modal

目录

bitset

```cpp
#include <bits/stdc++.h>
using i64 = long long;

const int N = 1030;

//std::bitset<10000> s;
//std::bitset<1000005> f[110];
std::bitset<N> f[2][N];
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    std::bitset<4> s(std::string("1001"));//填充字符串
    //std::cout << s << '\n';
    int t = 5;
    std::bitset<10> k(t); //不足补零
    //std::cout << k << "\n";

    //std::bitset<1000>f[100]; //支持多维，代表有100个长度为1000的01串（默认为0）
    //s.count() 返回有多少个1
    //s.any() 至少有一个1返回true，反之false
    //s.none() 全为0返回true，反之false
    //s.set() 将每位全部赋值为1
    //s.set(u，v) 将第u位赋值为v， v只能取值0或者1
    //s.reset() 将每位全部赋值为0
    //s.reset(k) 将第k位赋值为0

    /*
    //example：有n个数，x可以取值li-ri，问sumXi可能的值有多少。（背包问题）

    // f[i] |= f[i - 1] << (x * x)
    int n;
    int l[110], r[110];
    std::cin >> n;
    for (int i = 1; i <= n; i ++) {
        std::cin >> l[i] >> r[i];
    }
    f[0].set(0);
    for (int i = 1; i <= n; i ++) {
        for (int j = l[i]; j <= r[i]; j ++) {
            f[i] |= (f[i - 1] << (j * j));
        }
    }
    std::cout << f[n].count() << "\n";
    */
    //example2: 优化DP时空复杂度;
        n个数, 背包容量m, 问装满背包时候, 背包里面异或值最大可能是多少?
    //朴素方程 f[i][j][k] 前i个数, 异或值为j, 体积为k的方案是否存在 ,  滚动只能优化掉 一维 i,

    int T;
    std::cin >> T;
    while (T --) {
      int n, m;
      std::cin >> n >> m;
      for (int i = 0; i < 1024; i ++) f[0][i] = f[1][i] = 0;
      f[0][0][0] = 1;
      for (int i = 1, x = 1; i <= n; i ++, x ^= 1) {
        int v, w;
        std::cin >> v >> w;
        for (int j = 0; j < 1024; j ++) {
          f[x][j] = f[x ^ 1][j ^ w] << v | f[x ^ 1][j];
        }
      }
      int ok = -1;
      for (int i = 0; i < 1024; i ++)
      if (f[n & 1][i][m]) ok = i;
      std::cout << ok << "\n";
    }
    return 0;
  }
```

最大流

```
#include<bits/stdc++.h>
using i64 = long long;

//Dicnic
//------------------------------------------
const int V = 1010;
const int E = 10100;
template<typename T>
struct FlowGraph {
  int s, t, vtot, etot, head[V], dis[V], cur[V];
  struct edge {
    int v, nxt;
    T f;
  } e[E * 2];
  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];
    }
    std::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, std::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 dicnic() {
    T flow = 0;
    while (bfs()) {
      flow += dfs(s, std::numeric_limits<T>::max());
    }
    return flow;
    }
    void init(int _s, int _t, int _vtot) {
        s = _s;
        t = _t;
        vtot = _vtot;
        for (int i = 1; i <= vtot; i ++) {
            head[i] = -1;
        }
    }
};

//------------------------------------------
void solve() {
    int n, m, s, t;
    std::cin >> n >> m >> s >> t;
    FlowGraph<i64> g;
    g.init(s, t, n);
    for (int i = 1; i <= m; i ++) {
        int u, v, w;
        std::cin >> u >> v >> w;
        g.addedge(u, v, w);
    }
    std::cout << g.dicnic() << '\n';
}
int main(){
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int t = 1;
    //std::cin >> t;
    while (t --) {
        solve();
    }
    return 0;
}
```

最流费用流

```
#include<bits/stdc++.h>
using i64 = long long;
//最小费用最大流 - 最大费用最大流
// MCMF :  minimum cost maximum flow

const int V = 2010;
const int E = 20100;
template<typename T>
```

```cpp
struct MinCostGraph {
  int s, t, vtot, etot, cur[V], head[V], pre[V];
  T dis[V], flow, cost;
  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 = std::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;
    std::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 = std::numeric_limits<T>::max();
    while (~ pre[u]) {
      f = std::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;
    }
  }

  std::array<T, 2> 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;
    }
  }
};

void solve() {
  int n, m, s, t;
  std::cin >> n >> m >> s >> t;
  MinCostGraph<int> g;
  g.init(s, t, n);
  for (int i = 1; i <= m; i ++) {
    int u, v, f, c;
    std::cin >> u >> v >> f >> c;
    g.addedge(u, v, f, c);
  }
  auto it = g.solve();
  std::cout << it[0] << ' ' << it[1] << '\n';
}
int main() {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);
  int t = 1;
  //std::cin >> t;
```

```
    while (t --) {
        solve();
    }
    return 0;
}
```

Fenwick

```
struct Fenwick {
    int n;
    std::vector<int> a;

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

线段树 tag

```
struct SegmentTree {
    int n;
    struct node {
        int l, r, x;
        int sum, add, max;
    };
    std::vector<node> t;
    std::vector<int> a;
    SegmentTree() {}
    void init(int n) {
        this -> n = n;
        t.assign(4 * n + 1, (node){0, 0, 0, 0, 0, 0});
        a.assign(n + 1, 0);
        build(1, 1, n);
    }
    void init(int n, std::vector<int> v) {
        this -> n = n;
        t.assign(4 * n + 1, (node){0, 0, 0, 0, 0, 0});
        a.assign(v.begin(), v.end());
        build(1, 1, n);
    }
    void pushup(int u) {
        t[u].sum = t[u << 1].sum + t[u << 1 | 1].sum;
        t[u].max = std::max(t[u << 1].max, t[u << 1 | 1].max);
    }
    void pushdown(int u) {
        if (! t[u].add) return ;
        t[u << 1].sum += (t[u << 1].r - t[u << 1].l + 1) * t[u].add;
        t[u << 1 | 1].sum += (t[u << 1 | 1].r - t[u << 1 | 1].l + 1) * t[u].add;
        t[u << 1].add += t[u].add, t[u << 1 | 1].add += t[u].add;
        t[u << 1].max += t[u].add, t[u << 1 | 1].max += t[u].add;
        t[u].add = 0;
    }
    void build(int u, int l, int r) {
        t[u] = {l, r, 0, 0, 0, 0};
        if (l == r) {
            t[u] = {l, r, a[l], a[l], 0, a[l]};
```

