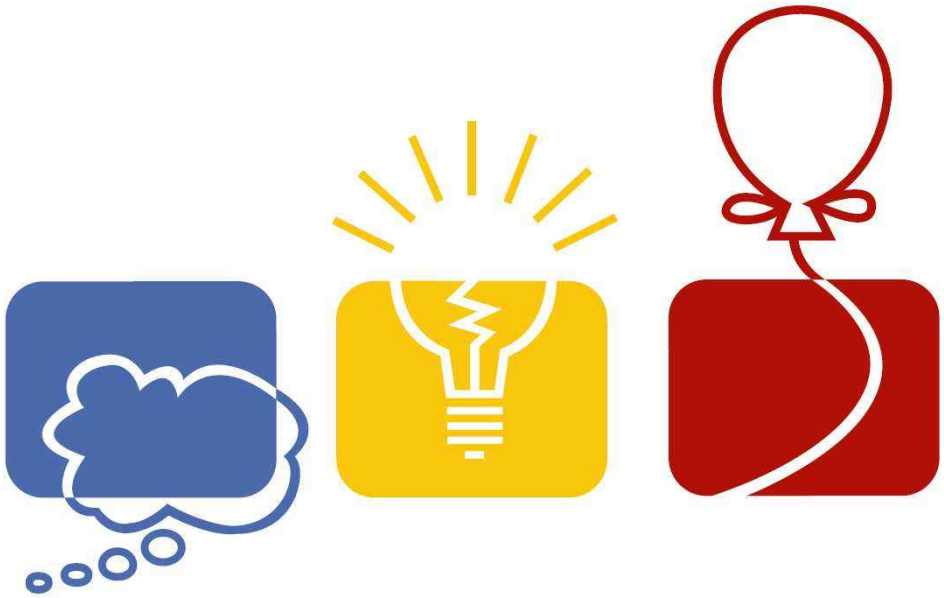

ACM/ICPC Template Runtime Error

浙江工商大学

锟斤拷之烫

September 16, 2021

Contents

0 头文件

0.1 Header

```cpp
#include "bits/stdc++.h"

using namespace std;

typedef long long ll;
typedef long double ld;
typedef unsigned long long ull;
typedef vector<ll> VI;
typedef pair<int, int> pii;
typedef pair<double, double> pdd;
typedef pair<ll, ll> pll;

#define endl "\n"
#define fi first
#define se second
#define eb emplace_back
#define mem(a, b) memset(a , b , sizeof(a))

const ll INF = 0x3f3f3f3f;
const ll mod = 998244353;
// const ll mod = 1e9 + 7;
const double eps = 1e-6;
const double PI = acos(-1);
const double R = 0.57721566490153286060651209;


void solve() {

}

signed main() {
    ios_base::sync_with_stdio(false);
    // cin.tie(nullptr);
    // cout.tie(nullptr);
#ifdef FZT_ACM_LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
    signed test_index_for_debug = 1;
    char acm_local_for_debug = 0;
    do {
        if (acm_local_for_debug == '$') exit(0);
        if (test_index_for_debug > 20)
            throw runtime_error("Check the stdin!!!");
        auto start_clock_for_debug = clock();
        solve();
        auto end_clock_for_debug = clock();
        cout << "Test " << test_index_for_debug << " successful" << endl;
        cerr << "Test " << test_index_for_debug++ << " Run Time: "
             << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC << "s" << endl;
        cout << "--------------------------------------------------" << endl;
    } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
#else
    solve();
#endif
```

```
55      return 0;
56  }
```

1 字符串

1.1 字符串哈希

```
// 区间哈希

const int N = 100010, P = 131;  // 这里的P 是经验值 131 , 或者13331

int n,m;
char str[N];
// 这里用unsigned long long 存储就相当于mod 2 ^ 64, 因为超过了会溢出的
ULL h[N], p[N];   // h[]是存储字符串哈希值的  p[] 是存储p次方的

ULL get(int l, int r) {
    return h[r] - h[l -1] * p[r - l + 1];     // 区间hash 的公式
}

int main() {
    scanf("%d%d%s", &n, &m, str + 1);

    p[0] = 1;
    for(int i=1; i<=n; i++) {
        p[i] = p[i-1] * P;  // p数组保存 计算的次方数
        h[i] = h[i-1] * P +str[i];   // 计算字符串的前缀,   后面的是0次 所以直接加上str[i]就行了
    }

    while(m -- ) {
        int l1, r1, l2, r2;
        scanf("%d%d%d%d", &l1, &r1, &l2, &r2)   ;
        if(get(l1, r1) == get(l2, r2)) cout << "Yes" << endl;
        else cout << "No" << endl;
    }
    return 0;
}

// 线段树维护区间哈希
struct Tree {
    int l, r;
    ull sum;
    Tree operator + (const Tree &a) const {
        Tree ans;
        ans.l = l, ans.r = a.r;
        ans.sum = sum * p[a.r - a.l + 1] + a.sum;
        return ans;
    }
}t[N << 2];
```

1.2 KMP

```
const int maxn = 10000005;
int Next[maxn];
int slen, tlen;
char S[maxn], T[maxn];
void GetNext() {
    int j, k;
    j = 0;
    k = -1;
```

```
    Next[0] = -1;
    while(j < tlen) {
        if(k == -1 || T[j] == T[k]) {
            Next[++j] = ++k;
        }
        else
            k = Next[k];
        }
}

/*返回模式串T在主串S中c首次出现的位置 ， 返回的位置是从0开始的*/
int KMP_Index() {
    int i = 0, j = 0;
    GetNext();
    while(i < slen && j < tlen) {
        if(j == -1 || S[i] == T[j]) i++, j++;
        else j = Next[j];
    }
    if(j == tlen) return i - tlen;
    else return -1;
}

 /*返回模式串T在字串中出现的次数*/
int KMP_Count() {
    int ans = 0;
    if(slen == 1 && tlen == 1) {
        if(S[0] == T[0])
            return 1;
        else
            return 0;
    }
    GetNext();
    int j = 0;
    for(int i = 0;i < slen; i++) {
        while(j > 0 && S[i] != T[j]) {
            j = Next[j];
        }
        if(S[i] == T[j]) j++;
        if(j == tlen) {
            ans++;
            j = Next[j];
        }
        /*if(j ==tlen)
        {
            printf("%d ",i - j);
        }
        模式串在主串中所有的位置*/
        }
    return ans;
}
int main()
{
    cin >> S >> T;
    slen = strlen(S);
    tlen = strlen(T);
    cout << "返回模式串T在主串S中c首次出现的位置是: " << KMP_Index() << endl;
    cout << "返回模式串T在字串中出现的次数是: "<< KMP_Count() << endl;
    return 0;
}
```

```

/*---------------------------------*/

//预处理前缀数组,前缀数组用于在字符串失配时进行转跳
#define MAXN 35000
int pi[MAXN];//前缀函数
void prefix_function(string s){
    memset(pi,0,sizeof(pi));
    for(int i=1;i<s.size();i++){
        int p = pi[i-1];
        while(p>0&&s[p]!=s[i])p=pi[p-1];
        if(s[i]==s[p])  pi[i]=++p;
    }
    return ;
}

//KMP写法一,不需要预处理版本
//将两个字符串拼接,如果从前往后求前缀,如果前缀为模式串长度,则匹配成功
vector<int> kmp(string s1,string s2){
    vector<int> appear;
    string s = s2+'%'+s1;//ATTENTION!!这个百分号应该是不存在于s1和s2中的任字符
    for (int i = 1; i < s.size(); i++) {
        int p = pi[i - 1];
        while (p > 0 && s[i] != s[p]) p = pi[p - 1];
        if (s[i] == s[p]) p++;
        pi[i] = p;
        if(pi[i]==s2.size())
            appear.push_back(i-2-s2.size());//注意这里要根据字符串首字母为止为0或1进行微调
                                            //此时为1
    }
    return appear;
}

//KMP写法2,需要对模式串进行预处理
vector<int> kmp(string s1,string s2){
    vector<int> appear;
    int p=0;
    for(int i=0;i<s1.size();i++){
        while (p > 0 && s1[i] != s2[p]) p = pi[p - 1];
        if (s1[i] == s2[p]) p++;
        if(p==s2.size())
            appear.push_back(i+2-s2.size());//要注意同理
    }
    return appear;
}
```

1.3 AC 自动机

```
typedef long long ll;

// 指针版

typedef struct _node {
    _node *child[26]; // 儿子节点
    _node *fail; // ！！！
    vector<int> exist; // 该节点是否存在完整的单词
    bool flag;
}node;
```

```
11
12  char T[1000005];
13  int n;
14
15  void insert(node *root, char *q) {
16      node *tmp = root;
17      int len = strlen(q);
18      for(int i = 0;i < len; i++) {
19          int v = q[i] - 'a';
20          if(tmp -> child[v] == NULL) {
21              node *temp = new node();
22              tmp -> child[v] = temp;
23          }
24          tmp = tmp -> child[v];
25      }
26      tmp -> exist.push_back(len);
27  }
28
29  void build(node *root) {
30      for(int i = 0;i < n; i++) {
31          char p[1000005];
32          cin >> p;
33          insert(root, p); // 建立字典树
34      }
35      queue<node*> q; // BFS层次遍历
36
37      for(int i = 0;i < 26; i++) { // 确定第一层的fail指针全部指向root节点
38          if(root -> child[i]) {
39              q.push(root -> child[i]);
40              root -> child[i] -> fail = root;
41          }
42      }
43
44      while(!q.empty()) {
45          node *x = q.front();
46          q.pop();
47          for(int i = 0;i < 26; i++) {
48              if(x -> child[i]) {
49                  node *y = x -> child[i];
50                  node *fafail = x -> fail;
51
52                  while(fafail && fafail -> child[i] == NULL) {
53                      fafail = fafail -> fail;
54                  }
55
56                  if(fafail == NULL)
57                      y -> fail = root;
58                  else
59                      y -> fail = fafail -> child[i];
60
61                  q.push(y);
62              }
63          }
64      }
65  }
66
67  ll query(node *root) {
68      int len = strlen(T);
69      ll ans = 0;
```

```
70      node *tmp = root;
71      for(int i = 0;i < len; i++) {
72          int c = T[i] - 'a';
73          while(tmp -> child[c] == NULL && tmp -> fail) {
74              tmp = tmp -> fail;
75          }
76          if(tmp -> child[c] != NULL) {
77              tmp = tmp -> child[c];
78          }
79          else
80              continue;
81
82          if(!tmp -> flag) {
83              ans += tmp->exist.size();
84              tmp->flag = true;
85
86              if (tmp->exist.size()) {
87                  for (int j = 0; j < tmp->exist.size(); j++) {
88                      int m = tmp->exist[j];
89                      for (int k = i - m + 1; k <= i; k++) {
90                          cout << T[k];
91                      }
92                      cout << endl;
93                  }
94              }
95          }
96      }
97      return ans;
98  }
99
100 void AC() {
101     node *root = new node();
102     build(root);
103     query(root);
104 }
105
106 const int CHAR_NUM = 26; // 仅小写
107 const int N = 1e6 + 10;
108
109 int trie[N][CHAR_NUM], cnt;
110
111 int fail[N];
112
113 int val[N];
114
115 char T[N];
116 int n;
117
118 void Insert(char *s) {
119     int len = strlen(s);
120     int root = 0;
121     for(int i = 0;i < len; i++) {
122         int v = s[i] - 'a';
123         if(!trie[root][v])
124             trie[root][v] = ++cnt;
125         root = trie[root][v];
126     }
127     val[root]++;
128 }
```

```

void Build() {
    for(int i = 0;i < n; i++) {
        char p[N];
        cin >> p;
        Insert(p);
    }

    queue<int> q;

    fail[0] = 0;
    for(int i = 0;i < 26; i++) {
        if(trie[0][i]) {
            q.push(trie[0][i]);
            fail[trie[0][i]] = 0;
        }
    }

    while(!q.empty()) {
        int now = q.front();
        q.pop();

        for(int i = 0;i < 26; i++) {
            if(trie[now][i]) {
                fail[trie[now][i]] = trie[fail[now]][i];
                q.push(trie[now][i]);
            }
            else {
                trie[now][i] = trie[fail[now]][i];
            }
        }
    }
}

ll Query() {
    ll ans = 0;
    int len = strlen(T);
    int root = 0;
    for(int i = 0;i < len; i++) {
        int v = T[i] - 'a';
        root = trie[root][v];

        for(int j = root; j && val[j] != -1; j = fail[j]) {
            ans += val[j];
            val[j] = -1;
        }
    }
    return ans;
}

void Ac() {
    Build();
    Query();
}
```

1.4 马拉车

```
#include <iostream>
```

```cpp
#include <cstdio>
#include <cstring>
#define Min(a,b) a>b?b:a
#define Max(a,b) a>b?a:b
using namespace std;
const int N = 2e6 + 10 ;
int Len[N] , a[N] ;
char str[N] ;
int n,mx,id,len;
string s ;
void init(int l , int r){

    int k=0;
    str[k++] = '$';
    for(int i=l;i<=r;i++){

        str[k++]='#';
        str[k++]=s[i];
    }
    str[k++]='#';
    len=k;
    str[k] = '\0' ;
}
int ans = 0 , flag ;
int Manacher(){

  Len[0] = 0;
  int sum = 0;
  mx = 0;
  id = 0 ;
  // cout << str << endl ;
  for(int i=1;i<len;i++){

    if(i < mx) Len[i] = Min(mx - i, Len[2 * id - i]);
    else Len[i] = 1;
    while(str[i - Len[i]]== str[i + Len[i]]) Len[i]++;
    if(Len[i] + i > mx){
            // 更新最长的回文串
      mx = Len[i] + i;           // mx是回文串右边一个位置
      id = i;              //id是回文串的中心
      sum = Max(sum, Len[i]);         // sum 是回文串的长度 + 1

    }
    // cout << i << " " << Len[i] << endl ;
    // if(Len[i] == i)  表示前缀是回文的
    //  {

    //    if(ans < Len[i])
    //     ans = Len[i] - 1 , flag = 1 ;
    //  }
    // if(Len[i] + i == len)  表示后缀是回文的
    //  if(ans < Len[i])
    //   ans = Len[i] - 1 , flag = 2 ;
  }
  return sum - 1 ;
}

int main()
{
```

```
61
62    scanf("%d",&n);
63    while(n--){
64
65      cin >> s ;
66      len = s.size();
67      int l = 0 , r = len - 1 ;
68      init(l , r);
69      ans = 0 , flag = 0 ;
70      // cout << Manacher() << endl ;
71    }
72    return 0;
73  }
```

2 动态规划

2.1 01 背包

```
int v[1005], w[1005];
int f[1005];

void fsbag() {
    int n, V;
    cin >> n >> V;
    for(int i = 1;i <= n; i++) {
        cin >> v[i] >> w[i];
    }

    for(int i = 1;i <= n; i++) {
        for(int j = V;j >= v[i]; j--) {
            f[j] = max(f[j], f[j - v[i]] + w[i]);
        }
    }

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

2.2 简单多重背包

```
int v[1005], w[1005], s[1005];
int f[1005];

void mulbag() {
    int N, T;
    cin >> N >> T;
    for(int i = 1;i <= N; i++)
        cin >> v[i] >> w[i] >> s[i];
    for(int i = 1;i <= N; i++) {
        for(int j = T;j >= 0; j--) {
            for(int k = 1;k <= s[i]; k++) {
                if(j - k * v[i] >= 0)
                    f[j] = max(f[j], f[j - k * v[i]] + k * w[i]);
            }
        }
    }
    cout << f[T] << endl;
}
```

2.3 二进制优化多重背包

```

int f[2005];
struct Good{
     int v, w;
};

void mulbag2() {
    int N, T;
    cin >> N >> T;
    vector<Good> goods;
    for(int i = 1;i <= N; i++) {
```

```
        int v, w, s;
        cin >> v >> w >> s;
        for(int k = 1;k <= s; k *= 2) {
            s -= k;
            goods.push_back({v * k, w * k});
        }
        if(s > 0)
            goods.push_back({s * v, s * w});
    }
    for(auto good : goods) {
        for(int j = T;j >= good.v; j--) {
            f[j] = max(f[j], f[j - good.v] + good.w);
        }
    }
    cout << f[T] << endl;
}
```

2.4 单调队列优化多重背包

```
const int N = 1e5 + 10;

int que[N], head, tail;
int f[N], pre[N];
void solve() {
    int n, m; cin >> n >> m;
    for(int i = 1;i <= n; i++) {
        memcpy(pre, f, sizeof(f));
        int v, w, s; cin >> v >> w >> s;
        for(int j = 0;j < v; j++) {
            head = 1; tail = 0;
            for(int k = j;k <= m; k += v) {
                while(head <= tail && que[head] < k - s * v) head++;
                if(head <= tail) f[k] = max(pre[k], pre[que[head]] + (k - que[head]) / v * w);
                while(head <= tail && pre[k] >= pre[que[tail]] + (k - que[head]) / v * w) tail--;
                que[++tail] = k;
            }
        }
    }
    cout << f[m] << endl;
}
```

2.5 混合背包

```
struct Good {
    int v, w, kind;
};

int f[2005];
void mulbag2() {
    int n, V;
    cin >> n >> V;
    vector<Good> goods;
    for(int i = 1;i <= n; i++) {
        int v, w, s;
        cin >> v >> w >> s;
```

```
        if(s < 0) {
            goods.push_back({v, w, -1});
        }
        else if(s == 0) {
            goods.push_back({v, w, 0});
        }
        else {
            for(int k = 1;k <= s; k *= 2) {
                s -= k;
                goods.push_back({v * k, w * k, -1});
            }
            if(s > 0)
                goods.push_back({v * s, w * s, -1});
        }
    }
    for(auto good: goods) {
        if(good.kind == -1) {
            for(int j = V;j >= good.v; j--) {
                f[j] = max(f[j], f[j - good.v] + good.w);
            }
        }
        else {
            for(int j = good.v;j <= V; j++) {
                f[j] = max(f[j], f[j - good.v] + good.w);
            }
        }
    }
    cout << f[V] << endl;
}
```

2.6 完全背包

```
int v[1005], w[1005];
int f[1005];

void cptbag() {
    int N, T;
    cin >> N >> T;
    for(int i = 1;i <= N; i++)
        cin >> v[i] >> w[i];
    for(int i = 1;i <= N; i++) {
        for(int j = 0;j <= T; j++) {
            if(j - v[i] >= 0)
                f[j] = max(f[j], f[j - v[i]] + w[i]);
        }
    }
    cout << f[T] << endl;
}
```

2.7 分组背包

```
int v[2005], w[2005];
int f[2005];

void Divbag() {
    int n, V;
    cin >> n >> V;
```

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

2.8 二维费用背包

```
int f[2005][2005];

void TwoCostbag() {
    int n, M, V;
    cin >> n >> V >> M;
    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--) {
                f[j][k] = max(f[j][k], f[j - v][k - m] + w);
            }
        }
    }
    cout << f[V][M] << endl;
}
```

2.9 树上背包

```
//加了剪枝后复杂度为O(NM)

void dfs(int u, int fa) {
    siz[u] = 1;
    for (auto &v: g[u])
        if (v != fa) {
            dfs(v, u);

            int now = min(siz[u] + siz[v] + 1, M);
            int t[MAX_M]; for (int i = 0; i <= M; i++) t[i] = INF/-INF;//初始化
            for (int i = 0; i <= siz[u]; i++)
                for (int j = 0; j <= siz[v] && i + j <= M; j++) {
                    //...转移方程
                }
            for (int i = 0; i <= now; i++) f[u][i] = min/max(f[u][i], t[i]);
            siz[u] = now;
        }
}
```

2.10 LIS

```
typedef long long ll;
const int N = 1e5 + 10;
const int INF = 0x3f3f3f3f;

int a[N], f[N];

int n;
int ans;

int DP() {
    for(int i = 1;i <= n; i++) {
        f[i] = 1;
    }
    for(int i = 1;i <= n; i++) {
        for(int j = 1;j < i; j++) {
            if(a[j] < a[i])
                f[i] = max(f[i], f[j] + 1);
        }
    }
    for(int i = 1;i <= n; i++) {
        ans = max(ans, f[i]);
    }
    return ans;
}

int stl() {
    int d[N];
    fill(f, f + n, INF);
    for(int i = 0;i < n; i++) {
        int j = lower_bound(f, f + n, a[i]) - f;
        d[i] = j + 1;
        if(ans < d[i]) {
            ans = d[i];
        }
        f[j] = a[i];
    }
    return ans;
}

int t[N];
int maxn;

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

void update(int k, int val) {
    while(k <= maxn) {
        t[k] = max(t[k], val);
        k += lowbit(k);
    }
}

int query(int k) {
    int res = 0;
    while(k) {
        res = max(res, t[k]);
```

```
58          k -= lowbit(k);
59      }
60      return res;
61  }
62
63  int Tree() {
64      for(int i = 1;i <= n; i++) {
65          maxn = max(maxn, a[i]);
66      }
67
68      int ans = 0;
69      for(int i = 1;i <= n; i++) {
70          int q = query(a[i]) + 1; // 保证j < i，找f[j]最大的
71          update(a[i], q);
72          ans = max(ans, q);
73      }
74      return ans;
75  }
```

2.11 数位 DP

```
1   // 输出0~n中存在49的个数
2
3   int dight[105];
4   ll dp[105][10]; // 第一维是数位，第二维是上一个数位上的数字
5
6   ll DFS(int pos, int pre, bool limit) {// pos为该数位，pre为上一数位上的数字，limit表示是否上
        界，上一位pre是否已到达最大
7       if(pos == 0) return 1; // 每一数位都遍历完，说明这一个数是可取的
8       if(!limit && dp[pos][pre] != -1) return dp[pos][pre]; // 剪枝，记忆化搜索
9       int up = limit ? dight[pos] : 9; // 如果有上界，则pos该数位只能取到dight[pos]这么大
10      ll ans = 0;
11      for(int i = 0;i <= up; i++) {// 遍历该数位1~up的每一位
12          if(pre == 4 && i == 9) // 如果上一位是4且这一位是9，那就组成了49，不可取
13              continue;
14          ans += DFS(pos - 1, i, limit && i == dight[pos]); // ans+搜索下一位
15      }
16      if(!limit)
17          dp[pos][pre] = ans;
18      return ans;
19  }
20
21  ll sum(ll n) {
22      int k = 0;
23      while(n) {
24          dight[++k] = n % 10;
25          n /= 10;
26      }
27      return DFS(k, 0, 1);
28  }
29
30  void solve() {
31      mem(dp, -1); // 初始化所有的dp
32      ll n; cin >> n;
33      cout << n - sum(n) + 1 << endl; // +1是因为sum(n)把0也算进去了
34  }
```

2.12 状压 DP

```
// 杭电1565

int n;
int a[22][22];
int dp[22][1 << 18]; // 第一维是行数，第二位是该行的方案数，继承了前面所有行数的方案数
int tot[1 << 18]; // 方案数

int calc(int i, int k)
{
    int cnt = 1, res = 0;
    while(k)
    {
        if(k & 1) res += a[i][cnt];
        k >>= 1;
        cnt++;
    }
    return res;
}

void solve()
{
    while(cin >> n) {
        mem(dp, 0);
        int cnt = 0;

        for (int i = 0; i <= (1 << n) - 1; i++) { // 预处理
            if ((i & (i >> 1)) == 0) // 判断i这个二进制是否满足相邻没有两个1的条件
                tot[++cnt] = i;
        }

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

        for (int i = 1; i <= n; i++) { // 行遍历
            for (int k = 1; k <= cnt; k++) { // 第i行k的二进制排列的数，与下面的j进行&
                int val = calc(i, tot[k]); // 计算k的二进制中1所在a数组里的权值
                for (int j = 1; j <= cnt; j++) { // 第i-1行j的二进制排列的数，与上面的k进行&并进行状态转移
                    if ((tot[j] & tot[k]) == 0)
                        dp[i][k] = max(dp[i][k], dp[i - 1][j] + val);
                }
            }
        }

        int ans = -1;
        for (int j = 1; j <= cnt; j++)
            ans = max(ans, dp[n][j]);

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

2.13 四边形优化

```
//四边形优化区间dp(n^3 -> n^2)
```

```
//a < b < c < d, f[l][r] = min(f[l][k] + f[k + 1][r] + cost(l, r))
//1. cost(b, c) <= cost(a, d)
//2. cost(a, c) + cost(b, d) <= cost(a, d) + cost(b, c), 即交叉小于包含

void solve() {
    for (int len = 2; len <= n; len++) {
        for (int l = 1, r; l + len - 1 <= n; l++) {
            r = l + len - 1;
            mn[l][r] = 0x3f3f3f3f;
            for (int k = m[l][r - 1]; k <= m[l + 1][r]; k++)
                if (mn[l][k] + mn[k + 1][r] + cost(l, r) < mn[l][r]) {
                    mn[l][r] = mn[l][k] + mn[k + 1][r] + cost(l, r);
                    m[l][r] = k;
                }
        }
    }
}
```

2.14 环基树 dp

```
int flag, S, E;//flag是否找到环, SE为环上两个点

void findCircle(int u, int fa) {
    vis[u] = 1;
    for (int i = head[u], v; i; i = e[i].nxt)
        if ((v = e[i].to) != fa) {
            if (vis[v]) flag = 1, S = u, E = v;
            else findCircle(v, u);
        }
}

void dp(int u, int fa) {
    //dp过程

    for (int i = head[u], v; i; i = e[i].nxt)
        if ((v = e[i].to) != fa && v) {
            dp(v, u);

        }
}

ll calc(int u) {
    flag = 0;
    findCircle(u, 0);
    if (flag) {
        for (int i = head[S], v; i; i = e[i].nxt)
            if ((v = e[i].to) == E) {
                e[i].to = e[i ^ 1].to = 0;//删边操作, 注意e[tot]中tot从2开始
                break;
            }
        ll res = 0;
        dp(S, 0); res = max(res, ...);
        dp(E, 0); res = max(res, ...);
        return res;
    }
    else {
        dp(u, 0);
        return ...;
```

```
    }
}
```

2.15 线段树优化 dp

```
const ll INF = 0x3f3f3f3f;

#define lc u << 1
#define rc u << 1 | 1
#define mid (t[u].l + t[u].r) / 2

const int N = 3e5 + 10;
const int K = 100 + 10;
int dp[N][K];

struct Tree {
    int l, r;
    int mn;
    int tag;
}t[N << 1];

inline void push_up(int u) {
    t[u].mn = min(t[lc].mn, t[rc].mn);
}

inline void push_down(int u) {
    if(!t[u].tag) return ;
    t[lc].tag += t[u].tag;
    t[rc].tag += t[u].tag;
    t[lc].mn  += t[u].tag;
    t[rc].mn  += t[u].tag;
    t[u].tag = 0;
}

void build(int u, int l, int r, int k) {
    t[u].l = l; t[u].r = r;
    t[u].tag = 0; t[u].mn = INF;
    if(l == r) {
        t[u].mn = dp[l][k];
        return ;
    }
    int m = (l + r) / 2;
    build(lc, l, m, k);
    build(rc, m + 1, r, k);
    push_up(u);
}

void modify(int u, int ql, int qr, int val) {
    if(ql <= t[u].l && t[u].r <= qr) {
        t[u].mn += val;
        t[u].tag += val;
        return ;
    }
    push_down(u);
    if(ql <= mid) modify(lc, ql, qr, val);
    if(qr >  mid) modify(rc, ql, qr, val);
    push_up(u);
}
```

```cpp

int query(int u, int ql, int qr) {
    if(ql <= t[u].l && t[u].r <= qr) return t[u].mn;
    push_down(u);
    int ans = INF;
    if(ql <= mid) ans = min(ans, query(lc, ql, qr));
    if(qr  > mid) ans = min(ans, query(rc, ql, qr));
    return ans;
}


void solve() {
    int n, k; cin >> n >> k;
    vector<int> a(n + 1), pre(n + 1, -1), from(n + 1);
    for(int i = 1;i <= n; i++) cin >> a[i];

    for(int i = 1;i <= n; i++) {
        dp[i][1] = dp[i - 1][1];
        from[i] = pre[a[i]];
        if(from[i] != -1) dp[i][1] += i - from[i];
        pre[a[i]] = i;
    }

    for(int i = 2;i <= k; i++) {
        build(1, 1, n, i - 1);
        for(int j = i;j <= n; j++) {
            if(from[j] != -1) modify(1, 1, from[j] - 1, j - from[j]);
            dp[j][i] = min(dp[j - 1][i - 1], query(1, i - 1, j - 1));
            // cout << j << " " << "分成" << i << "段: " << " " << dp[j][i] << endl;
        }
    }
    cout << dp[n][k] << endl;
}

signed main() {
    solve();
}
```

3 数据结构

3.1 BTree

```
template<class T>

struct TreeNode {
    T value;
    TreeNode *left;
    TreeNode *right;
};

template<class T>
TreeNode<T> *createTree(const T *pre, const T *in, const int len) {
    TreeNode<T> *t = NULL;
    if (len > 0) {
        t = new TreeNode<T>;
        t->value = pre[0];
        int index;
        for (index = 0; index < len; index++) {
            if (in[index] == pre[0]) {
                break;
            }
        }
        if (index == len) {
            index = -1;
        }
        t->left = createTree(pre + 1, in, index);
        t->right = createTree(pre + index + 1, in + index + 1, len - index - 1);
    }
    return t;
}

template<class T>
int preOrder(TreeNode<T> *root, queue<T> &out) {
    if (root) {
        int count = 1;
        out.push(root->value);
        count += preOrder(root->left, out);
        count += preOrder(root->right, out);
        return count;
    } else {
        return 0;
    }
}

template<class T>
int inOrder(TreeNode<T> *root, queue<T> &out) {
    if (root) {
        int count = 1;
        count += inOrder(root->left, out);
        out.push(root->value);
        count += inOrder(root->right, out);
        return count;
    } else {
        return 0;
    }
}

```

```
56  template<class T>
57  void postOrder(TreeNode<T> *root, queue<T> &out) {
58      if (root) {
59          postOrder(root->left, out);
60          postOrder(root->right, out);
61          out.push(root->value);
62      } else {
63          return;
64      }
65  }
66
67  template<class T>
68  T *convertQueueToArray(queue<T> &out, int len) {
69      T *list = new T[len];
70      int now = 0;
71      while (!out.empty() && now < len) {
72          list[now] = out.front();
73          out.pop();
74          now++;
75      }
76      return list;
77  }
78
79  template<class T>
80  void destroyTree(TreeNode<T> *root) {
81      if (root) {
82          destroyTree(root->left);
83          destroyTree(root->right);
84          delete root;
85      } else return;
86  }
87
88  template<class T>
89  void insertIntoBSTree(TreeNode<T> *root, const T &value) {
90      if (!root) {
91          return;
92      }
93      if (value < root->value) {
94          if (root->left) {
95              insertIntoTree(root->left, value);
96          } else {
97              root->left = new TreeNode<T>;
98              root->left->value = value;
99              root->left->left = NULL;
100             root->left->right = NULL;
101         }
102     } else if (value > root->value) {
103         if (root->right) {
104             insertIntoTree(root->right, value);
105         } else {
106             root->right = new TreeNode<T>;
107             root->right->value = value;
108             root->right->left = NULL;
109             root->right->right = NULL;
110         }
111     }
112 }
113
114 template<class T>
```

```
TreeNode<T> *createBSTree(T *list, int len) {
    if (len < 1) {
        return NULL;
    }
    TreeNode<T> *root = new TreeNode<char>;
    root->value = list[0];
    root->left = NULL;
    root->right = NULL;
    for (int i = 1; i < len; i++) {
        insertIntoBSTree(root, list[i]);
    }
    return root;
}
```

3.2 pbds-bbtree

```
// RBTree 红黑树
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
// 红黑树
__gnu_pbds::tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update> t;
// null_type无映射(低版本g++为null_mapped_type)
// 类似multiset
__gnu_pbds::tree<int, null_type, less_equal<int>, rb_tree_tag,
    tree_order_statistics_node_update> t;
find_by_order(size_t order);
// 结点更新
tree_order_statistics_node_update
insert(p);
erase(it);
// 求k在树中是第几小,假设插入当前值判断当前值是第几小,最小为第0小
order_of_key(p);
// 找到第order小的迭代器
find_by_order(order);
// 前驱
lower_bound(p);
// 后驱
upper_bound(p);
// 合并
a.join(b);
// 分割 key小于等于v的元素属于a，其余的属于b
a.split(v, b);

// 优先队列
#include <ext/pb_ds/priority_queue.hpp>
#include <ext/pb_ds/assoc_container.hpp>
// 操作类似于stl的优先队列
typedef __gnu_pbds::priority_queue<node, greater<node>, __gnu_pbds::thin_heap_tag> heap
    ;
heap::point_iterator; // 指向元素的指针
```

3.3 BIT

```
// 树状数组和二进制紧紧联系

int tree[100005];
```

```
4  int n;
5
6  int lowbit(int x) {
7      return x & (-x); // 计算机中负数使用对应正数的补码表示
8  }
9
10 int getsum(int x) {
11     int ans = 0;
12     while(x > 0) {
13         ans += tree[x];
14         x -= lowbit(x);
15     }
16     return ans;
17 }
18
19 void update(int x,int value) {
20     while(x <= n) {
21         tree[x] += value;
22         x += lowbit(x);
23     }
24 }
25
26 /*--------区间最大值--------*/
27 int t[N], a[N];
28 void modify(int x) {
29     while(x <= n) {
30         t[x] = a[x];
31         for(int i = 1;i < lowbit(x); i <<= 1) {
32             t[x] = max(t[x], t[x - i]);
33         }
34         x += lowbit(x);
35     }
36 }
37
38 int query(int l, int r) {
39     int ans = 0;
40     while(l <= r) {
41         ans = max(ans, a[r]);
42         for (r-- ;r - lowbit(r) >= l; r -= lowbit(r)) {
43             ans = max(ans, t[r]);
44         }
45     }
46     return ans;
47 }
```

3.4 二维 BIT

```
1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5  const int N = 1005;
6  int t[N][N];
7  int n;
8
9  int lowbit(int x) {
10     return x & (-x);
11 }
```

```

void update(int x, int y, int val) {
    for(int i = x;i > 0; i -= lowbit(x)) {
        for(int j = y;j > 0; y -= lowbit(y)) {
            t[i][j] += val;
        }
    }
}

int query(int x, int y) {
    int res = 0;
    for(int i = x;i <= n; i += lowbit(x)) {
        for(int j = y;j <= n; j += lowbit(y)) {
            res += t[i][j];
        }
    }
    return res;
}

int main() {
    int m;
    cin >> n >> m;
    while(m--) {
        int opt;
        cin >> opt;
        if(opt == 1) {
            int x1, y1, x2, y2, val;
            cin >> x1 >> y1 >> x2 >> y2 >> val;
            update(x2, y2, val);
            update(x1 - 1, y2, -val);
            update(x2, y1 - 1, -val);
            update(x1 - 1, y1 - 1, val);
        }
        else {
            int x, y;
            cin >> x >> y;
            cout << query(x, y) << endl;
        }
    }
}
```

3.5 线段树

```
const int N = 1e5 + 10;

#define lc u << 1
#define rc u << 1 | 1
#define mid (t[u].l + t[u].r) / 2

struct Tree {
    int l, r;
    int sum, tag;
}t[N << 1];

void push_up(int u) {
    t[u].sum = t[lc].sum + t[rc].sum;
}

```

```
void push_down(int u) {
    if(!t[u].tag) return ;
    t[lc].sum += t[u].tag * (t[lc].r - t[lc].l + 1);
    t[rc].sum += t[u].tag * (t[rc].r - t[rc].l + 1);
    t[lc].tag += t[u].tag;
    t[rc].tag += t[u].tag;
    t[u].tag = 0;
}

void modify(int u, int ql, int qr, int v) {
    if(ql <= t[u].l && t[u].r <= qr) return ;
    push_down(u);
    if(ql <= mid) modify(lc, ql, qr, v);
    if(qr > mid)  modify(rc, ql, qr, v);
    push_up(u);
}

int query(int u, int ql, int qr) {
    if(ql <= t[u].l && t[u].r <= qr) return t[u].sum;
    push_down(u);
    int ans = 0;
    if(ql <= mid) ans += query(lc, ql, qr);
    if(qr > mid)  ans += query(rc, ql, qr);
    return ans;
}
```

3.6 二维线段树

```
#define lc u << 1
#define rc u << 1 | 1
#define m (l + r) / 2

const int N = 1e3 + 10;

struct Tree_y {
    int mx, mn;

    Tree_y operator + (const Tree_y &rhs) {
        Tree_y ans;
        ans.mx = max(mx, rhs.mx);
        ans.mn = min(mn, rhs.mn);
        return ans;
    }
};

int leafx[N], leafy[N];

struct Tree_x {
    Tree_y ty[N << 2];

    void build(int u, int l, int r) {
        ty[u].mx = -INF; ty[u].mn = INF;
        if(l == r) {
            leafy[l] = u;
            return ;
        }
        build(lc, l, m);
        build(rc, m + 1, r);
```

```
    }

    Tree_y query(int u, int l, int r, int ql, int qr) {
        if(qr < l || r < ql) return (Tree_y) {-INF, INF};
        if(ql <= l && r <= qr) return ty[u];
        return query(lc, l, m, ql, qr) + query(rc, m + 1, r, ql, qr);
    }
}tx[N << 2];

int n;

void build(int u, int l, int r) {
    tx[u].build(1, 1, n);
    if(l == r) {
        leafx[l] = u;
        return ;
    }
    build(lc, l, m);
    build(rc, m + 1, r);
}

// (x,y)单点更新，首先更新叶子节点，然后向上合并父亲节点
void modify(int x, int y, int val) {
    int valx = leafx[x];
    int valy = leafy[y];
    tx[valx].ty[valy].mn = tx[valx].ty[valy].mx = val;
    for(int i = valx; i; i >>= 1) {
        for(int j = valy; j; j >>= 1) {
            if(i == valx && j == valy) continue ;
            if(j == valy) {
                // 如果当前更新的列就是需要更新的叶子节点，那么由当前行的两个儿子节点来更新
                tx[i].ty[j] = tx[i << 1].ty[j] + tx[i << 1 | 1].ty[j];
            }
            else {
                tx[i].ty[j] = tx[i].ty[j << 1] + tx[i].ty[j << 1 | 1];
            }
        }
    }
}

Tree_y query(int u, int l, int r, int ql, int qr, int qx, int qy) {
    if(qr < l || r < ql) return (Tree_y) {-INF, INF};
    if(ql <= l && r <= qr) return tx[u].query(1, 1, n, qx, qy);
    return query(lc, l, m, ql, qr, qx, qy) + query(rc, m + 1, r, ql, qr, qx, qy);
}
```

3.7 权值线段树

```
// 权值线段树，相当于一个桶，每个节点用来表示一个区间的数***出现的次数***。

#include <bits/stdc++.h>

using namespace std;

#define lc u << 1
#define rc u << 1 | 1
#define m (l + r) / 2
#define mid (t[u].l + t[u].r) / 2
```

```

const int N = 1e5 + 10;

int t[N << 2];

int a[N];

void push_up(int u) {
    t[u] = t[lc] + t[rc];
}

void build(int u, int l, int r) {
    if(l == r) {
        t[u] = a[l]; // a[l]表示数为l的个数
        return ;
    }
    build(lc, l, m);
    build(rc, m + 1, r);
    push_up(u);
}

void update(int u, int l, int r, int k, int cnt) { // k这个数的个数增加cnt
    if(l == r) {
        t[u] += cnt;
        return ;
    }
    if(k <= m)
        update(lc, l, m, k, cnt);
    else
        update(rc, m + 1, r, k, cnt);
    push_up(u);
}

int query(int u, int l, int r, int k) { // 查询k这个数的个数
    if(l == r) {
        return t[u];
    }
    if(k <= m)
        return query(lc, l, m, k);
    else
        return query(rc, m + 1, r, k);
}

int k_max_th(int u, int l, int r, int k) { // 查询第k大的值
    if(l == r) {
        return l;
    }
    if(t[lc] >= k) return k_max_th(lc, l, m, k);
    else return k_max_th(rc, m + 1, r, k - t[lc]);
}
```

