (cp){0,0.25}*(f[i]+!f[j])*
(g[i]+!g[j]);
    }
    dft(t,n,-1); dft(r,n,-1);
    rep(i,0,n)a[i]=( (ll(t[i].a+0.5)%mo<<15) + ll(r[i].a+0.5) + (ll(r[i].b+0.5)%mo<<30)
)%mo;
}

```

3.18.2 NTT

jly板子

Z函数下不要引入std库

```
constexpr int P = 998244353;
using i64 = long long;
// assume -P <= x < 2P
int norm(int x) {
    if (x < 0) {
        x += P;
    }
    if (x >= P) {
        x -= P;
    }
    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)) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(P - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, P - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = i64(x) * rhs.x % P;
        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) {
        Z res = lhs;
        res -= rhs;
        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res /= rhs;
        return res;
    }
    friend std::istream &operator>>(std::istream &is, Z &a) {
        i64 v;
        is >> v;
        a = Z(v);
        return is;
    }
    friend std::ostream &operator<<(std::ostream &os, const Z &a) {
        return os << a.val();
    }
};

std::vector<int> rev;
std::vector<Z> roots{0, 1};
void dft(std::vector<Z> &a) {
    int n = a.size();

```

```
    if (int(rev.size()) != n) {
        int k = __builtin_ctz(n) - 1;
        rev.resize(n);
        for (int i = 0; i < n; i++) {
            rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
        }
    }

    for (int i = 0; i < n; i++) {
        if (rev[i] < i) {
            std::swap(a[i], a[rev[i]]);
        }
    }
    if (int(roots.size()) < n) {
        int k = __builtin_ctz(roots.size());
        roots.resize(n);
        while ((1 << k) < n) {
            Z e = power(Z(3), (P - 1) >> (k + 1));
            for (int i = 1 << (k - 1); i < (1 << k); i++) {
                roots[2 * i] = roots[i];
                roots[2 * i + 1] = roots[i] * e;
            }
            k++;
        }
    }
    for (int k = 1; k < n; k *= 2) {
        for (int i = 0; i < n; i += 2 * k) {
            for (int j = 0; j < k; j++) {
                Z u = a[i + j];
                Z v = a[i + j + k] * roots[k + j];
                a[i + j] = u + v;
                a[i + j + k] = u - v;
            }
        }
    }
}
void idft(std::vector<Z> &a) {
    int n = a.size();
    std::reverse(a.begin() + 1, a.end());
    dft(a);
    Z inv = (1 - P) / n;
    for (int i = 0; i < n; i++) {
        a[i] *= inv;
    }
}
struct Poly {
    std::vector<Z> a;
    Poly() {}
    Poly(const std::vector<Z> &a) : a(a) {}
    Poly(const std::initializer_list<Z> &a) : a(a) {}
    int size() const {
        return a.size();
    }
    void resize(int n) {
        a.resize(n);
    }
    Z operator[](int idx) const {
        if (idx < size()) {
            return a[idx];
        } else {
            return 0;
        }
```

```
    }
    Z &operator[](int idx) {
        return a[idx];
    }
    Poly mulxk(int k) const {
        auto b = a;
        b.insert(b.begin(), k, 0);
        return Poly(b);
    }
    Poly modxk(int k) const {
        k = std::min(k, size());
        return Poly(std::vector<Z>(a.begin(), a.begin() + k));
    }
    Poly divxk(int k) const {
        if (size() <= k) {
            return Poly();
        }
        return Poly(std::vector<Z>(a.begin() + k, a.end()));
    }
    friend Poly operator+(const Poly &a, const Poly &b) {
        std::vector<Z> res(std::max(a.size(), b.size()));
        for (int i = 0; i < int(res.size()); i++) {
            res[i] = a[i] + b[i];
        }
        return Poly(res);
    }
    friend Poly operator-(const Poly &a, const Poly &b) {
        std::vector<Z> res(std::max(a.size(), b.size()));
        for (int i = 0; i < int(res.size()); i++) {
            res[i] = a[i] - b[i];
        }
        return Poly(res);
    }
    friend Poly operator*(Poly a, Poly b) {
        if (a.size() == 0 || b.size() == 0) {
            return Poly();
        }
        int sz = 1, tot = a.size() + b.size() - 1;
        while (sz < tot) {
            sz *= 2;
        }
        a.a.resize(sz);
        b.a.resize(sz);
        dft(a.a);
        dft(b.a);
        for (int i = 0; i < sz; ++i) {
            a.a[i] = a[i] * b[i];
        }
        idft(a.a);
        a.resize(tot);
        return a;
    }
    friend Poly operator*(Z a, Poly b) {
        for (int i = 0; i < int(b.size()); i++) {
            b[i] *= a;
        }
        return b;
    }
    friend Poly operator*(Poly a, Z b) {
        for (int i = 0; i < int(a.size()); i++) {
            a[i] *= b;
        }
```

```
        return a;
    }
    Poly &operator+=(Poly b) {
        return (*this) = (*this) + b;
    }
    Poly &operator-=(Poly b) {
        return (*this) = (*this) - b;
    }
    Poly &operator*=(Poly b) {
        return (*this) = (*this) * b;
    }
    Poly deriv() const {
        if (a.empty()) {
            return Poly();
        }
        std::vector<Z> res(size() - 1);
        for (int i = 0; i < size() - 1; ++i) {
            res[i] = (i + 1) * a[i + 1];
        }
        return Poly(res);
    }
    Poly integr() const {
        std::vector<Z> res(size() + 1);
        for (int i = 0; i < size(); ++i) {
            res[i + 1] = a[i] / (i + 1);
        }
        return Poly(res);
    }
    Poly inv(int m) const {
        Poly x{a[0].inv()};
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x * (Poly{2} - modxk(k) * x)).modxk(k);
        }
        return x.modxk(m);
    }
    Poly log(int m) const {
        return (deriv() * inv(m)).integr().modxk(m);
    }
    Poly exp(int m) const {
        Poly x{1};
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x * (Poly{1} - x.log(k) + modxk(k))).modxk(k);
        }
        return x.modxk(m);
    }
    Poly pow(int k, int m) const {
        int i = 0;
        while (i < size() && a[i].val() == 0) {
            i++;
        }
        if (i == size() || 1LL * i * k >= m) {
            return Poly(std::vector<Z>(m));
        }
        Z v = a[i];
        auto f = divxk(i) * v.inv();
        return (f.log(m - i * k) * k).exp(m - i * k).mulxk(i * k) * power(v, k);
    }
    Poly sqrt(int m) const {
```

```
        Poly x{1};
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x + (modxk(k) * x.inv(k)).modxk(k)) * ((P + 1) / 2);
        }
        return x.modxk(m);
    }
    Poly mulT(Poly b) const {
        if (b.size() == 0) {
            return Poly();
        }
        int n = b.size();
        std::reverse(b.a.begin(), b.a.end());
        return ((*this) * b).divxk(n - 1);
    }
    std::vector<Z> eval(std::vector<Z> x) const {
        if (size() == 0) {
            return std::vector<Z>(x.size(), 0);
        }
        const int n = std::max(int(x.size()), size());
        std::vector<Poly> q(4 * n);
        std::vector<Z> ans(x.size());
        x.resize(n);
        std::function<void(int, int, int)> build = [&](int p, int l, int r) {
            if (r - l == 1) {
                q[p] = Poly{1, -x[l]};
            } else {
                int m = (l + r) / 2;
                build(2 * p, l, m);
                build(2 * p + 1, m, r);
                q[p] = q[2 * p] * q[2 * p + 1];
            }
        };
        build(1, 0, n);
        std::function<void(int, int, int, const Poly &)> work = [&](int p, int l, int r,
const Poly &num) {
            if (r - l == 1) {
                if (l < int(ans.size())) {
                    ans[l] = num[0];
                }
            } else {
                int m = (l + r) / 2;
                work(2 * p, l, m, num.mulT(q[2 * p + 1]).modxk(m - l));
                work(2 * p + 1, m, r, num.mulT(q[2 * p]).modxk(r - m));
            }
        };
        work(1, 0, n, mulT(q[1].inv(n)));
        return ans;
    }
};