```
      return ;
    }
    int mid = l + r >> 1;
    build(u << 1, l, mid);
    build(u << 1 | 1, mid + 1, r);
    pushup(u);
  }
  void add(int u, int l, int r, int val) {
    if (t[u].l >= l && t[u].r <= r) {
      t[u].sum += (t[u].r - t[u].l + 1) * val;
      t[u].max += val;
      t[u].add += val;
      return ;
    }
    pushdown(u);
    int mid = t[u].l + t[u].r >> 1;
    if (l <= mid) add(u << 1, l, r, val);
    if (mid < r) add(u << 1 | 1, l, r, val);
    pushup(u);
  }
  int query(int u, int l, int r) {
    if (t[u].l >= l && t[u].r <= r) {
      return t[u].sum;
    }
    pushdown(u);
    int res = 0;
    int mid = t[u].l + t[u].r >> 1;
    if (l <= mid) res += query(u << 1, l, r);
    if (mid < r) res += query(u << 1 | 1, l, r);
    return res;
  }
  int query_max(int u, int l, int r) {
    if (t[u].l >= l && t[u].r <= r) {
      return t[u].max;
    }
    pushdown(u);
    int res = -1e9;
    int mid = t[u].l + t[u].r >> 1;
    if (l <= mid) res = std::max(res, query_max(u << 1, l, r));
    if (mid < r) res = std::max(res, query_max(u << 1 | 1, l, r));
    return res;
  }
  void add(int l, int r, int val) {
    add(1, l, r, val);
  }
  int query(int l, int r) {
    return query(1, l, r);
    }
    int Max(int l, int r) {
        return query_max(1, l, r);
    }
};
```

分块

```
void resort(int x,int n){
    v[x].clear();
    for(int i=(x-1)*b+1;i<=min(x*b,n);i++)v[x].push_back(a[i]);
    sort(v[x].begin(),v[x].end());
}
void change2(int l,int r,int c,int n){
    int p=blo[l],q=blo[r];
    for(int i=l;i<=min(r,p*b);i++)a[i]+=c;
    resort(p,n);
    if(p==q)return ;
    for(int i=(q-1)*b+1;i<=r;i++)a[i]+=c;
    resort(q,n);
    for(int i=p+1;i<q;i++)add[i]+=c;
}
void ask2(int l,int r,int c){
    int res=-1,p=blo[l],q=blo[r];
    for(int i=l;i<=min(r,p*b);i++)
    if(a[i]+add[p]<c)res=max(res,a[i]+add[p]);
    if(p==q){cout<<res<<endl;return ;}
    for(int i=(q-1)*b+1;i<=r;i++)
    if(a[i]+add[q]<c)res=max(res,a[i]+add[q]);
    for(int i=p+1;i<q;i++){
        int pos=lower_bound(v[i].begin(),v[i].end(),c-add[i])-v[i].begin();
        if(pos&&v[i][pos-1]+add[i]<c)res=max(res,v[i][pos-1]+add[i]);
    }
    cout<<res<<endl;
}
void Sblk2(){
    int n,op,l,r,c; cin>>n; b=sqrt(n);
    for(int i=1;i<=n;i++){
        cin>>a[i];blo[i]=(i-1)/b+1;
        v[blo[i]].push_back(a[i]);
    }
    for(int i=1;i<=blo[n];i++)sort(v[i].begin(),v[i].end());
    for(int i=1;i<=n;i++){
        cin>>op>>l>>r>>c;
        if(!op)change2(l,r,c,n);
    else ask2(l,r,c);
  }
}
```

area 扫描线

```
#include <bits/stdc++.h>
using i64 = long long;
#define int i64
struct SegmentTree {
  int n;
  struct node {
    int l, r, len;
    int sum, add, max; // x = len, sum = times, add = delt
  };
  // 考虑改变线段树存值的方式，因为 a[3] - a[1] -->
  // a[1 -- 2] + a[3 -- 3] 而 3 -- 3 nothing.
  //
  std::vector<node> t;
  std::vector<int> a;
  SegmentTree() {}
  void init(int n) {
    this -> n = n;
    t.assign(4 * n + 1, (node){0, 0, 0, 0, 0, 0});
    a.assign(n + 1, 0);
    build(1, 1, n);
  }
  void init(int n, std::vector<int> v) {
    this -> n = n;
    t.assign(4 * n + 1, (node){0, 0, 0, 0, 0, 0});
    a.assign(v.begin(), v.end());
    build(1, 1, n);
  }
  void pushup(int u) { // ub大集合!
    if (t[u].sum) {
      t[u].len = a[t[u].r] - a[t[u].l - 1];
    } else {
      if (t[u].l != t[u].r)
      t[u].len = t[u << 1].len + t[u << 1 | 1].len;
      else t[u].len = 0;
    }
  }
  void build(int u, int l, int r) {
    t[u] = {l, r, 0, 0, 0, 0};
    if (l == r) {
            return ;
        }
        int mid = l + r >> 1;
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
    }
    void add(int u, int l, int r, int val) {
        if (t[u].l >= l && t[u].r <= r) {
            t[u].sum += val;
            pushup(u);
            return ;
        }
        int mid = t[u].l + t[u].r >> 1;
        if (l <= mid) add(u << 1, l, r, val);
        if (mid < r) add(u << 1 | 1, l, r, val);
        pushup(u);
    }
    void add(int l, int r, int val) {
        add(1, l, r, val);
    }
};

signed main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    // 扫描线.
    int n;
    std::cin >> n;
    std::vector<int> X;
    std::vector<std::array<int, 4> > e(n * 2);
    for (int i = 0; i < n; i ++) {
        int x1, y1, x2, y2;
        std::cin >> x1 >> y1 >> x2 >> y2;
        X.push_back(x1);
        X.push_back(x2);
        e[i * 2] = {x1, x2, y1, 1};
        e[i * 2 + 1] = {x1, x2, y2, -1};
    }
    std::sort(X.begin(), X.end());
    X.erase(std::unique(X.begin(), X.end()), X.end());
    int m = X.size();
    std::vector<int> a(m);
    for (int i = 1; i < m; i ++) {
        a[i] = X[i] - X[i - 1];
    }
    m --;
    for (int i = 1; i <= m; i ++) {
        a[i] += a[i - 1];
    }
    std::sort(e.begin(), e.end(), [&](auto a, auto b) {
        return a[2] < b[2];
    });

    SegmentTree t;
    t.init(m, a);
    int ans = 0;
    for (int i = 0; i < 2 * n; i ++) {
        int l = std::lower_bound(X.begin(), X.end(), e[i][0]) - X.begin() + 1;
        int r = std::lower_bound(X.begin(), X.end(), e[i][1]) - X.begin() + 1;
        t.add(l, r - 1, e[i][3]);
        if (i + 1 < 2 * n) ans += t.t[1].len * (e[i + 1][2] - e[i][2]);
    }
    std::cout << ans << '\n';
    return 0;
}
```

DSU

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

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

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

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

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

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

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

HLD