3.8 线段树动态开点合并分裂

```
#define mid (l+r)/2
static const int MAX_N = N * 40;
int rt[N], now;
int lc[MAX_N], rc[MAX_N];
ll sum[MAX_N];
```

```
int tot, rub[MAX_N];
int newNode() { return rub[0] ? rub[rub[0]--] : ++tot; }
void remove(int &u) {
    lc[u] = rc[u] = sum[u] = 0;
    rub[++rub[0]] = u;
    u = 0;
}
void push_up(int u) { sum[u] = sum[lc[u]] + sum[rc[u]]; }
void build(int &u, int l, int r) {
    u = newNode();
    if (l == r) {
        sum[u] = cnt[l];
        return;
    }
    build(lc[u], l, mid); build(rc[u], mid + 1, r);
    push_up(u);
}
void update(int &u, int l, int r, int p, ll k) {
    if (!u) u = newNode();
    if (l == r) {
        sum[u] += k;
        return;
    }
    if (p <= mid) update(lc[u], l, mid, p, k);
    else update(rc[u], mid + 1, r, p, k);
    push_up(u);
}
ll querySum(int u, int l, int r, int ql, int qr) {
    if (!u) return 0;
    if (ql <= l && r <= qr) return sum[u];
    ll res = 0;
    if (ql <= mid) res += querySum(lc[u], l, mid, ql, qr);
    if (qr > mid) res += querySum(rc[u], mid + 1, r, ql, qr);
    return res;
}
int queryKth(int u, int l, int r, ll k) {
    if (l == r) return l;
    if (k <= sum[lc[u]]) return queryKth(lc[u], l, mid, k);
    else return queryKth(rc[u], mid + 1, r, k - sum[lc[u]]);
}
void merge(int u, int v, int l, int r) {
    if (l == r) {
        sum[u] += sum[v];
        return;
    }
    if (lc[u] && lc[v]) merge(lc[u], lc[v], l, mid), remove(lc[v]);
    else if (lc[v]) lc[u] = lc[v], lc[v] = 0;
    if (rc[u] && rc[v]) merge(rc[u], rc[v], mid + 1, r), remove(rc[v]);
    else if (rc[v]) rc[u] = rc[v], rc[v] = 0;
    push_up(u);
}
void split(int &newp, int &u, int l, int r, int ql, int qr) {//分裂出[ql, qr]间的点
    if (!u) return;
    if (ql <= l && r <= qr) {
        newp = u;
        u = 0;
        return;
    }
    if (!newp) newp = newNode();
```

```
65      if (ql <= mid) split(lc[newp], lc[u], l, mid, ql, qr);
66      if (qr > mid) split(rc[newp], rc[u], mid + 1, r, ql, qr);
67      push_up(u);
68      push_up(newp);
69
70  }
71  #undef mid
```

3.9 线段树维护 LIS 方案数

```
1   // 线段树维护序列总LIS的长度mx.fi和方案数mx.se
2   // 以及对于每个点，可以存在于多少个LIS种
3
4   // https://nanti.jisuanke.com/t/39611
5
6   namespace Tree_LIS {
7       const int N = 1e6 + 10;
8
9   #define lc t[u].l
10  #define rc t[u].r
11  #define mid (l + r) / 2
12
13      struct Tree {
14          int l, r;
15          ll len; // 长度
16          ll sum; // 个数
17      }t[N << 2];
18
19      int root, cnt;
20      void init() {
21          mem(t, 0);
22          cnt = root = 0;
23      }
24
25      void push_up(int u) {
26          if(t[lc].len == t[rc].len) {
27              t[u].len = t[lc].len;
28              t[u].sum = (t[lc].sum + t[rc].sum) % mod;
29          }
30          else if(t[lc].len < t[rc].len) {
31              t[u].len = t[rc].len;
32              t[u].sum = t[rc].sum;
33          }
34          else {
35              t[u].len = t[lc].len;
36              t[u].sum = t[lc].sum;
37          }
38      }
39
40      void modify(int &u, int l, int r, int p, int le, int su) {
41          if(!u) u = ++cnt;
42          if(l == r) {
43              if(t[u].len == le) t[u].sum = (t[u].sum + su) % mod;
44              else if(t[u].len < le) {
45                  t[u].len = le;
46                  t[u].sum = su;
47              }
48              return ;
```

```
        }
        if(!lc) lc = ++cnt;
        if(!rc) rc = ++cnt;
        if(p <= mid) modify(lc, l, mid, p, le, su);
        else modify(rc, mid + 1, r, p, le, su);
        push_up(u);
    }

    pll query(int u, int l, int r, int ql, int qr) {
        if(ql <= l && r  <= qr) return pll{t[u].len, t[u].sum};
        pll lson = {0, 0}, rson = {0, 0};
        if(!lc) lc = ++cnt;
        if(!rc) rc = ++cnt;
        if(ql <= mid) lson = query(lc, l, mid, ql, qr);
        if(qr  > mid) rson = query(rc, mid + 1, r, ql, qr);
        if(lson.fi == rson.fi) return pll{lson.fi, (lson.se + rson.se) % mod};
        else if(lson.fi < rson.fi) return rson;
        else return lson;
    }
};

using namespace Tree_LIS;

ll quick_pow(ll a, ll b) {
    ll ans = 1;
    while(b) {
        if(b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans % mod;
}

void solve() {
    int n; cin >> n;
    int L = 0, R = 1e9 + 7;
    vector<int> a(n + 1);
    vector<pll> l(n + 1), r(n + 1);
    for(int i = 1;i <= n; i++) cin >> a[i];
    init();
    modify(root, L, R, a[1], 1, 1);
    l[1] = {1, 1};
    for(int i = 2;i <= n; i++) {
        pll temp = query(root, L, R, 0, a[i] - 1);
        if(temp.fi == 0) temp = {0, 1};
        modify(root, L, R, a[i], temp.fi + 1, temp.se);
        l[i] = {temp.fi + 1, temp.se};
    }
    pll mx = query(root, L, R, 0, R);
    init();
    modify(root, L, R, R - a[n], 1, 1);
    r[n] = {1, 1};
    for(int i = n - 1;i >= 1; i--) {
        pll temp = query(root, L, R, 0, R - a[i] - 1);
        if(temp.fi == 0) temp = {0, 1};
        modify(root, L, R, R - a[i], temp.fi + 1, temp.se);
        r[i] = {temp.fi + 1, temp.se};
    }
    for(int i = 1;i <= n; i++) {
```

```
108         if(r[i].fi + l[i].fi - 1 == mx.fi) {
109             cout << (r[i].se * l[i].se % mod * quick_pow(mx.se, mod - 2) % mod + mod) %
     mod << " ";
110         }
111         else cout << 0 << " ";
112     }
113     cout << endl;
114 }
```

3.10 线段树维护最小字典序 LIS

```
1  // 线段树维护LIS输出字典序最小的路径
2  const int N = 1e5 + 10;
3
4  #define lc u << 1
5  #define rc u << 1 | 1
6  #define mid (t[u].l + t[u].r) / 2
7
8  struct Tree {
9      int l, r;
10     int mx;
11     int id;
12 }t[N << 2];
13
14 inline void push_up(int u) {
15     if (t[lc].mx > t[rc].mx)
16         t[u].mx = t[lc].mx, t[u].id = t[lc].id;
17     else if (t[lc].mx < t[rc].mx)
18         t[u].mx = t[rc].mx, t[u].id = t[rc].id;
19     else
20         t[u].mx = t[lc].mx, t[u].id = min(t[lc].id, t[rc].id);
21 }
22
23 void build(int u, int l, int r) {
24     t[u].l = l;
25     t[u].r = r;
26     t[u].mx = t[u].id = 0;
27     if (l == r)
28         return;
29     int m = (l + r) >> 1;
30     build(lc, l, m);
31     build(rc, m + 1, r);
32     push_up(u);
33 }
34
35 void modify(int u, int ql, int qr, int val, int id) {
36     if (ql <= t[u].l && t[u].r <= qr) {
37         if (t[u].mx < val || (t[u].mx == val && t[u].id > id)) {
38             t[u].mx = val;
39             t[u].id = id;
40         }
41         return;
42     }
43     if (ql <= mid)
44         modify(lc, ql, qr, val, id);
45     if (qr > mid)
46         modify(rc, ql, qr, val, id);
47     push_up(u);
```

```
}

pii query(int u, int ql, int qr) {
    if (ql <= t[u].l && t[u].r <= qr)
        return pii{t[u].mx, t[u].id};
    pii lson = {-1, -1}, rson = {-1, -1};
    if (ql <= mid)
        lson = query(lc, ql, qr);
    if (qr > mid)
        rson = query(rc, ql, qr);
    if (lson.x > rson.x)
        return lson;
    else if (lson.x < rson.x)
        return rson;
    else
        return {lson.x, min(lson.y, rson.y)};
}

void solve() {
    int n;
    cin >> n;
    assert(1 <= n && n <= 1e5);
    build(1, 1, 1e5);
    vector<int> a(n + 1), ans(n + 1), fa(n + 1);
    pii res = {0, 0};
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        assert(1 <= a[i] && a[i] <= 1e5);
        if (a[i] == 1) {
            fa[i] = 0;
            ans[i] = 1;
            modify(1, a[i], a[i], 1, i);
            continue;
        }
        pii temp = query(1, 1, a[i] - 1);
        ans[i] = temp.x + 1;
        fa[i] = temp.y;
        modify(1, a[i], a[i], ans[i], i);
        if (res.x < ans[i])
            res = pii{ans[i], i};
    }
    vector<int> v;
    int tt = res.second;
    while (tt) {
        v.push_back(tt);
        tt = fa[tt];
    }
    cout << v.size() << endl;
    for (int i = v.size() - 1; i >= 0; i--) {
        cout << v[i] << (i == 0 ? endl : " ");
    }
}


// 线段树维护LIS方案数
```

3.11 线段树维护插队问题

```cpp
// n个人，每个人a_i要顺序坐在pos_i，问最终的序列如何
// 最后一个人一定坐在自己喜欢坐的位置，去掉该位置，倒数第二个人成为最后一个人，所以就是查找空位置的第pos位置

const int N = 4e5 + 10;

#define lc u << 1
#define rc u << 1 | 1
#define mid (l + r) / 2
int sum[N << 2], ans[N];

void push_up(int u) {
    sum[u] = sum[lc] + sum[rc];
}

void build(int u, int l, int r) {
    if(l == r) {
        sum[u] = 1;
        ans[l] = 0;
        return ;
    }
    build(lc, l, mid);
    build(rc, mid + 1, r);
    push_up(u);
}

void modify(int u, int l, int r, int k, int val) {
    if(l == r) {
        ans[l] = val;
        sum[u] = 0;
        return ;
    }
    if(sum[lc] >= k) modify(lc, l, mid, k, val);
    else modify(rc, mid + 1, r, k - sum[lc], val);
    push_up(u);
}

void solve() {
    int n;
    while(~scanf("%d", &n)) {
        vector<pii> p(n + 1);
        for(int i = 1;i <= n; i++) {
            scanf("%d%d",&p[i].fi, &p[i].se);
        }
        build(1, 1, n);
        for(int i = n;i >= 1; i--) {
            modify(1, 1, n, p[i].fi + 1, p[i].se);
        }
        for(int i = 1;i <= n; i++) {
            printf("%d ", ans[i]);
        }
        printf("\n");
    }
}
```

3.12 线段树维护连续区间异或值

```
// 当[l,r]^x时，很不幸异或出来的结果不一定连续，而是分成多个连续区间，所以需要用线段树来构造一个区间异或x之后还是连续区间

// [0, 7]可以分成[0, 3]和[4, 7]，这样区间异或x还是连续区间

// 主要操作:

/*
把低pos位全为0
int ql = (l ^ val) & (((1 << 30) - 1) ^ (1 << pos) - 1);
int qr = ql + (1 << pos) - 1;
把低pos位全为1

高pos不管
*/

vector<pii> g[N], len;
int l[N], r[N];

void modify(int pos, int l, int r, int L, int R, int val) {
    if(L <= l && r <= R) {
        // 把低pos设置为0
        int ql = (l ^ val) & (((1 << 30) - 1) ^ (1 << pos) - 1);
        int qr = ql + (1 << pos) - 1;
        len.push_back(pii{ql, qr});
        return ;
    }
    int mid = (l + r) / 2;
    if(L <= mid) modify(pos - 1, l, mid, L, R, val);
    if(R  > mid) modify(pos - 1, mid + 1, r, L, R, val);
}

void dfs(int u, int fa, int w) {
    modify(30, 0, (1 << 30) - 1, l[u], r[u], w);
    for(auto e : g[u]) {
        if(e.fi == fa) continue ;
        dfs(e.fi, u, e.se ^ w);
    }
}
```

3.13 线段树维护区间异或

```
const int N = 2e5 + 10;

#define lc u << 1
#define rc u << 1 | 1

struct Tree {
    int sum, tag;
}t[21][N << 2];
int a[N];

void push_up(int id, int u) {
    t[id][u].sum = t[id][lc].sum + t[id][rc].sum;
}

void push_down(int id, int u, int l, int r) {
    if(!t[id][u].tag) return ;
```

```
17      int m = (l + r) / 2;
18      t[id][lc].sum = (m - l + 1) - t[id][lc].sum;
19      t[id][rc].sum = (r - m) - t[id][rc].sum;
20      t[id][lc].tag ^= 1;
21      t[id][rc].tag ^= 1;
22      t[id][u].tag = 0;
23  }
24
25  void build(int id, int u, int l, int r) {
26      if(l == r) {
27          t[id][u].sum = (a[l] >> id) & 1;
28          t[id][u].tag = 0;
29          return ;
30      }
31      int m = (l + r) / 2;
32      build(id, lc, l, m);
33      build(id, rc, m + 1, r);
34      push_up(id, u);
35  }
36
37  void modify(int id, int u, int l, int r, int ql, int qr) {
38      if(ql <= l && r <= qr) {
39          t[id][u].sum = (r - l + 1) - t[id][u].sum;
40          t[id][u].tag ^= 1;
41          return ;
42      }
43      push_down(id, u, l, r);
44      int m = (l + r) / 2;
45      if(ql <= m) modify(id, lc, l, m, ql, qr);
46      if(qr  > m) modify(id, rc, m + 1, r, ql, qr);
47      push_up(id, u);
48  }
49
50  int query(int id, int u, int l, int r, int ql, int qr) {
51      if(ql <= l && r <= qr) return t[id][u].sum;
52      push_down(id, u, l, r);
53      int ans = 0;
54      int m = (l + r) / 2;
55      if(ql <= m) ans += query(id, lc, l, m, ql, qr);
56      if(qr  > m) ans += query(id, rc, m + 1, r, ql, qr);
57      return ans;
58  }
59
60  void solve() {
61      int n, m; cin >> n >> m;
62      for(int i = 1;i <= n; i++) cin >> a[i];
63      for(int i = 0;i <= 20; i++) {
64          build(i, 1, 1, n);
65      }
66      while(m--) {
67          int opt; cin >> opt;
68          if(opt == 1) {
69              int l, r; cin >> l >> r;
70              ll ans = 0;
71              for(int i = 0;i <= 20; i++) {
72                  ans += query(i, 1, 1, n, l, r) * (1ll << i);
73              }
74              cout << ans << endl;
75          }
```

```
        else {
            int l, r, k; cin >> l >> r >> k;
            for(int i = 0;i <= 20; i++) {
                if((k >> i) & 1) modify(i, 1, 1, n, l, r);
            }
        }
    }
}
```

3.14 树状数组维护逆序对

```
BITree t;
int n;
pii a[N];

void solve() {
    t.init(n);
    for (int i = 1; i <= n; i++) {
        int x;
        cin >> x;
        a[i] = make_pair(x, i);
    }
    sort(a + 1, a + n + 1);
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        t.change(a[i].second, 1);
        ans += (i - t.query(a[i].second));
    }
    cout << ans << endl;
}
```

3.15 ST

```
// 倍增思想加DP优化

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

const int N = 1e5 + 10;

int a[N];
int dp[N][105];

// 一维ST表
void One_ST(){
    int n;
    cin >> n;
    for(int i = 1;i <= n; i++) {
        cin >> a[i];
        dp[i][0] = a[i];

        // cin >> dp[i][0];
    }

    for(int j = 1;j <= log2(n); j++) {
        for(int i = 1;i <= n; i++) {
            if(i + (1 << (j - 1)) <= n)
```

```
                dp[i][j] = max(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
        }
    }

    int T;
    cin >> T;
    while(T--) {
        int l, r;
        cin >> l >> r;
        int len = log2(r - l + 1);
        int k = max(dp[l][len], dp[r - (1 << len) + 1][len]);
        cout << k << endl;
    }
}

// 二维ST表
int f[105][105][105];

void Two_ST() {
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= n; i++)
        for(int j = 1;j <= m; j++){
            cin >> a[i];
            f[i][j][0] = a[i];
        }

// 大矩阵分成四个小矩阵求最值

    for(int k = 1;k < log2(n); k++) {
        for(int i = 1;i <= n; i++) {
            for(int j = 1;j <= m; j++) {
                if((i + (1 << (k - 1)) <= n) && (j + (1 << (k - 1)) <= m))
                    f[i][j][k] = max(max(f[i][j + (1 << (k - 1))][k - 1], f[i + (1 << (
    k - 1))][j][k - 1]), max(f[i][j][k - 1], f[i + (1 << (k - 1))][j + (1 << (k - 1))][
    k - 1]));
        }
    }

    int T;
    cin >> T;
    while(T--) {
        int l, r, k;
        cin >> l >> r >> k;
        int len = log2(k);
        int s = max(max(f[l][r][len], f[l + k - (1 << len)][r + k - (1 << len)][len]),
    max(f[l + k - (1 << len)][r][len], f[l][r + k - (1 << len)][len]));
    }

}
```

3.16 笛卡尔树

```
// 笛卡尔树是一种由数列构造的特殊二叉搜索树，每个节点都有两个键值，first为下标，second为权值
// 笛卡尔树满足两个性质，在下标递增的情况下就是一个大/小根堆

// 笛卡尔树，静态建树，区间最值跳转
struct CartesianTree {
```

```
    int rt; // 根节点
    pii ch[N]; // 左右儿子
    int st[N]; // 单调栈

    void build(int n, int p[]) {
        rt = 0;
        int t = 0;
        for (int i = 1; i <= n; i++) {
            ch[i] = {0, 0};
            // 决定了大于还是小于
            while (t && p[st[t]] > p[i]) --t;
            if (t) {
                // 上一个点的右儿子作为自己的左儿子
                // 成为上一个点的右儿子
                ch[i].first = ch[st[t]].second;
                ch[st[t]].second = i;
            } else { // 自己作为根节点
                ch[i].first = rt;
                rt = i;
            }
            st[++t] = i;
        }
    }
} dika;
```

3.17 DancingLinks

```
// Dancing Links
struct DLX {
    int n, m, size;
    int U[MaxNode], D[MaxNode], L[MaxNode], R[MaxNode], Row[MaxNode], Col[MaxNode];
    int H[MaxN], S[MaxM];
    int ansd, ans[MaxN];

    void init(int _n, int _m) {
        n = _n;
        m = _m;
        for (int i = 0; i <= m; i++) {
            S[i] = 0;
            U[i] = D[i] = i;
            L[i] = i - 1;
            R[i] = i + 1;
        }
        R[m] = 0;
        L[0] = m;
        size = m;
        for (int i = 0; i <= n; i++) {
            H[i] = -1;
        }
    }

    void Link(int r, int c) {
        ++S[Col[++size] = c];
        Row[size] = r;
        D[size] = D[c];
        U[D[c]] = size;
        U[size] = c;
        D[c] = size;
```

```
32          if (H[r] < 0) {
33              H[r] = L[size] = R[size] = size;
34          } else {
35              R[size] = R[H[r]];
36              L[R[H[r]]] = size;
37              L[size] = H[r];
38              R[H[r]] = size;
39          }
40      }
41
42      void remove(int c) {
43          L[R[c]] = L[c];
44          R[L[c]] = R[c];
45          for (int i = D[c]; i != c; i = D[i]) {
46              for (int j = R[i]; j != i; j = R[j]) {
47                  U[D[j]] = U[j];
48                  D[U[j]] = D[j];
49                  --S[Col[j]];
50              }
51          }
52      };
53
54      void resume(int c) {
55          for (int i = U[c]; i != c; i = U[i])
56              for (int j = L[i]; j != i; j = L[j])
57                  ++S[Col[U[D[j]] = D[U[j]] = j]];
58          L[R[c]] = R[L[c]] = c;
59      }
60
61      bool Dance(int d) {
62          if (R[0] == 0) {
63              for (int i = 0; i < d; i++) {
64                  printf("%d%c", ans[i], " \n"[i == d - 1]);
65              }
66              return true;
67          }
68          int c = R[0];
69          for (int i = R[0]; i != 0; i = R[i]) if (S[i] < S[c]) c = i;
70          remove(c);
71          for (int i = D[c]; i != c; i = D[i]) {
72              ans[d] = Row[i];
73              for (int j = R[i]; j != i; j = R[j])remove(Col[j]);
74              if (Dance(d + 1))return true;
75              for (int j = L[i]; j != i; j = L[j])resume(Col[j]);
76          }
77          resume(c);
78          return false;
79      }
80  };
```

3.18 珂朵莉树

```
1   // 要先Split右端点(r+1)，在Split左端点(l)
2
3   #include <bits/stdc++.h>
4
5   using namespace std;
6
```

```
7  typedef long long ll;
8
9  struct node {
10     int l, r;
11     mutable ll v;
12     node (int L, int R = -1, ll V = 0) : l(L), r(R), v(V) {}
13     bool operator < (const node &rhs) const {
14         return l < rhs.l;
15     }
16 };
17
18 set<node> s;
19
20 auto Split(int pos) {
21     auto it = s.lower_bound(node(pos));
22     if(it != s.end() && it -> l == pos) return it;
23     --it;
24     int L = it -> l, R  = it -> r;
25     ll V = it -> v;
26     s.erase(it);
27     s.insert(node(L, pos - 1, V));
28     return s.insert(node(pos, R, V)).first;
29 }
30
31 void assign_val(int l, int r, ll val) { // 推平操作
32     auto itr = Split(r + 1);
33     auto itl = Split(l);
34     s.erase(itl, itr);
35     s.insert(node(l, r, val));
36 }
37
38 void add(int l, int r, ll val) { // 区间加
39     auto itr = Split(r + 1);
40     auto itl = Split(l);
41     for( ;itl != itr; ++itl) {
42         itl -> v += val;
43     }
44 }
45
46 ll kth(int l, int r, int k) { // 区间第k小
47     vector<pair<ll, int> > v;
48     auto itr = Split(r + 1);
49     auto itl = Split(l);
50     for( ;itl != itr; ++itl) {
51         v.push_back(pair<ll, int>{itl -> v, itl -> r - itl -> l + 1});
52     }
53     sort(v.begin(), v.end());
54     for(auto it = v.begin();it != v.end(); ++it) {
55         k -= it -> second;
56         if(k <= 0) return it -> first;
57     }
58 }
59
60 ll quick_pow(ll a, ll b, ll p) ;
61
62 ll qpow(int l, int r, int ex, int p) {
63     auto itr = Split(r + 1);
64     auto itl = Split(l);
65     ll ans = 0;
```

```
66      for( ;itl != itr; ++itl)
67          ans = (ans + ll(itl -> r - itl -> l + 1) * quick_pow(itl -> v, ll(ex), ll(p)) %
        ll(p)) % ll(p);
68      return ans % ll(p);
69  }
70  int main() {
71      int n, m;
72      cin >> n >> m;
73      for (int i = 1; i <= n; ++i){
74          ll x;
75          cin >> x;
76          s.insert(node(i,i,x));
77      }
78      s.insert(node(n + 1, n + 1, 0));
79      while(m--) {
80          int opt, l, r, x;
81          cin >> opt >> l >> r >> x;
82          if(opt == 1) add(l, r, x);
83          else if(opt == 2) assign_val(l, r, x);
84          else if(opt == 3) cout << kth(l, r, x)  << endl;
85          else {
86              int y;
87              cin >> y;
88              cout << qpow(l, r, x, y) << endl;
89          }
90      }
91  }
```

3.19 单调队列

```
1   const int N = 2e5 + 10;
2   int f[N], que[N], tail, head;
3
4   void solve() {
5       int n, k; cin >> n >> k;
6       vector<int> a(n + 1);
7       for(int i = 1;i <= n; i++) cin >> a[i];
8       // 维护最小值
9       tail = 1; head = 0;
10      for(int i = 1;i <= n; i++) {
11          while(tail >= head && f[tail] >= a[i]) tail--;
12          que[++tail] = i;
13          f[tail] = a[i];
14          while(que[head] + k <= i) head++;
15          if(i >= k) cout << f[head] << " ";
16      }
17      cout << endl;
18
19      // 维护最大值
20      tail = 1; head = 0;
21      for(int i = 1;i <= n; i++) {
22          while(tail >= head && f[tail] <= a[i]) tail--;
23          que[++tail] = i;
24          f[tail] = a[i];
25          while(que[head] + k <= i) head++;
26          if(i >= k) cout << f[head] << " ";
27      }
28      cout << endl;
```

```
}
```

3.20 单调栈

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

struct Monotone_stack {
    static const int N = 1e5 + 10;
    int a[N];
    stack<int> s;
    int n;

    void read() {
        cin >> n;
        for(int i = 1;i <= n; i++) cin >> a[i];
    }

    void Monotone_min() {
        for(int i = 1;i <= n; i++) {
            if(s.empty() || s.top() >= a[i])
                s.push(a[i]);
            else {
                while(!s.empty() && s.top() < a[i]) {
                    cout << s.top() << endl;
                    s.pop();
                }
                s.push(a[i]);
            }
        }
        while(!s.empty()) {
            cout << s.top() << endl;
            s.pop();
        }
    }

    void Monotone_max() {
        for(int i = 1;i <= n; i++) {
            if(s.empty() || s.top() <= a[i])
                s.push(a[i]);
            else {
                while(!s.empty() && s.top() > a[i]) {
                    cout << s.top() << endl;
                    s.pop();
                }
                s.push(a[i]);
            }
        }
        while(!s.empty()) {
            cout << s.top() << endl;
            s.pop();
        }
    }

}Worker;
```

3.21 差分

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

using namespace std;

/*--------------------一维差分---------------------*/

//可以区间修改，更改的值必须一样
int f[1005]; // 原始数组
int d[1005]; // 差分数组，原始数组的相邻元素之间的差值

int n;

void init() {
    for(int i = 1;i <= n; i++) {
        d[i] = f[i] - f[i - 1];
    }
}

void update(int l, int r, int k) {
    d[l] += k;
    d[r + 1] -= k;
}

void merge() {
    for(int i = 1;i <= n; i++) {
        d[i] = d[i] + d[i - 1];
    }
}

/*---------------二维差分------------------*/
const int N = 1005;

int mp[N][N];
int p[N][N];
int n, m;

void init() {

    for(int i = 1;i <= n; i++) {
        for(int j = 1;j <= m; j++) {
            p[i][j] = mp[i][j] - mp[i - 1][j] - mp[i][j - 1] + mp[i - 1][j - 1];
        }
    }
}

void update(int x1, int y1, int x2, int y2, int k) {
    p[x1][y1] += k;
    p[x1][y2 + 1] -= k;
    p[x2 + 1][y1] -= k;
    p[x2 + 1][y2 + 1] += k;
}

void merge() {

    for(int i = 1;i <= n; i++) {
        for(int j = 1;j <= m; j++) {
            p[i][j] = p[i][j] + p[i - 1][j] + p[i][j - 1] - p[i - 1][j - 1];
        }
    }
```

```
}

// 求区间交
// len存的是各线段长度
int Interval_crossing() {
    vector<pii> res;
    for(auto item : len) {
        res.push_back(pii{item.fi, 1});
        res.push_back(pii{item.se + 1, -1});
    }
    sort(res.begin(), res.end());
    int tag = 0, lst = 0, ans = 0;
    for(auto i : res) {
        if(tag == n) ans += i.fi - lst;
        tag += i.se;
        lst = i.fi;
    }
    return ans;
}
```

3.22 trie

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

const int N = 1e6 + 10;

int cnt[N], idx;
int trie[N][26];

void insert(string s) {
    int len = s.length();
    int root = 0;
    for(int i = 0;i < len; i++) {
        int v = s[i] - 'a';
        if(!trie[root][v]) trie[root][v] =++idx;
        root = trie[root][v];
    }
    cnt[root]++;
}

int query(string s) {
    int ans = 0;
    int len = s.length();
    int root = 0;
    for(int i = 0;i < len; i++) {
        int v = s[i] - 'a';
        if(!trie[root][v]) break;
        root = trie[root][v];
        ans += cnt[root];
    }
    return ans;
}

void solve()
{
    int n, m;
    scanf("%d%d",&n,&m);
```

```
    for(int i = 1;i <= n; i++) {
        char s[N];
        scanf("%s",s);
        insert(s);
    }
    while(m--) {
        char s[N];
        scanf("%s",s);
        printf("%d\n",query(s));
    }
}
```

3.23 HashTable

```
struct HashTable {
    static const int MOD = 1e7 + 10;
    struct edge {
        int nxt;
        ll num, val;
    } e[MOD];
    int head[MOD], tot;
    void clear() { tot = 0; memset(head, 0, sizeof(head)); }
    void insert(ll u, ll w) { e[++tot] = edge{head[u % MOD], u, w }, head[u % MOD] =
    tot; }
    int find(ll u) {
        for (int i = head[u % MOD]; i; i = e[i].nxt)
            if (e[i].num == u) return e[i].val;
        return -1;
    }
} hs;
```

3.24 2-4 维前缀和

```
// 统计(a,b)到(c,d)这个矩阵中的所有0子矩阵

const int N = 50 + 10;
int sum[N][N];
int Q[N][N][N][N];

void solve() {
    int n, m, q; cin >> n >> m >> q;
    for(int i = 1;i <= n; i++) {
        string s; cin >> s;
        for(int j = 1;j <= m; j++) {
            sum[i][j] = (s[j - 1] - '0') + sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j
     - 1];
        }
    }

    for(int a = 1;a <= n; a++) {
        for(int b = 1;b <= m; b++) {
            for(int c = a;c <= n; c++) {
                for(int d = b;d <= m; d++) {
                    if(sum[c][d] - sum[a - 1][d] - sum[c][b - 1] + sum[a - 1][b - 1] ==
     0) {
                        Q[a][b][c][d]++;
                    }
```

```
                }
            }
        }
    }

    for(int a = n;a >= 1; a--) {
        for(int b = m;b >= 1; b--) {
            for(int c = 1;c <= n; c++) {
                for(int d = 1;d <= m; d++) {
                    Q[a][b][c][d] += Q[a + 1][b][c][d];
                }
            }
        }
    }

    for(int a = n;a >= 1; a--) {
        for(int b = m;b >= 1; b--) {
            for(int c = 1;c <= n; c++) {
                for(int d = 1;d <= m; d++) {
                    Q[a][b][c][d] += Q[a][b + 1][c][d];
                }
            }
        }
    }

    for(int a = n;a >= 1; a--) {
        for(int b = m;b >= 1; b--) {
            for(int c = 1;c <= n; c++) {
                for(int d = 1;d <= m; d++) {
                    Q[a][b][c][d] += Q[a][b][c - 1][d];
                }
            }
        }
    }

    for(int a = n;a >= 1; a--) {
        for(int b = m;b >= 1; b--) {
            for(int c = 1;c <= n; c++) {
                for(int d = 1;d <= m; d++) {
                    Q[a][b][c][d] += Q[a][b][c][d - 1];
                }
            }
        }
    }

    while(q--) {
        int a, b, c, d; cin >> a >> b >> c >> d;
        cout << Q[a][b][c][d] << endl;
    }
}
```

3.25 普通并查集

```
const int N = 1e5 + 10;

int f[N], dep[N];

inline int find(int x) {
```

```
    return f[x] == x ? x : f[x] = find(f[x]);
}

inline void merge(int x, int y) {
    int u = find(x);
    int v = find(y);
    if(u == v) return ;
    else {
        if(dep[u] <= dep[v]) { f[u] = v; dep[v] = max(dep[v], dep[u] + 1); }
        else { f[v] = u; dep[u] = max(dep[u], dep[v] + 1); }
    }
}
```

3.26 带权并查集

```
// 根据题目时候并成链还是树
const int N = 1e5 + 10;
int d[N], f[N];

int find(int x) {
    if(f[x] != x) {
        int fa = f[x];
        f[x] = find(f[x]);
        d[x] += d[fa];
    }
    return f[x];
}

void merge(int x, int y, int v) { // 两个点和权值
    int px = find(x);
    int py = find(y);
    if(px == py) return ;
    else {
        f[px] = py;
        d[px] = d[y] + v - d[x];
    }
}

int main() {
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= n; i++) f[i] = i;
    while(m--) {
        int x, y, v;
        cin >> x >> y >> v;
        merge(x, y, v);
    }
}
```

3.27 可撤销并查集

```
struct node {
    int x, y, z;
};

struct UnionFind {
private:
```

```
    int rk[N], pre[N], siz[N], totNode;//N为最大点数
    stack<node> st;//node记录上次修改的内容
public:
    void init(int tot) {
        totNode = tot;
        for (int i = 1; i <= totNode; i++)
            pre[i] = i, siz[i] = rk[i] = 1;
    }
    int find(int x) { while (x ^ pre[x]) x = pre[x]; return x; }
    void merge(int x, int y) {//按秩合并
        x = find(x), y = find(y);
        if (x == y) return;
        if (rk[x] < rk[y]) swap(x, y);
        st.push(node{ y, rk[x], siz[y] });
        pre[y] = x, rk[x] += rk[x] == rk[y], siz[x] += siz[y];
    }
    int start() { return st.size(); }
    void end(int last) {//撤回merge操作
        while (st.size() > last) {
            node tp = st.top();
            rk[pre[tp.x]] -= tp.y, siz[pre[tp.x]] -= tp.z;
            pre[tp.x] = tp.x;
            st.pop();
        }
    }
    bool judge() { return siz[find(1)] == totNode; }
} uf;
```

3.28 种类并查集

```
// 根据题目要求，若是集合中有n个关系，那需要开n倍的并查集大小

const int N = 1e5 + 10;
int f[N * 2], dep[N * 2];

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

void merge(int x, int y) {
    x = find(x);
    y = find(y);
    if(x == y) return ;
    if(dep[x] <= dep[y]) {
        f[x] = y; dep[y] = max(dep[x] + 1, dep[y]);
    }
    else {
        f[y] = x; dep[x] = max(dep[y] + 1, dep[x]);
    }
}

int main() {
    int n, q;
    cin >> n >> q;
    for(int i = 1;i <= 2 * n; i++) f[i] = i;// **

    while(q--) {
        int flag, x, y;
        cin >> flag >> x >> y;
        if(flag) { // 敌人
```

```
            merge(x + n, y);
            merge(y + n, x);
        }
        else {
            merge(x, y); // 同伴
        }
    }
    int ans = 0;
    for(int i = 1;i <= n; i++) {
        if(f[i] == i) ans ++;
    }
    cout << ans << endl;
}
```

3.29 KD 求矩阵权值和

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

const int MAX = 200005;
const double alpha = 0.75;
//查包含在x1,y1,x2,y2为左下角和右上角的矩形里面权值之和
//K-D Tree 二维划分树
int n, ans, rt, WD, tot, top, rub[MAX];

struct node {
    int x[2], w;
} p[MAX];

struct K_D_tree {
    int ls, rs, siz, mn[2], mx[2], sum;
    //mn[0], mx[0] -> x的取值范围
    //mn[1], mx[1] -> y的取值范围
    node tmp;
} t[MAX];

int operator < (const node &a, const node &b) { return a.x[WD] < b.x[WD]; }

int newnode() {
    if (top) return rub[top--];
    else return ++tot;
}

void push_up(int u) {
    for (int i = 0; i <= 1; i++) {//更新x, y的取值范围
        t[u].mn[i] = t[u].mx[i] = t[u].tmp.x[i];
        if (t[u].ls) {//左子树的最大最小值
            t[u].mn[i] = min(t[u].mn[i], t[t[u].ls].mn[i]);
            t[u].mx[i] = max(t[u].mx[i], t[t[u].ls].mx[i]);
        }
        if (t[u].rs) {//右子树的最大最小值
            t[u].mn[i] = min(t[u].mn[i], t[t[u].rs].mn[i]);
            t[u].mx[i] = max(t[u].mx[i], t[t[u].rs].mx[i]);
        }
    }
    t[u].sum = t[t[u].ls].sum + t[t[u].rs].sum + t[u].tmp.w;
    t[u].siz = t[t[u].ls].siz + t[t[u].rs].siz + 1;
}
```

```
43
44  int build(int l, int r, int wd) {
45      if (l > r) return 0;
46      int u = newnode();
47      int m = (l + r) >> 1;
48      WD = wd; nth_element(p + l, p + m, p + r + 1);
49      t[u].tmp = p[m];
50      t[u].ls = build(l, m - 1, wd ^ 1);
51      t[u].rs = build(m + 1, r, wd ^ 1);
52      push_up(u);
53      return u;
54  }
55
56  void pia(int u, int num) {//拍扁回炉重做
57      if (t[u].ls) pia(t[u].ls, num);
58      p[t[t[u].ls].siz + num + 1] = t[u].tmp, rub[++top] = u;
59      if (t[u].rs) pia(t[u].rs, t[t[u].ls].siz + num + 1);
60  }
61
62  void check(int &u, int wd) {//检查是否平衡，不平衡则需要重建
63      if (t[u].siz * alpha < t[t[u].ls].siz || t[u].siz * alpha < t[t[u].rs].siz) pia(u,
        0), u = build(1, t[u].siz, wd);
64  }
65
66  void insert(int &u, node tp, int wd) {//插入点
67      if (!u) {
68          u = newnode();
69          t[u].ls = t[u].rs = 0, t[u].tmp = tp;
70          push_up(u);
71          return;
72      }
73      if (tp.x[wd] <= t[u].tmp.x[wd]) insert(t[u].ls, tp, wd ^ 1);
74      else insert(t[u].rs, tp, wd ^ 1);
75      push_up(u);
76      check(u, wd);
77  }
78
79  bool in(int x1, int y1, int x2, int y2, int X1, int Y1, int X2, int Y2) {//完全被包含
80      return (x1 <= X1 && X2 <= x2 && y1 <= Y1 && Y2 <= y2);
81  }
82
83  bool out(int x1, int y1, int x2, int y2, int X1, int Y1, int X2, int Y2) {//完全无交集
84      return (x1 > X2 || x2 < X1 || y1 > Y2 || y2 < Y1);
85  }
86
87  int query(int u, int x1, int y1, int x2, int y2) {
88      if (!u) return 0;
89      int res = 0;
90      if (in(x1, y1, x2, y2, t[u].mn[0], t[u].mn[1], t[u].mx[0], t[u].mx[1])) return t[u
        ].sum;
91      if (out(x1, y1, x2, y2, t[u].mn[0], t[u].mn[1], t[u].mx[0], t[u].mx[1])) return 0;
92      if (in(x1, y1, x2, y2, t[u].tmp.x[0], t[u].tmp.x[1], t[u].tmp.x[0], t[u].tmp.x[1]))
         res += t[u].tmp.w;
93      res += query(t[u].ls, x1, y1, x2, y2) + query(t[u].rs, x1, y1, x2, y2);
94      return res;
95  }
96
97  void init() {
98      ans = rt = top = tot = 0;
```

```
}


void solve() {
    scanf("%d", &n);
    while(1) {
        int opt; scanf("%d",&opt);
        if(opt == 1) {
            int x, y, w; scanf("%d%d%d",&x,&y,&w);
            insert(rt, node{x ^ ans, y ^ ans, w ^ ans}, 0);
        }
        else if(opt == 2) {
            int x1, y1, x2, y2; scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            ans = query(rt, x1 ^ ans, y1 ^ ans, x2 ^ ans, y2 ^ ans);
            printf("%d\n",ans);
        }
        else break;
    }
}
```