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

    int n;
    std::cin >> n;

    std::vector<Z> fac(n + 1), invfac(n + 1), p(n + 1);
    fac[0] = p[0] = 1;
```

```
    for (int i = 1; i <= n; i++) {
        fac[i] = fac[i - 1] * i;
        p[i] = p[i - 1] * 26;
    }
    invfac[n] = fac[n].inv();
    for (int i = n; i > 0; i--) {
        invfac[i - 1] = invfac[i] * i;
    }

    std::vector<Z> f(n / 3 + 1);
    for (int i = 0; i <= n / 3; i++) {
        f[i] = fac[n - 2 * i] * invfac[i] * invfac[n - 3 * i] * p[n - 3 * i];
    }
    for (int i = 0; i <= n / 3; i++) {
        f[i] *= fac[i];
    }
    std::vector<Z> h(n / 3 + 1);
    for (int i = 0; i <= n / 3; i++) {
        h[i] = invfac[i] * (i % 2 ? -1 : 1);
    }
    auto g = Poly(f).mulT(Poly(h));

    for (int i = 0; i <= n; i++) {
        if (3 * i <= n) {
            g[i] *= invfac[i];
            std::cout << g[i];
        } else {
            std::cout << 0;
        }
        std::cout << " \n"[i == n];
    }

    return 0;
}

```

无模NTT

```
//https://ac.nowcoder.com/acm/contest/31454/E
using i64 = long long;
constexpr int NTT_N = (1 << 18) + 5;
constexpr i64 P = 3152519739159347311;
i64 mul (i64 a, i64 b) {
  return (__int128)a * (__int128)b % P;
  a %= P, b %= P;
  return ((a * b - P * (i64)((long double)a / P * b + 0.5)) % P + P) % P;
}
i64 power (i64 a, i64 b) {
  i64 ans = 1;
  while (b) {
    if (b & 1) ans = mul(ans, a);
    a = mul(a, a);
    b >>= 1;
  }
  return ans;
}
int ntt_n;
i64 omega[NTT_N], omega_inv[NTT_N];

void NTT_init(int n) {
```

```cpp
  ntt_n = n;
  i64 wn = power(3, (P - 1) / n);
  omega[0] = omega_inv[0] = 1;
  for (int i = 1; i < n; i++)
    omega_inv[n - i] = omega[i] = mul(omega[i - 1], wn);
}
void NTT (vector<i64> &a, int n, int tp) {
  for (int i = 1, j = 0, k; i < n - 1; i++) {
    k = n;
    do {
      j ^= (k >>= 1);
    } while (j < k);
    if (i < j) swap(a[i], a[j]);
  }
  for (int k = 2, m = ntt_n / 2; k <= n; k *= 2, m /= 2) {
    for (int i = 0; i < n; i += k) {
      for (int j = 0; j < k / 2; j++) {
        i64 w = (tp > 0 ? omega : omega_inv)[m * j];
        i64 u = a[i + j], v = mul(w, a[i + j + k / 2]);
        a[i + j] = (u + v) % P;
        a[i + j + k / 2] = (u - v + P) % P;
      }
    }
  }
  if (tp < 0) {
    i64 inv = power(n, P - 2);
    for (int i = 0; i < n; i++) a[i] = mul(a[i], inv);
  }
}
vector<i64> convo (const vector<i64> &a, const vector<i64> &b) {
  if (a.empty() || b.empty()) return {};
  int s = a.size() + b.size() - 1, B = 32 - __builtin_clz(s), n = 1 << B;
  i64 inv = power(n, P - 2);
  vector<i64> L(a), R(b), out(n);
  L.resize(n), R.resize(n);
  NTT(L, n, 1), NTT(R, n, 1);
  for (int i = 0; i < n; i++) {
    out[i] = mul(L[i], R[i]);
  }
  NTT(out, n, -1);
  return {out.begin(), out.begin() + s};
}

vector<i64> ans[NTT_N];
void Solve() {
  int n, q;
  cin >> n >> q;
  vector<int> p(n + 1), vis(n + 1);
  for (int i = 1; i <= n; i++) {
    cin >> p[i];
  }
  vector<i64> h(n + 1);
  set<int> cyclen;
  for (int i = 1; i <= n; i++) if (!vis[i]) {
    int u = i;
    vector<int> cyc;
    while (!vis[u]) {
      vis[u] = true;
      cyc.emplace_back(u);
      u = p[u];
    }
    int m = cyc.size();
```

```
    vector<i64> f(m), g(m);
    for (int j = 0; j < m; j++) {
      f[j] = cyc[j], g[j] = cyc[(m - j) % m];
      h[j] = 0;
    }
    int L = 1;
    while (L < 2 * m) L <<= 1;
    NTT_init(L);
    auto h2 = convo(f, g);
    for (int j = 0; j < h2.size(); j++) {
      h[j % m] += h2[j];
    }
    if (!cyclen.count(m)) {
      cyclen.emplace(m);
      ans[m] = vector<i64> (m, 0);
    }
    for (int j = 0; j < m; j++) {
      ans[m][j] += h[j];
    }
  }

  while (q--) {
    int x;
    cin >> x;
    i64 ret = 0;
    for (auto m : cyclen) {
      ret += ans[m][x % m];
    }
    cout << ret << '\n';
  }
}
```

3.18.3 分治NTT

```
std::function<Poly(int, int)> convo = [&] (int l, int r) {
    if (r - l == 1) {
      return Poly({1, v[l]});
    }
    int m = l + r >> 1;
    return convo(l, m) * convo(m, r);
};
```

3.19 拓展中国剩余定理 excrt

```
constexpr int N = 1e5 + 10;
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (!b) {
        x = 1; y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}
ll qmul (ll a, ll b, ll m) {
    ll ret = 0;
    while (b) {
        if (b & 1) ret = (ret + a) % m;
        a = (a + a) % m;
```

```
        b >>= 1;
    }
    return ret % m;
}
ll excrt (const vector<ll> &m, const vector<ll> &r, int n) {
    ll ans = r[0], M = m[0], t, y;
    for (int i = 1; i < n; i++) {
        ll mi = m[i], ret = ((r[i] - ans) % mi + mi) % mi;
        ll g = exgcd(M, mi, t, y);
        if (ret % g) {
            return -1;
        }
        t = qmul(t, ret / g, mi);
        ans += t * M;
        M = mi / g * M;
        ans = (ans % M + M) % M;
    }
    return ans;
}
```

3.20 Point

```
using T = double;
struct Point {
    T x;
    T y;
    Point(T x = 0, T y = 0) : x(x), y(y) {}

    Point &operator+=(const Point &p) {
        x += p.x, y += p.y;
        return *this;
    }
    Point &operator-=(const Point &p) {
        x -= p.x, y -= p.y;
        return *this;
    }
    Point &operator*=(const T &v) {
        x *= v, y *= v;
        return *this;
    }
    friend Point operator-(const Point &p) {
        return Point(-p.x, -p.y);
    }
    friend Point operator+(Point lhs, const Point &rhs) {
        return lhs += rhs;
    }
    friend Point operator-(Point lhs, const Point &rhs) {
        return lhs -= rhs;
    }
    friend Point operator*(Point lhs, const T &rhs) {
        return lhs *= rhs;
    }
};

T dot(const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y;
}

T cross(const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
```

```
}
```

4 动态规划

4.1 线性DP

4.1.2 最长上升子序列 O(nlogn)

```
// O(n^2)
int f[N], q[N]; // q原数组,f[i]表示以结尾的最长上升序列
int mx = 1;

for (int i = 1; i <= n; i ++ ) {
    f[i] = 1; // 设f[i]默认为1，找不到前面数字小于自己的时候就为1
    for (int j = 1; j < i; j ++ ) {
        if (q[i] > q[j]) f[i] = max(f[i], f[j] + 1);
    }
    mx = max(mx, f[i]);
}

// O(nlogn)
vector<int> stk;
stk.push_back(q[1]);
for (int i = 2; i <= n; i ++ ) {
    if (q[i] > stk.back()) stk.push_back(q[i]); // //如果该元素大于栈顶元素,将该元素入栈
    else *lower_bound(stk.begin(), stk.end(), q[i]) = q[i]; // //替换掉第一个大于或者等于这个数字的那个数
}
cout << stk.size() << endl;
```

4.1.3 最长公共子序列 O(n^2)

```
int a[N], b[N];
int f[N][N]; // f[i][j]表示a串前i个元素和b串前j个元素的最长公共子序列长度，注意下标从1开始

cin >> n >> m >> a + 1 >> b + 1;
for (int i = 1; i <= n; i ++ ) {
    for (int j = 1; j <= m; j ++ ) {
        if (a[i] == b[j]) {
            f[i][j] = f[i - 1][j - 1] + 1;
        }
        else {
            f[i][j] = max(f[i - 1][j], f[i][j - 1]);
        }
    }
}

// LCS + 输出字符串
cin >> n >> m >> a + 1 >> b + 1;
int A = strlen(a + 1), B = strlen(b + 1); // a串b串长度
for (int i = A; i >= 1; i -- ) {
    for (int j = B; j >= 1; j -- ) {
        if (a[i] == b[j]) f[i][j] = f[i + 1][j + 1] + 1;
        else f[i][j] = max (f[i + 1][j], f[i][j + 1]);
    }
}

int i = 1, j = 1;
```

```
while (i <= A && j <= B) {
    if (a[i] == b[j]) {
        cout << a[i]; // 输出最长公共子串
        i ++ , j ++ ;
    }
    else if (f[i][j] == f[i + 1][j]) i ++ ;
    else j ++ ;
}
```

4.1.4 最长公共上升子序列 O(n^2)

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

const int maxn = 3010;
int a[maxn], b[maxn];
int f[maxn][maxn]; // f[i] [j] 代表所有a[1 ~ i]和b[1 ~ j]中以b[j]结尾的公共上升子序列的长度最大值;
int n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i ++ ) cin >> a[i];
    for (int i = 1; i <= n; i ++ ) cin >> b[i];

    for (int i = 1; i <= n; i ++ ){
        int maxv = 1;
        for (int j = 1; j <= n; j ++ ){
            f[i][j] = f[i - 1][j];
            if (a[i] == b[j]) f[i][j] = max(f[i][j], maxv);
            if (a[i] > b[j]) maxv = max(maxv, f[i - 1][j] + 1);
        }
    }

    int ans = 0;
    for (int i = 1; i <= n; i ++ ) ans = max(ans, f[n][i]);
    cout << ans << endl;
}
```