```
#include<bits/stdc++.h>
using i64 = long long;
const int N = 1e5 + 10, M = 2e5 + 10;

int a[N];
int tot, head[M], nxt[M], to[M];
void addedge(int u, int v) {
  nxt[++ tot] = head[u];
  head[u] = tot;
  to[tot] = v;
}

int lowbit(int x) {return x & (- x);}
struct Fenwick {
  int n;
  i64 t1[N], t2[N];
  void add(int x, i64 val) {
    for (int i = x; i <= n; i += lowbit(i)) {
      t1[i] = t1[i] + val, t2[i] = t2[i] + val * x;
    }
  }
  i64 sum(int x) {
    i64 ans = 0;
    for (int i = x; i; i -= lowbit(i)) {
      ans = ans + (x + 1) * t1[i] - t2[i];
    }
```

```
      return ans;
    }
    void update(int l, int r, i64 val) {
      add(l, val), add(r + 1, -val);
    }
    i64 query(int l, int r) {
      return sum(r) - sum(l - 1);
    }
  } arr;

  struct TreeList {
    int cnt = 0;
    int dep[N], top[N], son[N], siz[N], fa[N], dfn[N];

    void dfs1(int cur) {
      dep[cur] = dep[fa[cur]] + (siz[cur] = 1);
      for (int i = head[cur]; i; i = nxt[i]) {
        if (to[i] == fa[cur]) continue;
        fa[to[i]] = cur;
        dfs1(to[i]);
        siz[cur] += siz[to[i]];
        if (! son[cur] || siz[son[cur]] < siz[to[i]]) {
          son[cur] = to[i];
        }
      }
    }
    void dfs2(int cur, int t) {
      dfn[cur] = ++ cnt;
      top[cur] = t;
      if (son[cur]) {
        dfs2(son[cur], t);
      }
      for (int i = head[cur]; i; i = nxt[i]) {
        if (to[i] != fa[cur] && to[i] != son[cur]) {
          dfs2(to[i], to[i]);
        }
      }
    }

    void update(int x, int y, i64 v) {
      while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) std::swap(x, y);
        arr.update(dfn[top[x]], dfn[x], v);
        x = fa[top[x]];
      }
      if (dfn[x] > dfn[y]) std::swap(x, y);
      arr.update(dfn[x], dfn[y], v);
    }

    i64 query(int x, int y) {
      i64 ans = 0;
      while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) std::swap(x, y);
        ans += arr.query(dfn[top[x]], dfn[x]);
        x = fa[top[x]];
      }
      if (dfn[x] > dfn[y]) std::swap(x, y);
      return ans + arr.query(dfn[x], dfn[y]);
    }
    void update_subtree(int x, i64 v) {
      arr.update(dfn[x], dfn[x] + siz[x] - 1, v);
    }
    i64 query_subtree(int x) {
      return arr.query(dfn[x], dfn[x] + siz[x] - 1);
    }
  } tr;

  void solve() {
    int n, m, r, p;
    std::cin >> n >> m >> r >> p;
    for (int i = 1; i <= n; i ++) {
      std::cin >> a[i];
    }
    for (int i = 1; i < n; i ++) {
      int u, v;
      std::cin >> u >> v;
      addedge(u, v);
      addedge(v, u);
    }
    //
    tr.dfs1(r);
    tr.dfs2(r, r);
    arr.n = n;
    for (int i = 1; i <= n; i ++) {
      tr.update(i, i, a[i]);
    }
    while (m --) {
      int s, x, y, v;
      std::cin >> s >> x;
      if (s == 1) {
        std::cin >> y >> v;
        tr.update(x, y, v);
      } else if (s == 2) {
        std::cin >> y;
```

```
            std::cout << (tr.query(x, y) % p) << '\n';
        } else if (s == 3) {
            std::cin >> v;
            tr.update_subtree(x, v);
        } else if (s == 4) {
            std::cout << (tr.query_subtree(x) % p) << '\n';
        }
    }
}
int main(){
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int t = 1;
    //std::cin >> t;
    while (t --) {
        solve();
    }
    return 0;
}
```

LCA

```
struct LCA { // 倍增
    int n, rt = 1, deep[N], ff[16][N];
    std::vector<int> e[N];
    LCA(int n) : n(n) {}
    void dfs(int u, int p) {
        ff[0][u] = p;
        for (auto v : e[u]) {
            if (v == p) continue;
            deep[v] = deep[u] + 1;
            dfs(v, u);
        }
    }
    void work() {
        dfs(rt, 0);
        for (int i = 1; i <= 15; i ++) {
            for (int j = 1; j <= n; j ++) {
                ff[i][j] = ff[i - 1][ff[i - 1][j]];
            }
        }
    }
    int lca(int u, int v) {
        if (deep[u] < deep[v]) std::swap(u, v);
        for (int i = 15; i >= 0; i --) {
```

```
      if (deep[u] - deep[v] >= (1 << i)) {
        u = ff[i][u];
      }
    }
    if (u == v) {
      return u;
    }
    for (int i = 15; i >= 0; i --) {
      if (ff[i][u] != ff[i][v]) {
        u = ff[i][u], v = ff[i][v];
      }
    }
    u = ff[0][u];
    return u;
  }
};
```

虚树

```
struct Xushu {
  int n, rt = 1, dn, res = 0, num = 0;
  int ff[20][N];
  struct node {
    int to, nxt;
  };
  std::vector<std::vector<int> > H;
  std::vector<int> dp, col, dfn, h;
  std::vector<node> e;

  Xushu(int n, std::vector<int> a) {
    H.assign(n + 1, {});
    this -> n = n;
    h.assign(n + 1, 0);
    col.assign(a.begin(), a.end());
    dp.assign(n + 1, 0);
    dfn.assign(n + 1, 0);
    e.assign(2 * (n + 1), (node){0, 0});
  }
  void addEdge(int a, int b) {
    e[++ num] = {b, h[a]}; h[a] = num;
    e[++ num] = {a, h[b]}; h[b] = num;
  }

  void dfs(int u, int p) {
    ff[0][dfn[u] = ++ dn] = p;
    for (int i = h[u]; i; i = e[i].nxt) {
      if (e[i].to == p) continue;
      dfs(e[i].to, u);
    }
  }
  int get(int x, int y) {
    if (dfn[x] < dfn[y]) return x;
    else return y;
  }
  void work() {
    dfs(rt, 0);
    for (int i = 1; i <= std::__lg(n); i ++) { //RMQ deep less
      for (int j = 1; j + (1 << i) - 1 <= n; j ++) {
        ff[i][j] = get(ff[i - 1][j], ff[i - 1][j + (1 << i - 1)]);
      }
    }
  }
  int lca(int u, int v) {
    if (u == v) {
      return u;
    }
    if ((u = dfn[u]) > (v = dfn[v])) std::swap(u, v);
    int d = std::__lg(v - u);
    return get(ff[d][++ u], ff[d][v - (1 << d) + 1]);
  }

  void build(std::vector<int> a) {
    //H.assign(n + 1, {});
    int nn = a.size();
    std::sort(a.begin(), a.end(), [&](int x, int y) {
      return dfn[x] < dfn[y];
    });

    for (int i = 0; i + 1 < nn; i ++) {
      a.push_back(lca(a[i], a[i + 1]));
    }
    a.push_back(1);

    std::sort(a.begin(), a.end(), [&](int x, int y) {
      return dfn[x] < dfn[y];
    });
    a.erase(std::unique(a.begin(), a.end()), a.end());
    int o = a.size();
    for (int i = 0; i + 1 < o; i ++) {
      H[lca(a[i], a[i + 1])].push_back(a[i + 1]);
    }
  }
```