3.30 KD 求最近点对距离

```
// 二维平面里最近点对距离
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const int N = 1e5 + 10;
struct node {
    ll x[2];
}a[N], b[N];

int now, n;
ll ans;
map<pair<ll, ll>, int> mp;

bool cmp(node a, node b) { return a.x[now] < b.x[now]; }

ll sqr(int x) { return (ll)x * x; }
ll dis(node a, node b) { return sqr(a.x[0] - b.x[0]) + sqr(a.x[1] - b.x[1]); }

void build(int l, int r, int d) {
    if(l >= r) return ;
    int m = (l + r) >> 1;
    now = d;
    nth_element(a + l, a + m, a + r, cmp);
    build(l, m, d ^ 1);
    build(m + 1, r, d ^ 1);
}

void query(int l, int r, int d, node p) {
    if(l >= r) return ;
    int m = (l + r) >> 1;
    int ql = l, qr = m;
    ll res = dis(a[m], p);
    if(ans == 0 || res && ans > res) ans = res;
    if(p.x[d] > a[m].x[d]) ql = m + 1, qr = r;
    query(ql, qr, d ^ 1, p);
```

```
    if(ans > sqr(a[m].x[d] - p.x[d]))
        query(l + m - ql + 1, m + r - qr, d ^ 1, p);
}

void solve() {
    scanf("%d",&n);
    ll sum = 5e18;
    for(int i = 0;i < n; i++) {
        scanf("%lld %lld",&a[i].x[0],&a[i].x[1]);
        if(mp[{a[i].x[0], a[i].x[1]}]) sum = 0;
        else mp[{a[i].x[0], a[i].x[1]}]++;
        b[i] = a[i];
    }
    build(0, n, 0);
    for(int i = 0;i < n; i++) {
        ans = 0;
        query(0, n, 0, b[i]);
        sum = min(ans, sum);
    }
    printf("%.4lf\n",sqrt(1.0 * sum));
}
```

3.31 CDQ 分治

```
// 原问题：“任意两个元素之间的贡献”

// 降维成的子问题：“左段元素对右段每一个元素的贡献”

// 分而治之，最基础的应用为归并排序
// 将一个区间分成两个区间[l,mid]和[mid+1,r]，然后排序使其有序
// 注意事项：
// 1、离散化
// 2、如果有多个点的参量完全相同，由于顺着添加，查出来的最后一个答案才是正确的
// 对于y排序，我们可以直接对两段区间Sort，然而归并排序本身就是分治，我们可以在CDQ的过程中进行归并排序，要比Sort少一个log

// 千万不要忘记离散化！！！！！！！！！！！！！！！！
```

3.32 cdq 处理逆序数

```
const int N = 1e5 + 10;
int ans[N], cnt[N];

struct star {
    int x, y, id;
}a[N], tmp[N];

bool cmp(star a, star b) {
    if(a.y == b.y) return a.x < b.x;
    return a.y < b.y;
}

void cdq(int l, int r) {
    if(l == r) return ;
    int m = (l + r) / 2;
    cdq(l, m);
    cdq(m + 1, r);
```

```
    int p = l, q = m + 1;
    for(int i = l;i <= r; i++) {
        if((p <= m && a[p].x <= a[q].x) || q > r) {
            tmp[i] = a[p++];
        }
        else {
            ans[a[q].id] += p - l;
            tmp[i] = a[q++];
        }
    }
    for(int i = l;i <= r; i++) a[i] = tmp[i];
}

void solve() {
    int n; cin >> n;
    for(int i = 1;i <= n; i++) cin >> a[i].x >> a[i].y, a[i].id = i;
    sort(a + 1, a + n + 1, cmp);
    cdq(1, n);
    for(int i = 1;i <= n; i++) cnt[ans[i]]++;
    for(int i = 0;i < n; i++) cout << cnt[i] << endl;
}
```

3.33 cdq 处理二维偏序

```

const int N = 1e5 + 10;
int ans[N], cnt[N];

struct star {
    int x, y, id;
}a[N], tmp[N];

bool cmp(star a, star b) {
    if(a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}

void cdq(int l, int r) {
    if(l == r) return ;
    int m = (l + r) / 2;
    cdq(l, m);
    cdq(m + 1, r);
    int p = l, q = m + 1;
    for(int i = l;i <= r; i++) {
        if((p <= m && a[p].x <= a[q].x) || q > r) {
            tmp[i] = a[p++];
        }
        else {
            ans[a[q].id] += i - l;
            tmp[i] = a[q++];
        }
    }
    for(int i = l;i <= r; i++) a[i] = tmp[i];
}

void solve() {
    int n; cin >> n;
    for(int i = 1;i <= n; i++) cin >> a[i].x >> a[i].y;
```

```
    sort(a + 1, a + n + 1, cmp);
    cdq(1, n);
    for(int i = 1;i <= n; i++) cnt[ans[i]]++;
    for(int i = 0;i < n; i++) cout << cnt[i] << endl;
}
```

3.34 cdq 套 cdq 处理三维偏序

```
// 一维排序、二维cdq、三维树状数组

const int N = 1e5 + 10;
struct node {
    int x, y, z;
    int id;
    int tag;
    bool operator < (const node &a)const{
        if(x != a.x) return x < a.x;
        if(y != a.y) return y < a.y;
        return z < a.z;
    }
    bool operator == (const node &a)const{
        return x == a.x && y == a.y && z == a.z;
    }
}a[N], b[N], tmp[N];

int ans[N];
int n;

void cdq2(int l, int r) {
    if(l == r) return ;
    int mid = (l + r) / 2;
    cdq2(l, mid); cdq2(mid + 1, r);
    int p = l, q = mid + 1, cnt = 0;
    for(int i = l;i <= r; i++) {
        if(q > r || (p <= mid && b[p].z <= b[q].z)) {
            if(b[p].tag == 0) cnt++;
            tmp[i] = b[p++];
        }
        else {
            if(b[q].tag == 1) ans[b[q].id] += cnt;
            tmp[i] = b[q++];
        }
    }
    for(int i = l;i <= r; i++) b[i] = tmp[i];
}

void cdq1(int l, int r) {
    if(l == r) return ;
    int mid = (l + r) / 2;
    cdq1(l, mid); cdq1(mid + 1, r);
    int p = l, q = mid + 1;
    /* 因为是计算左端元素对右端元素的影响，所以需要打个标记tag来记录他是左端还是右端元素 */
    for(int i = l;i <= r; i++) {
        if(q > r || (p <= mid && a[p].y <= a[q].y)) {
            a[p].tag = 0;
            b[i] = a[p++];
        }
        else {
```

```
            a[q].tag = 1;
            b[i] = a[q++];
        }
    }
    for(int i = l;i <= r; i++) a[i] = b[i];
    cdq2(l, r);
}

void solve() {
    n = read();
    for(int i = 1;i <= n; i++) {
        a[i].x = read(), a[i].y = read(), a[i].z = read();
        a[i].id = i;
    }
    sort(a + 1, a + n + 1);
    for(int i = n - 1;i >= 1; i--) {
        if(a[i + 1] == a[i]) ans[a[i].id] = ans[a[i + 1].id] + 1;
    }
    cdq1(1, n);
    for(int i = 1;i <= n; i++) cout << ans[i] << endl;
}
```

3.35 cdq 套树状数组处理三维偏序

```
// 一维排序、二维cdq、三维树状数组

const int N = 1e5 + 10;
struct node {
    int x, y, z;
    int id;
    bool operator < (const node &a)const{
        if(x != a.x) return x < a.x;
        if(y != a.y) return y < a.y;
        return z < a.z;
    }
    bool operator == (const node &a)const{
        return x == a.x && y == a.y && z == a.z;
    }
}a[N], b[N];
int n;


int ans[N];

struct BIT {
#define lowbit(x) (x & (-x))
    int n;
    int t[N];

    void init(int _n) {
        mem(t, 0);
        n = _n;
    }

    void update(int x, int val) {
        while (x <= n) {
            t[x] += val;
            x += lowbit(x);
```

```
        }
    }

    int query(int x) {
        int ans = 0;
        while (x) {
            ans += t[x];
            x -= lowbit(x);
        }
        return ans;
    }
}bit;

void cdq(int l, int r) {
    if(l == r) return ;
    int mid = (l + r) / 2;
    cdq(l, mid);
    cdq(mid + 1, r);
    int p = l, q = mid + 1;
    for(int i = l;i <= r; i++) {
        if(q > r || (p <= mid && a[p].y <= a[q].y)) {
            bit.update(a[p].z, 1);
            b[i] = a[p++];
        }
        else {
            ans[a[q].id] += bit.query(a[q].z);
            b[i] = a[q++];
        }
    }
    for(int i = l;i <= mid; i++) bit.update(a[i].z, -1);
    for(int i = l;i <= r; i++) a[i] = b[i];
}

void solve() {
    n = read();
    int mx = 0;
    for(int i = 1;i <= n; i++) {
        a[i].x = read(), a[i].y = read(), a[i].z = read();
        a[i].id = i;
        mx = max(mx, a[i].z);
    }
    bit.init(mx);
    sort(a + 1, a + n + 1);
    for(int i = n - 1;i >= 1; i--) {
        if(a[i] == a[i + 1]) ans[a[i].id] = ans[a[i + 1].id] + 1;
    }
    cdq(1, n);
    for(int i = 1;i <= n; i++) cout << ans[i] << endl;
}
```

3.36 cdq 维护矩阵内二维数点

```
// 求二维平面上(x1,y1)到(x2,y2)的矩阵中数点

// 利用前缀和思想，把问题划分成[x2,y2] - [x1-1,y2] - [x2,y2-1] + [x1-1,y1-1]

// 所有要建立4个虚点为查询点，而原本实点为修改点

```

```
7  const int N = 3e6 + 10;
8
9  struct node {
10     int x, y, opt, id;
11     // opt为操作类型，1为修改，0为查询
12     bool operator < (const node& o) const {
13         return x == o.x ? (y == o.y ? opt : y < o.y) : x < o.x;
14     }
15     // 注意排序顺序，坐标相同时，要使opt放前面，因为要先修改
16
17     bool operator == (const node &o) const {
18         return x == o.x && y == o.y;
19     }
20 }a[N], tmp[N];
21
22 int ans[N];
23
24 void cdq(int l, int r) {
25     if(l == r) return ;
26     int mid = (l + r) / 2;
27     cdq(l, mid); cdq(mid + 1, r);
28     int p = l, q = mid + 1, cnt = 0;
29     for(int i = l;i <= r; i++) {
30         if(q > r || (p <= mid && a[p].y <= a[q].y)) {
31             cnt += a[p].opt;
32             tmp[i] = a[p++];
33         }
34         else {
35             ans[a[q].id] += cnt;
36             tmp[i] = a[q++];
37         }
38     }
39     for(int i = l;i <= r; i++) a[i] = tmp[i];
40 }
41
42 void solve() {
43     int n = read(), m = read();
44     for(int i = 1;i <= n; i++) {
45         a[i].x = read(), a[i].y = read(), a[i].opt = 1;
46     }
47     int _n = 0;
48     for(int i = 1;i <= m; i++) {
49         int x1 = read(), y1 = read(), x2 = read(), y2 = read();
50         a[++n] = (node){x2, y2, 0, ++_n};
51         a[++n] = (node){x2, y1 - 1, 0, ++_n};
52         a[++n] = (node){x1 - 1, y2, 0, ++_n};
53         a[++n] = (node){x1 - 1, y1 - 1, 0, ++_n};
54     }
55     sort(a + 1, a + n + 1);
56     cdq(1, n);
57     for(int i = 1;i + 3 <= _n; i += 4) {
58         cout << ans[i] - ans[i + 1] - ans[i + 2] + ans[i + 3] << endl;
59     }
60 }
```

3.37 可持久化 01trie

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

```
2  using namespace std;
3
4  const int N = 2e7 + 10;
5
6  int t[N][2];
7  int cnt, root[N], sz[N][2];
8  int a[N];
9
10 void insert(int pre, int &now, int i, int x) {
11     if(i < 0) return ;
12     now = ++cnt;
13     int d = x >> i & 1;
14     t[now][d ^ 1] = t[pre][d ^ 1];
15     sz[now][d ^ 1] = sz[pre][d ^ 1]; sz[now][d] = sz[pre][d] + 1;
16     insert(t[pre][d], t[now][d], i - 1, x);
17 }
18
19 int query(int l, int r, int i, int x) {
20     if(i < 0) return 0;
21     int d = x >> i & 1;
22     int tmp = sz[r][d ^ 1] - sz[l][d ^ 1];
23     if(tmp > 0) return query(t[l][d ^ 1], t[r][d ^ 1], i - 1, x) + (1 << i);
24     else return query(t[l][d], t[r][d], i - 1, x);
25 }
26
27 int main() {
28     int n, m;
29     cin >> n >> m;
30     for(int i = 1;i <= n; i++) {
31         int x;
32         cin >> x;
33         insert(root[i - 1], root[i], 30, x);
34     }
35     while(m--) {
36         int l, r, x;
37         cin >> l >> r >> x;
38         cout << query(root[l - 1], root[r], 30, x) << endl;
39     }
40 }
```

3.38 可持久化数组

```
1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5  const int N = 1e5 + 10;
6
7  struct Node {
8      int l, r, val;
9  }hjt[N * 40];
10 int cnt, root[N];
11 int a[N];
12
13 inline void build(int &now, int l, int r) {
14     now = ++cnt;
15     if(l == r) {
16         hjt[now].val = a[l];
```

```
17          return ;
18      }
19      int m = (l + r) >> 1;
20      build(hjt[now].l, l, m);
21      build(hjt[now].r, m + 1, r);
22  }
23
24  inline void modify(int ver, int &now, int l, int r, int pos, int value) {
25      hjt[now = ++cnt] = hjt[ver];
26      if(l == r) {
27          hjt[now].val = value;
28          return ;
29      }
30      int m = (l + r) >> 1;
31      if(pos <= m) modify(hjt[ver].l, hjt[now].l, l, m, pos, value);
32      else modify(hjt[ver].r, hjt[now].r, m + 1, r, pos, value);
33  }
34
35  inline int query(int now, int l, int r, int pos) {
36      if(l == r) return hjt[now].val;
37      int m = (l + r) >> 1;
38      if(pos <= m) return query(hjt[now].l, l, m, pos);
39      else return query(hjt[now].r, m + 1, r, pos);
40  }
41
42  int main() {
43      int n, m;
44      cin >> n >> m;
45      for(int i = 1;i <= n; i++) cin >> a[i];
46      build(root[0], 1, n);
47      for(int i = 1;i <= m; i++) {
48          int ver, opt;
49          cin >> ver >> opt;
50          if(opt == 1) {
51              int pos, value;
52              cin >> pos >> value;
53              modify(root[ver], root[i], 1, n, pos, value);
54          }
55          else {
56              int pos;
57              cin >> pos;
58              root[i] = root[ver];
59              cout << query(root[i], 1, n, pos) << endl;
60          }
61      }
62  }
```

3.39 可持久化并查集

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 1e5 + 10;
5
6  struct Node {
7      int l, r, val;
8  }hjt[N * 40 * 2];
9
```

```
int cnt, rootfa[N], rootdep[N], tot;
int n;

inline void build(int &now, int l, int r) {
    now = ++cnt;
    if(l == r) {
        hjt[now].val = ++tot;
        return ;
    }
    int m = (l + r) >> 1;
    build(hjt[now].l, l, m);
    build(hjt[now].r, m + 1, r);
}

inline void modify(int ver, int &now, int l, int r, int pos, int value) {
    hjt[now = ++cnt] = hjt[ver];
    if(l == r) {
        hjt[now].val = value;
        return ;
    }
    int m = (l + r) >> 1;
    if(pos <= m) modify(hjt[ver].l, hjt[now].l, l, m, pos, value);
    else modify(hjt[ver].r, hjt[now].r, m + 1, r, pos, value);
}

inline int query(int now, int l, int r, int pos) {
    if(l == r) return hjt[now].val;
    int m = (l + r) >> 1;
    if(pos <= m) return query(hjt[now].l, l, m, pos);
    else return query(hjt[now].r, m + 1, r, pos);
}

inline int find(int ver, int x) {
    int fx = query(rootfa[ver], 1, n, x);
    return fx == x ? x : find(ver, fx);
}

inline void merge(int ver, int x, int y) {
    x = find(ver - 1, x);
    y = find(ver - 1, y);
    if(x == y) {
        rootfa[ver] = rootfa[ver - 1];
        rootdep[ver] = rootdep[ver - 1];
    }
    else {
        int depx = query(rootdep[ver - 1], 1, n, x);
        int depy = query(rootdep[ver - 1], 1, n, y);
        if(depx < depy) {
            modify(rootfa[ver - 1], rootfa[ver], 1, n, x, y);
            rootdep[ver] = rootdep[ver - 1];
        }
        else if(depx > depy) {
            modify(rootfa[ver - 1], rootfa[ver], 1, n, y, x);
            rootdep[ver] = rootdep[ver - 1];
        }
        else {
            modify(rootfa[ver - 1], rootfa[ver], 1, n, x, y);
            modify(rootdep[ver - 1], rootdep[ver], 1, n, y, depy + 1);
        }
```

```
    }
}

int main() {
    int m;
    cin >> n >> m;
    build(rootfa[0], 1, n);
    for(int ver = 1;ver <= m; ver++) {
        int opt, x, y;
        cin >> opt;
        if(opt == 1) {
            cin >> x >> y;
            merge(ver, x, y);
        }
        else if(opt == 2) {
            cin >> x;
            rootfa[ver] = rootfa[x];
            rootdep[ver] = rootdep[x];
        }
        else {
            cin >> x >> y;
            rootfa[ver] = rootfa[ver - 1];
            rootdep[ver] = rootdep[ver - 1];
            int u = find(ver, x);
            int v = find(ver, y);
            if(u == v) cout << 1 << endl;
            else cout << 0 << endl;
        }
    }
}
```

3.40 扫描线求面积并

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

const int N = 2e5 + 10;

#define lc u << 1
#define rc u << 1 | 1
#define mid (t[u].l + t[u].r) >> 1

int n, cnt;

double v[N];
struct L {
    double x, y1, y2;
    int state;
    bool operator < (L rhs) {return x < rhs.x; }
}line[N << 2];

struct Node {
    int l, r, cover;
    double len;
}t[N << 2];

inline void push_up(int u) {
```

```
26      if(t[u].cover) t[u].len = v[t[u].r + 1] - v[t[u].l];
27      else if(t[u].l == t[u].r) t[u].len = 0;
28      else t[u].len = t[lc].len + t[rc].len;
29  }
30
31  void build(int u, int l, int r) {
32      t[u].l = l; t[u].r = r;
33      if(l == r) {
34          t[l].cover = t[l].len = 0;
35          return ;
36      }
37      int m = (l + r) >> 1;
38      build(lc, l, m);
39      build(rc, m + 1, r);
40  }
41
42  void modify(int u, int ql, int qr, int state) {
43      if(ql <= t[u].l && t[u].r <= qr) {
44          t[u].cover += state;
45          push_up(u);
46          return ;
47      }
48      if(ql <= mid) modify(lc, ql, qr, state);
49      if(qr > mid)  modify(rc, ql, qr, state);
50      push_up(u);
51  }
52
53  void init() {
54      cin >> n;
55      for(int i = 1;i <= n; i++) {
56          double x1, y1, x2, y2;
57          scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
58          line[i] = L{x1, y1, y2, 1}; v[i] = y1;
59          line[n + i] = L{x2, y1, y2, -1}; v[n + i] = y2;
60      }
61      n <<= 1;
62      sort(line + 1, line + n + 1);
63      sort(v + 1, v + n + 1);
64      cnt = unique(v + 1, v + n + 1) - (v + 1);
65      build(1, 1, cnt);
66  }
67
68  void solve() {
69      double ans = 0;
70      for(int i = 1;i <= n; i++) {
71          int ql = lower_bound(v + 1, v + cnt + 1, line[i].y1) - v;
72          int qr = lower_bound(v + 1, v + cnt + 1, line[i].y2) - v - 1;
73          modify(1, ql, qr, line[i].state);
74          ans += t[1].len * (line[i + 1].x - line[i].x);
75      }
76      cout << ans << endl;
77  }
78
79  int main() {
80      init();
81      solve();
82  }
```

3.41 扫描线求周长并

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

const int N = 2e5 + 10;

#define INF 0x3fffff
#define lc u << 1
#define rc u << 1 | 1
#define mid (t[u].l + t[u].r) >> 1

int n;

struct L {
    int y, x1, x2;
    int state;
    bool operator < (L rhs) {return y < rhs.y; }
}line[N << 2];

struct Node {
    int l, r, cover;
    bool ls, rs;
    int num;
    int len;
}t[N << 2];

inline void push_up(int u) {
    if(t[u].cover) {
        t[u].len = t[u].r - t[u].l + 1;
        t[u].ls = t[u].rs = 1;
        t[u].num = 1;
    }
    else if(t[u].l == t[u].r) {
        t[u].ls = t[u].rs = 0;
        t[u].len = t[u].num = 0;
    }
    else {
        t[u].len = t[lc].len + t[rc].len;
        t[u].ls = t[lc].ls; t[u].rs = t[rc].rs;
        t[u].num = t[lc].num + t[rc].num - (t[lc].rs & t[rc].ls);
    }
}

void build(int u, int l, int r) {
    t[u].l = l; t[u].r = r;
    if(l == r) {
        t[u].len = t[u].cover = t[u].ls = t[u].rs = t[u].num = 0;
        return ;
    }
    int m = (l + r) >> 1;
    build(lc, l, m);
    build(rc, m + 1, r);
}

void modify(int u, int ql, int qr, int state) {
    if(ql <= t[u].l && t[u].r <= qr) {
        t[u].cover += state;
```

```
58          push_up(u);
59          return ;
60      }
61      if(ql <= mid) modify(lc, ql, qr, state);
62      if(qr > mid)  modify(rc, ql, qr, state);
63      push_up(u);
64  }
65
66  void init() {
67      cin >> n;
68      int mx = -INF, mn = INF;
69      for (int i = 1; i <= n; i++) {
70          int x1, x2, y1, y2;
71          cin >> x1 >> y1 >> x2 >> y2;
72          mx = max(mx, max(x1, x2));
73          mn = min(mn, min(x1, x2));
74          line[i] = L{y1, x1, x2, 1};
75          line[n + i] = L{y2, x1, x2, -1};
76      }
77      n <<= 1;
78      sort(line + 1, line + n + 1);
79      build(1, mn, mx);
80  }
81
82  void solve() {
83      int ans = 0;
84      int last = 0;
85      for(int i = 1;i <= n; i++) {
86          modify(1, line[i].x1, line[i].x2 - 1, line[i].state);
87          ans += abs(t[1].len - last); // 横线
88          ans += (line[i + 1].y - line[i].y) * 2 * t[1].num; // 竖线
89          last = t[1].len;
90      }
91      printf("%d\n",ans);
92  }
93
94  int main() {
95      init();
96      solve();
97  }
```

3.42 区间第 k 小\

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   const int N = 1e5 + 10;
5
6   vector<int> v;
7   struct Node {
8       int l, r;
9       int val;
10  }hjt[N * 40];
11  int root[N], cnt;
12
13  int get_id(int x) { return lower_bound(v.begin(), v.end(), x) - v.begin() + 1; }
14
15  void insert(int pre, int &now, int l, int r, int p) {
```

```
    hjt[++cnt] = hjt[pre];
    now = cnt;
    hjt[now].val++;
    if(l == r) return ;
    int m = (l + r) >> 1;
    if(p <= m) insert(hjt[pre].l, hjt[now].l, l, m, p);
    else insert(hjt[pre].r, hjt[now].r, m + 1, r, p);
}

int query(int L, int R, int l, int r, int k) {
    if(l == r) return l;
    int m = (l + r) >> 1;
    int tmp = hjt[hjt[R].l].val - hjt[hjt[L].l].val;
    if(k <= tmp) return query(hjt[L].l, hjt[R].l, l, m, k);
    else return query(hjt[L].r, hjt[R].r, m + 1, r, k - tmp);
}

int main() {
    int n, q;
    cin >> n >> q;
    vector<int> a(n + 1);
    for(int i = 1;i <= n; i++) { cin >> a[i]; v.push_back(a[i]); }
    sort(v.begin(), v.end());
    v.erase(unique(v.begin(), v.end()), v.end());

    for(int i = 1;i <= n; i++) {
        insert(root[i - 1], root[i], 1, n, get_id(a[i]));
    }

    while(q--) {
        int l, r, k;
        cin >> l >> r >> k;
        cout << v[query(root[l - 1], root[r], 1, n, k) - 1] << endl;
    }
}
```

3.43 区间前 k 大

```

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

typedef long long ll;
const int N = 1e6 + 10;
int a[N];
vector<int> v;
int cnt, root[N];

struct Node {
    int l, r;
    ll sum;
    int num;
    int val;
}hjt[N * 40];

int getid(int x) { return lower_bound(v.begin(), v.end(), x) - v.begin() + 1; }

void insert(int pre, int &now, int l, int r, int p, int val) {
```

```
    now = ++cnt;
    hjt[now] = hjt[pre];
    hjt[now].num++; hjt[now].sum += val;
    if(l == r) {
        hjt[now].val = val;
        return ;
    }
    int m = (l + r) >> 1;
    if(p <= m) insert(hjt[pre].l, hjt[now].l, l, m, p, val);
    else insert(hjt[pre].r, hjt[now].r, m + 1, r, p, val);
}

ll query(int L, int R, int l, int r, int k) {
    if(l == r) return hjt[R].val * k;
    int m = (l + r) >> 1;
    int tmp = hjt[hjt[R].r] .num - hjt[hjt[L].r].num;
    if(k <= tmp) return query(hjt[L].r, hjt[R].r, m + 1, r, k);
    else return hjt[hjt[R].r].sum - hjt[hjt[L].r].sum + query(hjt[L].l, hjt[R].l, l, m,
     k - tmp);
}

void init(int n) {
    v.clear();
    cnt = 0;
    for(int i = 1;i <= n; i++) {
        scanf("%d",&a[i]);
        v.push_back(a[i]); root[i] = 0;
    }
    sort(v.begin(), v.end());
    v.erase(unique(v.begin(), v.end()), v.end());
}

int main() {
    int _;
    scanf("%d",&_);
    while(_--) {
        int n;
        scanf("%d",&n);
        init(n);
        for(int i = 1;i <= n; i++) {
            insert(root[i - 1], root[i], 1, n, getid(a[i]), a[i]);
        }
        int q;
        scanf("%d",&q);
        while(q--) {
            int l, r, k;
            scanf("%d%d%d",&l,&r,&k);
            int t = r - l + 1;
            ll ans = query(root[l - 1], root[r], 1, n, k);
            printf("%lld\n",1ll * t * (t + 1) * (2 * t + 1) / 6 + ans);
        }
    }
}
```

3.44 树套树维护三维偏序

```
const int N = 4e6 + 10;

```

```
3  int n, k;
4  struct node {
5      int a, b, c;
6      int operator < (const node &o) const {
7          return a != o.a ? (a < o.a) : (b != o.b ? (b < o.b) : (c < o.c));
8      }
9
10     int operator == (const node &o) const {
11         return a == o.a && b == o.b && c == o.c;
12     }
13 }p[N];
14
15
16 struct Tree1 {
17     int l, r;
18 }t1[N << 2];
19
20 struct Tree2 {
21     int l, r;
22     int num;
23 }t2[N << 2];
24
25 int root, root2[N];
26 int cnt1, cnt2;
27
28 void vec_insert(int &u, int l, int r, int pos, int val) {
29     if(!u) u = ++cnt2;
30     t2[u].num += val;
31     if(l == r) return ;
32     int m = (l + r) / 2;
33     if(pos <= m) vec_insert(t2[u].l, l, m, pos, val);
34     else vec_insert(t2[u].r, m + 1, r, pos, val);
35 }
36
37 int vec_query(int u, int l, int r, int ql, int qr) {
38     if(!u) return 0;
39     if(ql <= l && r <= qr) return t2[u].num;
40     int ans = 0;
41     int mid = (l + r) / 2;
42     if(ql <= mid) ans += vec_query(t2[u].l, l, mid, ql, qr);
43     if(qr  > mid) ans += vec_query(t2[u].r, mid + 1, r, ql, qr);
44     return ans;
45 }
46
47 // 在第一维权值线段树在[1,k]根据p[x].b插入，第二维权值线段树在[1,k]根据p[x].c插入
48 void tree_insert(int &u, int l, int r, int x, int val) {
49     if(!u) u = ++cnt1;
50     vec_insert(root2[u], 1, k, p[x].c, val);
51     if(l == r) return ;
52     int m = (l + r) / 2;
53     if(p[x].b <= m) tree_insert(t1[u].l, l, m, x, val);
54     else tree_insert(t1[u].r, m + 1, r, x, val);
55 }
56
57 int tree_query(int u, int l, int r, int x) {
58     if(!u) return 0;
59     if(1 <= l && r <= p[x].b) return vec_query(root2[u], 1, k, 1, p[x].c);
60     int mid = (l + r) / 2;
61     int ans = 0;
```

```
    if(1 <= mid) ans += tree_query(t1[u].l, l, mid, x);
    if(p[x].b > mid) ans += tree_query(t1[u].r, mid + 1, r, x);
    return ans;
}

void solve() {
    cin >> n >> k;
    for(int i = 1;i <= n; i++) cin >> p[i].a >> p[i].b >> p[i].c;
    sort(p + 1, p + n + 1);
    vector<int> ans(n + 1);
    int sum = 1;
    for(int i = 1;i <= n; i++) {
        // 因为这些个都一样，如果不这样操作，会使后面的不会对前面的有贡献
        if(p[i + 1] == p[i]) {
            sum++;
            continue;
        }
        tree_insert(root, 1, k, i, sum);
        int res = tree_query(root, 1, k, i);
        ans[res] += sum;
        sum = 1;
    }
    for(int i = 1;i <= n; i++) cout << ans[i] << endl;
}
```

3.45 线段树套主席树

```
const int N = 2e5 + 10;

int n, m, l, r, a, b, num[N], Last[N], pre[N], Hash[N], ans[N][2];
int cnt, root[N], sum[N * 20], lc[N * 20], rc[N * 20];
struct Query {
    int a, b, l, id;
};

struct data {
    int a, v;

    bool operator<(const data &b) const {
        return a < b.a;
    }
} d[N];

vector<Query> q[N * 4];

void addquery(int o, int l, int r, int L, int R, Query qry) {
    if (L <= l && R >= r) {
        q[o].push_back(qry);
        return;
    }
    int mid = (l + r) / 2;
    if (L <= mid) {
        addquery(o * 2, l, mid, L, R, qry);
    }
    if (R > mid) {
        addquery(o * 2 + 1, mid + 1, r, L, R, qry);
    }
}
```

```
32
33  void build(int y, int &x, int l, int r, int k) {
34      x = ++cnt;
35      sum[x] = sum[y] + 1;
36      lc[x] = lc[y];
37      rc[x] = rc[y];
38      if (l == r) {
39          return;
40      }
41      int mid = (l + r) / 2;
42      if (k <= mid) {
43          build(lc[y], lc[x], l, mid, k);
44      } else {
45          build(rc[y], rc[x], mid + 1, r, k);
46      }
47  }
48
49  int query(int y, int x, int l, int r, int k) {
50      if (!x || l == r) {
51          return 0;
52      }
53      int mid = (l + r) / 2;
54      if (k <= mid) {
55          return query(lc[y], lc[x], l, mid, k);
56      } else {
57          return sum[lc[x]] - sum[lc[y]] + query(rc[y], rc[x], mid + 1, r, k);
58      }
59  }
60
61  void insert(int o, int l, int r) {
62      if (q[o].size()) {
63          Hash[0] = 0;
64          for (int i = l; i <= r; i++) {
65              Hash[++Hash[0]] = num[i];
66          }
67          sort(Hash + 1, Hash + Hash[0] + 1);
68          int s = 0;
69          for (int i = l; i <= r; i++) {
70              d[++s].a = lower_bound(Hash + 1, Hash + Hash[0] + 1, num[i]) - Hash;
71              d[s].v = pre[i];
72          }
73          sort(d + 1, d + s + 1);
74          for (int i = 1; i <= s; i++) {
75              build(root[i - 1], root[i], 0, n, d[i].v);
76          }
77          int a, b;
78          for (int i = 0; i < q[o].size(); i++) {
79              a = lower_bound(Hash + 1, Hash + Hash[0] + 1, q[o][i].a) - Hash;
80              b = upper_bound(Hash + 1, Hash + Hash[0] + 1, q[o][i].b) - Hash - 1;
81              ans[q[o][i].id][0] += sum[root[b]] - sum[root[a - 1]];
82              ans[q[o][i].id][1] += query(root[a - 1], root[b], 0, n, q[o][i].l);
83          }
84          memset(root, 0, sizeof(int) * (Hash[0] + 1));
85          memset(sum, 0, sizeof(int) * (cnt + 1));
86          memset(lc, 0, sizeof(int) * (cnt + 1));
87          memset(rc, 0, sizeof(int) * (cnt + 1));
88          cnt = 0;
89      }
90      if (l == r) {
```

```
91          return;
92      }
93      int mid = (l + r) / 2;
94      insert(o * 2, l, mid);
95      insert(o * 2 + 1, mid + 1, r);
96  }
97
98  void solve() {
99      int _;
100     cin >> _;
101     while (_--) {
102         for (int i = 0; i < N * 4; i++) q[i].clear();
103         cnt = 0;
104         for (int i = 0; i < N; i++) ans[i][0] = ans[i][1] = pre[i] = Last[i] = 0;
105         cin >> n >> m;
106         for (int i = 1; i <= n; i++) {
107             cin >> num[i];
108             num[i]++;
109             pre[i] = Last[num[i]];
110             Last[num[i]] = i;
111         }
112         for (int i = 1; i <= m; i++) {
113             cin >> l >> a >> r >> b;
114             a++, b++;
115             addquery(1, 1, 1e5 + 1, l, r, (Query) {a, b, l, i});
116         }
117         insert(1, 1, 1e5 + 1);
118         for (int i = 1; i <= m; i++) {
119             cout << ans[i][1] << endl;
120         }
121     }
122 }
```

3.46 Scapegoat

```
1   // 无旋转平衡，暴力拍扁重构
2
3   #include <bits/stdc++.h>
4   using namespace std;
5
6   namespace Scapegoat_Tree {
7   #define MAXN (100000 + 10)
8       const double alpha = 0.75;
9       struct Node {
10      Node * ch[2];
11      int key, size, cover; // size为有效节点的数量，cover为节点总数量
12      bool exist; // 是否存在（即是否被删除）
13      void PushUp(void) {
14          size = ch[0]->size + ch[1]->size + (int)exist;
15          cover = ch[0]->cover + ch[1]->cover + 1;
16      }
17      bool isBad(void) { // 判断是否需要重构
18          return ((ch[0]->cover > cover * alpha + 5) ||
19                  (ch[1]->cover > cover * alpha + 5));
20          }
21      };
22      struct STree {
23      protected:
```

```
24          Node mem_poor[MAXN]; //内存池，直接分配好避免动态分配内存占用时间
25          Node *tail, *root, *null; // 用null表示NULL的指针更方便，tail为内存分配指针，root为根
26          Node *bc[MAXN]; int bc_top; // 储存被删除的节点的内存地址，分配时可以再利用这些地址
27
28          Node * NewNode(int key) {
29              Node * p = bc_top ? bc[--bc_top] : tail++;
30              p->ch[0] = p->ch[1] = null;
31              p->size = p->cover = 1; p->exist = true;
32              p->key = key;
33              return p;
34          }
35          void Travel(Node * p, vector<Node *>&v) {
36              if (p == null) return;
37              Travel(p->ch[0], v);
38              if (p->exist) v.push_back(p); // 构建序列
39              else bc[bc_top++] = p; // 回收
40              Travel(p->ch[1], v);
41          }
42          Node * Divide(vector<Node *>&v, int l, int r) {
43              if (l >= r) return null;
44              int mid = (l + r) >> 1;
45              Node * p = v[mid];
46              p->ch[0] = Divide(v, l, mid);
47              p->ch[1] = Divide(v, mid + 1, r);
48              p->PushUp(); // 自底向上维护，先维护子树
49              return p;
50          }
51          void Rebuild(Node * &p) {
52              static vector<Node *>v; v.clear();
53              Travel(p, v); p = Divide(v, 0, v.size());
54          }
55          Node ** Insert(Node *&p, int val) {
56              if (p == null) {
57                  p = NewNode(val);
58                  return &null;
59              }
60              else {
61                  p->size++; p->cover++;
62
63                  // 返回值储存需要重构的位置，若子树也需要重构，本节点开始也需要重构，以本节点为根重构
64                  Node ** res = Insert(p->ch[val >= p->key], val);
65                  if (p->isBad()) res = &p;
66                  return res;
67              }
68          }
69          void Erase(Node *p, int id) {
70              p->size--;
71              int offset = p->ch[0]->size + p->exist;
72              if (p->exist && id == offset) {
73                  p->exist = false;
74                  return;
75              }
76              else {
77                  if (id <= offset) Erase(p->ch[0], id);
78                  else Erase(p->ch[1], id - offset);
79              }
80          }
81      public:
82          void Init(void) {
```

```
83              tail = mem_poor;
84              null = tail++;
85              null->ch[0] = null->ch[1] = null;
86              null->cover = null->size = null->key = 0;
87              root = null; bc_top = 0;
88          }
89          STree(void) { Init(); }
90
91          void Insert(int val) {
92              Node ** p = Insert(root, val);
93              if (*p != null) Rebuild(*p);
94          }
95          int Rank(int val) {
96              Node * now = root;
97              int ans = 1;
98              while (now != null) { // 非递归求排名
99                  if (now->key >= val) now = now->ch[0];
100                 else {
101                     ans += now->ch[0]->size + now->exist;
102                     now = now->ch[1];
103                 }
104             }
105             return ans;
106         }
107         int Kth(int k) {
108             Node * now = root;
109             while (now != null) { // 非递归求第K大
110                 if (now->ch[0]->size + 1 == k && now->exist) return now->key;
111                 else if (now->ch[0]->size >= k) now = now->ch[0];
112                 else k -= now->ch[0]->size + now->exist, now = now->ch[1];
113             }
114         }
115         void Erase(int k) {
116             Erase(root, Rank(k));
117             if (root->size < alpha * root->cover) Rebuild(root);
118         }
119         void Erase_kth(int k) {
120             Erase(root, k);
121             if (root->size < alpha * root->cover) Rebuild(root);
122         }
123     }sTree;
124 #undef MAXN
125
126 }
127
128 int main() {
129     Scapegoat_Tree::sTree.Init();
130     int _; cin >> _;
131     while(_--) {
132         int opt, x;
133         cin >> opt >> x;
134         if(opt == 1) Scapegoat_Tree::sTree.Insert(x);
135         else if(opt == 2) Scapegoat_Tree::sTree.Erase(x);
136         else if(opt == 3) cout << Scapegoat_Tree::sTree.Rank(x) << endl;
137         else if(opt == 4) cout << Scapegoat_Tree::sTree.Kth(x) << endl;
138         else if(opt == 5) cout << Scapegoat_Tree::sTree.Kth(Scapegoat_Tree::sTree.Rank(
    x) - 1) << endl;
139         else if(opt == 6) cout << Scapegoat_Tree::sTree.Kth(Scapegoat_Tree::sTree.Rank(
    x + 1)) << endl;
```

```
    }
}
```