4.2 背包

4.2.1 01背包 O(nm)

```
// 求恰好装满的最优解,f[0] = 0, f[1~m] = -inf;

int n, m;
int v[N], w[N];
int f[N];

int main()
{
    cin >> n >> m; // n件物品，背包容量为m

    for (int i = 1; i <= n; i ++ ){
        int v, w;
        cin >> v >> w; // v质量，w价值
        for (int j = m; j >= v; j -- ) f[j] = max(f[j], f[j - v] + w);
```

```
    }

    cout << f[m] << endl;

    return 0;
}
```

4.2.2 完全背包 O(nm)

```
// 每件物品有无限个
int f[N];

int main()
{
    cin >> n >> m; // n件物品，背包容量为m
    for (int i = 0; i <= n; i ++ ){
        int v, w;
        cin >> v >> w;// v质量，w价值
        // 正序枚举
        for (int j = v; j <= m; j ++ ) f[j] = max(f[j], f[j - v] + w);
    }

    cout << f[m] << endl;
    return 0;
}
```

4.2.3 多重背包

4.2.3.1 二进制优化 O (NVlogs)

```
 // 每件物品最多有Si个，用二进制优化后再做01背包
int v[N], w[N];
int f[M];

int main()
{
    cin >> n >> m;

    int cnt = 0; // 表示下标
    for (int i = 1; i <= n; i ++ ){
        int a, b, s; // a质量，b价值，s最多个数
        cin >> a >> b >> s;
        int k = 1; // 幂次
        while (k <= s){
            cnt ++;
            v[cnt] = a * k;
            w[cnt] = b * k;
            s -= k;
            k *= 2;
        }

        if (s > 0){  // 表示有大于幂次的部分
            cnt ++ ;
            v[cnt] = a * s;
            w[cnt] = b * s;
        }
    }
```

```
28
29      //n = cnt;
30
31      for (int i = 1; i <= cnt; i ++ ){
32          for (int j = m; j >= v[i]; j -- ){
33              f[j] = max(f[j], f[j - v[i]] + w[i]);
34          }
35      }
36
37      cout << f[m] << endl;
38
39      return 0;
40  }
```

4.2.3.2 单调队列优化 O (NV)

```
1   int f[N], g[N], q[N]; // fj为前i个物品，在体积j下的最大价值，g为前i-1个物品,在体积j下的最大价值，q
    为单调队列.
2
3   int main()
4   {
5       cin >> n >> m;
6       for (int i = 0; i < n; i ++ )
7       {
8           int v, w, s;
9           cin >> v >> w >> s;
10          memcpy(g, f, sizeof f);
11          for (int j = 0; j < v; j ++ )
12          {
13              int hh = 0, tt = -1;
14              for (int k = j; k <= m; k += v)
15              {
16                  if (hh <= tt && q[hh] < k - s * v) hh ++ ;
17                  while (hh <= tt && g[q[tt]] - (q[tt] - j) / v * w <= g[k] - (k - j) / v *
    w) tt -- ;
18                  q[ ++ tt] = k;
19                  f[k] = g[q[hh]] + (k - q[hh]) / v * w;
20              }
21          }
22      }
23
24      cout << f[m] << endl;
25
26      return 0;
27  }
```

4.2.4 分组背包 O (nms)

```
1   int v[N][N], w[N][N], s[N];
2   int f[N];
3
4   int main()
5   {
6       cin >> n >> m;
7
8       for (int i = 1; i <= n; i ++ ){
9           cin >> s[i];
```

```
        for (int j = 0; j < s[i]; j ++ ){
            cin >> v[i][j] >> w[i][j];
        }
    }

    for (int i = 1; i <= n; i ++ ){
        for (int j = m; j >= 0; j -- ){
            for (int k = 0; k < s[i]; k ++ ){
                if (v[i][k] <= j) f[j] = max(f[j], f[j - v[i][k]] + w[i][k]);
            }
        }
    }

    cout << f[m] << endl;

    return 0;
}
```

4.2.5 超大背包

```
// 当物品质量过高，则将价值和质量转换
int f[maxn]; // f[i]为价值为i下的最小体积
void solve()
{
    cin >> n >> m;
    memset(f, 0x3f, sizeof f);
    f[0] = 0;
    for (int i = 1; i <= n; i ++ ){
        int w, v;
        cin >> v >> w;
        // 用价值上限暴力枚举
        for (int j = 100000; j >= w; j -- ) f[j] = min(f[j], f[j - w] + v);
    }
    int ans = 0;
    for (int i = 1; i <= 100000; i ++ ){
        if (f[i] <= m) ans = max(ans, i); // 暴力枚举所有价值，当体积能装下即为一组合法解
    }
    cout << ans << endl;
}
```

4.2.6 二维费用背包(有物品数量限制的背包) O(nmk)

```
int n, V, M;
int f[maxn][maxn]; // 表示不超过i的第一费用，j的第二费用的最大价值。

int main()
{
    cin >> n >> V >> M; // M为第二重最大费用，或为物品数量限制
    for (int i = 1; i <= n; i ++ ){
        int v, m, w;
        cin >> v >> m >> w;
        for (int j = V; j >= v; j -- ){
            for (int k = M; k >= m; k -- ){ // 第二重费用，或为物品数量限制
                f[j][k] = max(f[j][k], f[j - v][k - m] + w);
            }
        }
    }
```

```
    cout << f[V][M] << endl;
}
```

4.2.7 混合背包

```
// 01背包，完全背包，多重背包的混合
// 考虑将01背包和完全背包转化为多重背包，再使用二进制优化求解多重背包
int n, m, v[100010], w[100010], dp[100010];

int main()
{
    cin >> n >> m; // n件物品，容量为m
    int cnt = 1;
    for(int i = 1; i <= n; i ++)
    {
        int a, b, s;
        cin >> a >> b >> s; //a体积，b价值
        int k = 1; // 幂次
        if(s < 0) s = 1; // s<0，01背包，只能拿一个
        else if(s == 0) s = m / a; // s=0,完全背包，最多拿m/a个
        while(k <= s)
        {
            v[cnt] = a * k;
            w[cnt] = b * k;
            s -= k;
            k *= 2;
            cnt ++;
        }
        if(s > 0)
        {
            v[cnt] = s * a;
            w[cnt] = s * b;
            cnt ++ ;
        }
    }
    for(int i = 1; i <= cnt; i ++)
    {
        for(int j = m; j >= v[i]; j -- )
        {
            dp[j] = max(dp[j], dp[j - v[i]] + w[i]);
        }
    }
    cout << dp[m];
}
```

4.2.6 背包问题输出方案

```
//用res[i]表示背包容量为i时上次选择了第几个物品.

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

const int maxn = 101000;
int n, m;
int dp[maxn], res[maxn];
int v[maxn], w[maxn];

int main () {
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= n; i ++ ) scanf("%d%d", &v[i], &w[i]);

```

```
        for (int i = 1; i <= n; i ++ ) {
                for (int j = v[i]; j <= m; j ++ ) {
                        if (dp[i - 1][j] < dp[i - 1][j - v[i]] + w[i]) {
                                res[j] = v[i];
                                dp[j] = dp[j - v[i]] + w[i];
                        }
                }
        }

        printf("%d\n", dp[m]);

        vector<int> ans;
        int now = dp[m];
        while (res[now] != 0) {
            ans.push_back(res[now]);
            now -= res[now];
        }

        for (int i = ans.size() - 1; i >= 0; i -- ) {
            printf("%lld ", ans[i]);
        }

}
```

4.3 区间DP O(n^3)

```
int f[maxn][maxn], g[maxn][maxn]; // 合并i~j所需的最小/最大价值
int s[maxn]; // 原数组的前缀和
int w[maxn];

for (int i = 1; i <= n; i ++ ) {
        cin >> w[i];
        w[i + n] = w[i]; // 若有环形要求
    }

for (int i = 1; i <= n * 2; i ++ ) s[i] = s[i - 1] + w[i];

memset(f, 0x3f, sizeof f);
memset(g, -0x3f, sizeof g);

for (int len = 1; len <= n; len ++ ) { // 枚举区间长度
    for (int l = 1; l + len - 1 <= n * 2; l ++ ) { // 枚举左端点 (若无环形则到n)
        int r = l + len - 1; // 找到右端点
        if (l == r) f[l][r] = g[l][r] = 0;
        else {
            for (int k = l; k < r; k ++ ) { // 枚举分界点
                f[l][r] = min(f[l][r], f[l][k] + f[k + 1][r] + s[r] - s[l - 1]); //
minvalue
                g[l][r] = max(g[l][r], g[l][k] + g[k + 1][r] + s[r] - s[l - 1]); //
maxvalue
            }
        }
    }
}

int mi = inf, mx = -inf;
for (int i = 1; i <= n; i ++ ) {
    // 若无环形则直接f[1][n];
    mi = min(mi, f[i][i + n - 1]);
    mx = max(mx, g[i][i + n - 1]);
```

```
}
cout << mi << endl;
cout << mx << endl;

/*
将ax和ax+1合并为ax*ax+1，获得(ax-ax+1)^2
注：不用考虑合并的和得到的分数计算方式不相等，合并完后就是改变ax和ax+1的值，直接照题目计算即可
*/
void solve() {

    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= n; i++) {
        s[i][i] = a[i];
        for (int j = i + 1; j <= n; j++) {
            s[i][j] = s[i][j - 1] * a[j] % mod;
        }
    }

    for (int len = 2; len <= n; len++) {
        for (int l = 1; l <= n - len + 1; l++) {
            int r = l + len - 1;
            for (int k = l; k < r; k++) {
                dp[l][r] = max(dp[l][r], dp[l][k] + dp[k + 1][r] + (s[l][k] - s[k + 1][r]) * (s[l][k] - s[k + 1][r]));
            }
        }
    }
    cout << dp[1][n] << endl;


}
```