```
    void DFS(int u, int color) {
        int r = 1, sum = 1;
        for (auto v : H[u]) {
            DFS(v, color);
            r = r * (dp[v] + 1) % mod;
            sum = (sum + dp[v]) % mod;
        }
        res = (res + r) % mod;
        if (col[u] != color) {
            r --;
            res = (res - sum + mod) % mod;
        }
        dp[u] = r;
        //std::cout << dp[u] << '\n';
    }
    void solve(int cl) {
        DFS(1, cl);
        auto CLEAR = [&](auto self, int u) -> void {
            for (auto v : H[u]) self(self, v);
            H[u].clear();
        };
        CLEAR(CLEAR, 1);

    }
};
```

强连通分量 SCC

```
struct SCC {
    int n;
    std::vector<std::vector<int>> adj;
    std::vector<int> stk;
    std::vector<int> dfn, low, bel;
    int cur, cnt;

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

    void init(int n) {
        this->n = n;
        adj.assign(n + 1, {});
        dfn.assign(n + 1, -1);
        low.resize(n + 1);
        bel.assign(n + 1, -1);
        stk.clear();
        cur = cnt = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
    }

    void dfs(int x) {
        dfn[x] = low[x] = ++ cur;
        stk.push_back(x);

        for (auto y : adj[x]) {
            if (dfn[y] == -1) {
                dfs(y);
                low[x] = std::min(low[x], low[y]);
            } else if (bel[y] == -1) {
                low[x] = std::min(low[x], dfn[y]);
            }
        }

        if (dfn[x] == low[x]) {
            int y;
            do {
                y = stk.back();
                bel[y] = cnt;
                stk.pop_back();
            } while (y != x);
            cnt ++;
        }
    }

    std::vector<int> work() {
        for (int i = 1; i <= n; i ++) {
            if (dfn[i] == -1) {
                dfs(i);
            }
        }
        return bel;
    }
};
```

SCC 缩点

```cpp
struct SCC {
  int n;
  std::vector<std::vector<int>> adj;
  std::vector<int> stk;
  std::vector<int> dfn, low, bel;
  int cur, cnt;

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

  void init(int n) {
    this->n = n;
    adj.assign(n + 1, {});
    dfn.assign(n + 1, -1);
    low.resize(n + 1);
    bel.assign(n + 1, -1);
    stk.clear();
    cur = cnt = 0;
  }

  void addEdge(int u, int v) {
    adj[u].push_back(v);
  }

  void dfs(int x) {
    dfn[x] = low[x] = ++ cur;
    stk.push_back(x);

    for (auto y : adj[x]) {
      if (dfn[y] == -1) {
        dfs(y);
        low[x] = std::min(low[x], low[y]);
      } else if (bel[y] == -1) {
        low[x] = std::min(low[x], dfn[y]);
      }
    }

    if (dfn[x] == low[x]) {
      int y;
      do {
        y = stk.back();
        bel[y] = cnt;
```

```
                stk.pop_back();
            } while (y != x);
            cnt ++;
        }
    }

    std::vector<int> work() {
        for (int i = 1; i <= n; i ++) {
            if (dfn[i] == -1) {
                dfs(i);
            }
        }
        return bel;
    }
};
```

无向图割点

```
struct GeDian {
    int n, root;
    std::vector<std::vector<int>> adj;
    std::vector<int> dfn, low;
    std::vector<bool> bel;
    int cur;

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

    void init(int n) {
        this->n = n;
        adj.assign(n + 1, {});
        dfn.assign(n + 1, -1);
        low.resize(n + 1);
        bel.assign(n + 1, 0);
        cur = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void dfs(int x) {
        dfn[x] = low[x] = ++ cur;
        int ch = 0;

        for (auto y : adj[x]) {
            if (dfn[y] == -1) {
                ch ++;
                dfs(y);
                low[x] = std::min(low[x], low[y]);
                if (x != root && low[y] >= dfn[x] ) {
                    bel[x] = 1;
                }
            } else {
                low[x] = std::min(low[x], dfn[y]);
            }
        }

        if (x == root && ch >= 2) {
            bel[x] = 1;
        }
    }

    std::vector<bool> work() {
        for (int i = 1; i <= n; i ++) {
            if (dfn[i] == -1) {
                root = i;
                dfs(i);
            }
        }
        return bel;
    }
};
```

二分图

```
struct BipartGraph {
    int n, cnt, ans, nx, ny;
    std::vector<std::vector<int>> adj;
    std::vector<int> col;
    //-----------------------------------
    std::vector<int> link;
    std::vector<bool> vis;

    BipartGraph() {}

    void init1(int n) {
      this -> n = n;
      adj.assign(n + 1, {});
      col.assign(n + 1, 0);
      cnt = 0;
   }

   void addEdge1(int u, int v) {
      adj[u].push_back(v);
      adj[v].push_back(u);
   }

   bool bipart(int u) {
      for (auto v : adj[u]) {
         if (col[v] == col[u]) {
            return false;
         }
         if (col[v] == 0) {
            col[v] = 3 - col[u];
            if (! bipart(v)) return false;
         }
      }
      return true;
   }

   bool check_bipart() {
      bool f = 1;
      for (int i = 1; i <= n; i ++) {
         if (col[i] == 0) {
            cnt ++;
            col[i] = 1;
            if (! bipart(i)) {
               f = 0;
            }
         }
      }
      return f;
   }

   //----------------------------------------------------

   void init2(int nx, int ny) {
      this -> nx = nx;
      this -> ny = ny;
      adj.assign(nx + 1, {});
      link.assign(ny + 1, -1);
      vis.assign(nx + 1, false);
      ans = 0;
```

```
  }
  void addEdge2(int u, int v) {
    adj[u].push_back(v);
    adj[v].push_back(u);
  }

  bool dfs(int u) {
    for (auto v : adj[u]) {
      if (vis[v]) continue;
      vis[v] = true;
      if (link[v] == 0 || dfs(link[v])) {
        link[v] = u;
        return true;
      }
    }
    return false;
  }
  int Maxans() {
    for (int i = 1; i <= nx; i ++) {
      std::fill(vis.begin(), vis.end(), false);
      if (dfs(i)) ans ++;
    }
    return ans;
  }
};
```

TwoSat