3.47 Splay

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

const int N = 1e5 + 10;

struct Splay { // 有旋平衡树
#define rt  ch[0][1]
#define lc  ch[u][0]
#define rc  ch[u][1]
#define identify(u) (ch[fa[u]][1] == u)
    int tot, totElement;
    int val[N], cnt[N], fa[N], sum[N];
    int ch[N][2];
    Splay() { tot = totElement = 0; }
    void push_up(int u) { sum[u] = sum[lc] + sum[rc] + cnt[u]; }
    void connect(int u, int par, int son) { ch[par][son] = u, fa[u] = par; }
    void rotate(int u) {
        int fc = identify(u), f = fa[u];
        int gc = identify(f), g = fa[f];
        int uc = fc ^ 1, son = ch[u][uc];
        connect(son, f, fc);
        connect(f, u, uc);
        connect(u, g, gc);
        push_up(f);
        push_up(u);
    }
    void splay(int u, int v) {
        v = fa[v];
        while (fa[u] != v) {
            int f = fa[u];
            if (fa[f] != v)
                rotate(identify(u) ^ identify(f) ? u : f);
            rotate(u);
        }
    }
    int creat(int v, int par) {
        val[++tot] = v;
        fa[tot] = par;
        sum[tot] = cnt[tot] = 1;
        return tot;
    }
    void destory(int u) {
        val[u] = cnt[u] = fa[u] = sum[u] = lc = rc = 0;
        tot -= u == tot;
    }
    int find(int v) {
        int u = rt;
        while (1) {
            if (val[u] == v) {
                splay(u, rt);
                return u;
            }
            int nxt = v > val[u];
```

```
54              if (!ch[u][nxt]) return -1;
55              u = ch[u][nxt];
56          }
57      }
58      int insert(int v) {
59          totElement++;
60          if (totElement == 1) {
61              creat(v, 0);
62              return rt = tot;
63          }
64          int u = rt;
65          while (1) {
66              sum[u]++;
67              if (v == val[u]) {
68                  cnt[u]++;
69                  return u;
70              }
71              int nxt = v > val[u];
72              if (!ch[u][nxt]) {
73                  creat(v, u);
74                  splay(ch[u][nxt] = tot, rt);
75                  return tot;
76              }
77              u = ch[u][nxt];
78          }
79      }
80      void remove(int v) {
81          int u = find(v);
82          if (!u) return;
83          totElement--;
84          if (cnt[u] > 1) {
85              cnt[u]--, sum[u]--;
86              return;
87          }
88          if (!lc) fa[rt = rc] = 0;
89          else {
90              int now = lc;
91              while (ch[now][1]) now = ch[now][1];
92              splay(now, lc);
93              connect(rc, now, 1); connect(now, 0, 1);
94              push_up(now);
95          }
96          destory(u);
97      }
98      int getRank(int v) {
99          int k = 0, u = rt;
100         while (1) {
101             if (v == val[u]) {
102                 k += sum[lc] + 1;
103                 splay(u, rt);
104                 return k;
105             }
106             else if (v < val[u]) u = lc;
107             else {
108                 k += sum[lc] + cnt[u];
109                 u = rc;
110             }
111             if (!u) return 0;
112         }
```

```
113     }
114     int atRank(int k) {
115         if (k > totElement) return -1;
116         int u = rt;
117         while (1) {
118             if (k > sum[lc] && k <= sum[lc] + cnt[u]) {
119                 splay(u, rt);
120                 break;
121             }
122             if (k <= sum[lc]) u = lc;
123             else {
124                 k -= sum[lc] + cnt[u];
125                 u = rc;
126             }
127         }
128         return val[u];
129     }
130     int upper(int v) {
131         int u = rt, minn = 0x3f3f3f3f, p = 0;
132         while (u) {
133             if (val[u] > v && val[u] < minn) minn = val[u], p = u;
134             if (v >= val[u]) u = rc;
135             else u = lc;
136         }
137         if (!p) splay(p, rt);
138         return minn;
139     }
140     int lower(int v) {
141         int u = rt, maxx = -0x3f3f3f3f, p = 0;
142         while (u) {
143             if (val[u] < v && val[u] > maxx) maxx = val[u], p = u;
144             if (v <= val[u]) u = lc;
145             else u = rc;
146         }
147         if (!p) splay(p, rt);
148         return maxx;
149     }
150 }splay;
151
152 void solve() {
153     int _; cin >> _;
154     while(_--) {
155         int opt, x;
156         cin >> opt >> x;
157         if(opt == 1) splay.insert(x);
158         else if(opt == 2) splay.remove(x);
159         else if(opt == 3) cout << splay.getRank(x) << endl;
160         else if(opt == 4) cout << splay.atRank(x) << endl;
161         else if(opt == 5) cout << splay.lower(x) << endl;
162         else if(opt == 6) cout << splay.upper(x) << endl;
163     }
164 }
```

3.48 Splay 区间翻转

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

```
4  const int MAX = 4e5 + 10;
5  int n;
6
7  struct Splay {
8  #define rt  ch[0][1]
9  #define lc  ch[u][0]
10 #define rc  ch[u][1]
11     int tot, totElement;
12     int val[MAX], fa[MAX], sum[MAX], tag[MAX];
13     int ch[MAX][2];
14     Splay() { tot = totElement = 0; }
15     void push_up(int u) { sum[u] = sum[lc] + sum[rc] + 1; }
16     void push_down(int u) {
17         if (tag[u]) {
18             tag[lc] ^= 1;
19             tag[rc] ^= 1;
20             tag[u] = 0;
21             swap(lc, rc);
22         }
23     }
24     int identify(int u) { return ch[fa[u]][1] == u; }
25     void connect(int u, int par, int son) { ch[par][son] = u, fa[u] = par; }
26     void rotate(int u) {
27         int fc = identify(u), f = fa[u];
28         int gc = identify(f), g = fa[f];
29         int uc = fc ^ 1, son = ch[u][uc];
30         connect(son, f, fc);
31         connect(f, u, uc);
32         connect(u, g, gc);
33         push_up(f);
34         push_up(u);
35     }
36     void splay(int u, int v) {
37         v = fa[v];
38         while (fa[u] != v) {
39             int f = fa[u];
40             if (fa[f] != v)
41                 rotate(identify(u) ^ identify(f) ? u : f);
42             rotate(u);
43         }
44     }
45     int creat(int v, int par) {
46         val[++tot] = v;
47         fa[tot] = par;
48         sum[tot] = 1;
49         return tot;
50     }
51     int insert(int v) {
52         totElement++;
53         if (totElement == 1) {
54             creat(v, 0);
55             return rt = tot;
56         }
57         int u = rt;
58         while (1) {
59             sum[u]++;
60             if (v == val[u]) return u;
61             int nxt = v > val[u];
62             if (!ch[u][nxt]) {
```

```
                creat(v, u);
                splay(ch[u][nxt] = tot, rt);
                return tot;
            }
            u = ch[u][nxt];
        }
    }
    int queryKth(int k) {
        if (k > totElement) return -1;
        int u = rt;
        while (1) {
            push_down(u);
            if (k > sum[lc] && k <= sum[lc] + 1) {
                splay(u, rt);
                return val[u];
            }
            if (k <= sum[lc]) u = lc;
            else {
                k -= sum[lc] + 1;
                u = rc;
            }
        }
    }
    void reverse(int ql, int qr) {
        ql = queryKth(ql);
        qr = queryKth(qr + 2);
        splay(ql, rt);
        splay(qr, ch[ql][1]);
        tag[ch[qr][0]] ^= 1;
    }
    void DFS(int u) {
        push_down(u);
        if(lc) DFS(lc);
        if(val[u] > 1 && val[u] < n + 2)
            cout << val[u] - 1 << " ";
        if(rc) DFS(rc);
    }
} splay;

void solve() {
    int m; cin >> n >> m;
    for(int i = 1;i <= n + 2; i++) splay.insert(i);
    while(m--) {
        int l, r; cin >> l >> r;
        splay.reverse(l, r);
    }
    splay.DFS(splay.ch[0][1]);
    cout << endl;
}

/*--------------------------*/

typedef long long ll;

#define INF 0x3f3f3f3f

const int N = 1e5 + 10;

int a[N];
```

```
122 int n, cnt, root;
123
124 struct Node {
125     int ch[2];
126     int val, fa;
127     int siz, tag;
128 } t[N];
129
130 void update(int x) {
131     t[x].siz = t[t[x].ch[0]].siz + t[t[x].ch[1]].siz + 1;
132 }
133
134 void push_down(int x) {
135     if (x && t[x].tag) {
136         t[t[x].ch[0]].tag ^= 1;
137         t[t[x].ch[1]].tag ^= 1;
138         swap(t[x].ch[0], t[x].ch[1]);
139         t[x].tag = 0;
140     }
141 }
142
143 int id(int x) {
144     return x == t[t[x].fa].ch[1];
145 }
146
147 void connect(int fa, int x, int d) {
148     t[x].fa = fa;
149     t[fa].ch[d] = x;
150 }
151
152 void rotate(int x) {
153     int f = t[x].fa;
154     int ff = t[f].fa;
155     push_down(x);
156     push_down(f);
157     int fson = id(x);
158     int ffson = id(f);
159     int son = t[x].ch[fson ^ 1];
160     connect(f, son, fson);
161     connect(x, f, fson ^ 1);
162     connect(ff, x, ffson);
163     update(f), update(x);
164 }
165
166 void splay(int x, int to) { // 将 x 转到 to 的子节点位置
167     while (t[x].fa != to) {
168         int f = t[x].fa;
169         if (t[f].fa != to) {
170             rotate(id(x) == id(f) ? f : x);
171         }
172         rotate(x);
173     }
174     if (!to) { // 在 splay(l, 0) 时起作用
175         root = x;
176     }
177 }
178
179 int build(int l, int r, int fa) {
180     if (l > r) {
```

```
181         return 0;
182     }
183     int mid = (l + r) >> 1;
184     int x = ++cnt;
185     t[x].fa = fa;
186     t[x].siz = 1;
187     t[x].val = a[mid];
188     t[x].ch[0] = build(l, mid - 1, x);
189     t[x].ch[1] = build(mid + 1, r, x);
190     update(x);
191     return x;
192 }
193
194 int find(int rank) {
195     int x = root;
196     while (1) {
197         push_down(x);
198         if (rank <= t[t[x].ch[0]].siz) {
199             x = t[x].ch[0];
200         } else {
201             rank -= t[t[x].ch[0]].siz + 1;
202             if (!rank) {
203                 return x;
204             }
205             x = t[x].ch[1];
206         }
207     }
208 }
209
210 void reverse(int l, int r) {
211     l = find(l);
212     r = find(r);
213     splay(l, 0);
214     splay(r, l);
215     int pos = t[t[root].ch[1]].ch[0];
216     t[pos].tag ^= 1;
217 }
218
219 void print(int x) {
220     if (!x) {
221         return;
222     }
223     push_down(x);
224     print(t[x].ch[0]);
225     if (t[x].val != INF && t[x].val != -INF) {
226         printf("%d ", t[x].val);
227     }
228     print(t[x].ch[1]);
229 }
230
231 void solve() {
232     cin >> n >> m;
233     for(int i = 1;i <= n; i++) cin >> a[i + 1];
234     a[1] = -INF; a[n + 2] = INF;
235     root = build(1, n + 2, 0);
236     for(int i = 1;i <= m; i++) {
237         int l, r; cin >> l >> r;
238         reverse(l + 1, r + 1);
239     }
```

```
240     print(root);
241 }
```

3.49 01trie

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 1e3 * 32 + 10;
5
6  int t[N][2];
7  int num[N];
8  int idx;
9
10 void insert(ll x) {
11     int rt = 0;
12     for(int i = 32;i >= 0; i--) {
13         int v = x >> i & 1;
14         if(!t[rt][v]) {
15             t[idx][0] = t[idx][1] = 0;
16             num[idx] = 0;
17             t[rt][v] = idx++;
18         }
19         rt = t[rt][v];
20         num[rt]++;
21     }
22 }
23
24 void modify(ll x, int val) {
25     int rt = 0;
26     for(int i = 32;i >= 0; i--) {
27         int v = x >> i & 1;
28         rt = t[rt][v];
29         num[rt] += val;
30     }
31 }
32
33 ll query(ll x) {
34     ll ans = 0;
35     int rt = 0;
36     for(int i = 32;i >= 0; i--) {
37         int v = x >> i & 1;
38         if(t[rt][!v] && num[t[rt][!v]]) {
39             ans += 1 << i;
40             rt = t[rt][!v];
41         }
42         else rt = t[rt][v];
43     }
44     return ans;
45 }
46
47 // 不能有两个相同的异或，加一个num数组，表示访问次数
48
49 int main() {
50     int _ = read();
51     while(_--) {
52         idx = 1; t[0][0] = t[0][1] = 0;
53         int n = read();
```

```
54          vector<ll> a(n + 1);
55          for(int i = 1;i <= n; i++) insert(a[i] = read());
56          ll ans = 0;
57          for(int i = 1;i <= n; i++) {
58              modify(a[i], -1);
59              for(int j = i + 1;j <= n; j++) {
60                  modify(a[j], -1);
61                  ans = max(ans, query(a[i] + a[j]));
62                  modify(a[j], 1);
63              }
64              modify(a[i], 1);
65          }
66          cout << ans << endl;
67      }
68  }
```

4 博弈论

4.1 巴什博弈

```
// 一堆石头，一次只能拿m个
void BaShe() {
    int n, m;
    cin >> n >> m;
    if(n % (m + 1))
        cout << "first" << endl;
    else
        cout << "second" << endl;
}
```

4.2 斐波那契博弈

```
int fib[100001], n;
map<int, bool> p;
int main() {
    fib[1] = 1; fib[2] = 2;
    for(int i = 3;i <= 50; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
        p[fib[i]] = 1;
    }
    scanf("%d",&n);
    if(p[n]) printf("second");
    else printf("first");
}
```

4.3 尼姆博弈

```
//  尼姆博弈是巴什博弈的进化版
// 有n堆石子，两个人可以从任意一堆石子中拿任意多个石子(不能不拿)，没法拿的人失败

// 定理：当n堆石子的数量异或和等于0时，先手必胜，否则先手必败

const int N = 1e5 + 10;
int a[N];

void Nim() {
    int n, ans = 0;
    cin >> n;
    for(int i = 1;i <= n; i++) {
        cin >> a[i];
        ans ^= a[i];
    }
    if(ans == 0)
        cout << "second" << endl;
    else
        cout << "first" << endl;
}

// Auti-nim 反尼姆游戏
// 当先拿完所有石子时候输
// 当如下条件时，先手必胜
// □ ：所有堆的石子数均=1，且有偶数堆。
// □ ：至少有一个堆的石子数>1，且石子堆的异或和≠0。
```

4.4 威佐夫博弈

```
// 每次每个人可以从任意一堆石子中取任意多的石子或者从两堆石子中取同样多的石子，不能取得人输

void WZF() {
    int a, b;
    cin >> a >> b;
    if(a > b) swap(a, b);
    int temp = b - a;
    int ans = temp * (1.0 + sqrt(5.0)) / 2.0;
    if(ans == a)
        cout << "second" << endl;
    else
        cout << "first" << endl;
}

void EX_WZF() {
    int a, b, k;
    cin >> a >> b >> k;
    k++;
    if(a > b) swap(a, b);
    int temp = (b - a) / k;
    int ans1 = temp * (2 - k + sqrt(4.0 + k * k)) / 2;
    int ans2 = temp * (2 + k + sqrt(4.0 + k * k)) / 2;
    if(ans1 == a && ans2 == b)
        cout << "second" << endl;
    else
        cout << "first" << endl;
}
```

4.5 SG 函数

```
// SG函数
#define N 1001
//f[]: 可以取走的石子个数
//sg[]:0~n的SG函数值
int f[N], sg[N], mex[N];

void getSG(int n) {
    int i, j;
    memset(sg, 0, sizeof(sg));
    for (i = 1; i <= n; i++) {
        memset(mex, 0, sizeof(mex));
        for (j = 1; f[j] <= i; j++)
            mex[sg[i - f[j]]] = 1;
        for (j = 0; j <= n; j++) { //求mes{}中未出现的最小的非负整数
            if (mex[j] == 0) {
                sg[i] = j;
                break;
            }
        }
    }
}
```

5 树与森林

5.1 dp 树上直径

```cpp
// 树的最长路径

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

const int N = 2e5 + 10;

int f[N][2];
int down[N], up[N];
int ans;

vector<int> g[N];

void dfs1(int u, int fa) {
    for(auto v : g[u]) {
        if(v == fa) continue ;
        dfs1(v, u);
        if(f[v][0] + 1 > f[u][0]) {
            f[u][1] = f[u][0];
            f[u][0] = f[v][0] + 1;
            down[u] = v;
        }
        else if(f[v][0] + 1 > f[u][1]) f[u][1] = f[v][0] + 1;
    }
    ans = max(ans, f[u][0] + f[u][1]);
}

void dfs2(int u, int fa) {
    for(auto v : g[u]) {
        if(v == fa) continue ;
        up[v] = up[u] + 1;
        if(down[u] == v) up[v] = max(up[v], f[u][1] + 1);
        else up[v] = max(up[v], f[u][0] + 1);
        dfs2(v, u);
    }
}

int main() {
    int n; cin >> n;
    for(int i = 1;i < n; i++) {
        int u, v; cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    dfs1(0, -1);
    dfs2(0, -1);
    for(int i = 0;i < n; i++) {
        int d[3] = {f[i][1], f[i][0], up[i]};
        sort(d, d + 3);
        if(d[2] + d[1] == ans) cout << i << endl;
    }
}
```

5.2 重链剖分

```
const int maxn = 4e5 + 10;

struct Edge {
    int v, next;
}e[maxn << 1];

int head[maxn * 2], cnt;

inline void add(int u, int v) {
    e[++cnt].v = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

int fa[maxn], dep[maxn], siz[maxn], son[maxn];

void dfs1(int u, int par) {
    dep[u] = dep[fa[u] = par] + (siz[u] = 1);
    for(int i = head[u]; ~i; i = e[i].next) {
        int v = e[i].v;
        if(v == par) continue;
        dfs1(v, u);
        siz[u] += siz[v];
        if(!son[u] || siz[v] > siz[son[u]])
            son[u] = v;
    }
}

int dfn[maxn], top[maxn], nodeof[maxn], tim;

void dfs2(int u, int topf) {
    nodeof[dfn[u] = ++tim] = u;
    top[u] = topf;
    if(!son[u]) return ;
    dfs2(son[u], topf);
    for(int i = head[u]; ~i; i = e[i].next) {
        int v = e[i].v;
        if(v == fa[u] || v == son[u]) continue;
        dfs2(v, v);
    }
}

int w[maxn];

#define lc u << 1
#define rc u << 1 | 1
#define mid (t[u].l + t[u].r) / 2
struct Tree {
    int l, r, sum, tag;
}t[maxn << 2];
inline void push_up(int u) ;
inline void push_down(int u) ;
void build(int u, int l, int r) ;
void modify(int u, int ql, int qr, int v) ;
int query(int u, int ql, int qr) ;

void modify_chain(int x, int y, int val) {
```

```
    while(top[x] != top[y]) {
        if(dep[top[x]] < dep[top[y]]) swap(x, y);
        modify(1, dfn[top[x]], dfn[x], val);
        x = fa[top[x]];
    }
    if(dep[x] > dep[y]) swap(x, y);
    modify(1, dfn[x], dfn[y], val);
}

int query_chain(int x, int y) {
    int ans = 0;
    while(top[x] != top[y]) {
        if(dep[top[x]] < dep[top[y]]) swap(x, y);
        ans += query(1, dfn[top[x]], dfn[x]);
        x = fa[top[x]];
    }
    if(dep[x] > dep[y]) swap(x, y);
    ans += query(1, dfn[x], dfn[y]);
    return ans;
}

signed main() {
    memset(head, -1, sizeof(head));
    int n; cin >> n;
    for(int i = 1;i <= n; i++) cin >> w[i];
    for(int i = 1;i <= n - 1; i++) {
        int u, v; cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    dfs1(1, 0);
    dfs2(1, 1);
    build(1, 1, n);
    int m; cin >> m;
    while(m--) {
        int opt; cin >> opt;
        if(opt == 1) {
            int x, y, val; cin >> x >> y >> val;
            modify_chain(x, y, val);
        }
        else if(opt == 2) {
            int x, val; cin >> x >> val;
            modify(1, dfn[x], dfn[x] + siz[x] - 1, val);
        }
        else if(opt == 3) {
            int x, y; cin >> x >> y;
            cout << query_chain(x, y) << endl;
        }
        else if(opt == 4) {
            int x; cin >> x;
            cout << query(1, dfn[x], dfn[x] + siz[x] - 1) << endl;
        }
    }
}
```

5.3 倍增 LCA

```
const int N = 1e5 + 10;
```

```cpp
vector<int> g[N];

int fa[N][30];
int dep[N];

void dfs(int u, int par) {
    dep[u] = dep[fa[u][0] = par] + 1;
    for(int i = 1;i <= 20; i++) {
        fa[u][i] = fa[fa[u][i - 1]][i - 1];
    }
    for(auto v : g[u]) {
        if(v == par) continue;
        dfs(v, u);
    }
}

int LCA(int x, int y) {
    if(dep[x] < dep[y]) swap(x, y);
    for(int i = 20;i >= 0; i--) {
        if(dep[fa[x][i]] >= dep[y]) x = fa[x][i];
    }
    if(x == y) return x;
    for(int i = 20;i >= 0; i--) {
        if(fa[x][i] != fa[y][i]) {
            x = fa[x][i];
            y = fa[y][i];
        }
    }
    return fa[x][0];
}
```

5.4 树剖 LCA

```cpp
int son[N], siz[N], top[N], fa[N], dep[N];
void dfs(int u, int par) {
    dep[u] = dep[fa[u] = par] + (siz[u] = 1);
    int max_son = -1;
    for (auto &v: g[u])
        if (v != par) {
            dfs(v, u);
            siz[u] += siz[v];
            if (max_son < siz[v])
                son[u] = v, max_son = siz[v];
        }
}
void dfs2(int u, int topf) {
    top[u] = topf;
    if (!son[u]) return;
    dfs2(son[u], topf);
    for (auto &v: g[u])
        if (v != fa[u] && v != son[u]) dfs2(v, v);
}
int LCA(int x, int y) {
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        x = fa[top[x]];
    }
    return dep[x] < dep[y] ? x : y;
```

```
}
```

5.5 树的直径

```
const int N = 1e5 + 10;

struct Edge {
    int v, w;
};
vector<Edge> g[N];
int dp[N][2];
int ans;

void dfs(int u, int fa) {
    for(auto e : g[u]) {
        int v = e.v;
        if(v == fa) continue;
        dfs(v, u);
        if(dp[v][0] + e.w > dp[u][0]) {
            dp[u][1] = dp[u][0];
            dp[u][0] = dp[v][0] + e.w;
        }
        else if(dp[v][0] + e.w > dp[u][1]) dp[u][1] = dp[v][0] + e.w;
    }
    ans = max(ans, dp[u][0] + dp[u][1]);
}
```

5.6 树的重心

```
const int N = 1e5 + 10;

struct Edge {
    int v, next;
}e[N * 2];

int cnt, head[N * 2];

int d[N], R[2], root;
int n;

int balance;

inline void add(int u, int v) {
    e[++cnt].v = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

void DFS(int u, int fa) {
    d[u] = 1;
    int res = 0;
    for(int i = head[u] ; i != -1 ; i = e[i].next) {
        int v = e[i].v;
        if(v == fa) continue;
        DFS(v, u);
        d[u] += d[v];
        res = max(res, d[v]);
```

```
    }
    res = max(res, n - d[u]);
    if(res < balance) {
        R[root++] = u;
        balance = res;
    }
    else if(res == balance) {
        R[root++] = u;
    }
}

int main() {
    cin >> n;
    balance = n / 2;
    for(int i = 1;i < n; i++) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    DFS(1, 0);
    if(R[0]) cout << R[0] << endl;
    if(R[1]) cout << R[1] << endl;
}
```

5.7 树的最大匹配

```
// 设状态为f[u][1/0]表示以u为根的子树与儿子连边/不连边的最大匹配

const int N = 1e5 + 10;
vector<int> g[N];
int f[N][2];

void dfs(int u, int fa) {
    int mn = INF;
    for(auto v : g[u]) {
        if(v == fa) continue ;
        dfs(v, u);
        f[u][0] += f[v][1]; // u不与儿子连边，即加上所有与儿子连边的v
        f[u][1] += f[v][1]; // u与儿子连边，即加上一个不与儿子连边的v和其他所有与儿子连边的v
        mn = min(mx, f[v][1] - f[v][0]);
    }
    if(mn != INF) f[u][1] = dp[u][1] - mx + 1;
}
```

5.8 树分治-点分治

```
// 题意：n个节点的树，存在边权，范围1e18
// 求任意两点之间点集的子集中两点之间路径异或和为0的个数
// u<v,u'<v',(u',v') □  path(u,v),求path(u', v')异或和==0

struct Edge {
    int to, nxt;
    ll w;
};
const int N = int(1e5 + 10);
const int M = N << 1;
```

```

struct Grahp {
    int head[N];
    Edge eg[M];
    int tot;

    void init(int n) {
        memset(head, -1, sizeof(int) * ++n);
    }

    inline void addEdge(int u, int v, ll w) {
        eg[tot] = {v, head[u], w};
        head[u] = tot++;
    }
} gh;

bool vis[N];
// q队列, fa祖先, sz是子树大小, smx是子树最大
int q[N], fa[N], sz[N], smx[N];

int froot(int s) {
    int l, r, mn = N, rt = 0;
    q[l = r = 1] = s;
    while (l <= r) {
        int u = q[l++];
        sz[u] = 1;
        smx[u] = 0;
        for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
            int v = gh.eg[i].to;
            if (v == fa[u] || vis[v]) continue;
            fa[v] = u;
            q[++r] = v;
        }
    }
    // 反向遍历所有点算size
    while (--l) {
        int u = q[l];
        int mx = max(smx[u], r - sz[u]);
        if (mx < mn) mn = mx, rt = u;
        if (l == 1) break; // 根节点没有fa
        sz[fa[u]] += sz[u];
        smx[fa[u]] = max(smx[fa[u]], sz[u]);
    }
    return rt;
}

// sons子树方向节点个数, val根到该节点异或和, gc边后继方向的节点个数
int sons[N], gc[M];
ll val[N];
ll ans = 0;
int n;

const int MOD = int(1e9 + 7);

ll nums[N];
int cnt[N];

void go(int s, int rt) {
    fa[s] = rt;
```

```
70      val[s] = 0;
71      int l, r;
72      // 不计算s
73      q[l = r = 0] = s;
74      int m = 0;
75      while (l <= r) {
76          int u = q[l++];
77          nums[m++] = val[u];
78          for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
79              int v = gh.eg[i].to;
80              if (v == fa[u] || vis[v]) continue;
81              fa[v] = u;
82              q[++r] = v;
83              val[v] = val[u] ^ gh.eg[i].w;
84              // 这个点方向后面有多少点
85              sons[v] = gc[i];
86          }
87      }
88      sort(nums, nums + m);
89      m = unique(nums, nums + m) - nums;
90      mst(cnt, 0, m);
91      // 遍历分支
92      for (int j = gh.head[s]; ~j; j = gh.eg[j].nxt) {
93          // 分支的根
94          int du = gh.eg[j].to;
95          if (vis[du]) continue;
96          q[l = r = 1] = du;
97          while (l <= r) {
98              int u = q[l++];
99              int k = lower_bound(nums, nums + m, val[u]) - nums;
100             (ans += 1ll * sons[u] * cnt[k] % MOD) %= MOD;
101             if (val[u] == 0) {
102                 (ans += 1ll * sons[u] * (n - gc[j]) % MOD) %= MOD;
103             }
104             for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
105                 int v = gh.eg[i].to;
106                 if (v == fa[u] || vis[v]) continue;
107                 q[++r] = v;
108             }
109         }
110         // 增加这个方向的值
111         while (--l) {
112             int u = q[l];
113             int k = lower_bound(nums, nums + m, val[u]) - nums;
114             (cnt[k] += sons[u]) %= MOD;
115         }
116     }
117 }
118
119 void work(int u) {
120     // 换根
121     u = froot(u);
122     vis[u] = true;
123     go(u, 0);
124     for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
125         int v = gh.eg[i].to;
126         if (vis[v]) continue;
127         work(v);
128     }
```

```
}

// 预处理边后继节点个数
int pdfs(int u, int f) {
    int fg_id = -1;
    int s = 1;
    for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
        int v = gh.eg[i].to;
        if (v == f) { // 记录父边ID
            fg_id = i;
            continue;
        }
        int c = pdfs(v, u);
        gc[i] = c;
        s += c;
    }
    // 存在父边
    if (~fg_id) gc[fg_id] = n - s;
    return s;
}

void solve() {
    while (cin >> n) {
        gh.init(n);
        for (int i = 2; i <= n; i++) {
            int u, v;
            ll w;
            u = i;
            cin >> v >> w;
            gh.addEdge(u, v, w);
            gh.addEdge(v, u, w);
        }
        mst(vis, false, n + 1);
        pdfs(1, 0);
        ans = 0;
        work(1);
        cout << ans << endl;
    }
}
```

5.9 树上 dsu

```
const int N = 1e5 + 10;

vector<int> g[N];
int siz[N], dep[N], son[N], dfn[N], nodeof[N], tim;

void calc(int u, int w) {
    // ....对u这一节点进行单独处理
    if(w > 0) // ....计算贡献
    else // ....撤销影响
}

void dfs1(int u, int fa) {
    dep[u] = dep[fa] + (siz[u] = 1);
    nodeof[dfn[u] = ++tim] = u;
    for(auto v : g[u]) {
        if(v == fa) continue ;
```

```
        dfs1(v, u);
        siz[u] += siz[v];
        if(!son[u] || siz[v] > siz[son[u]]) son[u] = v;
    }
}

void dfs2(int u, int fa, bool keep) {
    for(auto v : g[u]) {
        if(v == fa || v == son[u]) continue ;
        dfs2(v, u, 0);
    }
    if(son[u]) {
        dfs2(son[u], u, 1);
    }
    for(auto v : g[u]) {
        if(v == fa || v == son[u]) continue ;
        for(int j = 0;j < siz[v]; j++) {
            // ....更新答案
        }
        for(int j = 0;j < siz[v]; j++) {
            calc(nodeof[dfn[v] + j], 1);
        }
    }
    calc(u, 1);
    // ....更新答案
    if(!keep) {
        for(int i = 0;i < siz[u]; i++) calc(nodeof[dfn[u] + i], -1);
    }
}

int main() {
    int n; cin >> n;
    for(int i = 1;i < n; i++) {
        int u, v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    dfs1(1, 0);
    dfs2(1, 0, 0);
}
```

5.10 树上 dsu

```
const int N = 2e5 + 10;

vector<int> g[N];

int siz[N], son[N], col[N];
int ans[N], cnt[N];
bool vis[N];
int maxx, sum;
// maxx为每棵子树里出现最多的颜色，sum为编号和

void calc(int u, int fa, int val) {
    /*
    针对不同问题，采取的操作
    */
    else if(val > 0 && cnt[col[u]] == maxx) sum += col[u];
```

```
    for(auto v : g[u]) {
        if(v != fa && !vis[v]) calc(v, u, w);
    }
}

void dfs1(int u, int fa) {
    siz[u] = 1;
    for(auto v : g[u]) {
        if(v == fa) continue ;
        dfs1(v, u);
        siz[u] += siz[v];
        if(!son[u] || siz[v] > siz[son[u]]) son[u] = v;
    }
}

void dfs2(int u, int fa, bool keep) {
    for(auto v : g[u]) {
        if(v != fa && v != son[u]) {
            dfs2(v, u, 0);
        }
    }
    if(son[u]) {
        dfs2(son[u], u, 1);
        vis[son[u]] = 1;
    }
    calc(u, fa, 1);
    ans[u] = sum;
    if(son[u]) vis[son[u]] = 0;
    if(!keep) {
        calc(u, fa, -1);
        maxx = sum = 0;
    }
}

int main() {
    int n; cin >> n;
    for(int i = 1;i <= n; i++) cin >> col[i];
    for(int i = 1;i < n; i++) {
        int u,  v; cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    dfs0(1, 0);
    dfs1(1, 0, false);
    for(int i = 1;i <= n; i++) cout << ans[i] << endl;
}
```

5.11 树上 K 祖先

```
//倍增KFA，空间大点，但是好写
vector<int> g[N];

int anc[N][20];
void dfs(int u, int fa) {
    anc[u][0] = fa;
    for (int i = 1; i <= 19; i++) anc[u][i] = anc[anc[u][i - 1]][i - 1];
    for (auto &v: g[u])
        if (v != fa) dfs(v, u);
```

```
}

int kthFa(int u, int k) {
    int bit = 0;
    while (k) {
        if (k & 1) u = anc[u][bit];
        k >>= 1;
        bit++;
    }
    return u;
}


//树剖KFA
int siz[N], son[N], dep[N], fa[N], top[N];
int id[N], nodeOf[N], cnt;
void dfs(int u, int par) {
    dep[u] = dep[fa[u] = par] + (siz[u] = 1);
    for (auto &v: g[u])
        if (v != par) {
            dfs(v, u);
            siz[u] += siz[v];
            if (!son[u] || siz[v] > siz[son[u]])
                son[u] = v;
        }
}

void dfs2(int u, int topf) {
    nodeOf[id[u] = ++cnt] = u, top[u] = topf;
    if (!son[u]) return;
    dfs2(son[u], topf);
    for (auto &v: g[u])
        if (v != fa[u] && v != son[u]) dfs2(v, v);
}

int kthFa(int u, int k) {
    while (k >= id[u] - id[top[u]] + 1 && u) {
        k -= id[u] - id[top[u]] + 1;
        u = fa[top[u]];
    }
    return nodeOf[id[u] - k];
}
```

5.12 虚树

```
//虚树可以处理多次询问，并且每次询问只需要树上的K个关键点
//建立的虚树能保证点数 < 2 * K
//如果对虚树做dp，总体复杂度和∑K有关
//考虑dp的时候，需要同时考虑非关键点对答案的影响

int n;

struct edge {
    int nxt, to;
} e[N << 1];
int head[N], tot;
void add(int u, int v) { e[++tot] = edge{ head[u], v }, head[u] = tot;}

```

```
14 int dep[N], fa[N], topfa[N], siz[N], son[N], dfn[N], cnt;
15 void dfs(int u, int par) {
16     dep[u] = dep[fa[u] = par] + (siz[u] = 1);
17     int max_son = -1;
18     for (int i = head[u], v; i; i = e[i].nxt)
19         if ((v = e[i].to) != par) {
20             dfs(v, u);
21             siz[u] += siz[v];
22             if (max_son < siz[v]) son[u] = v, max_son = siz[v];
23         }
24 }
25 void dfs2(int u, int topf) {
26     topfa[u] = topf, dfn[u] = ++cnt;
27     if (!son[u]) return;
28     dfs2(son[u], topf);
29     for (int i = head[u], v; i; i = e[i].nxt)
30         if ((v = e[i].to) != fa[u] && v != son[u]) dfs2(v, v);
31 }
32 int LCA(int x, int y) {
33     while (topfa[x] != topfa[y]) {
34         if (dep[topfa[x]] < dep[topfa[y]]) swap(x, y);
35         x = fa[topfa[x]];
36     }
37     return dep[x] < dep[y] ? x : y;
38 }
39 int getDis(int x, int y) { return dep[x] + dep[y] - 2 * dep[LCA(x, y)]; }
40
41 //建立虚树
42 int tag[N];//tag[u] = 1 <=> 关键点
43 vector<int> g[N];//虚树边
44 void add_edge(int u, int v) { g[u].push_back(v); }
45 int st[N], top, rt;//rt为虚树根
46 void insert(int u) {
47     if (top == 1) {
48         st[++top] = u;
49         return;
50     }
51     int lca = LCA(u, st[top]);
52     if (lca != st[top]) {
53         while (top > 1 && dfn[st[top - 1]] >= dfn[lca])
54             add_edge(st[top - 1], st[top]), top--;
55         if (lca != st[top]) add_edge(lca, st[top]), st[top] = lca;
56     }
57     st[++top] = u;
58 }
59 bool cmp(const int &x, const int &y) { return dfn[x] < dfn[y]; }
60 void build(vector<int> &v) {
61     st[top = 1] = rt;
62     sort(v.begin(), v.end(), cmp);
63     for (auto &i: v) {
64         tag[i] = 1;
65         if (i != rt) insert(i);
66     }
67     while (top > 1) add_edge(st[top - 1], st[top]), top--;
68 }
69
70
71 void dp(int u) {
72     //...
```

```
}
void clear(int u) {//清空虚树边和标记，也可以和dp合并
    for (auto &v: g[u]) clear(v);
    g[u].clear(); tag[u] = 0;
}
void solve() {
    //...
    dp(rt); clear(rt);
    //...
}

int main() {
    scanf("%d", &n);
    for (int i = 1; i < n; i++) {
        int u, v; scanf("%d%d", &u, &v);
        add(u, v); add(v, u);
    }
    //此处距离为1，所以用dep替代dis，dis[fa[rt] = 0] = -1
    dep[0] = -1, rt = 1;
    dfs(rt, 0); dfs2(rt, rt);


    int Q; scanf("%d", &Q);
    while (Q--) {
        int K; scanf("%d", &K);//读取关键点
        for (int i = 1; i <= K; i++) scanf("%d", &a[i]);
        //构建虚树
        build(a);
        solve();
    }

    return 0;
}
```