4.4 树形DP

```
// 1.树中最长路径，求距离型
int dfs(int u, int f) {
    int dist = 0;
    int d1 = 0, d2 = 0; // d1为最长路径，d2为次长路径

    for (int i = h[u]; i != -1; i = ne[i]) {
        int j = e[i];
        if (j == f) continue;
        int d = dfs(j, u) + w[i];
        dist = max(dist, d);
        if (d >= d1) d2 = d1, d1 = d;
        else if (d > d2) d2 = d;
    }
    ans = max(ans, d1 + d2);
    return dist;
}
int main () {
    cin >> n;
    memset(h, -1, sizeof h);
    for (int i = 1; i < n; i ++ ) {
        int a, b, c;
        cin >> a >> b >> c;
        add(a, b, c), add(b, a, c);
    }
    dfs(1, -1);
```

```
    cout << ans << endl;
}

//
```

4.5 状压DP

4.6 数位DP

```
// 统计0~N有多少位含i
int dgt(int n) {
    int res = 0;
    while (n) ++ res, n /= 10;
    return res;
}

int count (int n, int i) {
    int res = 0, d = dgt(n);
    for (int j = 1; j <= d; j ++ ) {
        int p = pow(10, j - 1), l = n / p / 10, r = n % p, dj = n / p % 10;
        if (i) res += l * p;
        if (!i && l) res += (l - 1) * p;
        if ( (dj > i) && (i || l) ) res += p;
        if ( (dj == i) && (i || l) ) res += r + 1;
    }
    return res;
}

// 统计数位上至少有k个某数
#include <bits/stdc++.h>
using namespace std;

const int maxn = 210;
int f[maxn][maxn];
int k, b;

void init () {
        for (int i = 0; i < maxn; i ++ ) {
                for (int j = 0; j <= i; j ++ ) {
                        if (!j) f[i][j] = 1;
                        else f[i][j] = f[i - 1][j] + f[i - 1][j - 1];
                }
        }
}

int dp (int n) {
        if (!n) return 0;

        vector<int> nums;
        while (n) nums.push_back(n % b), n /= b;

        int res = 0;
        int ls = 0;

        for (int i = nums.size() - 1; i >= 0; i -- ) {
                int x = nums[i];
                if (x) {
                        res += f[i][k - ls];
```

```
                        if (x > 1) {
                                if (k - ls - 1 >= 0) res += f[i][k - ls - 1];
                                break;
                        }
                        else {
                                ls ++ ;
                                if (ls > k) break;
                        }
                }
                if (!i && ls == k) res ++ ;
        }

        return res;
}

int main () {
        init ();
        int l, r;
        cin >> l >> r >> k >> b;
        cout << dp(r) << ' ' << dp(l - 1) << endl;
}


// 统计l~r中有多少个满足条件的数（单调增，减，不出现某数等）
#include <bits/stdc++.h>
using namespace std;

const int maxn = 15;

int f[maxn][maxn];

void init () {
        for (int i = 0; i <= 9; i ++ ) f[1][i] = 1;

        for (int i = 2; i < maxn; i ++ ) {
                for (int j = 0; j <= 9; j ++ ) {
                        for (int k = j; k <= 9; k ++ ) {
                                f[i][j] += f[i - 1][k];
                        }
                }
        }
}

int dp(int n) {
        if (!n) return 1;

        vector<int> nums;
        while (n) nums.push_back(n % 10), n /= 10;
        int res = 0;
        int ls = 0;

        for (int i = nums.size() - 1; i >= 0; i -- ) {
                int x = nums[i];
                for (int j = ls; j < x; j ++ ) {
                        res += f[i + 1][j];
                }

                if (x < ls) break;
                ls = x;

                if (!i) res ++ ;
        }
```

```
        return res;
}

int main () {
        init();
        int l, r;
        while (cin >> l >> r) {
                cout << dp(r) - dp(l - 1) << endl;
        }
}
```

4.7 单调队列优化DP

```
//单调队列常用来优化：i的前m个范围内区间的最值问题。（注意，此处的范围都是一个定值

//n个物品中，连续m个中至少选出一个，总价值最小。
//或   不能连续k个在一起，即至少从k+1个中选出一个不选。
#include <bits/stdc++.h>
using namespace std;

const int maxn = 200010;
int n, m;
int w[maxn], dp[maxn];
int q[maxn];

int main () {
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= n; i ++ ) scanf("%d", &w[i]);

        int hh = 0, tt = 0;
        for (int i = 1; i <= n; i ++ ) {
                while (hh <= tt && i - q[hh] > m) hh ++ ;
                dp[i] = dp[q[hh]] + w[i];
                while (hh <= tt && dp[q[tt]] >= dp[i]) tt -- ;
                q[ ++ tt] = i;
        }

        if (n + 1 - m > q[hh]) hh ++ ;
        printf("%d\n", dp[q[hh]]);
}

```

5. 字符串

5.1 KMP

```
const int mxn = 1e6 + 100;
struct KMP {
    int ne[mxn], len;
    string t;
    void clear() {
        len = ne[0] = ne[1] = 0;
    }
    void init (string s) { /*1-bas*/
        len = s.size() - 1;
        t = s;
        for (int i = 2; i <= len; i ++ ) {
```

```
            ne[i] = ne[i - 1];
            while (ne[i] && s[i] != s[ne[i] + 1]) ne[i] = ne[ne[i]];
            ne[i] += (s[i] == s[ne[i] + 1]);
        }
    }
    vector<int> match (string s) { /*get start_pos*/
        int len_s = s.size() - 1;
        vector<int> st_pos(0);
        for (int i = 1, j = 1; i <= len_s;) {
            while (j != 1 && s[i] != t[j]) j = ne[j - 1] + 1;
            if (s[i] == t[j]) j ++ , i ++ ;
            else i ++ ;
            if (j == len + 1) {
                st_pos.push_back(i - j + 1);
                j = ne[len] + 1;
            }
        }
        return st_pos;
    }
    void debug () {
        for (int i = 0; i <= len; i ++ ) {
            printf("[debug] nxt[%d]=%d\n", i, ne[i]);
        }
    }
   /* 循环周期 形如 acaca 中 ac 是一个合法周期 */
    vector<int> periodic() {
        vector<int> ret;
        int now = len;
        while (now) {
            now = ne[now];
            ret.push_back(len - now);
        }
        return ret;
    }
    /* 循环节 形如 acac 中ac、acac是循环节，aca不是*/
    vector<int> periodic_loop() {
        vector<int> ret;
        for (int x : periodic()) {
            if (len % x == 0) ret.push_back(x);
        }
        return ret;
    }
    int min_periodic_loop() {
        return periodic_loop()[0];
    }
}kmp;
```

5.2 Manacher

```
struct Manacher {
    int lc[maxn];
    string ch;
    int N;
    // Manacher(string s) {init(s); manacher();}
    /* s 1 bas */
    void init (string s) {
        int n = sz(s) - 1;
        ch.resize(n * 2 + 10);
        ch[n * 2 + 1] = '#';
        ch[0] = '@';
```

```
        ch[n * 2 + 2] = '\0';
        for (int i = n; i >= 1; i -- ) {
            ch[i * 2] = s[i]; ch[i * 2 - 1] = '#';
        }
        N = 2 * n + 1;
    }
    void work() {
        lc[1] = 1; int k = 1;
        for (int i = 2; i <= N; i ++ ) {
            int p = k + lc[k] - 1;
            if (i <= p) {
                lc[i] = min(lc[2 * k - i], p - i + 1);
            }
            else lc[i] = 1;
            while (ch[i + lc[i]] == ch[i - lc[i]]) lc[i] ++ ;
            if (i + lc[i] > k + lc[k]) k = i;
        }
    }
    void debug () {
        puts(ch.c_str());
        for (int i = 1; i <= N; i ++ ) {
            printf("lc[%d]=%d\n", i, lc[i]);
        }
    }
}manacher;

void solve() {
    string s; cin >> s;
    s = " " + s;
    manacher.init(s); manacher.work();
    manacher.debug();
}
```