```
struct TwoSat {
  int n;
  std::vector<std::vector<int>> e;
  std::vector<bool> ans;
  TwoSat(int n) : n(n), e(2 * n), ans(n) {}
  void addClause(int u, bool f, int v, bool g) {
    e[2 * u + !f].push_back(2 * v + g);
    e[2 * v + !g].push_back(2 * u + f);
  }
  bool satisfiable() {
    std::vector<int> id(2 * n, -1), dfn(2 * n, -1), low(2 * n, -1);
    std::vector<int> stk;
    int now = 0, cnt = 0;
    std::function<void(int)> tarjan = [&](int u) {
      stk.push_back(u);
      dfn[u] = low[u] = now++;
      for (auto v : e[u]) {
        if (dfn[v] == -1) {
          tarjan(v);
          low[u] = std::min(low[u], low[v]);
        } else if (id[v] == -1) {
          low[u] = std::min(low[u], dfn[v]);
        }
      }
      if (dfn[u] == low[u]) {
        int v;
        do {
          v = stk.back();
          stk.pop_back();
          id[v] = cnt;
        } while (v != u);
        ++cnt;
      }
    };
    for (int i = 0; i < 2 * n; ++i) if (dfn[i] == -1) tarjan(i);
    for (int i = 0; i < n; ++i) {
      if (id[2 * i] == id[2 * i + 1]) return false;
      ans[i] = id[2 * i] > id[2 * i + 1];
    }
    return true;
  }
  std::vector<bool> answer() { return ans; }
}; // start from 0
```

线筛

```
std::vector<int> minp, primes;
void sieve(int n) {
  minp.assign(n + 1, 0);
  primes.clear();

  for (int i = 2; i <= n; i++) {
    if (minp[i] == 0) {
      minp[i] = i;
      primes.push_back(i);
    }

    for (auto p : primes) {
      if (i * p > n) {
        break;
      }
      minp[i * p] = p;
      if (p == minp[i]) {
        break;
      }
    }
  }
}
```

扩展欧拉定理

```
int main() {
  scanf("%d%d", &a, &m);
  bool flag = false;
  int phi_m = getphi(m);
  cin >> s;
  for(int i = 0; i < s.length(); ++ i) {
    b = (b * 10 + s[i] - '0');
    if(b >= phi_m)
    flag = 1, b %= phi_m;
  }
  if(flag)
  b += phi_m;
  int ans = qpow(a, b, m);
  printf("%d\n", ans);
  return 0;
}
```

欧拉函数

```
int phi(int n) {
  int res = n;
  for (int i = 2; i * i <= n; i++) {
    if (n % i == 0) {
      while (n % i == 0) {
        n /= i;
      }
      res = res / i * (i - 1);
    }
  }
  if (n > 1) {
    res = res / n * (n - 1);
  }
  return res;
}
// 线性

void get_phi(int n){
    phi[1]=1;
    vis[1]=true;
    for(int i=2;i<=n;i++){
        if(!vis[i])p[++cnt]=i,phi[i]=i-1;
        for(int j=1;i*p[j]<=n;j++){
            vis[i*p[j]]=true;
            if(i%p[j]==0){phi[p[j]*i]=phi[i]*p[j];break;}
            phi[p[j]*i]=phi[i]*(p[j]-1);
        }
    }
}
```

Exgcd

```
int exgcd(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    int d = exgcd(b, a % b, x, y);
    int z = x;
    x = y;
    y = z - (a / b) * y;
    return d;
}
void solve() {
    int a, b, c;
    std::cin >> a >> b >> c;
    int x, y;
    int d = exgcd(a, b, x, y);
    if (c % d) {  // not exist.
        std::cout << "-1\n";
        return ;
    }
    int t = c / d, aa = a / d, bb = b / d;
    x *= t, y *= t;
    // xx = x + k * bb, yy = y - k * aa
    int minx = (x % bb + bb - 1) % bb + 1;
    int miny = (y % aa + aa - 1) % aa + 1;
    int maxx = (c - b * miny) / a;
    int maxy = (c - a * minx) / b;
    int cnt = (maxx - minx) / bb + 1; // (maxy - miny) / aa + 1
    if (maxx <= 0 && maxy <= 0); // no positive ans
}
```

静态凸包

```
struct Point {
    i64 x;
    i64 y;
    Point(i64 x = 0, i64 y = 0) : x(x), y(y) {}
};

bool operator==(const Point &a, const Point &b) {
    return a.x == b.x && a.y == b.y;
}

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

Point operator-(const Point &a, const Point &b) {
    return Point(a.x - b.x, a.y - b.y);
}

i64 dot(const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y;
}

i64 cross(const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
}

void norm(std::vector<Point> &h) {
    int i = 0;
    for (int j = 0; j < int(h.size()); j++) {
        if (h[j].y < h[i].y || (h[j].y == h[i].y && h[j].x < h[i].x)) {
            i = j;
        }
    }
    std::rotate(h.begin(), h.begin() + i, h.end());
}

int sgn(const Point &a) {
    return a.y > 0 || (a.y == 0 && a.x > 0) ? 0 : 1;
}
```

```
std::vector<Point> getHull(std::vector<Point> p) {
   std::vector<Point> h, l;
   std::sort(p.begin(), p.end(), [&](auto a, auto b) {
      if (a.x != b.x) {
         return a.x < b.x;
      } else {
         return a.y < b.y;
      }
   });
   p.erase(std::unique(p.begin(), p.end()), p.end());
   if (p.size() <= 1) {
      return p;
   }

   for (auto a : p) {
      while (h.size() > 1 && cross(a - h.back(), a - h[h.size() - 2]) <= 0) {
         h.pop_back();
      }
      while (l.size() > 1 && cross(a - l.back(), a - l[l.size() - 2]) >= 0) {
         l.pop_back();
      }
      l.push_back(a);
      h.push_back(a);
   }

   l.pop_back();
   std::reverse(h.begin(), h.end());
   h.pop_back();
   l.insert(l.end(), h.begin(), h.end());
   return l;
}
```

线性基

```
struct Basis {
   std::vector<int> a, map;
   int n, basecnt;
   int sizofarr; // must set.
   bool zero;
   Basis() {}
   void init(int n) {
      this -> n = n;
      a.assign(n, 0);
      map.clear();
```