5.13 LCT

```
int ch[N][2], fa[N], rev[N], siz[N];//基本内容
int sum[N], val[N], tag[N];//另外要维护的
#define lc  ch[u][0]
#define rc  ch[u][1]
#define identify(u) (ch[fa[u]][1] == u)
#define isRoot(u)   (u != ch[fa[u]][0] && u != ch[fa[u]][1])
void flip(int u) { swap(lc, rc); rev[u] ^= 1; }
void push_up(int u) {
    siz[u] = siz[lc] + siz[rc] + 1;
    //...
}
void push_down(int u) {
    if (rev[u]) {
        if (lc) flip(lc);
        if (rc) flip(rc);
        rev[u] = 0;
    }
    //...
}
void update(int u) {//当前点之上的所有点都push_down
    if (!isRoot(u)) update(fa[u]);
    push_down(u);
```

```
}
void rotate(int u) {
    int f = fa[u], fc = identify(u);
    int g = fa[f], gc = identify(f);
    int uc = fc ^ 1, c = ch[u][uc];
    if (!isRoot(f))
        ch[g][gc] = u; fa[u] = g;
    ch[f][fc] = c, fa[c] = f;
    ch[u][uc] = f, fa[f] = u;
    push_up(f); push_up(u);
}
void splay(int u) {//将u变为u所在的Splay的根
    update(u);
    for (int f; f = fa[u], !isRoot(u); rotate(u))
        if (!isRoot(f)) rotate(identify(f) ^ identify(u) ? u : f);
}
int access(int u) {//将(rt, u)之间的路径变为实链
    int pre = 0;
    for (; u; u = fa[pre = u])
        splay(u), rc = pre, push_up(u);
    return pre;
}
void makeRoot(int u) {//将u变为整棵树的根(注意:不一定是当前splay的根)
    u = access(u);
    flip(u);
}
int findRoot(int u) {
    access(u), splay(u);
    while (lc) push_down(u), u = lc;
    splay(u);
    return u;
}
void link(int u, int v) {
    makeRoot(u); splay(u);
    if (findRoot(v) != u) fa[u] = v;
}
void split(int u, int v) {
    makeRoot(u);
    access(v); splay(v);//加了这个就将v变为splay的根
}
void cut(int u, int v) {
    makeRoot(u); splay(u);
    if (findRoot(v) == u && fa[v] == u && !ch[v][0]) {
        fa[v] = ch[u][1] = 0;
        push_up(u);
    }
}
void fix(int u, int k) {
    splay(u); val[u] = k;
}
```

6 数学

6.1 快速幂

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

typedef long long ll;

ll quick_pow(ll a, ll b, ll mod) {
    ll ans = 1;
    while(b) {
        if(b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans % mod;
}

// 快速乘和中国剩余定理搭配更香哦
ll ksc(ll a, ll b, ll mod)   {
    ll ans = 0;
    while(b > 0) {
        if(b & 1) {
            ans = (ans + a) % mod;
        }
        a = (a << 1) % mod;
        b >>= 1;
    }
    return ans;
}

ll quick_mul(ll a,ll b,ll c) {
    return (a * b - (ll)((ld)a * b / c) * c + c) % c;
}


const int N = 100 + 10;

int n;

struct Martix {
    ll a[N][N];
    Martix operator * (const Martix &rhs) const {
        Martix ans;
        mem(ans.a, 0);
        for(int i = 1;i <= n; i++) {
            for(int j = 1;j <= n; j++) {
                for(int k = 1;k <= n; k++) {
                    ans.a[i][j] = (ans.a[i][j] + a[i][k] * rhs.a[k][j]) % mod;
                }
            }
        }
        return ans;
    }
};

Martix quick_pow(Martix a, ll b) {
    Martix ans; mem(ans.a, 0);
```

```
    for(int i = 1;i <= n; i++) ans.a[i][i] = 1;
    while(b) {
        if(b & 1) ans = a * ans;
        a = a * a;
        b >>= 1;
    }
    return ans;
}
```

6.2 光速幂

```

ll v_pow(ll a, ll b) {
    ll ans = 1;
    ll base = 65536, k = 1;
    while(1) {
        if((b % (k * base)) / k == 0) break;
        ans = ans * quick_pow(a, (b % (k * base)) / k) % mod;
        a = quick_pow(a, base) % mod;
        k = k * base;
    }
    return ans;
}
```

6.3 $\mathbf{EX}_GCD$

```
typedef long long ll;

ll ex_gcd(ll a, ll b, ll &x, ll &y) {
    ll res, t;
    if(!b) {
        x = 1;
        y = 0;
        return a;
    }
    res = ex_gcd(b, a % b, x, y);
    t = x;
    x = y;
    y = t - (a / b) * y;
    return res;
}

ll INV(ll a, ll mod) {
    ll x, y;
    ll d = ex_gcd(a, mod, x, y);
    return d ? (x % mod + mod) % mod : -1;
}

ll solve_ex_gcd(ll a, ll b, ll c, ll &x, ll &y) {
    ll d = ex_gcd(a, b, x, y);
    if(c % d)
    {
        x = -1;
        y = -1;
        return -1;
    }
    x *= (c / d);
```

```
    b = abs(b / d);
    x = (x % b + b) % b;
    y = (c - a * x) / (b * d);
    return 0;
}

int main() {
    ll x, y;
    cout << solve_ex_gcd(2, 3, 4, x, y) << endl;
}
```

6.4 素数筛

```
const int N = 1e6 + 10;
bool is_prime[N]; //is_prime[i]为true时，i为素数
int prime[N], cnt = 0;

void sieve(int n) {
    is_prime[0] = is_prime[1] = 1; //0和1都不是素数
    for(int i = 2;i <= sqrt(n); i++) {
        if(!is_prime[i]) {
            for(int j = 2 * i;j <= n; j += i) {
                is_prime[j] = 1; //删除所有的素数的倍数
            }
        }
    }
}


void euler() {
    is_prime[0] = is_prime[1] = 1;
    for(int i = 2;i < N; i++) {
        if(!is_prime[i]) {
            prime[++cnt] = i;
        }
        for(int j = 1;j <= cnt && prime[j] * i < N; j++) {
            is_prime[i * prime[j]] = 1;
            if(i % prime[j] == 0) break;
        }
    }
}
```

6.5 整除分块

```
int calc(int n, int m) {
    //\sum_{i = 1} ^ {m} n / i
    //向下取整
    for (int l = 1, r; l <= m; l = r + 1) {
        if (n / l) r = min(m, n / (n / l));
        else r = m;
        //[l, r]之间的 n / l 都相等
    }

    //向上取整
    for (int l = 1, r; l <= m; l = r + 1) {
        int t = (n + l - 1) / l;
        if (t == 1) r = m;
```

```
14          else r = min(m, (n - 1) / (t - 1));
15          //[l, r]之间的 (n + l - 1) / l 都相等
16      }
17
18  }
```

6.6 线性递推逆元

```
1   // 乘法逆元的线性递推，复杂度O(n)
2
3   typedef long long ll;
4
5   ll inv[3000005] = {0 , 1}; //n等于1时，关于mod的逆元就为1
6
7   int main()
8   {
9       ll n, mod;
10      cin >> n >> mod;
11      cout << 1 << endl;
12      for(int i = 2;i <= n; i++)
13      {
14          inv[i] = mod - (mod / i) * inv[mod % i] % mod;
15          cout << inv[i] << endl;
16      }
17      return 0;
18  }
```

6.7 线性求任意 n 个数的逆元

```
1   typedef long long ll;
2
3   ll a[10005]; // n个数
4   ll s[10005]; // 前缀积
5   ll invs[10005]; // 前缀积的逆元
6   ll inv[10005]; // a[i]的逆元
7
8   ll qucik_pow(ll a, ll b, ll mod) ;
9
10  void Line_INV() {
11      s[0] = 1;
12      ll p;
13      for(int i = 1;i <= n; i++)
14          cin >> a[i]; // 任意n个数
15      cin >> p;
16      for (int i = 1; i <= n; ++i)
17          s[i] = s[i - 1] * a[i] % p;
18      sv[n] = quick_pow(s[n], p - 2);
19      for (int i = n; i >= 1; --i)
20          sv[i - 1] = sv[i] * a[i] % p;
21      for (int i = 1; i <= n; ++i)
22          inv[i] = sv[i] * s[i - 1] % p;
23  }
```

6.8 算术基本定理

```
ll get_Count(ll n) {
    ll ans = 1;
    for(int i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            int a = 0;
            while(n % i == 0) {
                a++;
                n /= i;
            }
            ans *= (a + 1);
        }
    }
    if(n > 1)  ans *= 2;
    return ans;
}

ll get_Sum(ll n) {
    ll ans = 1;
    for(int i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            ll a = 1;
            while(n % i == 0) {
                n /= i;
                a *= i;
            }
            ans = ans * (a * i - 1) / (i - 1);
        }
    }
    if(n > 1) ans *= (n + 1);
    return ans;
}
```

6.9 筛 phi

```
int is_prime[N], prime[N], cnt, phi[N];
void makePhi() {
    phi[1] = 1, cnt = 0;
    for (int i = 2; i < N; i++) {
        if (!is_prime[i]) prime[++cnt] = i, phi[i] = i - 1;
        for (int j = 1; j <= cnt && i * prime[j] < N; j++) {
            is_prime[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            else phi[i * prime[j]] = phi[i] * phi[prime[j]];
        }
    }
}
```

6.10 筛 mobius

```
const int N = 1e5 + 10;
bool is_prime[N];
int prime[N], mu[N], cnt;

void makeMobius() {
```

```
    mu[1] = 1; is_prime[0] = is_prime[1] = true;
    for(int i = 2;i < N; i++) {
        if (!is_prime[i]) {
            mu[i] = -1;
            prime[++cnt] = i;
        }
        for (int j = 1; j <= cnt && i * prime[j] < N; j++) {
            is_prime[i * prime[j]] = true;
            if (i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                break;
            }
            mu[i * prime[j]] = -mu[i];
        }
    }
}
```

6.11 筛积性函数

```
//只需要计算f(p ^ k)即可
//其余的都可以通过积性函数的性质来计算

int vis[N], prime[N], num;
int f[N], low[N];

void makeF(int siz) {//f为积性函数
    num = 0, low[1] = f[1] = 1;
    for (int i = 2; i <= siz; i++) {
        if (!vis[i]) prime[++num] = i, low[i] = i, f[i] = ...;//这里是f(p)的答案
        for (int j = 1; j <= num && i * prime[j] <= siz; j++) {
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                low[i * prime[j]] = low[i] * prime[j];
                if(low[i] == i) {//i = prime[j] ^ k
                    //只需要这里算一下
                    //考虑 p ^ 1 , p ^ 2, p ^ 3...
                }
                else f[i * prime[j]] = 1ll * f[i / low[i]] * f[prime[j] * low[i]] % mod
    ;
                break;
            }
            low[i * prime[j]] = prime[j];
            f[i * prime[j]] = 1ll * f[i] * f[prime[j]] % mod;
        }
    }
}
```

6.12 欧拉函数

```
// 求解单个正整数的欧拉函数
int Get_phi(int n) {
    int ans = n;
    for(int i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            ans = ans - ans / i;
            while(n % i == 0)
                n /= i;
```

```
        }
    }
    if(n > 1)
        ans = ans - ans / n;
    return ans;
}

// 埃拉托斯特尼筛求欧拉函数
int phi[10005];

void Euler_sieve(int n) {
    phi[1] = 1;
    for(int i = 2;i <= n; i++) {
        if(!phi[i]) {
            for(int j = i;j <= n; j += i)    {
                if(!phi[j])
                    phi[j] = j;
                phi[j] = phi[j] / i * (i - 1);
            }
        }
    }
}

// 欧拉筛求欧拉函数

const int N = 5e6 + 10;
bool is_prime[N];
int prime[N], phi[N], tot;

void Euler() {
    phi[1] = 1; is_prime[1] = true;
    for(int i = 2;i < N; i++){
        if(!is_prime[i]) {
            phi[i] = i - 1;
            prime[++tot] = i;
        }
        for(int j = 1;j <= tot && i * prime[j] < N; j++){
            is_prime[i * prime[j]] = true;
            if(i % prime[j]) {
                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
            }
            else{
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
        }
    }
}
```

6.13 原根

```
typedef long long ll;

vector<ll> YG;
ll p, n; // p是模数，n是p的欧拉函数值

ll gcd(ll a, ll b) {
    return b ? gcd(b, a % b) : a;
```

```
}

ll quick_pow(ll a, ll b, ll p) ;

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

vector<ll> PrimeFac(ll n) { // n的素因子
    vector<ll> fac;
    fac.clear();
    for(ll i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            fac.push_back(i);
            while(n % i == 0)
                n /= i;
        }
    }
    if(n > 1)
        fac.push_back(n);
    return fac;
}

bool is_Protogen(ll p) { // 原根p = 2、4、p^k、2*p^k(p为非2的质数，k为任意数)
    if(p == 2 || p == 4) return true;
    if(p <= 1 || p % 4 == 0) return false;
    ll num = 0;
    while(p % 2 == 0) // 2的倍数先筛掉
        p /= 2;
    for(int i = 3;i * i <= p; i++) { // p只能是一个非2的素数的倍数构成，否则没有原根
        if(p % i == 0) {
            num++;
            while(p % i == 0)
                p /= i;
        }
    }
    if(p > 1) num++;
    if(num == 1) return true;
    return false;
}

ll Protogen(ll p) {
    if(!is_Protogen(p)) // 先判断是否存在原根
        return -1;
    n = phi(p);
    if(p == 2) return 1;
    if(p == 3) return 2;
    if(p == 4) return 3;
```

```
    vector<ll> fac = PrimeFac(n); // f(p)的素因子
    for(int i = 2;i <= p - 1; i++) {
        if(gcd(i, p) != 1) // n是模p的欧拉函数值，i要和n互质
            continue;
        bool flag = true;
        for(ll j = 0;j < fac.size(); j++) {
            if(quick_pow(i, n / fac[j] , p) == 1)
                flag = 0;
        }
        if(flag) // i就是原根
            return i;
    }
    return -1;
}

void Sum_Protogen(ll k) { // 找出n的所有原根
    YG.push_back(k);
    for(int i = 2;i < n; i++) {
        if(gcd(i, n) == 1) // i要与f(n)互质
            YG.push_back(quick_pow(k, i, p));
    }
}

int main() {
    cin >> p;
    ll k = Protogen(p); // p的原根
    cout << k << endl;
    Sum_Protogen(k);
    for(int i = 0;i < YG.size(); i++) {
        cout << YG[i] << " ";
    }
    cout << endl;
    return 0;
}
```

6.14 原根表

```
mod                                         原根
r*2^k+1   r              k              g
3   1   1   2
5   1   2   2
17  1   4   3
97  3   5   5
193 3   6   5
257 1   8   3
7681    15  9   17
12289   3   12  11
40961   5   13  3
65537   1   16  3
786433  3   18  10
5767169 11  19  3
7340033 7   20  3
23068673    11  21  3
104857601   25  22  3
167772161   5   25  3
469762049   7   26  3
998244353   119 23  3   这个数常用
1004535809  479 21  3   加起来不会爆int
```

```
2013265921  15  27  31
2281701377  17  27  3    这个数平方刚好不会爆ll
3221225473  3   30  5
75161927681 35  31  3
77309411329 9   33  7
206158430209    3   36  22
2061584302081   15  37  7
2748779069441   5   39  3
6597069766657   3   41  5
39582418599937  9   42  5
79164837199873  9   43  5
263882790666241 15  44  7
1231453023109121    35  45  3
1337006139375617    19  46  3
3799912185593857    27  47  5
4222124650659841    15  48  19
7881299347898369    7   50  6
31525197391593473   7   52  3
180143985094819841  5   55  6
1945555039024054273 27  56  5
4179340454199820289 29  57  3
```

6.15 阶乘逆元

```cpp
const int N = 5e6 + 10;
const ll mod = 1e9 + 7;

ll F[N], invn[N], invF[N];

void Init() {
    F[0] = F[1] = invn[0] = invn[1] = invF[0] = invF[1] = 1;
    for(int i = 2;i < N; i++){
        F[i] = F[i - 1] * i % mod;
        invn[i] = (mod - mod / i) * invn[mod % i] % mod;
        invF[i] = invF[i - 1] * invn[i] % mod;
    }
}
```

6.16 常见积性函数

```cpp
//phi
//phi[i * j] = phi[i] * phi[j] * gcd(i, j) / phi[gcd(i, j)]

// d
// d[i * j] = \sum_{x|i} * \sum_{y|j} * [gcd(x, y) = 1]
```

6.17 $\text{Miller}_R abin$

```cpp
// 二次探测定理：对素数p,满足x^2≡1(modp)的小于p的正整数解x只有1或p-1.

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5 + 7;
const int times = 10;

```

```cpp
ll ksc(ll a, ll b, ll mod)  {
    ll ans = 0;
    while(b > 0) {
        if(b & 1) {
            ans = (ans + a) % mod;
        }
        a = (a << 1) % mod;
        b >>= 1;
    }
    return ans;
}

ll quick_pow(ll a, ll b, ll mod) {
    ll ans = 1, base = a;
    while(b != 0) {
        if(b & 1) {
            ans = ans * base % mod;
        }
        base = base * base % mod;
        b >>= 1;
    }
    return ans;
}

bool Miller_Pabin(ll n)//Miller测试的主体结构
{
    if(n < 2) return false;
    if(n == 2) return true;
    if(n & 1 == 0) return false;//对于偶数的优化
    ll k = 0,u = n - 1;//p为Miller测试的k, u为Miller测试的m

    while(u & 1 == 0){ // 把x拆成u*2^k
        u >>= 1;
        k++;
    }
    srand(time(NULL));

    ll x, pre; // pre为上次探测的x的值

    for(int i = 1;i <= times; i++) {
        x = rand() % (n - 1) + 1;
        x = quick_pow(x, u, n); // 先求出x^u(mod n)
        pre = x;
        for(int j = 1;j <= k; j++) {
            x = ksc(x, x, n);
            if(x == 1 && pre != 1 && pre != n - 1)
                return false;
            pre = x;
        }
        if(x != -1)
        return false;
    }
    return true;
}

int main() {
    ll n; cin >> n;
    cout << (Miller_Pabin(n) ? "Prime" : "Not a Prime") << endl;
}
```

6.18 二次剩余

```
typedef long long ll;

typedef struct{
    ll x, y; // 把求出来的w作为虚部，则为a + bw
}num;

ll quick_pow(ll a, ll b, ll p) {
    ll ans = 1;
    while(b) {
        if(b & 1) ans = ans * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return ans % p;
}


num num_mul(num a, num b, ll w, ll p) {// 复数乘法
    num ans = {0, 0};
    ans.x = (a.x * b.x % p + a.y * b.y % p * w % p + p) % p;
    ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
    return ans;
}

ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
    num ans = {1, 0};
    while(b) {
        if(b & 1)
            ans = num_mul(ans, a, w, p);
        a = num_mul(a, a, w, p);
        b >>= 1;
    }
    return ans.x % p;
}

ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
    return quick_pow(a, (p - 1) >> 1, p);
}

ll Cipolla(ll n, ll p) {// 输入a和p，是否存在x使得x^2 = a (mod p)，存在二次剩余返回x，存在二次非剩余返回-1     注意：p是奇质数
    n %= p;
    if(n == 0)
        return 0;
    if(p == 2)
        return 1;
    if(legendre(n, p) + 1 == p) // 二次非剩余
        return -1;

    ll a, w;

    while(true) {// 找出a，求出w，随机成功的概率是50%，所以数学期望是2
        a = rand() % p;
        w = ((a * a - n) % p + p) % p;
        if(legendre(w, p) + 1 == p) // 找到w，非二次剩余条件
            break;
    }
```

```
57      num x = {a, 1};
58      return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x,一个解是x，另一个解是p-x，这里的w其实
        要开方，但是由拉格朗日定理可知虚部为0，所以最终答案就是对x的实部用快速幂求解
59  }
60
61  int main()
62  {
63      ll n, p;
64      cin >> n >> p;
65      srand((unsigned)time(NULL));
66      cout << Cipolla(n, p) << endl;
67      return 0;
68  }
```

6.19 BSGS

```
1   // 求解a^x = b (mod c)，要求gcd(a, c) = 1， 不要求p为素数，x的范围是0 <= x <= p-1
2
3   #include <iostream>
4   #include <map>
5
6   using namespace std;
7   typedef long long ll;
8
9   const int maxn = 5e5 + 10;
10
11  int Hash[maxn], id[maxn], head[maxn], Next[maxn], cnt; // 链式前向星，比map快log，但是需要极
        大空间
12
13  struct HASH{
14      void insert(ll x, ll y) {
15          ll k = x % maxn;
16          Hash[cnt] = x;
17          id[cnt] = y;
18          Next[cnt] = head[k];
19          head[k] = cnt++;
20      }
21
22      ll query(ll x) {
23          for(int i = head[x % maxn]; i != -1 ; i = Next[i]){
24              if(Hash[i] == x)
25                  return id[i];
26          }
27          return -1;
28      }
29  }HASH;
30
31  ll BSGS(ll a, ll b, ll c) {
32      a %= c;
33      b %= c;
34      cnt = 1;
35      if(b == 1)  return 0;
36
37      memset(head, -1, sizeof(head));
38
39      ll m = ceil(sqrt((double)c)); // 向上取整
40      ll x = 1, p = 1;
41
```

```
    for(ll j = 0;j < m; j++, p = p * a % c) {// 0 ~ m - 1
        HASH.insert(p * b % c, j); // 先枚举右边，把(b*a^j, j)放入hash中
    }

    for(ll i = 1, j ;i <= m; i++) { // 枚举a^im
        x = x * p % c;
        if((j = HASH.query(x)) != -1)
            return i * m - j; // 找到相匹配的哈希值
    }
    return -1;
}

int main() {
    ll a, b, c;
    cin >> a >> b >> c;
    cout << BSGS(a, b, c) << endl;
}
```

6.20 $\mathbf{EX}_BSGS$

```
// a和c不互质
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

ll gcd(ll a, ll b) {
    return b ? gcd(b, a % b) : a;
}

ll EX_BSGS(ll a, ll b, ll c) {
    a %= c;
    b %= c;

    if(b == 1) return 0;

    ll k = 0, tmp = 1, d;
    while(true) {
        d = gcd(a, c);
        if(d == 1)
            break;
        if(b % d) // 无解
            return -1;
        b /= d; c /= d;
        tmp = tmp * (a / d) % c;
        k++;
        if(tmp == b)
            return k;
    }

    map<ll, ll> mp;
    mp.clear();

    ll m = ceil(sqrt((double)c)); // 向上取整
    ll x = 1, p = 1;
    for(ll j = 0;j < m; j++, p = p * a % c) {// 0 ~ m - 1
        mp[p * b % c] = j;
    }
```

```

    x = tmp % c;

    for(ll i = 1 ;i <= m; i++) { // 枚举a^im
        x = x * p % c;
        if(mp[x]) {
            return k + i * m - mp[x];
        }
    }
    return -1;
}

int main() {
    ll a, b, c;
    cin >> a >> b >> c;
    cout << EX_BSGS(a, b, c) << endl;
}
```

6.21 CRT

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

typedef long long ll;
ll m[10005], a[10005], n; //a是余数，b是模数

void exgcd(ll a, ll b, ll &x, ll &y) {
    if(b == 0) {
        x = 1;
        y = 0;
        return;
    }
    exgcd(b, a % b , y , x);
    y -= a / b * x;
}

ll INV(ll a, ll mod) {
    ll x, y;
    exgcd(a , mod , x , y);
    x = (x % mod + mod) % mod;
    return x;
}

ll CRT() {
    ll ans = 0, M = 1;
    for(ll i = 1;i <= n; i++) {
        M *= m[i]; // M是所有除数的乘积
    }
    for(ll i = 1;i <= n; i++) {
        ll mm = M / m[i];
        ll ret = INV(mm , m[i]); // 先求逆元
        ans = (ans + a[i] * mm % M * ret % M) % M;
/*

 ans = (ans + quick_mul(quick_mul(m , ret , M) , b[i] , M)) % M;
 利用快速乘防止爆longlong

 */
```

```
    }
    return (ans + M) % M;
}

int main() {
    ll ans = 0;
    scanf("%lld",&n);
    for(ll i = 1;i <= n; i++) {
        scanf("%lld%lld",&m[i],&a[i]);
        a[i] = (a[i] % m[i] + m[i]) % m[i];// 防止b[i]为负
    }
    ans = CRT(); // 精髓
    printf("%lld",ans);
    return 0;
}
```

6.22 EX$_{CRT}$

```
#include <iostream>

using namespace std;

typedef long long ll;

ll c[100005], m[100005], n;

ll ksc(ll a, ll b, ll mod) {
    ll ans = 0;
    while(b > 0) {
        if(b & 1) {
            ans = (ans + a) % mod;
        }
        a = (a << 1) % mod;
        b >>= 1;
    }
    return ans;
}

ll gcd(ll a, ll b) {
    return b ? gcd(b, a % b) : a;
}

ll ex_gcd(ll a, ll b, ll &x, ll &y) {
    ll res, t;
    if(!b) {
        x = 1;
        y = 0;
        return a;
    }
    res = ex_gcd(b, a % b, x, y);
    t = x;
    x = y;
    y = t - (a / b) * y;
    return res;
}

ll INV(ll a, ll mod) {
    ll x, y;
```

```
    ll d = ex_gcd(a, mod, x, y);
    return d ? (x % mod + mod) % mod : -1;
}

ll EX_CRT() {
    ll x, y;
    ll ans = c[1];
    ll M = m[1];
    for(int i = 2; i <= n; i++) {
        ll C = ((c[i] - ans) % m[i] + m[i]) % m[i];
        ll T = ex_gcd(M, m[i], x, y);
        if((c[i] - ans) % T)
            return -1;
        x = ksc(x, C / T, m[i] / T);
        ans += M * x;
        M *= (m[i] / T);
        ans = (ans % M + M) % M;
    }
    return ans;
}

/*
ll EX_CRT() // 便于理解
{
    for(int i = 2;i <= n; i++)
    {
        ll M1 = m[i - 1], M2 = m[i], C1 = c[i - 1], C2 = c[i];
        ll T = gcd(M1, M2); // gcd(M1, M2)
        if((C2 - C1) % T) // 无解
            return -1;
        m[i] = (M1 * M2) / T; // 合并后新同余方程的模
        c[i] = INV(M1 / T, M2 / T) * (C2 - C1) / T % (M2 / T) * M1 + C1; // 可快速乘优化
        c[i] = (c[i] % m[i] + m[i]) % m[i]; // 合并后新同余方程的余
    }
    return c[n];
}
*/

int main()
{
    cin >> n;
    for(int i = 1;i <= n; i++)
    cin >> c[i] >> m[i];
    cout << EX_CRT() << endl;
}
```

6.23 Lucas

```
namespace Comb {
    const int N = 1e6 + 10;
    ll F[N], invF[N], inv[N];

    void init() {
        F[0] = F[1] = invF[0] = invF[1] = inv[0] = inv[1] = 1;
        for (int i = 2; i < N; i++) {
            F[i] = F[i - 1] * i % mod;
            inv[i] = (mod - mod / i) * inv[mod % i] % mod;
            invF[i] = invF[i - 1] * inv[i] % mod;
```

```
        }
    }

    ll C(ll m, ll n) {
        if (m < 0 || n < 0 || n > m) return 0;
        ll ans = F[m];
        ans = ans * invF[n] % mod;
        ans = ans * invF[m - n] % mod;
        return ans;
    }

    ll Lucas(ll m, ll n) {
        return n ? Lucas(m / mod, n / mod) * C(m % mod, n % mod) % mod : 1;
    }

}
```

6.24 $\mathbf{EX}_{L}ucas$

```
// p不为质数，利用中国剩余定理结合求解
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

const int N = 1e5 + 10;

ll quick_pow(ll a, ll b, ll P) {
    ll ans = 1;
    while(b) {
        if(b & 1)
            ans = ans * a % P;
        a = a * a % P;
        b >>= 1;
    }
    return ans % P;
}

ll ex_gcd(ll a, ll b, ll &x, ll &y) {
    ll res, t;
    if(!b) {
        x = 1;
        y = 0;
        return a;
    }
    res = ex_gcd(b, a % b, x, y);
    t = x;
    x = y;
    y = t - (a / b) * y;
    return res;
}

ll INV(ll a, ll mod) {
    ll x, y;
    ll d = ex_gcd(a, mod, x, y);
    return d ? (x % mod + mod) % mod : -1;
}

```

```
ll fac(ll n, ll P, ll pk) {// 阶乘除去质因子后模质数幂（n / p^a) % pk
    if(!n) return 1;
    ll ans = 1;
    for(int i = 1;i < pk; i++) {// 第三部分：n!与p互质的乘积
        if(i % P)
            ans = ans * i % pk;
    }
    ans = quick_pow(ans, n / pk, pk) % pk; // 第三部分：n!与p互质的乘积,ans循环的次数为n/pk
    for(int i = 1;i <= n % pk; i++) {// 第四部分：循环过后n!剩下的部分
        if(i % P) ans = ans * i % pk;
    }
    return ans * fac(n / P, P, pk) % pk;  // 第一部分，p的幂，个数为n/p;   第二部分：(n/p)!
}

ll C(ll m, ll n, ll P, ll pk) {// 组合数模质数幂
    if(n < 0 || m < 0 || n > m) return 0;
    ll f1 = fac(m, P, pk), f2 = fac(n, P, pk), f3 = fac(m - n, P, pk), tmp = 0; // tmp
    = pk1 - pk2 - pk3
    for(ll i = m; i ; i /= P)     tmp += i / P;
    for(ll i = n; i ; i /= P)     tmp -= i / P;
    for(ll i = m - n; i ; i /= P) tmp -= i / P;
    return f1 * INV(f2, pk) % pk * INV(f3, pk) * quick_pow(P, tmp, pk) % pk;
}

ll p[N], a[N];
int cnt;

ll CRT() {
    ll M = 1, ans = 0;
    for(int i = 1;i <= cnt; i++)  M *= p[i];
    for(int i = 1;i <= cnt; i++) {
        ll m = M / p[i];
        ans = (ans + a[i] * m % M * INV(m, p[i]) % M) % M;
    }
    return (ans % M + M) % M;
}

ll EX_Lucas(ll m, ll n, ll P) {
    for(int i = 2;i * i <= P; i++) {
        if(P % i == 0) {
            ll tmp = 1;
            while(P % i == 0) {
                tmp *= i;
                P /= i;
            }
            p[++cnt] = tmp;
            a[cnt] = C(m, n, i, tmp);
        }
    }
    if(P > 1) {
        p[++cnt] = P;
        a[cnt] = C(m, n, P, P);
    }
    return CRT();
}
int main() {
    ll m, n, P;
    cin >> m >> n >> P;
    cnt = 0;
```

```
    cout << EX_Lucas(m, n, P) << endl;
}
```

6.25 Min25 筛

```
typedef long long ll;

const int N = 1e5 + 10;


namespace Min25 {
    int prime[N], id1[N], id2[N], flag[N], ncnt, m;

    ll g[N], sum[N], a[N], T, n;

    inline int ID(ll x) {
        return x <= T ? id1[x] : id2[n / x];
    }

    inline ll calc(ll x) {
        return x * (x + 1) / 2 - 1;
    }

    inline ll f(ll x) {
        return x;
    }

    inline void init() {
        ncnt = 0, m = 0;
        T = sqrt(n + 0.5);
        for (int i = 2; i <= T; i++) {
            if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
            for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {
                flag[i * prime[j]] = 1;
                if (i % prime[j] == 0) break;
            }
        }
        for (ll l = 1; l <= n; l = n / (n / l) + 1) {
            a[++m] = n / l;
            if (a[m] <= T) id1[a[m]] = m; else id2[n / a[m]] = m;
            g[m] = calc(a[m]);
        }
        for (int i = 1; i <= ncnt; i++)
            for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j]; j++)
                g[j] = g[j] - (ll)prime[i] * (g[ID(a[j] / prime[i])] - sum[i - 1]);
    }

    inline ll Solve(ll x) {
        if (x <= 1) return x;
        return n = x, init(), g[ID(n)];
    }

}
```

6.26 杜教 BM

```
typedef long long ll;
```

```
const ll mod = 1e9 + 7;

typedef vector<ll> VI;

ll quick_pow(ll a, ll b) ;

namespace linear_seq {
    const ll N = 1e5 + 10;
    ll res[N], base[N], _c[N], _md[N];

    vector<ll> Md;
    void mul(ll *a, ll *b, ll k) {
        for (ll i = 0; i < 2 * k; i++)
            _c[i] = 0;
        for (ll i = 0; i < k; i++) {
            if (a[i]) {
                for (int j = 0; j < k; j++) {
                    _c[i + j] = (_c[i + j] + a[i] * b[j]) % mod;
                }
            }
        }
        for (ll i = 2 * k - 1; i >= k; i--) {
            if (_c[i]) {
                for (ll j = 0; j < Md.size(); j++) {
                    _c[i - k + Md[j]] = (_c[i - k + Md[j]] - _c[i] * _md[Md[j]]) % mod;
                }
            }
        }
        for (ll i = 0; i < k; i++)
            a[i] = _c[i];
    }

    ll solve(ll n, VI a, VI b) {
        // a 系数 b 初值 b[n + 1] = a[0] * b[n] + ...
        // cout << b.size() << endl;
        ll ans = 0, pnt = 0;
        ll k = a.size();
        assert(a.size() == b.size());
        for (ll i = 0; i < k; i++)
            _md[k - i - 1] = -a[i];
        _md[k] = 1;
        Md.clear();
        for (ll i = 0; i < k; i++) {
            if (_md[i] != 0)
                Md.push_back(i);
        }
        for (ll i = 0; i < k; i++)
            res[i] = base[i] = 0;
        res[0] = 1;
        while ((1ll << pnt) <= n)
            pnt++;
        for (ll p = pnt; p >= 0; p--) {
            mul(res, res, k);
            if ((n >> p) & 1) {
                for (ll i = k - 1; i >= 0; i--)
                    res[i + 1] = res[i];
                res[0] = 0;
                for (ll i = 0; i < Md.size(); i++)
                    res[Md[i]] = (res[Md[i]] - res[k] * _md[Md[i]]) % mod;
```

```
61              }
62          }
63          for (ll i = 0; i < k; i++)
64              ans = (ans + res[i] * b[i]) % mod;
65          return ans;
66      }
67
68      VI BM(VI s) {
69          VI C(1, 1), B(1, 1);
70          ll L = 0, m = 1, b = 1;
71          for (ll n = 0; n < s.size(); n++) {
72              ll d = 0;
73              for (ll i = 0; i < L + 1; i++)
74                  d = (d + (ll)C[i] * s[n - i]) % mod;
75              if (d == 0)
76                  m++;
77              else if (2 * L <= n) {
78                  VI T = C;
79                  ll c = mod - d * quick_pow(b, mod - 2) % mod;
80                  while (C.size() < B.size() + m)
81                      C.push_back(0);
82                  for (int i = 0; i < B.size(); i++)
83                      C[i + m] = (C[i + m] + c * B[i]) % mod;
84                  L = n + 1 - L;
85                  B = T;
86                  b = d;
87                  m = 1;
88              }
89              else {
90                  ll c = mod - d * quick_pow(b, mod - 2) % mod;
91                  while (C.size() < B.size() + m)
92                      C.push_back(0);
93                  for (ll i = 0; i < B.size(); i++)
94                      C[i + m] = (C[i + m] + c * B[i]) % mod;
95                  m++;
96              }
97          }
98          return C;
99      }
100
101     ll gao(VI a, ll n) {
102         VI c = BM(a);
103         c.erase(c.begin());
104         for (ll i = 0; i < c.size(); i++)
105             c[i] = (mod - c[i]) % mod;
106         return solve(n, c, VI(a.begin(), a.begin() + c.size()));
107     }
108 }
109
110 void solve() {
111     int n;
112     while (~scanf("%d", &n)) {
113         VI v = VI{1,2,4,7,13,24};
114             printf("%d\n", linear_seq::gao(v, n - 1));
115     }
116 }
```

6.27 杜教筛

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

using namespace std;

typedef long long ll;
const int N = 1e6 + 10;

unordered_map<int, ll> smu, sphi;
bool isPrime[N];
int prime[N], num;
ll mu[N], phi[N];

void makeMobiusAndEuler(int siz) {
    mu[1] = phi[1] = 1;
    for (int i = 2; i <= siz; i++) {
        if (!isPrime[i]) prime[++num] = i, mu[i] = -1, phi[i] = i - 1;
        for (int j = 1; j <= num && i * prime[j] <= siz; j++) {
            isPrime[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            else {
                phi[i * prime[j]] = phi[prime[j]] * phi[i];
                mu[i * prime[j]] = -mu[i];
            }
        }
    }
    for (int i = 1; i <= siz; i++) mu[i] += mu[i - 1], phi[i] += phi[i - 1];
}

ll getSmu(int n) {
    if (n < N) return mu[n];
    if (smu[n]) return smu[n];
    ll res = 1;
    for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
        r = n / (n / l);
        res -= 1ll * (r - l + 1) * getSmu(n / l);
    }
    return smu[n] = res;
}

ll getSphi(int n) {
    if (n < N) return phi[n];
    if (sphi[n]) return sphi[n];
    ll res = 1ll * n * (n + 1) / 2;
    for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
        r = n / (n / l);
        res -= 1ll * (r - l + 1) * getSphi(n / l);
    }
    return sphi[n] = res;
}
```

6.28 反演相关

```
/*
莫比乌斯反演
g[n] = \sum_{d | n} f[d]
f[d] = \sum_{d | n} g[d] * mu[n / d]
二项式反演
g[n] = \sum{i = 1}^{n} C(n, i) * f[i]
f[n] = \sum{i = 1}^{n} C(n, i) * g[i] * (-1)^{n - i}
子集反演
f(S) = \sum_{T \belong S} g(T)
g(S) = \sum_{T \belong S} f(T) * (-1) ^ {|S| - |T|}
*/
```

6.29 整数拆分

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

typedef long long ll;

//递归
ll PartitionCount(ll n, ll m) {
    if(n == 1 || m == 1)  return 1;
    else if(n < m) return PartitionCount(n , n);
    else if(n == m) return PartitionCount(n , n - 1) + 1;
    else return PartitionCount(n - m , m) + PartitionCount(n , m - 1);
}

//DP
ll dp[1005][1005];

void Partition_DP(ll n, ll m) {
    for(ll i = 1;i <= n + 1; i++) {
        for(ll j = 1;j <= m + 1; j++) {
            if(i == 1 || j == 1) dp[i][j] = 1;
            else if(i == j) dp[i][j] = 1 + dp[i][j - 1];
            else if(i < j) dp[i][j] = dp[i][i];
            else dp[i][j] = dp[i - j][j] + dp[i][j - 1];
         }
     }
}
```

6.30 自适应 Simpson 积分

```
// 求一个函数在一个区间上的数值积分

double f(double x) { // 题目中要求的辛普森积分函数，这里简单写一下f(x)=x*x
    return x * x;
}

double Simpson(double a, double b) {
    double mid = (a + b) / 2.0;
    return (b - a) *(f(a) + f(b) + 4.0 * f(mid)) / 6.0;
}

double DFS(double a, double b, double eps)
{
    double mid = (a + b) / 2.0;
```

```
    double SA = Simpson(a, mid), SM = Simpson(a, b), SB = Simpson(mid, b);
    if(fabs(SA + SB - SM) <= 15.0 * eps)
        return SA + SB + (SA + SB - SM) / 15.0;
    return DFS(a, mid, eps / 2.0) + DFS(mid, b, eps / 2.0);
}

// 求一个函数在0~无穷的上的数值积分，若收敛输出答案，若发散输出orz
```

6.31 Bell

```

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

typedef long long ll;

const int N = 20;

ll S2[N][N];
ll B[N];

void Stirling2() {
    S2[0][0] = 1;

    for(int i = 1;i < N; i++) {
        for(int j = 1;j <= i; j++) {
            S2[i][j] = S2[i - 1][j - 1] + j * S2[i - 1][j];
        }
    }

}

// 根据第二类斯特林数

void Bell1() {

    for(int i = 0;i < N; i++) {
        for(int j = 0;j <= i; j++) {
            B[i] += S2[i][j];
        }
    }
}

// Bell三角形递推

ll b[N][N];

void Bell2() {
    b[1][1] = 1;
    for(int i = 2;i < N; i++) {
        b[i][1] = b[i - 1][i - 1];

        for(int j = 2;j < N; j++) {
            b[i][j] = b[i][j - 1] + b[i - 1][j - 1];
        }
    }
}