5.3 Hash

```
using ull = unsigned long long;
struct Hash {
    static constexpr int sigma = 60 * 60; /* 字符集大小 */
    static constexpr int HASH_CNT = 1; /* hash次数 */
    ull Prime_Pool[5] = {1998585857ul, 23333333333ul};
    ull Seed_Pool[5] = {146527, 911, 19260817, 91815541};
    ull Mod_Pool[5] = {998244353, 29123, 1000000009, 4294967291ull};
    ull seed, mod;
    vector<array<ull, HASH_CNT>> base, sum;
    int perm[sigma];

    Hash (const string &s) {
        Init(s);
    }
    void Init (const string &s) {
        int n = s.size() - 1;
        base.resize(n + 1, {}), sum.resize(n + 1, {});
        for (int _ = 0; _ < HASH_CNT; _++) {
            seed = Seed_Pool[_], mod = Mod_Pool[_];
            base[0][_] = 1;
            for (int i = 1; i <= n; i++) {
                base[i][_] = base[i - 1][_] * seed % mod;
                sum[i][_] = (sum[i - 1][_] * seed % mod + s[i]) % mod;
            }
        }
```

```
    }
    array<ull, HASH_CNT> hash (int l, int r) {
        array<ull, HASH_CNT> rt = {};
        for (int _ = 0; _ < HASH_CNT; _++) {
            int mod = Mod_Pool[_];
            rt[_] = (sum[r][_] - sum[l - 1][_] * base[r - l + 1][_] % mod + mod) % mod;
        }
        return rt;
    }
};
```

```
const int sigma = 60 * 60; /* 字符集大小 */
const int HASH_CNT = 2; /* hash次数 */
int s[maxn];
 /* char* 1-bas
 * sum[i] = s[i]+s[i-1]*Seed+s[i-2]*Seed^2+...+s[1]*Seed^(i-1)*/
ULL Prime_Pool[] = {1998585857ul, 23333333333ul};
ULL Seed_Pool[] = {911, 146527, 19260817, 91815541};
ULL Mod_Pool[] = {29123, 998244353, 1000000009, 4294967291ull};
struct Hash {
    ULL seed, mod;
    ULL base[maxn], sum[maxn];
    int perm[sigma];
    void init (int seedindex, int modindex) {
        seed = Seed_Pool[seedindex], mod = Mod_Pool[modindex];
        base[0] = 1;
        for (int i = 1; i <= n; i ++ ) {
            base[i] = base[i - 1] * seed % mod;
        }
        for (int i = 1; i <= n; i ++ ) {
            sum[i] = (sum[i - 1] * seed % mod + s[i]) % mod;
        }
    }
    /*random_shuffle 离散化id，防止kill_hash*/
    void index_init(int seedindex, int modindex) {
        seed = Seed_Pool[seedindex], mod = Mod_Pool[modindex];
        base[0] = 1;
        for (int i = 1; i <= n; i ++ ) {
            base[i] = base[i - 1] * seed % mod;
        }
        iota (perm + 1, perm + sigma + 1, 1);
        random_shuffle(perm + 1, perm + sigma + 1);
        for (int i = 1; i <= n; i ++ ) {
            sum[i] = (sum[i - 1] * seed % mod + perm[s[i]]) % mod;
        }
    }
    ULL gethash (int l, int r) {
        return (sum[r] - sum[l - 1] * base[r - l + 1] % mod + mod) % mod;
    }
}hasher[HASH_CNT];
pair<ULL, ULL> hashrange(int l, int r) {
    return mkp(hasher[0].gethash(l, r), hasher[1].gethash(l, r));
}
map<char, int> id; int idcnt;

void solve() {
    read(n), read(m);
    string a; cin >> a;
    for (int i = 0; i < n; i ++ ) {
        if (!id.count(a[i])) id[a[i]] = ++ idcnt;
```

```
        s[i + 1] = id[a[i]];
    }
    for (int i = 0; i < HASH_CNT; i ++ ) hasher[i].index_init(i, i);

    while (m -- ) {
        int l1, r1, l2, r2;
        read(l1), read(r1), read(l2), read(r2);
        if (hashrange(l1, r1) == hashrange(l2, r2)) puts("Yes");
        else puts("No");
    }
}
```

二维哈希

```
constexpr int N = 2e3 + 10;
int a[N + 5][N + 5], n, m;
struct Hash1 {
    const int p1 = 31, p2 = 1331, mod = 1e9 + 9;
    void add (int& x, int y) {
        if ((x += y) >= mod)
            x -= mod;
    }
    void sub (int& x, int y) {
        if ((x -= y) < 0)
            x += mod;
    }
    int invth1[N + 5], invth2[N + 5];
    int Pow1[N + 5], Pow2[N + 5];
    int sum1[N + 5][N + 5], sum2[N + 5][N + 5];
    int power (int a, int x) {
        int ans = 1;
        while (x) {
            if (x & 1)
                ans = 1ll * ans * a % mod;
            a = 1ll * a * a % mod;
            x >>= 1;
        }
        return ans;
    }
    int inv (int a) {
        return power (a, mod - 2);
    }
    void init () {
        Pow1[0] = Pow2[0] = invth1[0] = invth2[0] = 1;
        int invp1 = inv (p1), invp2 = inv (p2);
        for (int i = 1; i < N + 5; i++) {
            Pow1[i] = 1ll * Pow1[i - 1] * p1 % mod;
            Pow2[i] = 1ll * Pow2[i - 1] * p2 % mod;
            invth1[i] = 1ll * invth1[i - 1] * invp1 % mod;
            invth2[i] = 1ll * invth2[i - 1] * invp2 % mod;
        }
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++) {
                sum1[i][j] = 1ll * Pow1[i] * Pow2[j] % mod * a[i][j] % mod;
                sum2[i][j] = 1ll * Pow1[n - i + 1] * Pow2[m - j + 1] % mod * a[i][j] %
mod;
            }
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++) {
                add (sum1[i][j], sum1[i - 1][j]);
                add (sum1[i][j], (sum1[i][j - 1] - sum1[i - 1][j - 1] + mod) % mod);
```

```
                add (sum2[i][j], sum2[i - 1][j]);
                add (sum2[i][j], (sum2[i][j - 1] - sum2[i - 1][j - 1] + mod) % mod);
            }
    }
    bool check (int l, int r, int u, int d) {
        int g1 = 0;
        add (g1, sum1[u][d]);
        sub (g1, sum1[l - 1][d]);
        sub (g1, sum1[u][r - 1]);
        add (g1, sum1[l - 1][r - 1]);
        int g2 = 0;
        add (g2, sum2[u][d]);
        sub (g2, sum2[l - 1][d]);
        sub (g2, sum2[u][r - 1]);
        add (g2, sum2[l - 1][r - 1]);
        g1 = 1ll * g1 * invth1[l - 1] % mod * invth2[r - 1] % mod;
        g2 = 1ll * g2 * invth1[n - u] % mod * invth2[m - d] % mod;
        return g1 == g2;
    }
};
Hash1 h1;
struct Hash2 {
    const int p1 = 131, p2 = 911, mod = 1e9 + 9;
    void add (int& x, int y) {
        if ((x += y) >= mod)
            x -= mod;
    }
    void sub (int& x, int y) {
        if ((x -= y) < 0)
            x += mod;
    }
    int invth1[N + 5], invth2[N + 5];
    int Pow1[N + 5], Pow2[N + 5];
    int sum1[N + 5][N + 5], sum2[N + 5][N + 5];
    int power (int a, int x) {
        int ans = 1;
        while (x) {
            if (x & 1)
                ans = 1ll * ans * a % mod;
            a = 1ll * a * a % mod;
            x >>= 1;
        }
        return ans;
    }
    int inv (int a) {
        return power (a, mod - 2);
    }
    void init () {
        Pow1[0] = Pow2[0] = invth1[0] = invth2[0] = 1;
        int invp1 = inv (p1), invp2 = inv (p2);
        for (int i = 1; i < N + 5; i++) {
            Pow1[i] = 1ll * Pow1[i - 1] * p1 % mod;
            Pow2[i] = 1ll * Pow2[i - 1] * p2 % mod;
            invth1[i] = 1ll * invth1[i - 1] * invp1 % mod;
            invth2[i] = 1ll * invth2[i - 1] * invp2 % mod;
        }
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++) {
                sum1[i][j] = 1ll * Pow1[i] * Pow2[j] % mod * a[i][j] % mod;
                sum2[i][j] = 1ll * Pow1[n - i + 1] * Pow2[m - j + 1] % mod * a[i][j] %
mod;
            }
```

```
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++) {
                add (sum1[i][j], sum1[i - 1][j]);
                add (sum1[i][j], (sum1[i][j - 1] - sum1[i - 1][j - 1] + mod) % mod);
                add (sum2[i][j], sum2[i - 1][j]);
                add (sum2[i][j], (sum2[i][j - 1] - sum2[i - 1][j - 1] + mod) % mod);
            }
    }
    bool check (int l, int r, int u, int d) {
        int g1 = 0;
        add (g1, sum1[u][d]);
        sub (g1, sum1[l - 1][d]);
        sub (g1, sum1[u][r - 1]);
        add (g1, sum1[l - 1][r - 1]);
        int g2 = 0;
        add (g2, sum2[u][d]);
        sub (g2, sum2[l - 1][d]);
        sub (g2, sum2[u][r - 1]);
        add (g2, sum2[l - 1][r - 1]);
        g1 = 1ll * g1 * invth1[l - 1] % mod * invth2[r - 1] % mod;
        g2 = 1ll * g2 * invth1[n - u] % mod * invth2[m - d] % mod;
        return g1 == g2;
    }
};
Hash2 h2;
bool check (int a, int b, int c, int d) {
    if (!(a >= 1 && a <= n && b >= 1 && b <= m && c >= 1 && c <= n && d >= 1 && d <= m
&& a <= c && b <= d))
        return false;
    return h1.check (a, b, c, d) && h2.check (a, b, c, d);
}
```