```
  }
  bool insert(int x) {
    for (int i = n - 1; i >= 0; i --) {
      if (! (x >> i & 1)) continue;
      if (a[i]) x ^= a[i];
      else {
        for (int j = 0; j < i; j ++) {
          if (a[j] && (x >> j & 1)) {
            x ^= a[j];
          }
        }
        for (int j = i + 1; j < n; j ++) {
          if (a[j] >> i & 1) {
            a[j] ^= x;
          }
        }
        a[i] = x;
        basecnt ++;
        return true;
      }
    }
    return false;
  }
  void build(std::vector<int> b, int siz) {
    sizofarr = siz;
    basecnt = 0, zero = 0;
    for (int i = 1; i <= siz; i ++) {
      insert(b[i]);
    }
    zero = (basecnt != sizofarr); // 是否有零行
    for (int i = 0; i < n; i ++) {
      if (a[i]) map.push_back(i);
    }// 对应下标，方便
  }

  int query_max() {
    int res = 0;
    for (int i = 0; i < n; i ++) {
      res ^= a[i];
    }
    return res;
  }

  int query_min() {
    zero = (basecnt != sizofarr);
    if (zero) return 0;
    if (! map.size()) return -1;
```

```
        return a[map[0]];
    }

    int query_kth(int k) { // kth_small
        zero = (basecnt != sizofarr);
        int res = 0;
        if (zero) k --;
        int siz = map.size();
        if (k >= (1ll << siz)) return -1;
        for (int i = 0; i < siz; i ++) {
            if (k >> i & 1) {
                res ^= a[map[i]];
            }
        }
        return res;
    }

    int query_rank(int x) {
        zero = (basecnt != n);
        int siz = map.size(), res = zero;
        res = 0; // 本题,空为0
        for (int i = 0; i < siz; i ++) {
            if (x >> map[i] & 1) {
                res += 1ll << i;
            }
        }
        return res;
    }

    bool check_in(int x) {
        for (int i = n - 1; i >= 0; i --) {
            if (x >> i & 1) x ^= a[i];
        }
        return x == 0;
    }

    void mergeFrom(const Basis &other) {
        for (int i = 0; i < n; i ++) {
            int it = other.a[i];
            if (it) insert(it);
        }
    }
    friend Basis operator +(const Basis &a, const Basis &b) {
        Basis res = a;
        int X = res.n;
        for (int i = 0; i < X; i ++) {
            int it = b.a[i];
            if (it) res.insert(it);
        }
        return res;
    } // 用于线段树维护线性基
};
```

组合数

```
int qmi(int a, int b){
    if (a == 0) return 0;
    int res = 1;
    while (b) {
        if (b & 1) {
            res = 1ll * res * a % mod;
        }
        a = 1ll * a * a % mod;
        b >>= 1;
    }
    return res;
}
struct Comb {
    int n;
    std::vector<int> _fac, _invfac, _inv;
    Comb() : n{0}, _fac{1}, _invfac{1}, _inv{0} {}
    void init(int m) {
        _fac.resize(m + 1);
        _invfac.resize(m + 1);
        _inv.resize(m + 1);

        for (int i = n + 1; i <= m; i ++) {
            _fac[i] = 1ll * _fac[i - 1] * i % mod;
        }
        _invfac[m] = qmi(_fac[m], mod - 2);
        for (int i = m; i > n; i --) {
            _invfac[i - 1] = 1ll * _invfac[i] * i % mod;
            _inv[i] = 1ll * _invfac[i] * _fac[i - 1] % mod;
        }
        n = m;
    }
    int fac(int m) {
        return _fac[m];
    }
    int invfac(int m) {
        return _invfac[m];
    }
    int inv(int m) {
        return _inv[m];
    }
    int binom(int n, int m) {
        if (n < m || m < 0) return 0;
        return 1ll * fac(n) * invfac(m) % mod * invfac(n - m) % mod;
    }
} comb;
```

Lucas

```
//需要用到逆元处理的模板。
ll C(int a,int b,int mod){
    if(b>a)return 0;//重要
    return fact[a]*infact[b]%mod*infact[a-b]%mod;
}
ll Lucas(ll a,ll b,ll p){
    if(b==0)return 1;
    //if(a<p&&b<p)return C(a,b,p);
    return C(a%p,b%p,p)*Lucas(a/p,b/p,p)%p;
}

//-----扩展Lucas
#include<algorithm>
#include<iostream>
using namespace std;
typedef long long ll;
long long n,m,p,cnt,pr[1010],al[1010];
void exgcd(ll a,ll b,ll& x,ll& y){
    if(b==0){x=1,y=0;return;}
    exgcd(b,a%b,x,y);
    ll z=x;x=y;y=z-y*(a/b);
}
ll ny(ll a,ll p){//逆元
    ll x,y;exgcd(a,p,x,y);
    return (x+p)%p;
}
ll qmi(ll a,ll k,ll p){ ll res=1;
    while(k){if(k&1)res=res*a%p;a=a*a%p;k>>=1;}
    return res;
}
ll fac(ll n,ll p,ll ppa){//计算n!
    if (n==0) return 1; ll cir=1/*循环节*/,rem=1/*余数*/;
    for(ll i=1;i<=ppa;i++)if(i%p)cir=cir*i%ppa;
    cir=qmi(cir,n/ppa,ppa);
```

```
  for(ll i=ppa*(n/ppa);i<=n;i++)if(i%p)rem=rem*(i%ppa)%ppa;
  return fac(n/p,p,ppa)*cir%ppa*rem%ppa;
}
ll sum_fac(ll n, ll p){ ll count = 0;
  do {n /= p;count += n;}while(n);
  return count;
}
ll C(ll n,ll m,ll p,ll ppa){//计算Cnm%pi^ai
  ll fz=fac(n,p,ppa),fm1=ny(fac(m,p,ppa),ppa),fm2=ny(fac(n-m,p,ppa),ppa);
  ll mi=qmi(p,sum_fac(n,p)-sum_fac(m,p)-sum_fac(n-m,p),ppa);
  return fz*fm1%ppa*fm2%ppa*mi%ppa;
}
void pfd(ll n, ll m) {//分解p
  ll P = p;
  for (ll i = 2; i * i <= p; i ++) {
    if (! (P % i)) {
      ll ppa = 1;
      while (! (P % i)) ppa *= i, P /= i;
      pr[++ cnt] = ppa;
      al[cnt] = C(n, m, i, ppa);
    }
  }
  if(P != 1) pr[++cnt]=P,al[cnt]=C(n,m,P,P);
}
ll crt(){//中国剩余定理
  ll ans=0;
  for(ll i=1;i<=cnt;i++){
    ll M=p/pr[i],T=ny(M,pr[i]);
    ans=(ans+al[i]*M%p*T%p)%p;
  }return ans;
}
ll exlucas(ll n, ll m) {//扩展卢卡斯
  pfd(n, m);
  return crt();
}
int main() { // 模数不一定为质数
  cin >> n >> m >> p;
  cout << exlucas(n, m);
}
```

Manacher