```

```
// 自身递推

ll fac[N];

ll C(ll m, ll n) {
    return fac[m] / (fac[n] * fac[m - n]);
}

void Bell3() {

    fac[1] = 1;
    for(int i = 2;i < N; i++)
        fac[i] = fac[i - 1] * i;

    B[0] = 1;

    for(int i = 1;i < N; i++) {
        for(int k = 0;k <= i; k++) {
            B[i] += C(i, k) * B[k];
        }
    }
}
```

6.32 Catalan

```

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

typedef long long ll;

const int N = 1e5 + 10;

int C[N];

// 线性递推

void Calc1() {
    C[0] = 1;
    for(int i = 1;i < N; i++) {
        C[i] = C[i - 1] * (4 * i - 2) / (i + 1);
    }
}

// 组合数求解

int f[N];

void fac() {
    f[0] = 1;
    for(int i = 1;i < N; i++) {
        f[i] = f[i - 1] * i;
    }
}

void Calc2(int n) {
    C[n] = f[2 * n] / f[n + 1];
}
```

```

// 多项式求解

void Calc3(int n) {
    if(n == 1)
        C[n] = 1;

    for(int i = 1;i <= n; i++) {
        C[n] += C[n - i] * C[i - 1];
    }
}
```

6.33 Lucas

```
// mod一定为质数

namespace Comb {
    ll mod;
    const int N = 1e6 + 10;
    ll F[N], invF[N], inv[N];

    void init() {
        F[0] = F[1] = invF[0] = invF[1] = inv[0] = inv[1] = 1;
        for (int i = 2; i < N; i++) {
            F[i] = F[i - 1] * i % mod;
            inv[i] = (mod - mod / i) * inv[mod % i] % mod;
            invF[i] = invF[i - 1] * inv[i] % mod;
        }
    }

    ll C(ll m, ll n) {
        if (m < 0 || n < 0 || n > m) return 0;
        ll ans = F[m];
        ans = ans * invF[n] % mod;
        ans = ans * invF[m - n] % mod;
        return ans;
    }

    ll Lucas(ll m, ll n) {
        return n ? Lucas(m / mod, n / mod) * C(m % mod, n % mod) % mod : 1;
    }

}

// Comb::Lucas(m, n)
```

6.34 伯努利数

```
namespace BNL {
    const int N = 1e7 + 10, M = 1e6 + 10;
    struct Complex {
        double x, y;
        Complex(double a = 0, double b = 0): x(a), y(b) {}
        Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y);
     }
        Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y);
     }
```

```
8         Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y,
      x * rhs.y + y * rhs.x); }
9         Complex conj() { return Complex(x, -y); }
10    } w[N];
11
12    int tr[N];
13    ll F[N], G[N];
14
15    ll quick_pow(ll a, ll b, ll p) {
16        ll ans = 1;
17        while(b) {
18            if(b & 1) ans = ans * a % p;
19            a = a * a % p;
20            b >>= 1;
21        }
22        return ans % p;
23    }
24
25    void FFT(Complex *A, int len) {
26        for (int i = 0; i < len; i++) if(i < tr[i]) swap(A[i], A[tr[i]]);
27        for (int i = 2, lyc = len >> 1; i <= len; i <<= 1, lyc >>= 1)
28            for (int j = 0; j < len; j += i) {
29                Complex *l = A + j, *r = A + j + (i >> 1), *p = w;
30                for (int k = 0; k < i >> 1; k++) {
31                    Complex tmp = *r * *p;
32                    *r = *l - tmp, *l = *l + tmp;
33                    ++l, ++r, p += lyc;
34                }
35            }
36    }
37
38    inline void MTT(ll *x, ll *y, ll *z, int n) {
39        int len = 1; while (len <= n) len <<= 1;
40        for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
      0);
41        for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
      (2 * PI * i / len));
42
43        for (int i = 0; i < len; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
44        static Complex a[N], b[N];
45        static Complex dfta[N], dftb[N], dftc[N], dftd[N];
46
47        for (int i = 0; i < len; i++) a[i] = Complex(x[i] & 32767, x[i] >> 15);
48        for (int i = 0; i < len; i++) b[i] = Complex(y[i] & 32767, y[i] >> 15);
49        FFT(a, len), FFT(b, len);
50        for (int i = 0; i < len; i++) {
51            int j = (len - i) & (len - 1);
52            static Complex da, db, dc, dd;
53            da = (a[i] + a[j].conj()) * Complex(0.5, 0);
54            db = (a[i] - a[j].conj()) * Complex(0, -0.5);
55            dc = (b[i] + b[j].conj()) * Complex(0.5, 0);
56            dd = (b[i] - b[j].conj()) * Complex(0, -0.5);
57            dfta[j] = da * dc;
58            dftb[j] = da * dd;
59            dftc[j] = db * dc;
60            dftd[j] = db * dd;
61        }
62        for (int i = 0; i < len; i++) a[i] = dfta[i] + dftb[i] * Complex(0, 1);
63        for (int i = 0; i < len; i++) b[i] = dftc[i] + dftd[i] * Complex(0, 1);
```

```
64          FFT(a, len), FFT(b, len);
65          for (int i = 0; i < len; i++) {
66              int da = (ll)(a[i].x / len + 0.5) % mod;
67              int db = (ll)(a[i].y / len + 0.5) % mod;
68              int dc = (ll)(b[i].x / len + 0.5) % mod;
69              int dd = (ll)(b[i].y / len + 0.5) % mod;
70              z[i] = (da + ((ll)(db + dc) << 15) + ((ll)dd << 30)) % mod;
71          }
72      }
73
74      int getLen(int n) {
75          int len = 1; while (len < (n << 1)) len <<= 1;
76          for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
        0);
77          for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
        (2 * PI * i / len));
78          return len;
79      }
80
81      void Get_Inv(ll *f, ll *g, int n) {
82          if(n == 1) { g[0] = quick_pow(f[0], mod - 2, mod); return ; }
83          Get_Inv(f, g, (n + 1) >> 1);
84          int len = getLen(n);
85          static ll c[N];
86          for(int i = 0;i < len; i++) c[i] = i < n ? f[i] : 0;
87          MTT(c, g, c, len); MTT(c, g, c, len);
88          for(int i = 0;i < n; i++) g[i] = (2ll * g[i] - c[i] + mod) % mod;
89          for(int i = n;i < len; i++) g[i] = 0;
90          for(int i = 0;i < len; i++) c[i] = 0;
91      }
92
93      ll ff[N], invff[N], inv[N];
94      ll B[N];
95
96      ll C(ll m, ll n) {
97          if(m < 0 || n < 0 || n > m)
98              return 0;
99          ll ans = ff[m];
100         ans = ans * invff[n] % mod;
101         ans = ans * invff[m - n] % mod;
102         return ans;
103     }
104
105     void init(int m) {
106         ff[0] = ff[1] = inv[0] = inv[1] = invff[0] = invff[1] = 1;
107         for(int i = 2;i < M; i++)
108         {
109             ff[i] = ff[i - 1] * i % mod;
110             inv[i] = mod - (mod / i) * inv[mod % i] % mod;
111             invff[i] = invff[i - 1] * inv[i] % mod;
112         }
113
114         for(int i = 0;i <= m + 10; i++) F[i] = invff[i + 1];
115         Get_Inv(F, G, m + 10);
116         for(int i = 0;i <= m + 10; i++) B[i] = G[i] * ff[i] % mod;
117     }
118
119     ll solve(ll n, int k) {
120         init(k);
```

```
        ll ans = 0, prod = n % mod;
        for(int i = k; ~i ; i--) {
            ans = (ans + prod * B[i] % mod * C(k + 1, i) % mod) % mod;
            prod = prod * n % mod;
        }
        ans = ans * quick_pow(k + 1, mod - 2, mod) % mod;
        return ans;
    }
}

void solve() {
    ll n; int k; cin >> n >> k;
    cout << BNL::solve(n + 1, k) << endl;
}
```

6.35 步移

```
// S(n, m) = \sum_{i=0}^mC(n, i)

int x, y;
ll s;
ll S(int n, int m) {
    while(y < m) (s = s + C(x, ++y)) %= mod;
    while(y > m) (s = s - C(x, y--)) %= mod;
    while(x < n) (s = s * 2 - C(x++, y)) %= mod;
    while(x > n) (s = (s + C(--x, y)) * inv2) %= mod;
    return s;
}
```

6.36 康托展开

```
#include  <iostream>
#include <vector>
#include <algorithm>

using namespace std;

typedef long long ll;
const int mod = 1e9 + 7;
const int N = 1e5 + 10;

ll fac[N];
int a[N]; // 排列，康托展开求解
int n;
ll x; // 逆康托展开求解

void Get_F() {
    fac[0] = 1;
    for(int i = 1;i < N; i++)
        fac[i] = fac[i - 1] * i % mod;
}

ll CanTor() {
    ll ans = 0;
    for(int i = 1;i <= n; i++) {
        ll smaller = 0;
        for(int j = i + 1;j <= n; j++) {
```

```
27              if(a[j] < a[i])
28                  smaller++;
29          }
30          ans = (ans + fac[n - i] * smaller % mod) % mod;
31      }
32      return ans + 1;
33  }
34
35  void DeCantor() {
36      vector<int> v; // 存放当前可选数
37      vector<int> a; // 所求的排列组合序
38      for(int i = 1;i <= n; i++) {
39          v.push_back(i);
40      }
41      for(int i = n;i >= 1; i--) {
42          int r = x % fac[i - 1];
43          int t = x / fac[i - 1];
44          x = r;
45          sort(v.begin(), v.end());
46          a.push_back(v[t]);
47          v.erase(v.begin() + t);
48      }
49      for(int i = 0;i < a.size(); i++)
50          cout << a[i] << " ";
51      cout << endl;
52  }
53
54  // 线段树优化
55
56  const int N = 1000010;
57
58  ll fac[N];
59  int a[N]; // 排列，康托展开求解
60  int n;
61
62  struct SegmentTree {
63      int ls, rs;
64      int sum;
65  }t[N << 2];
66
67  int cnt, root;
68
69  void push_up(int u) {
70      t[u].sum = (t[lc].sum + t[rc].sum) % mod;
71  }
72
73  void build(int &u, int l, int r) {
74      if(!u) u = ++cnt;
75      if(l == r) {
76          t[u].sum = 1;
77          return ;
78      }
79      build(lc, l, m);
80      build(rc, m + 1, r);
81      push_up(u);
82  }
83
84  void update(int &u, int l, int r, int k) {
85      if(!u) u = ++cnt;
```

```
    if(l == r) {
        t[u].sum = 0;
        return ;
    }
    if(k <= m) update(lc, l, m, k);
    else update(rc, m + 1, r, k);
    push_up(u);
}

ll query(int u, int l, int r, int ql, int qr) {
    if(ql > qr) return 0;
    if(ql == l && qr == r) {
        return t[u].sum;
    }
    if(qr <= m) return query(lc, l, m, ql, qr) % mod;
    else if(ql > m) return query(rc, m + 1, r, ql, qr) % mod;
    else return (query(lc, l, m, ql, m) + query(rc, m + 1, r, m + 1, qr)) % mod;
}

void Get_F() {
    fac[0] = 1;
    for(int i = 1;i < N; i++)
        fac[i] = fac[i - 1] * i % mod;
}

void solve()
{
    Get_F();
    cin >> n;
    build(root, 1, n);
    ll ans = 0;
    for(int i = 1;i <= n; i++) {
        cin >> a[i];
        update(root, 1, n, a[i]);
        ans = (ans + query(root, 1, n, 1, a[i] - 1) * fac[n - i]) % mod;
    }
    cout << (ans + 1) % mod << endl;
}
```

6.37 模数非质数的组合

```
// 模数非质数情况下的组合问题
// one way, use CRT merge ans
// https://ac.nowcoder.com/discuss/655940?type=101&order=0&pos=2&page=1&channel=-1&
    source_id=discuss_tag_nctrack
// another way
// https://ac.nowcoder.com/acm/contest/view-submission?submissionId=47754622

#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int N = 1e6 + 10;

ll qpow(ll a, ll b, ll mod) {
    ll res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
```

```
17          a = a * a % mod;
18          b >>= 1;
19      }
20      return res;
21  }
22
23  ll exgcd(ll a, ll b, ll &x, ll &y) {
24      if (!b) {
25          x = 1, y = 0;
26          return a;
27      }
28      ll res = exgcd(b, a % b, x, y);
29      ll t = y;
30      y = x - a / b * y;
31      x = t;
32      return res;
33  }
34
35  ll inv(ll a, ll b) {
36      ll x = 0, y = 0;
37      exgcd(a, b, x, y);
38      return x = (x % b + b) % b;
39  }
40
41  //r[]为余数，m为模数，其中模数互质
42  //M = pi(mi), Mi = M / mi, invMi = Mi % mi
43  //ni满足是除了mi之外的倍数，且模mi为ri
44  //利用逆元性质，即ri * Mi * invMi = ri (mod mi)
45  //res = (sigma(ri * Mi * invMi)) % M
46
47  ll china(ll r[], ll m[], int n) {
48      ll M = 1, res = 0;
49      for (int i = 1; i <= n; i++) M *= m[i];
50      for (int i = 1; i <= n; i++) {
51          ll Mi = M / m[i], invMi = inv(Mi, m[i]);
52          res = (res + r[i] * Mi % M * invMi % M) % M;
53          //res = (res + mul(mul(r[i], Mi, M), invMi, M)) % M;按位乘
54      }
55      return (res % M + M) % M;
56  }
57
58  int f[N], g[N], F[N], G[N], invF[N];
59
60  int calc(int n, int p, int k) {
61      ll mod = qpow(p, k, LONG_LONG_MAX);
62      F[0] = 1, G[0] = 0;
63      for (int i = 1; i <= n; i++) {
64          g[i] = 0, f[i] = i;
65          while (f[i] % p == 0) f[i] /= p, g[i]++;
66          F[i] = 1ll * F[i - 1] * f[i] % mod;
67          G[i] = G[i - 1] + g[i];
68      }
69      invF[n] = inv(F[n], mod);
70      for (int i = n; i >= 1; i--) invF[i - 1] = 1ll * invF[i] * f[i] % mod;
71      int ans = 0;
72      for (int i = 0; i <= n / 2; i++) {
73          int t = 1ll * F[n] * invF[n - 2 * i] % mod * invF[i] % mod * invF[i] % mod *
74                  qpow(p, G[n] - G[n - 2 * i] - 2 * G[i], LONG_LONG_MAX) % mod;
75          ans = (ans + 1ll * t) % mod;
```

```
    }
    return ans;
}

ll r[20], m[20];

int main() {
#ifdef ACM_LOCAL
    freopen("input.in", "r", stdin);
    freopen("output.out", "w", stdout);
#endif
    int n, p;
    scanf("%d%d", &n, &p);
    int num = 0;
    for (int i = 2; i * i <= p; i++)
        if (p % i == 0) {
            int k = 0;
            m[++num] = 1;
            while (p % i == 0) p /= i, k++, m[num] *= i;
            r[num] = calc(n, i, k);
        }
    if (p > 1) {
        m[++num] = p;
        r[num] = calc(n, p, 1);
    }
    printf("%lld\n", china(r, m, num));

    return 0;
}
```

6.38 k 次最小置换复原

```

void solve() {
    int n; cin >> n;
    vector<int> a(n + 1), vis(n + 1);
    for(int i = 1;i <= n; i++) cin >> a[i];
    ll ans = 1;
    for(int i = 1;i <= n; i++) {
        if(!vis[i]) {
            int cnt = 0;
            int x = i;
            while(!vis[x]) {
                vis[x] = 1;
                cnt++;
                x = a[x];
            }
            ans = lcm(ans, cnt);
        }
    }
    cout << ans << endl;
}
```

6.39 Striling

```
typedef long long ll;
const int N = 20;
```

```

// 第一类斯特林数
ll S1[N][N];
void Stirling1() {
    S1[0][0] = 1;
    for(int i = 1;i < N; i++) {
        for(int j = 1;j <= i; j++) {
            S1[i][j] = S1[i - 1][j - 1] + (i - 1) * S1[i - 1][j];
        }
    }
}

// 第二类斯特林数
ll S2[N][N];
void Stirling2() {
    S2[0][0] = 1;
    for(int i = 1;i < N; i++) {
        for(int j = 1;j <= i; j++) {
            S2[i][j] = S2[i - 1][j - 1] + j * S2[i - 1][j];
        }
    }
}
```

6.40 第二类斯特林数

```
// {n,m}->n个不同元素划分成m个相同的集合中（不能有空集）的方案数。

// {n,m}={n-1,m-1}+k{n-1,m}

// {n,m}=\sum_{i=0}^n \frac{i^n}{i!} * {(-1)^{m-i}}{(m-i)!}

const int N = 1e6 + 10;
const ll mod = 167772161;

ll F[N], invF[N];
void init() ;

ll qpow(ll a, ll b, ll mod) ;

const ll G = 3;
const ll invG = qpow(G, mod - 2, mod);
int tr[N];

void NTT(ll *A, int len, int type) ;
void mul(ll *a, ll *b, int n) ;

ll a[N], b[N];

void solve() {
    init();
    int n; cin >> n;
    for(int i = 0;i <= n; i++) {
        a[i] = qpow(i, n, mod) * invF[i] % mod;
        if(i & 1) b[i] = mod - invF[i];
        else b[i] = invF[i];
    }
    mul(a, b, 2 * n);
    for(int i = 0;i <= n; i++) cout << a[i] << (i == n ? endl : " ");
```

```
}
```

6.41 第二类斯特林数

```
// 把n个不同元素划分成m个相同的集合（不能有空集）的方案数。

// k!\sum_{i=0}\frac{{i,k}x^i}{i!}=(e^x-1)^k

const int N = 6e5 + 10;
const ll mod = 167772161;

ll quick_pow(ll a, ll b) ;

const ll G = 3;
const ll invG = quick_pow(G, mod - 2);

int tr[N];
bool flag;

void NTT(ll *A, int len, int type) ;
void mul(ll *a, ll *b, int len) ;
void Get_Der(ll *f, ll *g, int len) ;
void Get_Int(ll *f, ll *g, int len) ;
void Get_Inv(ll *f, ll *g, int n) ;
void Get_Ln(ll *f, ll *g, int n) ;
void Get_Exp(ll *f, ll *g, int n) ;
void Get_Pow(ll *f, ll *g, int n, ll k1, ll k2);

ll a[N], ans[N];

ll F[N], invF[N], inv[N];
void init() ;


void solve() {
    init();
    int n; ll k; cin >> n >> k; n++;
    if(k >= mod) flag = 1;
    for(int i = 1;i < n; i++) a[i] = invF[i];
    Get_Pow(a, ans, n, k % mod, k % (mod - 1));
    for(int i = 0;i < n; i++) {
        cout << ans[i] * invF[k] % mod * F[i] % mod << (i == n - 1 ? endl : " ");
    }
}
```

6.42 第一类斯特林数

```
#include <algorithm>
#include <cstdio>
#include <cstring>

typedef long long LL;
const int N = 550050;
const int mod = 167772161;

LL pow_mod(LL a, LL b) {
  LL ans = 1;
```

```
  for (; b; b >>= 1, a = a * a % mod)
    if (b & 1) ans = ans * a % mod;
  return ans;
}

int L, rev[N];
LL w[N], inv[N], fac[N], ifac[N];

void Init(int n) {
  L = 1;
  while (L <= n) L <<= 1;
  for (int i = 1; i < L; ++i)
    rev[i] = (rev[i >> 1] >> 1) | ((i & 1) * L / 2);
  LL wn = pow_mod(3, (mod - 1) / L);
  w[L >> 1] = 1;
  for (int i = L >> 1; i < L; ++i) w[i + 1] = w[i] * wn % mod;
  for (int i = (L >> 1) - 1; i; --i) w[i] = w[i << 1];
}

void DFT(LL *A, int len) {
  int k = __builtin_ctz(L) - __builtin_ctz(len);
  for (int i = 1; i < len; ++i) {
    int j = rev[i] >> k;
    if (j > i) std::swap(A[i], A[j]);
  }
  for (int h = 1; h < len; h <<= 1)
    for (int i = 0; i < len; i += (h << 1))
      for (int j = 0; j < h; ++j) {
        LL t = A[i + j + h] * w[j + h] % mod;
        A[i + j + h] = A[i + j] - t;
        A[i + j] += t;
      }
  for (int i = 0; i < len; ++i) A[i] %= mod;
}

void IDFT(LL *A, int len) {
  std::reverse(A + 1, A + len);
  DFT(A, len);
  int v = mod - (mod - 1) / len;
  for (int i = 0; i < len; ++i) A[i] = A[i] * v % mod;
}

void offset(const LL *f, int n, LL c, LL *g) {
  // g(x) = f(x + c)
  // g[i] = 1/i! sum_{j=i}^n j!f[j] c^(j-i)/(j-i)!
  static LL tA[N], tB[N];
  int l = 1; while (l <= n + n) l <<= 1;
  for (int i = 0; i < n; ++i) tA[n - i - 1] = f[i] * fac[i] % mod;
  LL pc = 1;
  for (int i = 0; i < n; ++i, pc = pc * c % mod) tB[i] = pc * ifac[i] % mod;
  for (int i = n; i < l; ++i) tA[i] = tB[i] = 0;
  DFT(tA, l); DFT(tB, l);
  for (int i = 0; i < l; ++i) tA[i] = tA[i] * tB[i] % mod;
  IDFT(tA, l);
  for (int i = 0; i < n; ++i)
    g[i] = tA[n - i - 1] * ifac[i] % mod;
}

void Solve(int n, LL *f) {
```

```
  if (n == 0) return void(f[0] = 1);
  static LL tA[N], tB[N];
  int m = n / 2;
  Solve(m, f);
  int l = 1; while (l <= n) l <<= 1;
  offset(f, m + 1, m, tA);
  for (int i = 0; i <= m; ++i) tB[i] = f[i];
  for (int i = m + 1; i < l; ++i) tA[i] = tB[i] = 0;
  DFT(tA, l); DFT(tB, l);
  for (int i = 0; i < l; ++i) tA[i] = tA[i] * tB[i] % mod;
  IDFT(tA, l);
  if (n & 1)
    for (int i = 0; i <= n; ++i)
      f[i] = ((i ? tA[i - 1] : 0) + (n - 1) * tA[i]) % mod;
  else
    for (int i = 0; i <= n; ++i)
      f[i] = tA[i];
}

LL f[N];

int main() {
  int n;
  scanf("%d", &n);
  Init(n * 2);
  inv[1] = 1;
  for (int i = 2; i <= n; ++i) inv[i] = -(mod / i) * inv[mod % i] % mod;
  fac[0] = ifac[0] = 1;
  for (int i = 1; i <= n; ++i) {
    fac[i] = fac[i - 1] * i % mod;
    ifac[i] = ifac[i - 1] * inv[i] % mod;
  }
  Solve(n, f);
  for (int i = 0; i <= n; ++i)
    printf("%lld ", (f[i] + mod) % mod);
  return 0;
}
```

6.43 第一类斯特林数

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

#define Int register int
#define mod 167772161
#define MAXN 531072
#define Gi 3

int quick_pow (int a,int b,int c)
{
    int res = 1;
    while (b){
        if (b & 1) res = 1ll * res * a % c;
        a = 1ll * a * a % c;
        b >>= 1;
    }
    return res;
}
```

```
19
20  int limit = 1,l,r[MAXN];
21
22  void NTT (int *a,int type)
23  {
24      for (Int i = 0;i < limit;++ i) if (i < r[i]) swap (a[i],a[r[i]]);
25      for (Int mid = 1;mid < limit;mid <<= 1){
26          int Wn = quick_pow (Gi,(mod - 1) / (mid << 1),mod);
27          if (type == -1) Wn = quick_pow (Wn,mod - 2,mod);
28          for (Int R = mid << 1,j = 0;j < limit;j += R){
29              for (Int k = 0,w = 1;k < mid;++ k,w = 1ll * w * Wn % mod)
30              {
31                  int x = a[j + k],y = 1ll * w * a[j + k + mid] % mod;
32                  a[j + k] = (x + y) % mod,a[j + k + mid] = (x + mod - y) % mod;
33              }
34          }
35      }
36      if (type == 1) return ;
37      int Inv = quick_pow (limit,mod - 2,mod);
38      for (Int i = 0;i < limit;++ i) a[i] = 1ll * a[i] * Inv % mod;
39  }
40
41  int c[MAXN];
42
43  void Solve (int len,int *a,int *b)
44  {
45      if (len == 1) return b[0] = quick_pow (a[0],mod - 2,mod),void ();
46      Solve ((len + 1) >> 1,a,b);
47      limit = 1,l = 0;
48      while (limit < (len << 1)) limit <<= 1,l ++;
49      for (Int i = 0;i < limit;++ i) r[i] = (r[i >> 1] >> 1) | ((i & 1) << (l - 1));
50      for (Int i = 0;i < len;++ i) c[i] = a[i];
51      for (Int i = len;i < limit;++ i) c[i] = 0;
52      NTT (c,1);NTT (b,1);
53      for (Int i = 0;i < limit;++ i) b[i] = 1ll * b[i] * (2 + mod - 1ll * c[i] * b[i] %
        mod) % mod;
54      NTT (b,-1);
55      for (Int i = len;i < limit;++ i) b[i] = 0;
56  }
57
58  void deravitive (int *a,int n){
59      for (Int i = 1;i <= n;++ i) a[i - 1] = 1ll * a[i] * i % mod;
60      a[n] = 0;
61  }
62
63  void inter (int *a,int n){
64      for (Int i = n;i >= 1;-- i) a[i] = 1ll * a[i - 1] * quick_pow (i,mod - 2,mod) % mod
        ;
65      a[0] = 0;
66  }
67
68  int b[MAXN];
69
70  void Ln (int *a,int n){
71      memset (b,0,sizeof (b));
72      Solve (n,a,b);deravitive (a,n);
73      while (limit <= n) limit <<= 1,l ++;
74      for (Int i = 0;i < limit;++ i) r[i] = (r[i >> 1] >> 1) | ((i & 1) << (l - 1));
75      NTT (a,1),NTT (b,1);
```

```
76      for (Int i = 0;i < limit;++ i) a[i] = 1ll * a[i] * b[i] % mod;
77      NTT (a,-1);
78      inter (a,n);
79      for (Int i = n + 1;i < limit;++ i) a[i] = 0;
80  }
81
82  int F0[MAXN];
83
84  void Exp (int *a,int *B,int n){
85      if (n == 1) return B[0] = 1,void ();
86      Exp (a,B,(n + 1) >> 1);
87      for (Int i = 0;i < limit;++ i) F0[i] = B[i];
88      Ln (F0,n);
89      F0[0] = (a[0] + 1 + mod - F0[0]) % mod;
90      for (Int i = 1;i < n;++ i) F0[i] = (a[i] + mod - F0[i]) % mod;
91      NTT (F0,1);NTT (B,1);
92      for (Int i = 0;i < limit;++ i) B[i] = 1ll * F0[i] * B[i] % mod;
93      NTT (B,-1);
94      for (Int i = n;i < limit;++ i) B[i] = 0;
95  }
96
97  int read ()
98  {
99      int x = 0;char c = getchar();int f = 1;
100     while (c < '0' || c > '9'){if (c == '-') f = -f;c = getchar();}
101     while (c >= '0' && c <= '9'){x = (int)((int)(x << 3) % mod + (int)(x << 1) % mod +
        c - '0') % mod;c = getchar();}
102     return x * f;
103 }
104
105 void write (int x)
106 {
107     if (x < 0){x = -x;putchar ('-');}
108     if (x > 9) write (x / 10);
109     putchar (x % 10 + '0');
110 }
111
112 int n,k;
113 int fac[MAXN],A[MAXN],B[MAXN];
114
115 signed main()
116 {
117     n = read (),k = read ();
118     for (Int i = 0;i < n;++ i) A[i] = quick_pow (i + 1,mod - 2,mod);
119     Ln (A,n);
120     for (Int i = 0;i < n;++ i) A[i] = 1ll * A[i] * k % mod;
121     Exp (A,B,n);fac[0] = 1;
122     for (Int i = 1;i <= max (n,k);++ i) fac[i] = 1ll * fac[i - 1] * i % mod;
123     for (Int i = n;i >= k;-- i) B[i] = B[i - k];
124     for (Int i = 0;i < k;++ i) B[i] = 0;int Inv = quick_pow (fac[k],mod - 2,mod);
125     for (Int i = 0;i <= n;++ i) write (1ll * B[i] * fac[i] % mod * Inv % mod),putchar (
        ' ');
126     putchar ('\n');
127     return 0;
128 }
```

6.44 整数拆分多项式求逆

```

// NTT求法，任意模数复杂度较高
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 1e5 + 10;

struct Complex {
    double x, y;
    Complex(double a = 0, double b = 0): x(a), y(b) {}
    Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
    Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
    Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
     rhs.y + y * rhs.x); }
    Complex conj() { return Complex(x, -y); }
} w[N];

ll mod, g;
int tr[N];
ll F[N], G[N];

ll quick_pow(ll a, ll b) ;

int getLen(int n) ;

void NTT(ll *a, int len, int opt) ;

void mul(ll *a, ll *b, ll *z, int n) ;

void Get_Inv(ll *f, ll *g, int n) ;

void ChaiFen(ll *f, ll *g, int n) {
    int len = getLen(n);
    for(int i = 0;i < len; i++) {
        ll a = 1ll * i * (3 * i - 1) / 2; ll b = 1ll * i * (3 * i + 1) / 2;
        if(a > len && b > len) break;
        ll tmp;
        if(i & 1) tmp = mod - 1;
        else tmp = 1;
        if(a < len) f[a] = tmp;
        if(b < len) f[b] = tmp;
    }

    Get_Inv(f, g, n);
}

int main() {
    int n;
    cin >> n;
    ChaiFen(F, G, n);
    for(int i = 1;i <= n; i++) cout << G[i] << endl;
}
```

6.45 整数拆分生成函数

```

// O(n*sqrt(n))
```

```

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 1e5 + 5;
const ll mod = 1e9 + 7;
int f1[270], f2[270];

ll ans[N];

void init() {
    for (int i = 1; ; i++) {
        f1[i] = (3 * i * i - i) >> 1;
        if (f1[i] > 100000) break;
        f2[i] = (3 * i * i + i) >> 1;
        if (f2[i] > 100000) break;
    }
    ans[0] = 1;
    for (int i = 1; i <= 100000; i++) {
        for (int j = 1; ; j++) {
            if (f1[j] <= i) ans[i] += j & 1 ? ans[i-f1[j]] : -ans[i-f1[j]];
            else break;
            if (f2[j] <= i) ans[i] += j & 1 ? ans[i-f2[j]] : -ans[i-f2[j]];
            else break;
        }
        ans[i] = (ans[i] % mod + mod) % mod;
    }
}
```

6.46 普通型母函数

```
// 普通型母函数: (1+x^1+x^2+...)(1+x^2+x^4)(1+x^3+x^6..)(...)(...)... 类似整数拆分

// a_n=1,1,1,1... = \frac{1}{1-x}
// a_n=1,0,1,0... = \frac{1}{1-x^2}
// a_n=1,2,3,4... = \frac{1}{(1-x)^2}
// a_n=C(m,n)     = (1+x)^m
// a_n=C(m+n,n)   = \frac{1}{(1-x)^{m+1}}

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

typedef long long ll;

// 求解硬币等普通问题

const int N = 1e5 + 10;

int a[N]; // 权重为i的组合数，a[P]为答案
int b[N]; // 辅助数组
int P; // 需要被分解的数
int k; // 物品个数
int v[N]; // 每个物品的权重
int n1[N]; // 对于每种物品起始的因子（所需要的每个物品最小个数），最小为0
int n2[N]; // 对于每种物品最终的因子（所需要的每个物品最大个数），最大为INF
```

```

// 模板一(标准)

void Calc1() {
    memset(a, 0, sizeof(a));
    a[0] = 1;

    for(int i = 1;i <= k; i++) { // 枚举每个物品因子
        memset(b, 0, sizeof(b));
        for(int j = n1[i];j <= n2[i] && j * v[i] <= P; j++) { // 每个物品从最小因子到最大因子循环,如果n2是无穷的，则j<=n2[i]可以删去
            for(int m = 0;m + j * v[i] <= P; m++) { // 循环a的每个项
                b[m + j * v[i]] += a[m]; // 把结果加到对应项里，有点dp的味道
            }
        }
        memcpy(a, b, sizeof(b));
    }
}

// 模板二（数据量大的时候可以用，快速）

void Calc2() {
    memset(a, 0, sizeof(a));
    a[0] = 1;
    int last = 0;
    for(int i = 1; i <= k; i++) {
        int last2 = min(last + n2[i] * v[i], P);//计算下一次的last
        memset(b, 0, sizeof(int) * (last2 + 1));//只清空b[0..last2]
        for(int j = n1[i]; j <= n2[i] && j * v[i] <= last2; j++) //last2
            for(int m = 0; m <= last && m + j * v[i] <= last2; m++) //一个是last，一个是last2
                b[m + j * v[i]] += a[m];
        memcpy(a, b, sizeof(int) * (last2 + 1));//b赋值给a，只赋值0..last2
        last = last2;//更新last
    }
}
```

6.47 指数型母函数

```

// 需要借助e^x的泰勒展开，一般求解多重排列数，即有 种物品，已知每种物品的数量为 k1,k2,...,kn 个，求从中选出m件物品的排列数。

// 对n个元素全排列，方案数为n!/(n1!n2!...nk!)，对n个中的r个元素进行全排列，这里就用到了指数型母函数，即G(x)=(1+x/1!+x^2/2!+...+x^k1/k1!)(1+x/1!+x^2/2!+...+x^k2/k2!)...(1+x/1!+x^2/2!+...+x^kn/kn!)

// 化简得G(x)=a0 + a1*x+a2*x^2/2!+...+ap*x^p/p!   (p = k1+k2+k3+...) ai为选出i个物品的排列方案数

//  若题目有规定条件，比如需要物品i出现非0的偶数次，即原式为(x^2/2!+x^4/4!+...+x^ki/ki!)