5.4 ACAM

```
const int maxn = 300010;
queue<int> q;
//int vis[maxn]; 用于做出现次数
struct Aho_Corasick_Automaton {
  int c[maxn][26], fail[maxn], cnt, val[maxn];
//  int flag[maxn], in[maxn], ans[maxn];  // 用于做出现次数

  void ins (string s) { // 不加x,即为出现个数
    int len = s.size();
    int now = 0;
    for (int i = 0; i < len; i ++ ) {
      int v = s[i] - 'a';
      if (!c[now][v]) c[now][v] = ++ cnt;
      now = c[now][v];
    }
//      if (!flag[now]) flag[now] = x; // 用于做出现次数
//      val[x] = flag[now];

//      val[now] ++ ; // 用于做出现个数

  }

  void build() {
    for (int i = 0; i < 26; i ++ ) {
      if (c[0][i]) {
        fail[c[0][i]] = 0;
```

```
        q.push(c[0][i]);
      }
    }
    while (q.size()) {
      int t = q.front(); q.pop();
      for (int i = 0; i < 26; i ++ ) {
        if (c[t][i]) {
          fail[c[t][i]] = c[fail[t]][i];
//                  in[fail[c[t][i]]] ++ ; // 用于做出现次数
          q.push(c[t][i]);
        }
        else c[t][i] = c[fail[t]][i];
      }
    }
  }
  void query (char *s) {
    int len = strlen(s);
    int now = 0;
//      int ans = 0;
    for (int i = 0; i < len; i ++ ) {
      now = c[now][s[i] - 'a'];
//      ans[now] ++ ; // 用于做出现次数
//          for (int t = now; t && ~val[t]; t = fail[t]) {  // 用于做出现个数
//              ans += val[t], val[t] = -1;
//          }
    }
//      return ans; 最终出现个数
  }

//  void topsort() { // 用于做出现次数
//      for (int i = 1; i <= cnt; i ++ ) {
//          if (!in[i]) q.push(i);
//      }
//      while (q.size()) {
//          int t = q.front(); q.pop();
//          vis[flag[t]] = ans[t];
//          int v = fail[t];
//          in[v] -- ;
//          ans[v] += ans[t];
//          if (!in[v]) q.push(v);
//      }
//  }

  void clear () {
    cnt = 0;
    memset(c, 0, sizeof c);
    memset(val, 0, sizeof val);
    memset(fail, 0, sizeof fail);
//    memset(flag, 0, sizeof flag); // 用于做出现次数
//    memset(ans, 0, sizeof ans);
//    memset(in, 0, sizeof in);
  }
}acam;
int n;
string p[200];
char t[1000010];

// vis[AC.val[i]] 第i个串的出现个数.
int main () {
  while (scanf("%d", &n), n) {
    acam.clear();
    for (int i = 1; i <= n; i ++ ) {
```

```
      cin >> p[i];
      acam.ins(p[i], i);
    }
    acam.build();
    scanf("%s", t);
    acam.query(t);
    acam.topsort();

    int mx = 0;
    for (int i = 1; i <= n; i ++ ) {
      mx = max(mx, vis[acam.val[i]]);
      //        printf("%d\n", vis[AC.val[i]]);
    }

    printf("%d\n", mx);
    for (int i = 1; i <= n; i ++ ) {
      if (vis[acam.val[i]] == mx) cout << p[i] << endl;
    }
  }
}
```

5.5 最小表示法

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

char s[1000];

int main () {
    scanf("%s", s + 1);
    int n = strlen(s + 1);
    for (int i = 1; i <= n; i ++ ) {
        s[n + i] = s[i];
    }
    int i = 1, j = 2, k;
    while (i <= n && j <= n) {
        for (k = 0; k < n && s[i + k] == s[j + k]; k ++ );
        if (k == n) break;
        if (s[i + k] > s[j + k]) {
            i += k + 1;
            if (i == j) i ++ ;
        }
        else {
            j += k + 1;
            if (i == j) j ++ ;
        }
    }
    int ans = min(i, j);
    for (int i = ans; i <= ans + n - 1; i ++ ) printf("%c", s[i]);
}
```

5.6 Trie

```
// 字典树
const int mxn = 1e6 + 100;
struct Trie {
    int nxt[mxn << 1][26], cnt[mxn << 1];
    int c = 0;
    void clear() {
        c = 0;
```

```
        ms(nxt[0], 0);
        ms(cnt, 0);
    }

    inline int newnode () {
        c ++ ;
        return c;
    }

    void insert (string s) {
        int u = 0, now = 0;
        while (now < sz(s)) {
            u = insert(u, s[now] - 'a');
            now ++ ;
        }
        cnt[u] ++ ;
    }
    inline int insert (int pre, int ch) {
        return nxt[pre][ch] ? nxt[pre][ch] : nxt[pre][ch] = newnode();
    }

    inline int query (string s) {
        int u = 0, now = 0;
        while (now < sz(s)) {
            if (!nxt[u][s[now] - 'a']) return 0;
            else u = nxt[u][s[now] - 'a'];
            now ++ ;
        }
        return cnt[u];
    }
}trie;

void solve() {
        read(n);
        for (int i = 1; i <= n; i ++ ) {
                string op, a;
                cin >> op >> a;
                if (op == "I") trie.insert(a);
                else printf("%lld\n", trie.query(a));
        }
}


// 01trie 最大区间异或和
const int mxn = 5e5 + 100;
struct Trie {
    int nxt[mxn << 2][2], l[mxn << 2];
    int cnt, ansl, ansr, ansv;

    inline void init () {
        cnt = 0; ansv = -1;
        ms(nxt[0], 0);
        ms(l, 0x3f);
    }

    inline int create () {
        cnt ++ ;
        ms(nxt[cnt], 0);
        return cnt;
    }

    inline void insert (int id, int x) {
```

```
        int u = 0;
        for (int i = 31; i >= 0; i -- ) {
            int t = ((x >> i) & 1);
            if (!nxt[u][t]) nxt[u][t] = create();
            u = nxt[u][t];
        }
        l[u] = id;
//      l[u] = min(l[u], id);
    }
    inline void query (int id, int x) {
        int u = 0, res = 0;
//      de(ansv); de(x);
        for (int i = 31; i >= 0; i -- ) {
            int t = ((x >> i) & 1);
            if (nxt[u][!t]) {
                u = nxt[u][!t];
                res += 1ll << i;
            }
            else u = nxt[u][t];
        }
//      de(id); de(res);
//      if (res == ansv) {
//          if (l[u] < ansl) {
//              ansl = l[u]; ansr = id;
//          }
//      }
        if (res > ansv) {
            ansv = res; ansl = l[u]; ansr = id;
        }
    }
    /*
    带删除的操作，需要另开tag数组
    查询时需额外判断 tag[nxt[u][!t]]
    inline void erase (int x) {
        int u = 0;
        for (int i = 31; i >= 0; i -- ) {
            int t = ((x >> i) & 1);
            u = nxt[u][t];
            tag[u] -- ;
        }
    }
    */
}trie;

void solve() {
    read(n);
    trie.init(); trie.insert(0, 0);
    int sum = 0;
    for (int i = 1; i <= n; i ++ ) {
        int x; read(x); sum ^= x;
        trie.query(i, sum); trie.insert(i, sum);
    }
    printf("%lld %lld %lld\n", trie.ansv, trie.ansl + 1, trie.ansr);
}


// 可持久化01trie
const int mxn = 6e5 + 100;
struct Tire {
    int nxt[mxn * 25][2], sum[mxn * 25];
    int root[mxn * 25];
    int cnt;
```

```
    inline void init () {
        cnt = 0;
        ms(nxt[0], 0);
        root[0] = root[1] = sum[0] = sum[1] = 0;
    }

    inline int create () {
        cnt ++ ;
        ms(nxt[cnt], 0);
        return cnt;
    }

    inline int insert (int x, int pre) {
        int u = ++ cnt, t = u;
        for (int i = 30; i >= 0; i -- ) {
            int t = ((x >> i) & 1);
            nxt[u][0] = nxt[pre][0], nxt[u][1] = nxt[pre][1];
            sum[u] = sum[pre] + 1;
            nxt[u][t] = ++ cnt;
            u = nxt[u][t], pre = nxt[pre][t];
        }
        sum[u] = sum[pre] + 1;
        return t;
    }

    inline int query (int x, int l, int r) {
        int res = 0;
        for (int i = 30; i >= 0; i -- ) {
            int t = !((x >> i) & 1);
            if (sum[nxt[r][t]] - sum[nxt[l][t]] > 0) {
                res |= (1ll << i);
                l = nxt[l][t], r = nxt[r][t];
            }
            else l = nxt[l][!t], r = nxt[r][!t];
        }
        return res;
    }
}trie;

void solve() {
    read(n), read(m);
    int now = 0;
    n ++ ;
    trie.root[1] = trie.insert(now, trie.root[0]);
    for (int i = 2; i <= n; i ++ ) {
        int x; read(x);
        now ^= x;
        trie.root[i] = trie.insert(now, trie.root[i - 1]);
    }

    while (m -- ) {
        char op[2];
        scanf("%s", op);
        if (*op == 'A') {
            int x; read(x); now ^= x;
            n ++ ;
            trie.root[n] = trie.insert(now, trie.root[n - 1]);
        }
        else {
            int l, r, x;
            read(l), read(r), read(x);
```

```
            int tmp = now ^ x;
            ll ans = trie.query(tmp, trie.root[l - 1], trie.root[r]);
            printf("%lld\n", ans);
        }
    }
}
```

5.7 PAM