```
std::vector<int> manacher(std::string s) {
  std::string t = "#";
  for (auto c : s) {
        t += c;
        t += '#';
    }
    int n = t.size();
    std::vector<int> r(n);
    for (int i = 0, j = 0; i < n; i ++) {
        if (2 * j - i >= 0 && j + r[j] > i) {
            r[i] = std::min(r[2 * j - i], j + r[j] - i);
        }
        while (i - r[i] >= 0 && i + r[i] < n && t[i - r[i]] == t[i + r[i]]) {
            r[i] += 1;
        }
        if (i + r[i] > j + r[j]) {
            j = i;
        }
    }
    return r; // radius
    // 结论 rad[l + r] 是区间 [l, r) 的中间位置的回文长度。
}
```

最小表示法

```
void get_min() {
    int n;
    std::cin >> n;
    for (int i = 1; i <= n; i ++) {
        std::cin >> a[i];
        a[i + n] = a[i];
    }
    int i = 1, j = 2, k = 0;
    while (i <= n && j <= n && k < n) {
        if (a[i + k] == a[j + k]) k ++;
        else {
            a[i + k] > a[j + k] ? (i = i + k + 1) : (j = j + k + 1);
            if (i == j) j++;
            k=0;
        }
    }
    k = std::min(i, j);
    for (int i = k; i < k + n; i ++) {
        std::cout << a[i] << " ";
    }
}
```

KMP

```
std::vector<int> kmp(std::string s) {
    int n = s.size();
    std::vector<int> f(n + 1);
    for (int i = 1, j = 0; i < n; i ++) {
        while (j && s[i] != s[j]) {
            j = f[j];
        }
        j += (s[i] == s[j]);
        f[i + 1] = j;
    }
    return f;
}
void solve() {
    std::string s, t;
    std::cin >> s >> t;
    auto p = kmp(t);

    int n = s.size(), m = t.size();

    std::string a = " " + s, b = " " + t;
    for (int i = 1, j = 0; i <= n; i ++) {
        while (j && a[i] != b[j + 1]) {
            j = p[j];
        }
        j += (a[i] == b[j + 1]);
        if (j == m) {
            std::cout << i - m + 1 << '\n';
            j = p[j];
        }
    }
    for (int i = 1; i <= m; i ++) {
        std::cout << p[i] << ' ';
    }
    std::cout << '\n';
}
```

Matrix

```
struct Matrix {
    int n;
    std::vector<std::vector<int>> M;
    Matrix() {}
    void init(int n) {
        this -> n = n;
        M.assign(n + 1, std::vector<int>(n + 1, 0));
    }
    void norm() {
        for (int i = 1; i <= n; i ++) {
            M[i][i] = 1;
        }
    }
    Matrix friend operator * (const Matrix &a, const Matrix &b) {
        Matrix ans;
        int n = a.n;
        ans.init(n);
        for (int i = 1; i <= n; i ++) {
            for (int j = 1; j <= n; j ++) {
                for (int k = 1; k <= n; k ++) {
                    ans.M[i][j] = (ans.M[i][j] + 1ll * a.M[i][k] * b.M[k][j]) % mod;
                }
            }
        }
        return ans;
    }
};
Matrix qmi(Matrix a, i64 b) {
    Matrix ans;
    int n = a.n;
    ans.init(n);
    ans.norm();
    while (b) {
        if (b & 1) ans = ans * a;
        a = a * a;
        b >>= 1;
    }
    return ans;
}
```

高斯消元

```
const db eps = 1e-5;
struct guess {
    int n;
    std::vector<std::vector<db>> a;
    std::vector<db> b;

    guess() {}
```

```
  void init(int n, std::vector<std::vector<db>> x, std::vector<db> y) {
    this -> n = n;
    a.assign(n + 1, {});
    b.assign(y.begin(), y.end());
    for (int i = 1; i <= n; i ++) {
      a[i].assign(x[i].begin(), x[i].end());
    }
  }
  int work() {
    int l = 1;
    for (int i = 1; i <= n; i ++) {
      int fg = l;
      for (int j = l; j <= n; j ++) {
        if (std::fabs(a[j][i]) > std::fabs(a[l][i])) {
          l = j;
        }
      }
      for (int k = i; k <= n; k ++) {
        std::swap(a[l][k], a[fg][k]);
      }
      std::swap(b[l], b[fg]);
      l = fg;
      if (std::fabs(a[l][i]) < eps) {
        continue;
      }

      for (int j = 1; j <= n; j ++) {
        if (j != l && std::fabs(a[j][i]) > eps) {
          db delta = a[j][i] / a[l][i];
          for (int k = i; k <= n; k ++) {
            a[j][k] -= delta * a[l][k];
          }
          b[j] -= delta * b[l];
        }
      }
      l ++;
    }

    int f = 0;
    for (int i = l; i <= n; i ++) { // 无解
      if (std::fabs(b[i]) > eps) {
        f = -1;
        break;
      }
    }
    if (f != -1 && l <= n) { // 无穷多解
      f = -2;
      }
      if (f == 0) {
        for (int i = 1; i <= n; i ++) {
          b[i] /= a[i][i]; // b[i] is the ans
          a[i][i] = 1.0;
        }
      }
      return f;
    }
};
```

矩阵求逆

```
#include<bits/stdc++.h>
using namespace std;
namespace X{
    #define int long long
    constexpr int mod(1e9+7);
    int qmi(int x,int y){int ret=1;
        for(;y;y&1&&(ret=ret*x%mod),x=x*x%mod,y>>=1);return ret;}
    int ny(int x){return qmi(x,mod-2);}
    constexpr int maxn(405);
    int mt[maxn][maxn*2],N;
    void inv(){
        for(int i=1;i<=N;i++){int pos=0;
            for(int j=i;j<=N;j++)
            if(mt[j][i]){pos=j;break;}
            if(!pos){cout<<"No Solution"<<endl;exit(0);}
            if(pos^i)swap(mt[i],mt[pos]);int v=ny(mt[i][i]);
            for(int j=i;j<=N*2;j++)mt[i][j]=mt[i][j]*v%mod;
            for(int j=1;j<=N;j++){
                if(j==i)continue;int v=mt[j][i];
                for(int k=i;k<=N*2;k++)
                mt[j][k]=(mt[j][k]-v*mt[i][k])%mod;
            }
        }
    }
    int MAIN(){cin>>N;
        for(int i=1;i<=N;i++){
            for(int j=1;j<=N;j++)scanf("%lld",&mt[i][j]);mt[i][N+i]=1;
        }inv();
        for(int i=1;i<=N;i++){
            for(int j=1;j<=N;j++)printf("%lld ",(mt[i][j+N]%mod+mod)%mod);
            puts("");
        }return 0;
    }
};
# undef int
using namespace X;
int main(){
    return MAIN();
}
```

第二类斯特林