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

typedef long double ld;

double num[15]; // 每种物品的数量，第i个物品有num[i]个
```

```

double a[15], b[15];

double f[120]; // 阶乘

void fac()
{
    f[0] = 1;
    for(int i = 1;i <= 105; i++)
        f[i] = f[i - 1] * i;
}

void Calc() {
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= n; i++)
        cin >> num[i];

    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));

    for(int i = 0;i <= num[1]; i++) {
        a[i] = 1.0 / f[i];
    }

    for(int i = 2;i <= n; i++) {
        for(int j = 0;j <= m; j++) {
            for(int k = 0;k <= num[i] && j + k <= m; k++) {
                b[j + k] += a[j] / f[k];
            }
        }

        for(int j = 0;j <= m; j++) {
            a[j] = b[j];
            b[j] = 0;
        }
    }

    cout << a[m] * f[m] << endl;
}
```

6.48 伯努利数求和

```
namespace BNL {
    const int N = 1e7 + 10, M = 1e6 + 10;
    struct Complex {
        double x, y;
        Complex(double a = 0, double b = 0): x(a), y(b) {}
        Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y);
    }
        Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y);
    }
        Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y,
    x * rhs.y + y * rhs.x); }
        Complex conj() { return Complex(x, -y); }
    } w[N];

    int tr[N];
```

```
13      ll F[N], G[N];
14
15      ll quick_pow(ll a, ll b, ll p) {
16          ll ans = 1;
17          while(b) {
18              if(b & 1) ans = ans * a % p;
19              a = a * a % p;
20              b >>= 1;
21          }
22          return ans % p;
23      }
24
25      void FFT(Complex *A, int len) {
26          for (int i = 0; i < len; i++) if(i < tr[i]) swap(A[i], A[tr[i]]);
27          for (int i = 2, lyc = len >> 1; i <= len; i <<= 1, lyc >>= 1)
28              for (int j = 0; j < len; j += i) {
29                  Complex *l = A + j, *r = A + j + (i >> 1), *p = w;
30                  for (int k = 0; k < i >> 1; k++) {
31                      Complex tmp = *r * *p;
32                      *r = *l - tmp, *l = *l + tmp;
33                      ++l, ++r, p += lyc;
34                  }
35              }
36      }
37
38      inline void MTT(ll *x, ll *y, ll *z, int n) {
39          int len = 1; while (len <= n) len <<= 1;
40          for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
        0);
41          for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
        (2 * PI * i / len));
42
43          for (int i = 0; i < len; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
44          static Complex a[N], b[N];
45          static Complex dfta[N], dftb[N], dftc[N], dftd[N];
46
47          for (int i = 0; i < len; i++) a[i] = Complex(x[i] & 32767, x[i] >> 15);
48          for (int i = 0; i < len; i++) b[i] = Complex(y[i] & 32767, y[i] >> 15);
49          FFT(a, len), FFT(b, len);
50          for (int i = 0; i < len; i++) {
51              int j = (len - i) & (len - 1);
52              static Complex da, db, dc, dd;
53              da = (a[i] + a[j].conj()) * Complex(0.5, 0);
54              db = (a[i] - a[j].conj()) * Complex(0, -0.5);
55              dc = (b[i] + b[j].conj()) * Complex(0.5, 0);
56              dd = (b[i] - b[j].conj()) * Complex(0, -0.5);
57              dfta[j] = da * dc;
58              dftb[j] = da * dd;
59              dftc[j] = db * dc;
60              dftd[j] = db * dd;
61          }
62          for (int i = 0; i < len; i++) a[i] = dfta[i] + dftb[i] * Complex(0, 1);
63          for (int i = 0; i < len; i++) b[i] = dftc[i] + dftd[i] * Complex(0, 1);
64          FFT(a, len), FFT(b, len);
65          for (int i = 0; i < len; i++) {
66              int da = (ll)(a[i].x / len + 0.5) % mod;
67              int db = (ll)(a[i].y / len + 0.5) % mod;
68              int dc = (ll)(b[i].x / len + 0.5) % mod;
69              int dd = (ll)(b[i].y / len + 0.5) % mod;
```

```
70              z[i] = (da + ((ll)(db + dc) << 15) + ((ll)dd << 30)) % mod;
71          }
72      }
73
74      int getLen(int n) {
75          int len = 1; while (len < (n << 1)) len <<= 1;
76          for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
        0);
77          for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
        (2 * PI * i / len));
78          return len;
79      }
80
81      void Get_Inv(ll *f, ll *g, int n) {
82          if(n == 1) { g[0] = quick_pow(f[0], mod - 2, mod); return ; }
83          Get_Inv(f, g, (n + 1) >> 1);
84          int len = getLen(n);
85          static ll c[N];
86          for(int i = 0;i < len; i++) c[i] = i < n ? f[i] : 0;
87          MTT(c, g, c, len); MTT(c, g, c, len);
88          for(int i = 0;i < n; i++) g[i] = (2ll * g[i] - c[i] + mod) % mod;
89          for(int i = n;i < len; i++) g[i] = 0;
90          for(int i = 0;i < len; i++) c[i] = 0;
91      }
92
93      ll ff[N], invff[N], inv[N];
94      ll B[N];
95
96      ll C(ll m, ll n) {
97          if(m < 0 || n < 0 || n > m)
98              return 0;
99          ll ans = ff[m];
100         ans = ans * invff[n] % mod;
101         ans = ans * invff[m - n] % mod;
102         return ans;
103     }
104
105     void init(int m) {
106         ff[0] = ff[1] = inv[0] = inv[1] = invff[0] = invff[1] = 1;
107         for(int i = 2;i < M; i++)
108         {
109             ff[i] = ff[i - 1] * i % mod;
110             inv[i] = mod - (mod / i) * inv[mod % i] % mod;
111             invff[i] = invff[i - 1] * inv[i] % mod;
112         }
113
114         for(int i = 0;i <= m + 10; i++) F[i] = invff[i + 1];
115         Get_Inv(F, G, m + 10);
116         for(int i = 0;i <= m + 10; i++) B[i] = G[i] * ff[i] % mod;
117     }
118
119     ll solve(ll n, int k) {
120         init(k);
121         ll ans = 0, prod = n % mod;
122         for(int i = k; ~i ; i--) {
123             ans = (ans + prod * B[i] % mod * C(k + 1, i) % mod) % mod;
124             prod = prod * n % mod;
125         }
126         ans = ans * quick_pow(k + 1, mod - 2, mod) % mod;
```

```
        return ans;
    }
}

void solve() {
    ll n; int k; cin >> n >> k;
    cout << BNL::solve(n + 1, k) << endl;
}
```

6.49 斐波那契大数

```
int a[505][505];

void solve() {
    memset(a,0,sizeof(a));
    a[1][1]  = 1;
    a[2][1] = 1;
    int s,plus = 0;
    for(int i = 3;i < 491;i++){
        for(int j = 1;j < 491;j++){
            s = a[i-2][j]+a[i-1][j]+plus;
            a[i][j] = s % 10;
            plus = s / 10;
        }
    }
    int n;
    while(scanf("%d",&n) && n != -1) {
        int k = 491;
        while(1) {
            if(a[n][k]) break;
            k--;
        }
        while(k >= 1) {
            printf("%d",a[n][k]);
            k--;
        }
    }
}
```

6.50 斐波那契循环节

```
// 斐波那契循环节

typedef long long ll;
typedef long double ld;
typedef pair<int, int> pdd;

#define INF 0x7f7f7f
#define mem(a,b) memset(a , b , sizeof(a))
#define FOR(i, x, n) for(int i = x;i <= n; i++)

const ll mod = 1e9 + 7;
const int maxn = 5e6 + 10;

bool is_prime[maxn];
ll prime[maxn];
int p;
```

```
17
18  void sieve() // 素数筛
19  {
20      p = 0;
21      mem(is_prime, true);
22      is_prime[0] = is_prime[1] = false;
23      for(int i = 2;i < maxn; i++)
24      {
25          if(is_prime[i])
26          {
27              prime[++p] = i;
28              for(int j = i + i;j < maxn; j += i)
29              {
30                  is_prime[j] = false;
31              }
32          }
33      }
34  }
35
36  ll gcd(ll a, ll b)
37  {
38      return b ? gcd(b, a % b) : a;
39  }
40
41  ll quick_pow(ll a, ll b)
42  {
43      a %= mod;
44      ll ans = 1;
45      ll base = a;
46      while(b)
47      {
48          if(b&1)
49          {
50              ans = ans * base % mod;
51          }
52          base = base * base % mod;
53          b >>= 1;
54      }
55      return ans % mod;
56  }
57
58  ll num[maxn]; // 所有质数的循环节
59  ll f[maxn]; // 斐波那契数列
60
61  void Fib_Cyclic_node()
62  {
63      num[1] = 3;
64      for(int i = 2;i <= p; i++) // 找每个素数的循环节num[1~p]
65      {
66          f[1] = 1;
67          f[2] = 2;
68          int x = 3;
69          while(true)
70          {
71              f[x] = f[x - 1] + f[x - 2];
72              f[x] %= prime[i];
73              if(f[x] == 1 && f[x - 1] == 0) // f[x] % prime[i] == f[1]
74                  break;
75              x++;
```

```
76          }
77          num[i] = x;
78      }
79
80      ll n;
81      cin >> n; // 如果是质数，那循环节就是num[n] ;   如果是合数，那循环节就是n的素因子的最小公倍数
82      ll ans = 1;
83      ll x;
84      for(int i = 1;i <= p; i++)
85      {
86          if(n % prime[i] == 0)
87          {
88              x = 0;
89              while(n % prime[i] == 0)
90              {
91                  n /= prime[i];
92                  x++;
93              }
94          }
95          ll k = num[i] * quick_pow(prime[i], x - 1);
96          ans = ans * k / gcd(ans, k); // 最小公倍数
97      }
98      cout << ans << endl; // 最小循环节
99  }
100
101 // 广义斐波那契循环节
102
103 // fib(n) = a * fib(n - 1) + b * fib(n - 2)
104 // fib(1) = c   fib(2) = d
105 // 求f(n) mod p的循环节
106 //c = a * a - 4b是模p的二次剩余时枚举p-1的因子，否则枚举(p+1)(p-1)的因子
107
108 typedef long long ll;
109 typedef long double ld;
110 typedef pair<int, int> pdd;
111
112 #define INF 0x7f7f7f
113 #define mem(a,b) memset(a , b , sizeof(a))
114 #define FOR(i, x, n) for(int i = x;i <= n; i++)
115
116 ll fac[2][505];
117 ll cnt, ct;
118
119 ll pri[6] = {2, 3, 7, 109, 167, 500000003};
120 ll num[6] = {4, 2, 1, 2, 1, 1};
121
122 const ll mod = 1e9 + 7;
123 const int maxn = 5e6 + 10;
124
125 struct Matrix{
126     ll m[2][2];
127 };
128
129 Matrix A;
130
131 Matrix I = {1, 0, 0 , 1}; // 单位矩阵
132
133 Matrix multi(Matrix a, Matrix b) // 矩阵乘法
134 {
```

```
135     Matrix C;
136     for(int i = 0;i < 2; i++)
137     {
138         for(int j = 0;j < 2; j++)
139         {
140             C.m[i][j] = 0;
141             for(int k = 0;k < 2; k++)
142             {
143                 C.m[i][j] = (C.m[i][j] + a.m[i][k] * b.m[k][j] % mod) % mod;
144             }
145             C.m[i][j] %= mod;
146         }
147     }
148     return C;
149 }
150 
151 Matrix quick_Matrix(Matrix a, ll b) // 矩阵快速幂
152 {
153     Matrix ans = I, base = a;
154     while(b)
155     {
156         if(b & 1)
157         {
158             ans = multi(a, base);
159         }
160         base = multi(base, base);
161         b >>= 1;
162     }
163     return ans;
164 }
165 
166 ll quick_pow(ll a, ll b) ;
167 
168 ll legendre(ll a, ll p) // 勒让德符号 = {1, -1, 0}
169 {
170     ll k = quick_pow(a, (p - 1) >> 1);
171     if(k == 1)
172         return 1;
173     else
174         return -1;
175 }
176 
177 void DFS(int dept,ll product = 1)
178 {
179     if(dept == cnt)
180     {
181         fac[1][ct++] = product;
182         return;
183     }
184     for(int i=0; i<=num[dept]; i++)
185     {
186         DFS(dept+1,product);
187         product *= pri[dept];
188     }
189 }
190 
191 bool Fib_node(Matrix a, ll n) // n是否为循环节
192 {
193     Matrix ans = quick_Matrix(a, n);
```

```
194     return (ans.m[0][0] == 1 && ans.m[0][1] == 0 && ans.m[1][0] == 0 && ans.m[1][1] ==
        1); // 是否为单位矩阵I
195 }
196
197 ll Fib_Cyclic_node(ll a, ll b, ll c, ll d) // 广义斐波那契循环节斐波那契循环节
198 {
199     fac[0][0] = 1;
200     fac[0][1] = 2;
201     fac[0][3] = 500000003;
202     fac[0][3] = 1000000006;
203     ll c = a * a - 4 * b;
204     A.m[0][0] = a;
205     A.m[0][1] = b;
206     A.m[1][0] = 1;
207     A.m[1][1] = 0;
208     if(legendre(c, mod) == 1) // c是否为1e9+7的二次剩余
209     {
210         for(int i = 0;i < 4; i++)
211         {
212             if(Fib_node(A, fac[0][i]))
213                 return fac[0][i];
214         }
215     }
216     else
217     {
218         ct = 0;
219         cnt = 6;
220         DFS(0, 1);
221         sort(fac[1], fac[1] + ct);
222         for(int i = 0;i < ct; i++)
223         {
224             if(Fib_node(A, fac[1][i]))
225                 return fac[1][i];
226         }
227     }
228
229 }
230
231 int main()
232 {
233     ll a, b, c, d;
234     cin >> a >> b >> c >> d;
235     ll n = Fib_Cyclic_node(a, b, c, d); // 广义斐波那契循环节循环节长度
236     cout << n << endl;
237 }
```

6.51 矩阵快速幂求解

```
1  // 斐波那契循环节
2
3  typedef long long ll;
4
5  const ll mod = 1e9 + 7;
6
7  struct Martix {
8      int n, m;
9      ll a[70][70];
10     Martix operator * (const Martix& rhs) const {
```

```
        Martix ans;
        memset(&ans, 0, sizeof(ans));
        ans.n = n; ans.m = rhs.m;
        for(int i = 0;i <= ans.n; i++) {
            for(int j = 0;j <= ans.m; j++) {
                for(int k = 0;k <= m; k++) {
                    ans.a[i][j] = (ans.a[i][j] + a[i][j] * rhs.a[i][j] % mod) % mod;
                }
            }
        }
        return ans;
    }
    Martix() {n = m = 0; memset(a, 0, sizeof(a)); }
    void operator *= (const Martix& rhs) {*this = *this * rhs; }
};

struct Matrix{
    ll m[2][2];
};

Matrix ans;

Matrix I = {1, 1, 1 , 0}; // 单位矩阵

Matrix multi(Matrix a, Matrix b) {
    Matrix C;
    for(int i = 0;i < 2; i++) {
        for(int j = 0;j < 2; j++) {
            C.m[i][j] = 0;
            for(int k = 0;k < 2; k++) {
                C.m[i][j] = (C.m[i][j] + a.m[i][k] * b.m[k][j] % mod) % mod;
            }
            C.m[i][j] %= mod;
        }
    }
    return C;
}

void quick_Matrix(ll n) {
    Matrix base = I;
    ans.m[0][0] = 1;
    ans.m[1][1] = 1;
    while(n) {
        if(n & 1) ans = multi(ans, base);
        base = multi(base, base);
        n >>= 1;
    }
}

int main()
{
    ll n;
    cin >> n;
    quick_Matrix(n);
    cout << ans.m[0][1] << endl;
}
```

6.52 拉格朗日插值求和

```
namespace polysum {
#define rep(i, a, n) for (int i=a;i<n;i++)
#define per(i, a, n) for (int i=n-1;i>=a;i--)
    const int D = 1010000;///可能需要用到的最高次
    ll a[D], f[D], g[D], p[D], p1[D], p2[D], b[D], h[D][2], C[D], num[D];

    ll powmod(ll a, ll b) {
        ll res = 1;
        a %= mod;
        assert(b >= 0);
        for (; b; b >>= 1) {
            if (b & 1)res = res * a % mod;
            a = a * a % mod;
        }
        return res;
    }

    ///函数用途：给出数列的（d+1）项，其中d为最高次方项
    ///求出数列的第n项，数组下标从0开始
    ll calcn(int d, ll *a, ll n) { /// a[0].. a[d]  a[n]
        if (n <= d) return a[n];
        p1[0] = p2[0] = 1;
        rep(i, 0, d + 1) {
            ll t = (n - i + mod) % mod;
            p1[i + 1] = p1[i] * t % mod;
        }
        rep(i, 0, d + 1) {
            ll t = (n - d + i + mod) % mod;
            p2[i + 1] = p2[i] * t % mod;
        }
        ll ans = 0;
        rep(i, 0, d + 1) {
            ll t = g[i] * g[d - i] % mod * p1[i] % mod * p2[d - i] % mod * a[i] % mod;
            if ((d - i) & 1) ans = (ans - t + mod) % mod;
            else ans = (ans + t) % mod;
        }
        return ans;
    }

    void init(int M) {///用到的最高次
        f[0] = f[1] = g[0] = g[1] = 1;
        rep(i, 2, M + 5) f[i] = f[i - 1] * i % mod;
        g[M + 4] = powmod(f[M + 4], mod - 2);
        per(i, 1, M + 4) g[i] = g[i + 1] * (i + 1) % mod;///费马小定理筛逆元
    }

    ///函数用途：给出数列的（m+1）项，其中m为最高次方
    ///求出数列的前（n-1）项的和（从第0项开始）
    ll polysum(ll m, ll *a, ll n) { /// a[0].. a[m] \sum_{i=0}^{n-1} a[i]
        for (int i = 0; i <= m; i++) b[i] = a[i];

        ///前n项和，其最高次幂加1
        b[m + 1] = calcn(m, b, m + 1);
        rep(i, 1, m + 2) b[i] = (b[i - 1] + b[i]) % mod;
        return calcn(m + 1, b, n - 1);
    }

```

```
58      ll qpolysum(ll R, ll n, ll *a, ll m) { /// a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
59          if (R == 1) return polysum(n, a, m);
60          a[m + 1] = calcn(m, a, m + 1);
61          ll r = powmod(R, mod - 2), p3 = 0, p4 = 0, c, ans;
62          h[0][0] = 0;
63          h[0][1] = 1;
64          rep(i, 1, m + 2) {
65              h[i][0] = (h[i - 1][0] + a[i - 1]) * r % mod;
66              h[i][1] = h[i - 1][1] * r % mod;
67          }
68          rep(i, 0, m + 2) {
69              ll t = g[i] * g[m + 1 - i] % mod;
70              if (i & 1) p3 = ((p3 - h[i][0] * t) % mod + mod) % mod, p4 = ((p4 - h[i][1]
        * t) % mod + mod) % mod;
71              else p3 = (p3 + h[i][0] * t) % mod, p4 = (p4 + h[i][1] * t) % mod;
72          }
73          c = powmod(p4, mod - 2) * (mod - p3) % mod;
74          rep(i, 0, m + 2) h[i][0] = (h[i][0] + h[i][1] * c) % mod;
75          rep(i, 0, m + 2) C[i] = h[i][0];
76          ans = (calcn(m, C, n) * powmod(R, n) - c) % mod;
77          if (ans < 0) ans += mod;
78          return ans;
79      }
80
81      ll solve(ll n, int k) {
82          init(k + 10);
83          for (int i = 0; i <= k + 1; i++) num[i] = powmod((ll) i + 1, k);
84          ll ans = polysum(k + 1, num, n) % mod;
85          return ans;
86      }
87  }
88
89  void solve() {
90      ll n;
91      int k;
92      cin >> n >> k;
93      cout << polysum::solve(n, k) << endl;
94  }
```

6.53 四方定理

```
1
2   // 暴力穷举查找
3
4   void square(int n)
5   {
6       int ans = 0;
7       for(int i = 1;i <= n / 2; i++)
8           for(int j = 0;j <= i; j++)
9               for(int k = 0;k <= j; k++)
10                  for(int q = 0;q <= k; q++)
11                      if(i * i + j * j + k * k + q * q == n)
12                          ans++;
13  }
14
15
16  // 动规背包优化
17
```

```
const int N = 1e5 + 10;

int dp[5][N];

void square() {
    dp[0][0] = 1;
    for(int i = 1;i <= 200; i++) {
        for(int j = 1;j <= 4; j++) {
            for(int k = 1;k <= 32768; k++) {
                if(i * i <= k)
                    dp[j][k] += dp[j - 1][k - i * i];
            }
        }
    }
}
```

6.54 高斯消元

```
#include <bits/stdc++.h>
using namespace std;
const int N=210;
int a[N][N];//增广矩阵
int x[N];//解集
int freeX[N];//自由变元
// equ:方程个数 var:变量个数
int Gauss(int equ, int var){//返回自由变元个数
    /*初始化*/
    for(int i = 0;i <= var; i++){
        x[i] = 0;
        freeX[i] = 0;
    }

    /*转换为阶梯阵*/
    int col = 0;//当前处理的列
    int num = 0;//自由变元的序号
    int k;//当前处理的行
    for(k = 0;k < equ && col < var; k++, col++) {//枚举当前处理的行
        int maxr = k;//当前列绝对值最大的行
        for(int i = k + 1;i < equ; i++){//寻找当前列绝对值最大的行
            if(a[i][col] > a[maxr][col]){
                maxr = i;
                swap(a[k], a[maxr]);//与第k行交换
                break;
            }
        }
        if(a[k][col] == 0){//col列第k行以下全是0，处理当前行的下一列
            freeX[num++] = col;//记录自由变元
            k--;
            continue;
        }

        for(int i = k + 1;i < equ; i++){
            if(a[i][col] != 0){
                for(int j = col;j < var + 1; j++){//对于下面出现该列中有1的行，需要把1消掉
                    a[i][j] ^= a[k][j];
                }
            }
        }
```

```
    }

    /*求解*/
    //无解: 化简的增广阵中存在(0,0,...,a)这样的行，且a!=0
    for(int i = k;i < equ; i++)
        if(a[i][col] != 0)
            return -1;

    //无穷解: 在var*(var+1)的增广阵中出现(0,0,...,0)这样的行
    if(k < var)//返回自由变元数
        return var - k;//自由变元有var-k个

    //唯一解: 在var*(var+1)的增广阵中形成严格的上三角阵
    for(int i = var - 1;i >= 0; i--){//计算解集
        x[i] = a[i][var];
        for(int j = i + 1;j < var; j++)
            x[i] ^= (a[i][j] && x[j]);
    }
    return 0;
}
```

6.55 矩阵求逆

```
//原始矩阵A[0, n - 1][0, n - 1]
//右边一个单位阵I, 在a[0, n - 1][n, (n << 1) - 1]
//将左边A变成单位阵时，右边的I变为A^-1

ll a[MAX][MAX];
bool Gauss(int n) {
    for (int i = 0, r; i < n; i++) {
        r = i;
        for (int j = i + 1; j < n; j++)
            if (a[j][i] > a[r][i]) r = j;
        if (r != i) swap(a[i], a[r]);
        if (!a[i][i]) return false;//无解

        ll inv = qpow(a[i][i], mod - 2);
        for (int k = 0; k < n; k++) {
            if (k == i) continue;
            ll t = a[k][i] * inv % mod;
            for (int j = i; j < (n << 1); j++)
                a[k][j] = (a[k][j] - t * a[i][j] % mod + mod) % mod;
        }
        for (int j = 0; j < (n << 1); j++) a[i][j] = a[i][j] * inv % mod;
    }
    return true;
}

int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        a[i][i + n] = 1;
        for (int j = 0; j < n; j++)
            scanf("%lld", &a[i][j]);
    }
    if(Gauss(n)) {
        for(int i = 0;i < n; i++) {
            for(int j = n;j < n * 2; j++) {
```

```
                cout << a[i][j] << " ";
            }
            cout << endl;
        }
    }
    else cout << "No Solution" << endl;
}
```

6.56 可删除线性基

```
// 离线删除操作，维护线性基中每个元素的最晚删除时间。

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

typedef long long ll;
const int maxl = 60;

struct LinearBasis {
    ll a[maxl + 10], tim[maxl + 10];
    int n, size; // 每个相同异或值有2^{n-size}个
    vector<ll> v;

    LinearBasis() {
        memset(a, 0, sizeof(a));
        size = n = 0;
        v.clear();
    }

    void insert(ll x, ll t) {
        n++;
        for(int i = maxl;i >= 0; i--) {
            if(!(x >> i & 1)) continue ;
            if(a[i]) {
                if(t > tim[i]) swap(t, tim[i]), swap(x, a[i]);
                x ^= a[i];
            }
            else {
                ++size;
                a[i] = x; tim[i] = t;
                return ;
            }
        }
    }

    void erase(ll t) {
        for(int i = maxl;i >= 0; i--) {
            if(tim[i] == t) {
                a[i] = tim[i] = 0; --size;
                return ;
            }
        }
    }
};

int main() {
    LinearBasis lb;
    int n, m; cin >> n >> m;
```

```
    vector<ll> opt(n + 10), a(n + 10), del(n + 10), pre(n + 10);
    for(int i = 1;i <= m; i++) {
        cin >> opt[i] >> a[i];
        if(opt[i] == 1) pre[a[i]] = i, del[i] = m + 1;
        else del[pre[a[i]]] = i;
    }
    ll ans = 0;
    for(int i = 1;i <= m; i++) {
        if(opt[i] == 1) lb.insert(a[i], del[i]);
        else lb.erase(i);
        ans ^= 1ll << (lb.n - lb.size);
    }
    cout << ans << endl;
}
```

6.57 普通线性基

```
// 每个异或值都相同的个数都为2^n-r,所以不同的异或值有2^r个.
#include <bits/stdc++.h>
using namespace  std;

typedef long long ll;
const int maxl = 60;
ll quick_pow(ll a, ll b) ;
struct LinearBasis {
    ll a[maxl + 10];
    int n, size; // 每个相同异或值有2^{n-size}个
    vector<ll> v;

    LinearBasis() {
        memset(a, 0, sizeof(a));
        size = n = 0;
        v.clear();
    }

    void insert(ll t) {
        n++;
        for (int i = maxl; i >= 0; --i) {
            if (!(t >> i & 1)) continue;
            if (a[i]) t ^= a[i];
            else {
                ++size;
                // Rebuild
                for (int j = i - 1; j >= 0; j--) if (t >> j & 1) t ^= a[j];
                for (int j = i + 1; j <= maxl; ++j) if (a[j] >> i & 1) a[j] ^= t;
                //
                a[i] = t;
                return;
            }
        }
    }

    // 与Rebuild配套使用
    void basis() {
        for (int i = 0; i <= maxl; ++i) if (a[i]) v.push_back(i);
    }

    // 查询能否xor出x这个数
```

```
42      bool find(ll x) {
43          for(int i = maxl;i >= 0; i--) {
44              if(x >> i & 1) {
45                  if(!a[i]) return 0;
46                  x ^= a[i];
47              }
48          }
49          return 1;
50      }
51
52      // 查询异或最大值
53      ll askmax() {
54          ll ans = 0;
55          for(int i = maxl;i >= 0; i--) ans = max(ans, ans ^ a[i]);
56          return ans;
57      }
58
59      // 查询异或最小值
60      ll askmin() {
61          for(int i = 0;i <= maxl; i++) if(a[i]) return a[i];
62          return 0;
63      }
64
65      // 查询异或第k大
66      ll askmaxk(ll x) {
67
68      }
69
70      // 查询异或第k小
71      ll askmink(ll x) {
72          if(v.size() != n) x--;
73          if(!x) return 0;
74          if(x >= (1ll << v.size())) return -1;
75          ll ans = 0;
76          for(int i = 0;i < v.size(); i++) {
77              if(x >> i & 1) ans ^= a[v[i]];
78          }
79          return ans;
80      }
81
82      ll rank(ll x) {
83          ll ret = 0;
84          for (int i = 0; i < v.size(); ++i) if (x >> v[i] & 1) ret += 1LL << i;
85          return ret;
86      }
87  };
88
89
90  void solve() {
91      int n, x, q;
92      scanf("%d", &n);
93      LinearBasis lb;
94      for (int i = 0; i < n; ++i) scanf("%d", &x), lb.insert(x);
95      lb.basis();
96      scanf("%d", &q);
97      ll num = quick_pow(2, n - lb.size);
98      printf("%lld\n", (lb.rank(q) * num + 1));
99  }
```

6.58 区间修改区间线性基

```
#include<bits/stdc++.h>
#define maxn 200005
using namespace std;
struct Base{
    int a[31],cnt;
    void clear(){memset(a,0,sizeof a),cnt=0;}
    void ins(int x){
        if(cnt==30) return;
        for(int i=29;i>=0&&x;i--) if(x>>i&1){
            if(a[i]) x^=a[i];
            else {a[i]=x,cnt++;return;}
        }
    }
    void merge(const Base &B){
        for(int i=29;i>=0;i--) if(B.a[i]) ins(B.a[i]);
    }
}t[maxn<<2];
int n,m,a[maxn],b[maxn];
int arr[maxn];
void upd(int i,int v){for(;i<=n;i+=i&-i) arr[i]^=v;}
int qxor(int i){int s=0;for(;i;i-=i&-i) s^=arr[i];return s;}
void upd(int i){t[i]=t[i<<1],t[i].merge(t[i<<1|1]);}
void build(int i,int l,int r){
    if(l==r) return t[i].ins(b[l]);
    int mid=(l+r)>>1;
    build(i<<1,l,mid),build(i<<1|1,mid+1,r);
    upd(i);
}
void mdf(int i,int l,int r,int x){
    if(l==r) {t[i].clear(),t[i].ins(b[x]);return;}
    int mid=(l+r)>>1;
    x<=mid?mdf(i<<1,l,mid,x):mdf(i<<1|1,mid+1,r,x);
    upd(i);
}
void qry(int i,int l,int r,int x,int y){
    if(x<=l&&r<=y) return t[0].merge(t[i]);
    int mid=(l+r)>>1;
    if(x<=mid) qry(i<<1,l,mid,x,y);
    if(y>mid) qry(i<<1|1,mid+1,r,x,y);
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]),b[i]=a[i]^a[i-1],upd(i,b[i]);
    build(1,1,n);
    for(int op,l,r,x;m--;){
        scanf("%d%d%d",&op,&l,&r);
        if(op==1){
            scanf("%d",&x);
            upd(l,x),upd(r+1,x),b[l]^=x,b[r+1]^=x;
            mdf(1,1,n,l); if(r<n) mdf(1,1,n,r+1);
        }
        else{
            t[0].clear(),t[0].ins(qxor(l)); if(l<r) qry(1,1,n,l+1,r);
            printf("%d\n",1<<t[0].cnt);
        }
    }
```

```
}
```

6.59 无修改区间线性基

```
// 扫描r，维护线性基中每个元素的最大左端点l。与删除操作类似。
// 这个可以强制在线，把每个r的线性基存下来即可。
#include <bits/stdc++.h>
using namespace std;

const int N = 2e5 + 10;
struct node {
    int l, r, id;
    bool operator < (const node &p) const {
        return r < p.r;
    }
}q[N];

const int maxl = 60;

struct LinearBasis {
    ll a[maxl + 10], pos[maxl + 10];
    int n, size; // 每个相同异或值有2^{n-size}个
    vector<ll> v;

    LinearBasis() {
        memset(a, 0, sizeof(a));
        size = n = 0;
        v.clear();
    }

    void insert(ll t, ll id) {
        n++;
        for(int i = maxl;i >= 0; i--) {
            if(!(t >> i & 1)) continue ;
            if(a[i]) {
                if(id > pos[i]) swap(id, pos[i]), swap(t, a[i]);
                t ^= a[i];
            }
            else {
                a[i] = t;
                pos[i] = id;
                return ;
            }
        }
    }

    int askmax(ll x) {
        ll ans = 0;
        for(int i = maxl;i >= 0; i--) {
//            if(pos[i] >= x && !(ans >> i & 1)) ans ^= a[i];
            if(pos[i] >= x) ans = max(ans, ans ^ a[i]);
        }
        return ans;
    }
};

// 给你n个数，每次查询［公式］这个区间，问着个区间的最大异或值。

```

```
55 void solve() {
56     LinearBasis lb;
57     int n; cin >> n;
58     VI a(n + 1);
59     for(int i = 1;i <= n; i++) cin >> a[i];
60     int m; cin >> m;
61     VI ans(m + 1);
62     for(int i = 1;i <= m; i++) cin >> q[i].l >> q[i].r, q[i].id = i;
63     sort(q + 1, q + m + 1);
64     for(int i = 1, j = 1;i <= n; i++) {
65         lb.insert(a[i], i);
66         for(; j <= m && q[j].r <= i; j++) ans[q[j].id] = lb.askmax(q[j].l);
67     }
68     for(int i = 1;i <= m; i++) cout << ans[i] << endl;
69 }
```

6.60 FFT

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const double PI = acos(-1);
4  const int N = 4e5 + 10;
5
6  struct Complex {
7      double a, b;
8      Complex(double a = 0, double b = 0): a(a), b(b) {}
9      Complex operator * (const Complex &rhs) { return Complex(a * rhs.a - b * rhs.b, a *
       rhs.b + b * rhs.a); }
10     Complex operator + (const Complex &rhs) { return Complex(a + rhs.a, b + rhs.b); }
11     Complex operator - (const Complex &rhs) { return Complex(a - rhs.a, b - rhs.b); }
12 };
13
14 int tr[N];
15
16 void FFT(Complex *A, int len, int type) {
17     for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
18     for (int i = 2; i <= len; i <<= 1) {//区间长度
19         int mid = i / 2;
20         Complex Wn(cos(2 * PI / i), type * sin(2 * PI / i));//单位根
21         for (int k = 0; k < len; k += i) {//每个子问题的起始点
22             Complex w(1, 0);//omega
23             for (int l = k; l < k + mid; l++) {
24                 Complex t = w * A[l + mid];
25                 A[l + mid] = A[l] - t;
26                 A[l] = A[l] + t;
27                 w = w * Wn;
28             }
29         }
30     }
31 }
32
33 void mul(ll *x, int n, ll *y, int m) {
34     static Complex a[N], b[N];
35     for(int i = 0;i <= n; i++) a[i].a = x[i];
36     for(int i = 0;i <= m; i++) b[i].a = y[i];
37     int len = 1; while (len <= (n + m)) len <<= 1;
38     for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
39     FFT(a, len, 1), FFT(b, len, 1);
```

```
    for (int i = 0; i < len; i++) a[i] = a[i] * b[i];
    FFT(a, len, -1);
    for (int i = 0; i < len; i++) x[i] = (ll)(a[i].a / len + 0.5);
}

ll a[N], b[N];

void solve() {
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 0; i <= n; i++) scanf("%lld", &a[i]);
    for (int i = 0; i <= m; i++) scanf("%lld", &b[i]);

    mul(a, n, b, m);
    for (int i = 0; i <= n + m; i++)
        printf("%lld ", a[i]);
}
```

6.61 FWT

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 998244353;

const int N = 1e5 + 10;
int a[N], b[N];

inline void FWT_OR(int *f, int n, int opt) {
    for (int o = 2, k = 1; o <= n; o <<= 1, k <<= 1)
        for (int i = 0; i < n; i += o)
            for (int j = 0; j < k; j++)
                f[i + j + k] = (f[i + j + k] + 1ll * f[i + j] * opt % mod + mod) % mod;
}

inline void FWT_AND(int *f, int n, int opt) {
    for (int o = 2, k = 1; o <= n; o <<= 1, k <<= 1)
        for (int i = 0; i < n; i += o)
            for (int j = 0; j < k; j++)
                f[i + j] = (f[i + j] + 1ll * f[i + j + k] * opt % mod + mod) % mod;
}

inline void FWT_XOR(int *f, int n, int opt) {
    for (int o = 2, k = 1; o <= n; o <<= 1, k <<= 1)
        for (int i = 0; i < n; i += o)
            for (int j = 0; j < k; j++) {
                ll a0 = f[i + j], a1 = f[i + j + k];
                f[i + j] = (a0 + a1) % mod * opt % mod;
                f[i + j + k] = (a0 - a1 + mod) % mod * opt % mod;
            }
}

inline void mul_OR(int *a, int *b, int n) {
    FWT_OR(a, n, 1); FWT_OR(b, n, 1);
    for(int i = 0;i < n; i++) a[i] = a[i] * b[i] % mod;
    FWT_OR(a, n, -1);
}

```

```
inline void mul_AND(int *a, int *b, int n) {
    FWT_AND(a, n, 1); FWT_AND(b, n, 1);
    for(int i = 0;i < n; i++) a[i] = a[i] * b[i] % mod;
    FWT_AND(a, n, -1);
}

ll quick_pow(ll a, ll b) ;

inline void mul_XOR(int *a, int *b, int n) {
    ll inv2 = quick_pow(mod, mod - 2);
    FWT_XOR(a, n, 1); FWT_XOR(b, n, 1);
    for(int i = 0;i < n; i++) a[i] = a[i] * b[i] % mod;
    FWT_XOR(a, n, inv2);
}

int main() {
    int n;
    cin >> n;
    n = 1 << n;
    for(int i = 0;i < n; i++) cin >> a[i];
    for(int i = 0;i < n; i++) cin >> b[i];

    mul_OR(a, b, n);
    mul_AND(a, b, n);
    mul_XOR(a, b, n);


}
```

6.62 NTT

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5 + 10;

//mod比最终答案要大
//如果中间乘法计算会爆ll，换成按位乘
//精度高，比3次FFT要快(复数运算=>整数运算)
//4倍空间

ll qpow(ll a, ll b, ll mod) ;

const ll mod = 998244353;
const ll G = 3;
const ll invG = qpow(G, mod - 2, mod);
int tr[N];

void NTT(ll *A, int len, int type) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {
        int mid = i / 2;
        ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
        for (int k = 0; k < len; k += i) {
            ll w = 1;
            for (int l = k; l < k + mid; l++) {
                ll t = w * A[l + mid] % mod;
                A[l + mid] = (A[l] - t + mod) % mod;
```

```
                A[l] = (A[l] + t) % mod;
                w = w * Wn % mod;
            }
        }
    }
    if (type == -1) {
        ll invn = qpow(len, mod - 2, mod);
        for (int i = 0; i < len; i++)
            A[i] = A[i] * invn % mod;
    }
}

void mul(ll *a, ll *b, int n) {
    int len = 1; while (len <= n) len <<= 1;
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
    NTT(a, len, 1), NTT(b, len, 1);
    for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
    NTT(a, len, -1);
}

int n, m;
ll a[N], b[N];

int main() {

    scanf("%d%d", &n, &m);
    for (int i = 0; i <= n; i++) scanf("%lld", &a[i]);
    for (int i = 0; i <= m; i++) scanf("%lld", &b[i]);

    mul(a, b, n + m);
    for (int i = 0; i <= n + m; i++)
        printf("%lld ", a[i]);

    return 0;
}
```

6.63 cdq 分治 FFT

```
// hdu 7054
// 求解(1+x^{a1})*(1+x^{a2})*...*(1+x^{an})
// \sum_{i=1}^n a_i <= 1e6.

// 可以f[i][j]，前i个数的和为j的方案数，可以用生成函数转换，并用多项式求解，同时分治FFT优化。

const int N = 1e5 + 10;
int tr[N];
int getLen(int n) ;

void FFT(Complex *A, int len) ;

inline void MTT(ll *x, ll *y, ll *z, int len) ;

struct Poly {
    ll *p;
    int len;
    void init(int len) {
        p = a + cnt;
        this -> len = len;
```

```
21          for(int i = 0;i <= len; i++) p[i] = read();
22          cnt += len + 1;
23      }
24
25      void mul(const Poly b) {
26          static ll x[N], y[N];
27          int LEN = getLen(len + b.len);
28          for(int i = 0;i <= len; i++) x[i] = p[i];
29          for(int i = 0;i <= b.len; i++) y[i] = b.p[i];
30          for(int i = len + 1;i <= LEN; i++) x[i] = 0;
31          for(int i = b.len + 1;i <= LEN; i++) y[i] = 0;
32          MTT(x, y, p, LEN);
33          this -> len += b.len;
34          // 不知道为啥要两倍，可能会有不为0的情况，管他呢
35          for(int i = len + 1;i <= 2 * LEN; i++) p[i] = 0;
36          for(int i = 0;i <= LEN; i++) x[i] = y[i] = 0;
37      }
38  };
39
40  Poly cdq(int l, int r) {
41      Poly res;
42      if(l == r) res.init(len); // 长度
43      else {
44          int mid = (l + r) / 2;
45          res = cdq(l, mid);
46          res.mul(cdq(mid + 1, r));
47      }
48      return res;
49  }
50
51  void solve() {
52      mem(a, 0);
53      int n = read();
54      cnt = 0;
55      ll ans = 1;
56      Poly res = cdq(1, n);
57      for(int i = 0;i < n; i++) cout << res.p[i] << " ";
58  }
```

6.64 求逆分治 FFT

```
1   // f[i] = \sum_{j=1}^i f[i-j] * g[j]
2   // g相同，可以用多项式求逆
3   #include <bits/stdc++.h>
4   using namespace std;
5   typedef long long ll;
6   const double PI = acos(-1);
7   const int N = 1e5 + 10;
8
9   struct Complex {
10      double x, y;
11      Complex(double a = 0, double b = 0): x(a), y(b) {}
12      Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
13      Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
14      Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
         rhs.y + y * rhs.x); }
15      Complex conj() { return Complex(x, -y); }
16  } w[N];
```

```

ll mod;
int tr[N];
ll F[N], G[N];

ll quick_pow(ll a, ll b) ;

int getLen(int n) ;

void FFT(Complex *A, int len) ;

inline void MTT(ll *x, ll *y, ll *z, int len) ;

void Get_Inv(ll *f, ll *g, int n) ;

void fenzhiFFT(ll *f, ll *g, int n) {
    static ll a[N];
    for(int i = 1;i < n; i++) a[i] = (mod - f[i]) % mod;
    a[0] = 1;
    Get_Inv(a, g, n);

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

int main() {
    int n;
    cin >> n;
    for(int i = 1;i < n; i++) cin >> G[i];
    fenzhiFFT(G, F, n);

    for(int i = 0;i < n; i++) cout << F[i] << " ";
}
```

6.65 多项式求逆

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 1e5 + 10;

struct Complex {
    double x, y;
    Complex(double a = 0, double b = 0): x(a), y(b) {}
    Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
    Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
    Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
     rhs.y + y * rhs.x); }
    Complex conj() { return Complex(x, -y); }
} w[N];

ll mod;
int tr[N];
ll F[N], G[N];

ll quick_pow(ll a, ll b) ;
```

```

int getLen(int n) {
    int len = 1; while (len < (n << 1)) len <<= 1;
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
    for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin(2 *
    PI * i / len));
    return len;
}

void FFT(Complex *A, int len) ;

inline void MTT(ll *x, ll *y, ll *z, int len) ;

void Get_Inv(ll *f, ll *g, int n) {
    if(n == 1) { g[0] = quick_pow(f[0], mod - 2); return ; }
    Get_Inv(f, g, (n + 1) >> 1);

    int len = getLen(n);
    static ll c[N];
    for(int i = 0;i < len; i++) c[i] = i < n ? f[i] : 0;
    MTT(c, g, c, len); MTT(c, g, c, len);
    for(int i = 0;i < n; i++) g[i] = (2ll * g[i] - c[i] + mod) % mod;
    for(int i = n;i < len; i++) g[i] = 0;
    for(int i = 0;i < len; i++) c[i] = 0;
}

int main() {
    int n;
    cin >> n;
    for(int i = 0;i < n; i++) cin >> F[i];
    Get_Inv(F, G, n);
    for(int i = 0;i < n; i++) cout << G[i] << " ";
}
```

6.66 多项式快速幂

```
// f(x)^k, k较小时, 可取, 每次FFT之后长度*2
#define maxfft 1048576+5

struct cp {
    double a, b;
    cp operator+(const cp &o) const { return (cp) {a + o.a, b + o.b}; }
    cp operator-(const cp &o) const { return (cp) {a - o.a, b - o.b}; }
    cp operator*(const cp &o) const { return (cp) {a * o.a - b * o.b, b * o.a + a * o.b
    }; }
    cp operator*(const double &o) const { return (cp) {a * o, b * o}; }
    cp operator!() const { return (cp) {a, -b}; }
} w[maxfft];

int pos[maxfft];

void fft_init(int len) {
    int j = 0;
    while ((1 << j) < len)j++;
    j--;
    for (int i = 0; i < len; i++)
        pos[i] = pos[i >> 1] >> 1 | ((i & 1) << j);
}
```