```
//max(len(t)*cnt(t)) t为s回文子串，cnt(t)=t出现次数
const int maxn = 300010;
struct PAM {
        int s[maxn], now;
        int nxt[maxn][26], fail[maxn], l[maxn], last, tot;
        int num[maxn]; /*节点代表的所有回文串出现次数*/
        il void clear() {
                s[0] = l[1] = -1;
                fail[0] = tot = now = 1;
                last = l[0] = 0;
                ms(nxt[0], 0); ms(nxt[1], 0);
        }
        PAM () { clear(); }
        il int new_node (int ll) {
                tot ++ ;
                ms(nxt[tot], 0);
                fail[tot] = num[tot] = 0;
                l[tot] = ll;
                return tot;
        }
        il int get_fail(int x) {
                while (s[now - l[x] - 2] != s[now - 1]) x = fail[x];
                return x;
        }
        il void add (int ch) {
                s[now ++ ] = ch;
                int cur = get_fail(last);
                if (!nxt[cur][ch]) {
                        int tt = new_node(l[cur] + 2);
                        fail[tt] = nxt[get_fail(fail[cur])][ch];
                        nxt[cur][ch] = tt;
                }
                last = nxt[cur][ch]; num[last] ++ ;
        }
        il void build () {
                for (int i = tot; i >= 2; i -- ) {
                        num[fail[i]] += num[i];
                }
                num[0] = num[1] = 0;
        }
        il void init (char *ss) {
                while (*ss) {
                        add(*ss - 'a'); ss ++ ;
                }
        }
        il void init (string ss) {
                for (auto x : ss) add(x - 'a');
        }
        il int query();
};
il int PAM::query () {
        int res = 1;
```

```
        for (int i = 2; i <= tot; i ++ ) res = max(res, num[i] * l[i]);
        return res;
}
PAM p;
void solve() {
        cin >> n;
        string s; cin >> s;
        p.init(s); p.build();
        printf("%lld\n", p.query());
}
```

5.8 Z

```
//result: ext[i] = LCP(S[i,lens],T)
//require: z[i] = LCP(T[i,lent],T)
int z[maxn], ext[maxn];
il void exkmp (string s, int lens, string t, int lent, int *ext, int *z) {
        ext[0] = 0;
        for (int i = 1, l = 0, r = 0; i <= lens; i ++ ) {
                ext[i] = (i <= r) ? min(z[i - l + 1], r - i + 1) : 0ll;
                while (i + ext[i] <= lens && ext[i] < lent && s[i + ext[i]] == t[ext[i] + 1]) ext[i] ++ ;
                if (i + ext[i] - 1 >= r && i != 1) l = i, r = i + ext[i] - 1;
        }
}

void solve() {
        string a, b; cin >> a >> b;
        a = " " + a, b = " " + b;
        exkmp(b, sz(b) - 1, b, sz(b) - 1, z, z);
        exkmp(a, sz(a) - 1, b, sz(b) - 1, ext, z);
}
```

5.9 序列自动机

$nxt[i][j]$表示i后的第一个字符j的位置

```
vector<array<int, 27>> nxt(n + 2, {0});
for (int j = 1; j <= 26; j++) {
  nxt[n][j] = nxt[n + 1][j] = n + 1;
}
for (int i = n - 1; i >= 0; i--) {
  for (int j = 1; j <= 26; j++) {
    nxt[i][j] = nxt[i + 1][j];
  }
  nxt[i][s[i + 1] - 'a' + 1] = i + 1;
}
```

6 杂项

6.1 高精度

__int128使用（kuangbin模板）（范围：-2^127 ~ 2^127， 约10^38 ， longlong范围10^19）

```
#include <bits/stdc++.h>
using namespace std;
inline __int128 read(){
```

```
        __int128 x = 0, f = 1;
        char ch = getchar();
        while(ch < '0' || ch > '9'){
                if(ch == '-') f = -1;
                ch = getchar();
        }
        while(ch >= '0' && ch <= '9'){
                x = x * 10 + ch - '0';
                ch = getchar();
        }
        return x * f;
}
inline void print(__int128 x){
        if(x < 0){
                putchar('-'); x = -x;
        }
        if(x > 9) print(x / 10);
        putchar(x % 10 + '0');
}
int main(void){
    __int128 a = read();
    __int128 b = read();
    print(a + b);
    cout << endl;
    return 0;
}
```

大整数类（压9位）

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
struct Wint:vector<ll>
{
    const static ll BIT=1e9;
    Wint(ll n=0) {push_back(n);check();}
    Wint& operator=(const char* num)
    {
        int Len=strlen(num)-1; clear();
        for(int i=Len;i>=0;i-=9)
        {
            push_back(0); ll w=1;
            for(int j=i;j>i-9&&j>=0;--j)
                back()+=(num[j]^48)*w,w*=10;
        }
        return *this;
    }
    Wint& check()
    {
        while(!empty()&&!back()) pop_back();
        if(empty()) return *this;
        for(int i=1;i<size();++i)
            (*this)[i]+=(*this)[i-1]/BIT,
            (*this)[i-1]%=BIT;
        while(back()>=BIT)
        {
            push_back(back()/BIT);
            (*this)[size()-2]%=BIT;
        }
        return *this;
```

```
    }
};
bool operator<(Wint a,Wint b)
{
    if(a.size()!=b.size()) return a.size()<b.size();
    for(int i=a.size()-1;i>=0;--i)
        if(a[i]!=b[i]) return a[i]<b[i];
    return 0;
}
bool operator>(Wint a,Wint b) {return b<a;}
bool operator<=(Wint a,Wint b) {return !(a>b);}
bool operator>=(Wint a,Wint b) {return !(a<b);}
bool operator!=(Wint a,Wint b) {return a<b||b<a;}
bool operator==(Wint a,Wint b) {return !(a<b)&&!(b<a);}
Wint& operator+=(Wint &a,Wint b)
{
    if(a.size()<b.size()) a.resize(b.size());
    for(int i=0;i<b.size();++i) a[i]+=b[i];
    return a.check();
}
Wint operator+(Wint a,Wint b) {return a+=b;}
Wint& operator-=(Wint &a,Wint b)
{
    for(int i=0;i<b.size();a[i]-=b[i],++i)
        if(a[i]<b[i])
        {
            int j=i+1;
            while(!a[j]) ++j;
            while(j>i) --a[j],a[--j]+=Wint::BIT;
        }
    return a.check();
}
Wint operator-(Wint a,Wint b) {return a-=b;}
Wint operator*(Wint a,Wint b)
{
    if(a.empty()&&b.empty()) return a;
    Wint 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();
}
Wint& operator*=(Wint &a,Wint b) {return a=a*b;}
Wint operator/(Wint a,int b)
{
    Wint n; bool wp=0; ll t=0;
    for(int i=a.size()-1;i>=0;--i)
    {
        t=t*Wint::BIT+a[i];
        if(wp||t/b) wp=1,n.push_back(t/b);
        t%=b;
    }
    reverse(n.begin(),n.end());
    return n;
}
Wint& operator/=(Wint &a,int b) {return a=a/b;}
void readX(Wint &n) {char s[100010]; scanf("%s",s); n=s;}
void writeX(Wint n)
{
    if(n.empty()) {putchar('0'); return;}
    int Len=n.size()-1; printf("%lld",n[Len]);
    for(int i=Len-1;i>=0;--i) printf("%09lld",n[i]);
```

```
}
```

6.2 二维前缀和

```
//构造前缀和矩阵
a[i][j]+=a[i][j-1]+a[i-1][j]-a[i-1][j-1];

//求x1, y1, x2, y2为边界的子矩阵之和
a[x2][y2]-a[x1-1][y2]-a[x2][y1-1]+a[x1-1][y1-1];
```

6.3 双指针

6.4 ST表

```
vector<vector<int>> dp(32, vector<int> (n + 1));
vector<int> logn(n + 1);
logn[1] = 0, logn[2] = 1;
for (int i = 3; i <= n; i++) {
    logn[i] = logn[i / 2] + 1;
}
for (int i = 0; i <= 25; i++) {
    for (int j = 1; j + (1 << i) - 1 <= n; j++) {
        if (!i) dp[i][j] = a[j];
        else dp[i][j] = max(dp[i - 1][j], dp[i - 1][j + (1 << i - 1)]);
    }
}
auto qry = [&] (int l, int r) {
    int k = logn[r - l + 1];
    return max(dp[k][l], dp[k][r - (1 << k) + 1]);
};
```

二维ST表

```
int val[N][N];
int f[N][N][15][15];
int n, m, k;

void prework () {
    for (int i = 0; i < 15; i++) {
        for (int j = 0; j < 15; j++) {
            if (i == 0 and j == 0) continue;
            for (int k = 1; k <= n - (1 << i) + 1; k++) {
                for (int p = 1; p <= m - (1 << j) + 1; p++) {
                    if (i == 0)
                        f[k][p][i][j] = std::max (f[k][p][i][j - 1], f[k][p + (1 << j - 1)][i][j - 1]);
                    else
                        f[k][p][i][j] = std::max (f[k][p][i - 1][j], f[k + (1 << i - 1)][p][i - 1][j]);
                }
            }
        }
    }
}

int query (int r1, int c1, int r2, int c2) {
    int k1 = log2 (r2 - r1 + 1);
    int k2 = log2 (c2 - c1 + 1);
```

```
    return std::min (f[r1][c1][k1][k2], std::max (f[r2 - (1 << k1) + 1][c1][k1][k2],
std::min (f[r1][c2 - (1 << k2) + 1][k1][k2], f[r2 - (1 << k1) + 1][c2 - (1 << k2) + 1]
[k1][k2])));
}
```