```
// n 不同， m 相同集合
int S(int n, int m) {
    i64 res = 0;
    for (int i = 0; i <= m; i ++) {
        res += ((m - i) % 2 ? -1 : 1) * comb.binom(m, i) % mod * qmi(i, n) % mod;
        res %= mod;
    }
    res = res * comb.invfac(m) % mod;
    return res;
}
```

ntt

```
using Poly = std::vector<int>;
//4179340454199820289
const int G = 3, mod = 998244353, Maxn = 2e6 + 10;
int qmi(int a, int b = mod - 2) {
    int res = 1;
    while (b) {
        if (b & 1) {
            res = 1ll * res * a % mod;
        }
        a = 1ll * a * a % mod;
        b >>= 1;
    }
    return res;
}
const int invG = qmi(G);
int tr[Maxn << 1], tf;
void tpre(int n) {
    if (tf == n) return ;
    tf = n;
```

```
  for (int i = 0; i < n; i ++) {
    tr[i] = (tr[i >> 1] >> 1) | ((i & 1) ? n >> 1 : 0);
  }
}

void NTT(int n, int *g, bool op) {
  tpre(n);
  static u64 f[Maxn << 1], w[Maxn << 1];
  w[0] = 1;
  for (int i = 0; i < n; i ++) {
    f[i] = (((i64)mod << 5) + g[tr[i]]) % mod;
  }
  for (int l = 1; l < n; l <<= 1) {
    u64 tG = qmi(op ? G : invG, (mod - 1) / (l + l));
    for (int i = 1; i < l; i ++) w[i] = w[i - 1] * tG % mod;
    for (int k = 0; k < n; k += l + l)
    for (int p = 0; p < l; p ++) {
      int tt = w[p] * f[k | l | p] % mod;
      f[k | l | p] = f[k | p] + mod - tt;
      f[k | p] += tt;
    }
    if (l == (1 << 10))
    for (int i = 0; i < n; i ++) f[i] %= mod;
  }
  if (! op) {
    u64 invn = qmi(n);
    for(int i = 0; i < n; ++ i) {
      g[i] = f[i] % mod * invn % mod;
    }
  } else {
    for (int i = 0; i < n; ++ i) {
      g[i] = f[i] % mod;
    }
  }
}

void px(int n, int *f, int *g) {
  for (int i = 0; i < n; ++ i) {
    f[i] = 1ll * f[i] * g[i] % mod;
  }
}

Poly operator +(const Poly &A, const Poly &B) {
  Poly C = A;
  C.resize(std::max(A.size(), B.size()));
  for (int i = 0; i < B.size(); i ++) {
    C[i] = (C[i] + B[i]) % mod;
  }
  return C;
}
Poly operator -(const Poly &A, const Poly &B) {
  Poly C = A;
  C.resize(std::max(A.size(),B.size()));
  for (int i = 0; i < B.size(); i ++) {
    C[i] = (C[i] + mod - B[i]) % mod;
  }
  return C;
}
Poly operator *(const int c, const Poly &A) {
  Poly C;
  C.resize(A.size());
  for (int i = 0; i < A.size(); i ++) {
    C[i] = 1ll * c * A[i] % mod;
  }
  return C;
}
int lim; // set.
Poly operator *(const Poly &A, const Poly &B) {
  static int a[Maxn << 1], b[Maxn << 1];
  for (int i = 0; i < A.size(); i ++) a[i] = A[i];
  for (int i = 0; i < B.size(); i ++) b[i] = B[i];
  Poly C;
  C.resize(std::min(lim, (int)(A.size() + B.size() - 1)));
  int n = 1;
  for(n; n < A.size() + B.size() - 1; n <<= 1);
  NTT(n, a, 1);
  NTT(n, b, 1);
  px(n, a, b);
  NTT(n, a, 0);
  for (int i = 0; i < C.size(); i ++) {
    C[i] = a[i];
  }
  for (int i = 0; i <= n; i ++) {
    a[i] = 0;
    b[i] = 0;
  }
  return C;
}
void pinv(int n, const Poly &A, Poly &B) {
  if (n == 1) B.push_back(qmi(A[0]));
  else if (n & 1){
    pinv(-- n, A, B);
    int sav = 0;
    for (int i = 0; i < n; i ++) {
      sav = (sav + 1ll * B[i] * A[n - i] % mod) % mod;
    }
    B.push_back(1ll * sav * qmi(mod - A[0]) % mod);
  } else {
    pinv(n / 2, A, B);
    Poly sA;
    sA.resize(n);
    for (int i = 0; i < n; i ++) {
      sA[i] = A[i];
    }
    B = 2 * B - B * B * sA;
    B.resize(n);
  }
}
Poly pinv(const Poly &A) { // P-inv
  Poly C;
  pinv(A.size(), A, C);
  return C;
}
int inv[Maxn];
void Init() {
  inv[1] = 1;
  for (int i = 2; i <= lim; i ++) {
    inv[i] = 1ll * inv[mod % i] * (mod - mod / i) % mod;
  }
}
Poly dao(const Poly &A) { // P-qiu-dao
  Poly C = A;
  for (int i = 1; i < C.size(); i ++) {
    C[i - 1] = 1ll * C[i] * i % mod;
  }
  C.pop_back();
  return C;
}
Poly ints(const Poly &A) { // P-ji-fen
  Poly C = A;
  for (int i = C.size() - 1; i; i --)
  C[i] = 1ll * C[i - 1] * inv[i] % mod;
  C[0] = 0;
  return C;
}
Poly ln(const Poly &A) { // P-ln
  return ints(dao(A) * pinv(A));
}
void pexp(int n, const Poly &A, Poly &B) {
  if (n == 1) B.push_back(1);
  else if (n & 1) {
    pexp(n - 1, A, B);
    n -= 2;
    int sav = 0;
    for (int i = 0; i <= n; i ++) {
      sav = (sav + 1ll * (i + 1) * A[i + 1] % mod * B[n - i] % mod) % mod;
    }
    B.push_back(1ll * sav * inv[n + 1] % mod);
  } else {
    pexp(n / 2, A, B);
    Poly lnB = B;
    lnB.resize(n);
    lnB = ln(lnB);
    for (int i = 0; i < lnB.size(); i ++) {
      lnB[i] = (mod + A[i] - lnB[i]) % mod;
    }
    lnB[0] ++;
    B = B * lnB;
    B.resize(n);
  }
}
Poly pexp(const Poly &A) { // P-exp
  Poly C;
  pexp(A.size(), A, C);
  return C;
}


void solve() {
  int n;
  std::cin >> n;
  lim = n + n;
  Init();
  Poly F;
  F.resize(n);
  for (int i = 0; i < n; i ++) {
    std::cin >> F[i];
  }
  Poly ans = pexp(F);
  for (int i = 0; i < n; i ++) {
    std::cout << ans[i] << ' ';
  }
  std::cout << '\n';
}
```