6.5 快读

```
// 仅支持整型
template <typename _T>
inline void read(_T &f) {
    f = 0; _T fu = 1; char c = getchar();
    while (c < '0' || c > '9') { if (c == '-') { fu = -1; } c = getchar(); }
    while (c >= '0' && c <= '9') { f = (f << 3) + (f << 1) + (c & 15); c = getchar(); }
    f *= fu;
}

//重载流输入型
struct IOS{
    template<typename ATP>IOS& operator >> (ATP &x){
        x = 0; int f = 1; char c;
        for(c = getchar(); c < '0' || c > '9'; c = getchar()) if(c == '-')  f = -1;
        while(c >= '0' && c <= '9') x = x * 10 + (c ^ '0'), c = getchar();
        x*= f;
        return *this;
    }
}io;
io >> n;

```

6.6 模拟退火

```
/*
求费马点型

在二维平面上有 n 个点，第 i 个点的坐标为 (xi,yi)。

请你找出一个点，使得该点到这 n 个点的距离之和最小。

该点可以选择在平面中的任意位置，甚至与这 n 个点的位置重合。
*/

#define x first
#define y second
typedef pair<double, double> PII;
const int maxn = 110;
int n;
PII q[maxn];
double ans = 1e8;

double rand (double l, double r) {
        return (double)rand() / RAND_MAX * (r - l) + l;
}

double dis (PII a, PII b) {
        return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

```

```
double calc (PII p) {
        double res = 0;
        for (int i = 1; i <= n; i ++ ) {
                res += dis(p, q[i]);
        }
        ans = min(ans, res);
        return res;
}

void simulate_anneal () {
        PII cur(rand(0, 10000), rand(0, 10000));
        for (double t = 1e4; t > 1e-4; t *= 0.9) {
                PII np(rand(cur.x - t, cur.x + t), rand(cur.y - t, cur.y + t));
                double d = calc(np) - calc(cur);
                if (exp(-d / t) > rand(0, 1)) cur = np;
        }
}

int main () {
        srand(time(NULL));
        scanf("%d", &n);
        for (int i = 1; i <= n; i ++ ) scanf("%lf%lf", &q[i].first, &q[i].second);

        // 或 while ((double)clock()/CLOCKS_PER_SEC<0.8)
        for (int i = 1; i <= 100; i ++ )
                simulate_anneal();

        printf("%.0lf\n", ans);
}
```

6.7 STL

```
vector, 变长数组，倍增的思想
        size()  返回元素个数
        empty()  返回是否为空
        clear()  清空
        front()/back()
        push_back()/pop_back()
        begin()/end()
        []
        支持比较运算，按字典序

    pair<int, int>
        first, 第一个元素
        second, 第二个元素
        支持比较运算，以first为第一关键字，以second为第二关键字（字典序）

    string，字符串
        int has = s.find('xxxx');
        if (has != string::npos) 含xxx字符
        szie()/length()  返回字符串长度
        empty()
        clear()
        substr(起始下标，(子串长度))  返回子串
        c_str()  返回字符串所在字符数组的起始地址  代表可以用printf输出string类

    queue, 队列
        size()
        empty()
        push()  向队尾插入一个元素
```

```
        front()  返回队头元素
        back()  返回队尾元素
        pop()  弹出队头元素

    priority_queue，优先队列，默认是大根堆
        push()  插入一个元素
        top()  返回堆顶元素
        pop()  弹出堆顶元素
        定义成小根堆的方式：priority_queue<int, vector<int>, greater<int>> q;

    stack，栈
        size()
        empty()
        push()  向栈顶插入一个元素
        top()  返回栈顶元素
        pop()  弹出栈顶元素

    deque，双端队列
        size()
        empty()
        clear()
        front()/back()
        push_back()/pop_back()
        push_front()/pop_front()
        begin()/end()
        []

    set，map，multiset，multimap，基于平衡二叉树（红黑树），动态维护有序序列
        size()
        empty()
        clear()
        begin()/end()
        ++，-- 返回前驱和后继，时间复杂度 O(logn)

        set/multiset
            insert()  插入一个数
            find()  查找一个数
            count()  返回某一个数的个数
            erase()
                (1) 输入是一个数x，删除所有x   O(k + logn)
                (2) 输入一个迭代器，删除这个迭代器
            lower_bound()/upper_bound()
                lower_bound(x)  返回大于等于x的最小的数的迭代器
                upper_bound(x)  返回大于x的最小的数的迭代器
        map/multimap
            insert()  插入的数是一个pair
            erase()  输入的参数是pair或者迭代器
            find()
            []   时间复杂度是 O(logn)
            lower_bound()/upper_bound()

    unordered_set，unordered_map，unordered_multiset，unordered_multimap，哈希表
        和上面类似，增删改查的时间复杂度是 O(1)
        不支持 lower_bound()/upper_bound()， 迭代器的++，--



bit.size()        返回大小（位数）
bit.count()      返回1的个数
bit.any()        返回是否有1
bit.none()       返回是否没有1
bit.set()        全都变成1
```

```
bit.set(p)      将第p + 1位变成1（bitset是从第0位开始的！）
bit.set(p, x)   将第p + 1位变成x
bit.reset()     全都变成0
bit.reset(p)    将第p + 1位变成0
bit.flip()      全都取反
bit.flip(p)     将第p + 1位取反
bit.to_ulong()  返回它转换为unsigned long的结果，如果超出范围则报错
bit.to_ullong() 返回它转换为unsigned long long的结果，如果超出范围则报错
bit.to_string() 返回它转换为string的结果
```

6.8 离散化

```cpp
vector<int> v;
    for (int i = 1; i <= n; i ++ ) {
        read(q[i]); v.pb(q[i]);
    }
    sort(all(v));
    v.erase(unique(all(v)), v.end());
    for (int i = 1; i <= n; i ++ ) {
        q[i] = lower_bound(all(v), q[i]) - v.begin() + 1;
    }
```

6.9 随机数

```cpp
mt19937 sed(time(nullptr));
uniform_int_distribution<int> range(l, r); // l到r内随机数
cout << range(sed) << endl;
```

6.10 ModInt

Z

不要引入std

```cpp
constexpr int P = 998244353;
using i64 = long long;
// assume -P <= x < 2P
int norm(int x) {
    if (x < 0) {
        x += P;
    }
    if (x >= P) {
        x -= P;
    }
    return x;
}
template<class T>
T power(T a, i64 b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) {
            res *= a;
        }
    }
    return res;
}
```

```cpp
struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(i64 x) : x(norm(x % P)) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(P - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, P - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = i64(x) * rhs.x % P;
        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) {
        Z res = lhs;
        res -= rhs;
        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res /= rhs;
        return res;
    }
    friend std::istream &operator>>(std::istream &is, Z &a) {
        i64 v;
        is >> v;
        a = Z(v);
        return is;
    }
    friend std::ostream &operator<<(std::ostream &os, const Z &a) {
        return os << a.val();
    }
};


```

mint

```cpp
constexpr int P = 998244353;
template<const int T>
struct ModInt {
    const static int mod = T;
    int x;
    ModInt (int x = 0) : x (x% mod) {}
    int val () { return x; }
    constexpr int moded (int x) const {
        if (x < 0) {
            x += P;
        }
        if (x >= P) {
            x -= P;
        }
        return x;
    }
    ModInt operator + (const ModInt& a) const { int x0 = x; x0 += a.x; return ModInt (x0); }
    ModInt operator - (const ModInt& a) const { int x0 = x - a.x; return ModInt (x0 < mod ? x0 + mod : x0); }
    ModInt operator * (const ModInt& a) const { return ModInt (1LL * x * a.x % mod); }
    ModInt operator / (const ModInt& a) const { return *this * a.inv (); }
    void operator += (const ModInt& a) { x = moded (x + a.x); if (x >= mod) x -= mod; }
    void operator -= (const ModInt& a) { x -= a.x; if (x < 0) x += mod; }
    void operator *= (const ModInt& a) { x = 1LL * x * a.x % mod; }
    void operator /= (const ModInt& a) { *this = *this / a; }
    friend ostream& operator<<(ostream& os, const ModInt& a) { return os << a.x; }
    ModInt pow (int64_t n) const {
        ModInt res (1), mul (x);
        while (n) {
            if (n & 1) res *= mul;
            mul *= mul;
            n >>= 1;
        }
        return res;
    }
    ModInt inv () const {
        int a = x, b = mod, u = 1, v = 0;
        while (b) {
            int t = a / b;
            a -= t * b; swap (a, b);
            u -= t * v; swap (u, v);
        }
        if (u < 0) u += mod;
        return u;
    }
};
typedef ModInt<P> mint; // mod1 || mod2
```

6.11 重载哈希

```cpp
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
```

```
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM =
chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};
unordered_map<LL, LL, custom_hash> mp;
```

6.12 取模

```
int mol(int x) {
    if (x < 0) return x + P;
    if (x >= P) return x - P;
    return x;
}
```

```
#define mol(x) ((x) < 0 ? (x) + P : (x) >= P ? (x) - P : (x))
```

6.13 枚举子集

```
for (int i = 0; i < (1 << n); i++) {
    for (int j = i; j; j = (j - 1) & i) {
        cout << i << ' ' << j << '\n';
    }
}
```