```
22
23  void fft(cp *x, int len, int sta) {
24      for (int i = 0; i < len; i++)
25          if (i < pos[i])swap(x[i], x[pos[i]]);
26      w[0] = (cp) {1, 0};
27      for (unsigned i = 2; i <= len; i <<= 1) {
28          cp g = (cp) {cos(2 * PI / i), sin(2 * PI / i) * sta};
29          for (int j = i >> 1; j >= 0; j -= 2)w[j] = w[j >> 1];
30          for (int j = 1; j < i >> 1; j += 2)w[j] = w[j - 1] * g;
31          for (int j = 0; j < len; j += i) {
32              cp *a = x + j, *b = a + (i >> 1);
33              for (int l = 0; l < i >> 1; l++) {
34                  cp o = b[l] * w[l];
35                  b[l] = a[l] - o;
36                  a[l] = a[l] + o;
37              }
38          }
39      }
40      if (sta == -1)for (int i = 0; i < len; i++)x[i].a /= len, x[i].b /= len;
41  }
42
43  cp x[maxfft], y[maxfft], z[maxfft];
44
45  void FFT(int *a, int *b, int n, int m, int *c) {
46      int len = 1;
47      while (len < (n + m) >> 1)len <<= 1;
48      fft_init(len);
49      for (int i = n / 2; i < len; i++)x[i].a = x[i].b = 0;
50      for (int i = m / 2; i < len; i++)y[i].a = y[i].b = 0;
51      for (int i = 0; i < n; i++)(i & 1 ? x[i >> 1].b : x[i >> 1].a) = a[i];
52      for (int i = 0; i < m; i++)(i & 1 ? y[i >> 1].b : y[i >> 1].a) = b[i];
53      fft(x, len, 1), fft(y, len, 1);
54      for (int i = 0; i < len / 2; i++) {
55          int j = len - 1 & len - i;
56          z[i] = x[i] * y[i] - (x[i] - !x[j]) * (y[i] - !y[j]) * (w[i] + (cp) {1, 0}) *
        0.25;
57      }
58      for (int i = len / 2; i < len; i++) {
59          int j = len - 1 & len - i;
60          z[i] = x[i] * y[i] - (x[i] - !x[j]) * (y[i] - !y[j]) * ((cp) {1, 0} - w[i ^ len
         >> 1]) * 0.25;
61      }
62      fft(z, len, -1);
63      for (int i = 0; i < n + m; i++)
64          if (i & 1)c[i] = (int) (z[i >> 1].b + 0.5) ? 1 : 0;
65          else c[i] = (int) (z[i >> 1].a + 0.5) ? 1 : 0;
66  }
67
68  int n, k, f[maxfft], g[maxfft];
69
70  void Pow(int *f, int len, int k, int *g) {
71      g[0] = 1;
72      while (k) {
73          if (k & 1)FFT(g, f, len, len, g);
74          FFT(f, f, len, len, f);
75          k >>= 1;
76          len <<= 1;
77      }
78  }
```

6.67 多项式除法、取模

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 3e5 + 10;
ll mod;

struct Complex {
    double x, y;
    Complex(double a = 0, double b = 0): x(a), y(b) {}
    Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
    Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
    Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x * rhs.y + y * rhs.x); }
    Complex conj() { return Complex(x, -y); }
} w[N];

int tr[N];
ll F[N], G[N], D[N], R[N];

ll quick_pow(ll a, ll b) ;

int getLen(int n) ;

void FFT(Complex *A, int len) ;

inline void MTT(ll *x, ll *y, ll *z, int len) ;

void Get_Inv(ll *f, ll *g, int n) ;

void rever(ll *f, int n) { for(int i = 0, j = n - 1;i < j; i++, j--) swap(f[i], f[j]); }

void Get_Div(ll *f, ll *g, ll *d, ll *r, int n, int m) {
    static ll a[N], b[N], invb[N];
    for(int i = 0;i < n; i++) a[i] = f[i];
    for(int i = 0;i < m; i++) b[i] = g[i];
    rever(a, n); rever(b, m);
    //for(int i = 0;i < n - m + 1; i++) b[i] = i < m ? b[i] : 0;
    Get_Inv(b, invb, n - m + 1);

    int len = getLen(n);
    MTT(a, invb, a, len);
    rever(a, n - m + 1);
    for(int i = 0;i < len; i++) d[i] = i < n - m + 1 ? a[i] : 0;
    MTT(g, d, b, len);
    for(int i = 0;i < m; i++) { r[i] = (f[i] - b[i] + mod) % mod; }

    for(int i = m;i < len; i++) r[i] = 0;
    for(int i = 0;i < len; i++) a[i] = b[i] = invb[i] = 0;
}

int main() {
    int n, m;
    cin >> n >> m;
    for(int i = 0;i < n; i++) { cin >> F[i]; }
```

```
    for(int i = 0;i < m; i++) { cin >> G[i]; }
    Get_Div(F, G, D, R, n, m);

    for(int i = 0;i < n - m + 1; i++) cout << D[i] << " ";
    cout << endl;
    for(int i = 0;i < m - 1; i++) cout << R[i] << " ";
    cout << endl;
}
```

6.68 多项式 $\ln_{E}xp_{P}ow$

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);

const int N = 1e6 + 10;

ll quick_pow(ll a, ll b) {
    ll ans = 1;
    while(b) {
        if(b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans % mod;
}

const ll G = 3;
const ll invG = quick_pow(G, mod - 2);

int tr[N];
bool flag;

void NTT(ll *A, int len, int type) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {
        int mid = i / 2;
        ll Wn = quick_pow(type == 1 ? G : invG, (mod - 1) / i);
        for (int k = 0; k < len; k += i) {
            ll w = 1;
            for (int l = k; l < k + mid; l++) {
                ll t = w * A[l + mid] % mod;
                A[l + mid] = (A[l] - t + mod) % mod;
                A[l] = (A[l] + t) % mod;
                w = w * Wn % mod;
            }
        }
    }
    if (type == -1) {
        ll invn = quick_pow(len, mod - 2);
        for (int i = 0; i < len; i++)
            A[i] = A[i] * invn % mod;
    }
}

void mul(ll *a, ll *b, int len) {
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
```

```
48      NTT(a, len, 1), NTT(b, len, 1);
49      for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
50      NTT(a, len, -1); NTT(b, len, -1);
51  }
52
53  int getLen(int n) {
54      int len = 1; while (len <= (n << 1)) len <<= 1;
55      return len;
56  }
57
58  void Get_Der(ll *f, ll *g, int len) { for(int i = 1;i < len; i++) g[i - 1] = f[i] * i %
        mod; g[len - 1] = 0; }
59
60  void Get_Int(ll *f, ll *g, int len) { for(int i = 1;i < len; i++) g[i] = f[i - 1] *
        quick_pow(i, mod - 2) % mod; g[0] = 0; }
61
62  void Get_Inv(ll *f, ll *g, int n) {
63      if(n == 1) { g[0] = quick_pow(f[0], mod - 2); return ; }
64      Get_Inv(f, g, (n + 1) >> 1);
65
66      int len = getLen(n);
67      static ll c[N];
68      for(int i = 0;i < len; i++) c[i] = i < n ? f[i] : 0;
69      mul(c, g, len);
70      mul(c, g, len);
71      for(int i = 0;i < n; i++) g[i] = (2ll * g[i] - c[i] + mod) % mod;
72      for(int i = n;i < len; i++) g[i] = 0;
73      for(int i = 0;i < len; i++) c[i] = 0;
74  }
75
76  void Get_Ln(ll *f, ll *g, int n) {
77      static ll a[N], b[N];
78      Get_Der(f, a, n);
79      Get_Inv(f, b, n);
80      int len = getLen(n);
81      mul(a, b, len);
82      Get_Int(a, g, len);
83      for(int i = n;i < len; i++) g[i] = 0;
84      for(int i = 0;i < len; i++) a[i] = b[i] = 0;
85  }
86
87  void Get_Exp(ll *f, ll *g, int n) {
88      if(n == 1) return (void)(g[0] = 1);
89      Get_Exp(f, g, (n + 1) >> 1);
90
91      static ll a[N];
92      Get_Ln(g, a, n);
93      a[0] = (f[0] + 1 - a[0] + mod) % mod;
94      for(int i = 1;i < n; i++) a[i] = (f[i] - a[i] + mod) % mod;
95      int len = getLen(n);
96      mul(g, a, len);
97      for(int i = n;i < len; i++) g[i] = 0;
98      for(int i = 0;i < len; i++) a[i] = 0;
99  }
100
101 void Get_Pow(ll *f, ll *g, int n, ll k1, ll k2) {
102     static ll a[N], b[N], c[N];
103     ll deg = 0; for(int i = 0;i < n && f[i] == 0; i++) ++ deg;
104     if(deg * k1 > n || (flag && deg)) return ;
```

```
    ll f0 = f[deg], f0k = quick_pow(f0, k2), inv0 = quick_pow(f0, mod - 2);
    for(int i = deg;i < n; i++) a[i - deg] = f[i] * inv0 % mod;
    Get_Ln(a, b, n);
    for(int i = 0;i < n - deg * k1; i++) b[i] = b[i] * k1 % mod;
    Get_Exp(b, c, n);
    deg *= k1;
    for(int i = deg;i < n; i++) g[i] = (c[i - deg] * f0k % mod + mod) % mod;
    for(int i = 0;i < deg; i++) g[i] = 0;
    int len = getLen(n);
    for(int i = n;i < len; i++) g[i] = 0;
    for(int i = 0;i < len; i++) a[i] = b[i] = c[i] = 0;
}


ll a[N], ans[N];

void solve() {
    int n; string s; cin >> n >> s;
    ll k1 = 0, k2 = 0;
    for(int i = 0; i < s.length(); i++) {
        k1 = (k1 * 10 + s[i] - '0');
        flag |= (k1 >= mod);
        k1 %= mod;
        k2 = (k2 * 10 + s[i] - '0') % (mod - 1);
    }
    for(int i = 0; i < n; i++) cin >> a[i];
    Get_Pow(a, ans, n, k1, k2); // k1是底 % mod, k2是指数 % mod-1
    for(int i = 0;i < n; i++) cout << ans[i] << (i == n - 1 ? endl : " ");
}
```

6.69 任意模数 MTT

```
//将多项式拆成(a1 * mod + a2) * (b1 * mod + b2)的形式
//=>a1 * b1 * mod ^ 2 + (a2 * b1 + a1 * b2) * mod + a2 * b2
//在利用DFT合并、IDFT合并，最终只需要4次DFT即可
//精度10^14
//4倍空间

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 1e5 + 10;

struct Complex {
    double x, y;
    Complex(double a = 0, double b = 0): x(a), y(b) {}
    Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
    Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
    Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
     rhs.y + y * rhs.x); }
    Complex conj() { return Complex(x, -y); }
} w[N];

int tr[N];
ll a[N], b[N], ans[N];

int getLen(int n) {
```

```
26      int len = 1; while (len <= n) len <<= 1;
27      for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
28      for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin(2 *
        PI * i / len));
29      return len;
30  }
31
32  void FFT(Complex *A, int len) {
33      for (int i = 0; i < len; i++) if(i < tr[i]) swap(A[i], A[tr[i]]);
34      for (int i = 2, lyc = len >> 1; i <= len; i <<= 1, lyc >>= 1)
35          for (int j = 0; j < len; j += i) {
36              Complex *l = A + j, *r = A + j + (i >> 1), *p = w;
37              for (int k = 0; k < i >> 1; k++) {
38                  Complex tmp = *r * *p;
39                  *r = *l - tmp, *l = *l + tmp;
40                  ++l, ++r, p += lyc;
41              }
42          }
43  }
44
45  inline void MTT(ll *x, ll *y, ll *z, int len) {
46      for (int i = 0; i < len; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
47      static Complex a[N], b[N];
48      static Complex dfta[N], dftb[N], dftc[N], dftd[N];
49
50      for (int i = 0; i < len; i++) a[i] = Complex(x[i] & 32767, x[i] >> 15);
51      for (int i = 0; i < len; i++) b[i] = Complex(y[i] & 32767, y[i] >> 15);
52      FFT(a, len), FFT(b, len);
53      for (int i = 0; i < len; i++) {
54          int j = (len - i) & (len - 1);
55          static Complex da, db, dc, dd;
56          da = (a[i] + a[j].conj()) * Complex(0.5, 0);
57          db = (a[i] - a[j].conj()) * Complex(0, -0.5);
58          dc = (b[i] + b[j].conj()) * Complex(0.5, 0);
59          dd = (b[i] - b[j].conj()) * Complex(0, -0.5);
60          dfta[j] = da * dc;
61          dftb[j] = da * dd;
62          dftc[j] = db * dc;
63          dftd[j] = db * dd;
64      }
65      for (int i = 0; i < len; i++) a[i] = dfta[i] + dftb[i] * Complex(0, 1);
66      for (int i = 0; i < len; i++) b[i] = dftc[i] + dftd[i] * Complex(0, 1);
67      FFT(a, len), FFT(b, len);
68      for (int i = 0; i < len; i++) {
69          ll da = (ll)(a[i].x / len + 0.5) % mod;
70          ll db = (ll)(a[i].y / len + 0.5) % mod;
71          ll dc = (ll)(b[i].x / len + 0.5) % mod;
72          ll dd = (ll)(b[i].y / len + 0.5) % mod;
73          z[i] = (da + ((ll)(db + dc) << 15) + ((ll)dd << 30)) % mod;
74      }
75  }
76
77  int main() {
78
79      int n, m;
80      scanf("%d%d%lld", &n, &m, &mod);
81      for (int i = 0; i <= n; i++) scanf("%d", &a[i]);
82      for (int i = 0; i <= m; i++) scanf("%d", &b[i]);
83
```

```
    MTT(a, b, ans, n + m);
    for (int i = 0; i <= n + m; i++)
        printf("%s%d", i == 0 ? "" : " ", (ans[i] + mod) % mod);

    return 0;
}
```

6.70 任意模数 NTT

```
//要求选取的三个模数mod1 * mod2 * mod3 >= p^2*n
//优点是精度高，可达10^26
//缺点是常数大(9次NTT)，并且还使用了龟速乘
//4倍空间

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAX = 4e5 + 10;

ll qmul(ll a, ll b, ll mod) {
    ll res = 0;
    while (b) {
        if (b & 1)
            res = (res + a) % mod;
        a = (a << 1) % mod;
        b >>= 1;
    }
    return res;
}

ll qpow(ll a, ll b, ll mod) {
    ll res = 1;
    while (b) {
        if (b & 1) res = qmul(res, a, mod);
        a = qmul(a, a, mod);
        b >>= 1;
    }
    return res;
}

const ll mod1 = 998244353, mod2 = 1004535809, mod3 = 469762049, mod4 = mod1 * mod2;
const ll G = 3;
ll a[3][MAX], b[3][MAX], ans[MAX], p;
int tr[MAX];

void NTT(ll *A, int len, int type, ll mod) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {
        int mid = i / 2;
        ll Wn = qpow(type == 1 ? G : qpow(G, mod - 2, mod), (mod - 1) / i, mod);
        for (int k = 0; k < len; k += i) {
            ll w = 1;
            for (int l = k; l < k + mid; l++) {
                ll t = w * A[l + mid] % mod;
                A[l + mid] = (A[l] - t + mod) % mod;
                A[l] = (A[l] + t) % mod;
                w = w * Wn % mod;
            }
```

```
50          }
51      }
52      if (type != 1) {
53          ll invn = qpow(len, mod - 2, mod);
54          for (int i = 0; i < len; i++) A[i] = A[i] * invn % mod;
55      }
56  }
57
58  void mul(int i, int len, ll mod) {
59      NTT(a[i], len, 1, mod), NTT(b[i], len, 1, mod);
60      for (int j = 0; j < len; j++) a[i][j] = a[i][j] * b[i][j] % mod;
61      NTT(a[i], len, -1, mod);
62  }
63
64  void CRT(int len) {
65      ll inv1 = qpow(mod2, mod1 - 2, mod1);
66      ll inv2 = qpow(mod1, mod2 - 2, mod2);
67      ll inv3 = qpow(mod4 % mod3, mod3 - 2, mod3);
68      for (int i = 0; i < len; i++) {
69          ll t = 0;
70          t = (t + qmul(a[0][i] * mod2 % mod4, inv1, mod4)) % mod4;
71          t = (t + qmul(a[1][i] * mod1 % mod4, inv2, mod4)) % mod4;
72          a[1][i] = t;
73          t = (a[2][i] - a[1][i] % mod3 + mod3) % mod3 * inv3 % mod3;
74          ans[i] = (mod4 % p * t % p + a[1][i] % p) % p;
75      }
76  }
77
78  void doNTT(int n) {
79      int len = 1; while (len <= n) len <<= 1;
80      for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
81      mul(0, len, mod1), mul(1, len, mod2), mul(2, len, mod3);
82      CRT(len);
83  }
84
85  int main() {
86
87      int n, m;
88      scanf("%d%d%lld", &n, &m, &p);
89      for (int i = 0; i <= n; i++) {
90          ll x; scanf("%lld", &x);
91          a[0][i] = a[1][i] = a[2][i] = x % p;
92      }
93      for (int i = 0; i <= m; i++) {
94          ll x; scanf("%lld", &x);
95          b[0][i] = b[1][i] = b[2][i] = x % p;
96      }
97      doNTT(n + m);
98      for (int i = 0; i <= n + m; i++) printf("%lld ", ans[i]);
99
100     return 0;
101 }
```

6.71 多项式优化常系数齐次线性递推

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

```
4  typedef long long ll;
5  const double PI = acos(-1);
6  const int N = 3e5 + 10;
7
8  struct Complex {
9      double x, y;
10     Complex(double a = 0, double b = 0): x(a), y(b) {}
11     Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
12     Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
13     Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
        rhs.y + y * rhs.x); }
14     Complex conj() { return Complex(x, -y); }
15 } w[N];
16
17 ll mod;
18 int n, k, len, tr[N];
19 ll a[N], h[N], ans[N], s[N], invG[N], G[N];
20
21 ll quick_pow(ll a, ll b) ;
22
23 int getLen(int n) ;
24
25 void rever(ll *f, int n) ;
26
27 void FFT(Complex *A, int len) ;
28
29 inline void MTT(ll *x, ll *y, ll *z, int len) ;
30
31 void Get_Inv(ll *f, ll *g, int n) ;
32
33
34 void Mod(ll *f,ll *g) {
35     static ll tmp[N];
36     rever(f, k + k - 1);
37     for(int i = 0;i < k; i++) tmp[i] = f[i];
38     MTT(tmp, invG, tmp, len);
39     for(int i = k - 1; i < len; i++) tmp[i] = 0;
40     rever(f, k + k - 1); rever(tmp, k - 1);
41     MTT(tmp, G, tmp, len);
42     for(int i = 0;i < k; i++) g[i] = (f[i] + mod - tmp[i]) % mod;
43     for(int i = k;i < len; i++) g[i] = 0;
44     for(int i = 0;i < len; i++) tmp[i] = 0;
45 }
46 void fpow(int b) {
47     s[1] = 1; ans[0] = 1;
48     while(b) {
49         if(b & 1) { MTT(ans, s, ans, len); Mod(ans, ans); }
50         MTT(s, s, s, len); Mod(s, s);
51         b >>= 1;
52     }
53 }
54
55 ll DITI(ll *a, ll *h, ll *ans, int n, int k) {
56     G[k] = 1; for(int i = 1;i <= k; i++) G[k - i] = (mod - a[i]) % mod;
57     rever(G, k + 1);
58     len = getLen(k + 1);
59     Get_Inv(G, invG, k + 1);
60     for(int i = k + 1;i < len; i++) invG[i] = 0;
61     rever(G, k + 1);
```

```
    fpow(n);
    ll Ans = 0;
    for(int i = 0;i < k; i++) Ans = (Ans + 1ll * h[i] * ans[i] % mod) % mod;
    return Ans;
}

int main() {
    int n, k;
    cin >> n >> k;
    for(int i = 1;i <= k; i++){ cin >> a[i]; a[i] = a[i] < 0 ? a[i] + mod : a[i]; }
    for(int i = 0;i < k; i++) { cin >> h[i]; h[i] = h[i] < 0 ? h[i] + mod : h[i]; }

    ll Ans = DITI(a, h, ans, n, k);
    cout << Ans << endl;
}
```

6.72 FFT 加速带有通配符字符串匹配

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

// p[x] = \sum_{i=0}^{m-1} A[i]^3 * B[x-m+i+1] + \sum_{i=0}^{m-1} A[i] * B[x-m+i+1]^3 - 2 * \sum_{i=0}^{m-1} A[i]^2 * B[x-m+i+1]^2

const int N = 1e6 + 1e5;

ll qpow(ll a, ll b, ll mod) {
    ll ans = 1;
    while(b) {
        if(b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans % mod;
}

const ll G = 3;
const ll invG = qpow(G, mod - 2, mod);
int tr[N];

void NTT(ll *A, int len, int type) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {
        int mid = i / 2;
        ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
        for (int k = 0; k < len; k += i) {
            ll w = 1;
            for (int l = k; l < k + mid; l++) {
                ll t = w * A[l + mid] % mod;
                A[l + mid] = (A[l] - t + mod) % mod;
                A[l] = (A[l] + t) % mod;
                w = w * Wn % mod;
            }
        }
    }
    if (type == -1) {
        ll invn = qpow(len, mod - 2, mod);
```

```
        for (int i = 0; i < len; i++)
            A[i] = A[i] * invn % mod;
    }
}

void mul(ll *a, ll *b, int n) {
    int len = 1; while (len <= n) len <<= 1;
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
    NTT(a, len, 1), NTT(b, len, 1);
    for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
    NTT(a, len, -1);
}

ll a1[N], a2[N], a3[N], b1[N], b2[N], b3[N];

void solve() {
    int m, n; cin >> m >> n;
    string s, t; cin >> t >> s;
    for(int i = 0;i < m; i++) {
        if(t[i] == '*') continue ;
        int temp = t[i] - 'a' + 1;
        a1[i] = temp;
        a2[i] = temp * temp;
        a3[i] = temp * temp * temp;
    }
    for(int i = 0;i < n; i++) {
        if(s[i] == '*') continue;
        int temp = s[i] - 'a' + 1;
        b1[i] = temp;
        b2[i] = temp * temp;
        b3[i] = temp * temp * temp;
    }
    reverse(a1, a1 + m);
    reverse(a2, a2 + m);
    reverse(a3, a3 + m);
    mul(a1, b3, n + m);
    mul(a2, b2, n + m);
    mul(a3, b1, n + m);
    vector<int> ans;
    for(int x = m - 1;x < n; x++) {
        ll res = a1[x] + a3[x] - a2[x] * 2;
        if(!res) ans.push_back(x - m + 2);
    }
    cout << ans.size() << endl;
    for(int i = 0;i < ans.size(); i++) cout << ans[i] << (i == ans.size() - 1 ? endl :
    " ");
}
```

6.73 FFT 加速朴素字符串匹配

```
#include <bits/stdc++.h>
using namespace std;
const int N = 4e5 + 10;

// P[x] = \sum_{i=0}^{m-1} A[i] + \sum_{i=0}^{m-1} B[x - m + i + 1] - 2 * \sum_{i=0}^{m
    -1}A[i] * B[x - m + i + 1]

// reverse(a)
```

```

// 当串中的字符集较少时，可以针对每个字符进行FFT，计算每个字符对整个串的贡献

ll qpow(ll a, ll b, ll mod) ;

const ll mod = 998244353;
const ll G = 3;
const ll invG = qpow(G, mod - 2, mod);
int tr[N];

void NTT(ll *A, int len, int type) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {
        int mid = i / 2;
        ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
        for (int k = 0; k < len; k += i) {
            ll w = 1;
            for (int l = k; l < k + mid; l++) {
                ll t = w * A[l + mid] % mod;
                A[l + mid] = (A[l] - t + mod) % mod;
                A[l] = (A[l] + t) % mod;
                w = w * Wn % mod;
            }
        }
    }
    if (type == -1) {
        ll invn = qpow(len, mod - 2, mod);
        for (int i = 0; i < len; i++)
            A[i] = A[i] * invn % mod;
    }
}

void mul(ll *a, ll *b, int n) {
    int len = 1; while (len <= n) len <<= 1;
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
    NTT(a, len, 1), NTT(b, len, 1);
    for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
    NTT(a, len, -1);
}

ll a[N], b[N];

void solve() {
    string s, t; cin >> s >> t;
    int n = s.length(), m = t.length();
    for(int i = 0;i < n; i++) a[i] = s[i] - 'a' + 1;
    for(int i = 0;i < m; i++) b[i] = t[i] - 'a' + 1;
    reverse(b, b + m);
    mul(a, b, n + m - 2);
    double P = 0;
    for(int i = 0;i < m; i++) {
        P += (t[i] - 'a' + 1) * (t[i] - 'a' + 1);
    }
    vector<int> f(n + 1);
    for(int i = 1;i < n; i++) {
        f[i] = f[i - 1] + (s[i] - 'a' + 1) * (s[i] - 'a' + 1);
    }
    for(int x = m - 1;x < n; x++) {
        double res;
```

```
67          if(x == m - 1) res = P + f[x] - a[x] * 2;
68          else res = P + f[x] - f[x - m] - a[x] * 2;
69          if(!res) cout << x - m + 2 << endl;
70      }
71  }
```

6.74 x 不连续、暴力插值

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 3e5 + 10;

ll mod;
ll X[N], Y[N];

ll quick_pow(ll a, ll b) ;

ll Lagrange(ll *x, ll *y, int n, int k) {
    ll ans = 0;
    for(int i = 0;i < n; i++) {
        ll s1 = 1, s2 = 1;
        for(int j = 0;j < n; j++) {
            if(i == j) continue;
            s1 = s1 * (k - x[j] + mod) % mod;
            s2 = s2 * (x[i] - x[j] + mod) % mod;
        }
        ans = (ans + 1ll * y[i] * s1 % mod * quick_pow(s2, mod - 2) % mod) % mod;
    }
    return ans;
}

int main() {
    int n, k;
    cin >> n >> k;
    for(int i = 0;i < n; i++) cin >> X[i] >> Y[i];
    cout << Lagrange(X, Y, n, k) << endl;
}
```

6.75 x 连续、前缀优化

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5 + 10;

ll mod;
ll F[N];
ll pre[N], suf[N];
ll fac[N], invf[N];


ll quick_pow(ll a, ll b) ;

```

```
15 void init() {
16     fac[0] = 1;
17     for(int i = 1;i < N; i++) fac[i] = fac[i - 1] * i % mod;
18     invf[N - 1] = quick_pow(fac[N - 1], mod - 2);
19     for(int i = N - 1;i >= 1; i--) invf[i - 1] = invf[i] * i % mod;
20 }
21
22 ll Lagrange(ll *f, int k, int n) {
23     if(k <= n) return f[k];
24     pre[0] = suf[n] = 1;
25     for(int i = 1;i <= n; i++) pre[i] = pre[i - 1] * (k - i + 1) % mod;
26     for(int i = n;i >= 1; i--) suf[i - 1] = suf[i] * (k - i) % mod;
27     ll ans = 0;
28     for(int i = 0;i <= n; i++) {
29         int opt = (n - i) & 1 ? -1 : 1;
30         ans = (ans + 1ll * opt * pre[i] % mod * suf[i] % mod * invf[i] % mod * invf[n -
       i] % mod * f[i] % mod + mod) % mod;
31     }
32     return f[k] = ans;
33 }
34
35 int main() {
36     init();
37     int n, k;
38     cin >> n >> k;
39     for(int i = 0;i <= n; i++) cin >> F[i];
40     cout << Lagrange(F, k, n) << endl;
41
42 }
```

6.76 多项式 $\ln_e xp_p ow$ 1

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const double PI = acos(-1);
5  const int N = 1e5 + 10;
6
7  struct Complex {
8      double x, y;
9      Complex(double a = 0, double b = 0): x(a), y(b) {}
10     Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
11     Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
12     Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
       rhs.y + y * rhs.x); }
13     Complex conj() { return Complex(x, -y); }
14 } w[N];
15
16 ll mod, inv2;
17 int tr[N];
18 ll F[N], G[N];
19
20 ll quick_pow(ll a, ll b) ;
21
22 int getLen(int n) ;
23
24 void FFT(Complex *A, int len) ;
25
```

```
inline void MTT(ll *x, ll *y, ll *z, int len) ;

void Get_Inv(ll *f, ll *g, int n) ;

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

void Get_Int(ll *f, ll *g, int len) { for(int i = 1;i < len; i++) g[i] = f[i - 1] * quick_pow(i, mod - 2) % mod; g[0] = 0; }

void Get_Ln(ll *f, ll *g, int n) ;

void Get_Exp(ll *f, ll *g, int n) ;

void Get_Pow(ll *f, ll *g, int n, ll k) ;

void Get_Sqrt(ll *f, ll *g, int n) {
    static ll a[N];
    Get_Ln(f, a, n);
    for(int i = 0;i < n; i++) a[i] = a[i] * inv2 % mod;
    Get_Exp(a, g, n);
    int len = getLen(n);
    for(int i = n;i < len; i++) g[i] = 0;
    for(int i = 0;i < len; i++) a[i] = 0;
}
```

6.77 二次剩余处理边界不为 1

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 1e5 + 10;


struct Complex {
    double x, y;
    Complex(double a = 0, double b = 0): x(a), y(b) {}
    Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
    Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
    Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x * rhs.y + y * rhs.x); }
    Complex conj() { return Complex(x, -y); }
} w[N];

ll mod, inv2;
int tr[N];
ll F[N], G[N];

ll quick_pow(ll a, ll b) ;

typedef struct{
    ll x, y; // 把求出来的w作为虚部，则为a + bw
}num;

num num_mul(num a, num b, ll w, ll p) {// 复数乘法
    num ans = {0, 0};
```

```
30      ans.x = (a.x * b.x % p + a.y * b.y % p * w % p + p) % p;
31      ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
32      return ans;
33  }
34
35  ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
36      num ans = {1, 0};
37      while(b) {
38          if(b & 1)
39              ans = num_mul(ans, a, w, p);
40          a = num_mul(a, a, w, p);
41          b >>= 1;
42      }
43      return ans.x % p;
44  }
45
46  ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
47      return quick_pow(a, (p - 1) >> 1);
48  }
49
50  ll Cipolla(ll n, ll p) {// 输入a和p，是否存在x使得x^2 = a (mod p)，存在二次剩余返回x，存在二次
        非剩余返回-1     注意：p是奇质数
51      n %= p;
52      if(n == 0)
53          return 0;
54      if(p == 2)
55          return 1;
56      ll a, w;
57
58      while(true) {// 找出a，求出w，随机成功的概率是50%，所以数学期望是2
59          a = rand() % p;
60          w = ((a * a - n) % p + p) % p;
61          if(legendre(w, p) + 1 == p) // 找到w，非二次剩余条件
62              break;
63      }
64      num x = {a, 1};
65      return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x,一个解是x，另一个解是p-x，这里的w其实
        要开方，但是由拉格朗日定理可知虚部为0，所以最终答案就是对x的实部用快速幂求解
66  }
67
68  int getLen(int n) ;
69
70  void FFT(Complex *A, int len) ;
71
72  inline void MTT(ll *x, ll *y, ll *z, int len) ;
73
74  void Get_Inv(ll *f, ll *g, int n) ;
75
76  void Get_Sqrt(ll *f, ll *g, int n) {
77      if(n == 1) { ll t = Cipolla(f[0], mod); g[0] = min(mod - t, t); return ; }
78      Get_Sqrt(f, g, (n + 1) >> 1);
79
80      int len = getLen(n);
81      static ll c[N], invg[N];
82      for(int i = 0;i < len; i++) c[i] = i < n ? f[i] : 0;
83      Get_Inv(g, invg, n);
84      MTT(c, invg, c, len);
85      for(int i = 0;i < n; i++) g[i] = inv2 * (c[i] + g[i]) % mod;
86      for(int i = n;i < len; i++) g[i] = 0;
```

```
    for(int i = 0;i < len; i++) c[i] = invg[i] = 0;
}

int main() {
    inv2 = quick_pow(2, mod - 2);
    int n;
    cin >> n;
    for(int i = 0;i < n; i++) cin >> F[i];
    Get_Sqrt(F, G, n);
    for(int i = 0;i < n; i++) cout << G[i] << " ";
}
```

6.78 二维几何

```
#include <iostream>
#include <cmath>

using namespace std;

const double eps = 1e-6;
const double pi = acos(-1);

#define zero(x) (((x) > 0 ? (x) : -(x)) < eps)

int sgn(double d) {
    if(fabs(d) < eps)
        return 0;
    if(d > 0)
        return 1;
    else
        return -1;
}

int dcmp(double x, double y) {
    if(fabs(x - y) < eps)
        return 0;
    if(x > y)
        return 1;
    else
        return -1;
}

struct Point{ // 点
    double x, y;
    Point(double x = 0, double y = 0) : x(x), y(y) {}
};

struct line{
    Point a, b;
};

typedef Point Vector; // 向量

// 运算(向量之间)

Vector operator + (Vector A, Vector B) { // AB
    return Vector(A.x + B.x, A.y + B.y);
}
```

```
45
46  Vector operator - (Point A, Point B) { // BA
47      return Vector(A.x - B.x, A.y - B.y);
48  }
49
50  Vector operator * (Vector A, double p) { // A * p
51      return Vector(A.x * p, A.y * p);
52  }
53
54  Vector operator / (Vector A, double p) { // A / p
55      return  Vector(A.x / p, A.y / p);
56  }
57
58  bool operator < (const Point& a, const Point& b) { // 将点升序排列
59      if(a.x == b.x)
60          return a.y < b.y;
61      return a.x < b.x;
62  }
63
64  bool operator == (const Point& a, const Point& b) { // 判断是否为同一点
65      if(dcmp(a.x, b.x) == 0 && dcmp(a.y, b.y) == 0)
66          return true;
67      else
68          return false;
69  }
70
71  /*
      ------------------------------------------------------------------------------------------------
      */
72  // 向量
73
74  double Dot(Vector A, Vector B) { // 内积
75      return A.x * B.x + A.y * B.y;
76  }
77
78  double Cross(Vector A, Vector B) { // 外积
79      return A.x * B.y - A.y * B.x;
80  }
81
82  double Length(Vector A) { // 向量取模
83      return sqrt(Dot(A, A));
84  }
85
86  double Angle(Vector A, Vector B) { // 向量夹角
87      return acos(Dot(A, B) / Length(A) / Length(B));
88  }
89
90  double Area(Point A, Point B, Point C) { // 计算两向量构成的平行四边形有向面积
91      return Cross(B - A, C - A);
92  }
93
94  Vector Rotate(Vector A, double rad) { // 计算向量逆时针旋转后的向量
95      return Vector(A.x * cos(rad) - A.y * sin(rad), A.x * sin(rad) + A.y * cos(rad));
96  }
97
98  Vector Normal(Vector A) { // 计算向量逆时针转90度后的单位法向量
99      double L = Length(A);
100     return Vector(-A.y / L, A.x / L);
101 }
```

```
102
103 bool ToLeftTest(Point a, Point b, Point c) { // 判断bc是不是向ab的逆时针方向转向
104     return Cross(b - a, c - b) > 0;
105 }
106
107 /*
    ------------------------------------------------------------------------------------------------
    */
108 // 直线与线段
109
110 double Pow(double x) {
111     return x * x;
112 }
113
114 double distance (Point p1, Point p2) {// 两点距离
115     return sqrt(Pow(p1.x - p2.x) + Pow(p1.y - p2.y));
116 }
117
118 int dots_inline(Point p1, Point p2, Point p3) { // 判断三点共线
119     return Cross(p2 - p1, p3 - p1);
120 }
121
122 int dot_online_in(Point p, line l) { // 判断点在线段上（包含端点）
123     return zero(Cross(l.b - p, l.a - p) && ((l.a.x - p.x) * (l.b.x - p.x) < eps) && ((l
    .a.y - p.y) * (l.b.y - p.y) < eps));
124 }
125
126 int dot_online_ex(Point p, line l) { // 判断点在线段上（不包含端点）
127     return dot_online_in(p, l) && (!zero(p.x - l.a.x) || !zero(p.y - l.a.y)) && (!zero(
    p.x - l.b.x) || !zero(p.y - l.b.y));
128 }
129
130 int same_side(Point p1, Point p2, line l) { // 判断两点在线段同侧，点在线段上返回0
131     return Cross(l.a - l.b, p1 - l.b) * Cross(l.a - l.b, p2 - l.b) > eps;
132 }
133
134 int opposite_side(Point p1, Point p2, line l) { // 判断两点在线段异侧，点在线段上返回0
135     return Cross(l.a - l.b, p1 - l.b) * Cross(l.a - l.b, p2 - l.b) < -eps;
136 }
137
138 int parallel(line u, line v) { // 判断两直线平行
139     return zero((u.a.x - u.b.x) * (v.a.y - v.b.y) - (u.a.y - u.b.y) * (v.a.x - v.b.x));
140 }
141
142 int  perpendicular(line u, line v) { // 判断两直线垂直
143     return zero((u.a.x - u.b.x) * (v.a.x - v.b.x) + (u.a.y - u.b.y) * (v.a.y - v.b.y));
144 }
145
146 int intersect_in(line u, line v) {// 判断两线段相交，包括端点和部分重合
147     if(!dots_inline(u.a, u.b, v.a) || !dots_inline(u.a, u.b, v.b)) {
148         return !same_side(u.a, u.b, v) && !same_side(v.a, v.b, u);
149     }
150     return dot_online_in(u.a, v) || dot_online_in(u.b, v) || dot_online_in(v.a, u) ||
    dot_online_in(v.b, u);
151 }
152
153 int intersect_ex(line u, line v) {// 判断两线段相交，不包括端点和部分重合
154     return opposite_side(u.a, u.b, v) && opposite_side(v.a, v.b, u);
155 }
```

```
156
157 // 计算两直线交点，注意事先判断直线是否相交
158 // 计算两线段交点，注意事先判断线段相交和平行
159 Point intersection(line u, line v) {
160     Point ret = u.a;
161     double t = ((u.a.x - v.a.x) * (v.a.y - v.b.y) - (u.a.y - v.a.y) * (v.a.x - v.b.x))
        / ((u.a.x - u.b.x) * (v.a.y - v.b.y) - (u.a.y - u.b.y) * (v.a.x - v.b.x));
162     ret.x += (u.b.x - u.a.x) * t;
163     ret.y += (u.b.y - u.a.y) * t;
164     return ret;
165 }
166
167 Point ptoline(Point p, line l) { // 点到直线最近点
168     Point t = p;
169     t.x += l.a.y - l.b.y;
170     t.y += l.b.x - l.a.x;
171     line u = {p, t};
172     return intersection(u, l);
173 }
174
175 double disptoline(Point p, line l) { // 点到直线距离
176     return fabs(Cross(p - l.b, l.a - l.b) / distance(l.a, l.b));
177 }
178
179 Point ptoseg(Point p, line l) { // 点到线段最近点
180     Point t = p;
181     t.x += l.a.y - l.b.y;
182     t.y += l.b.x - l.a.x;
183     if(Cross(l.a - p, t - p) * Cross(l.b - p, t - p) > eps)
184         return distance(p, l.a) < distance(p, l.b) ? l.a : l.b;
185     line u = {p, t};
186     return intersection(u, l);
187 }
188
189 double disptoseg(Point p, line l) { // 点到线段距离
190     Point t = p;
191     t.x += l.a.y - l.b.y;
192     t.y += l.b.x - l.a.x;
193     if(Cross(l.a - p, t - p) * Cross(l.b - p, t - p) > eps) {
194         double dis1 = distance(p, l.a);
195         double dis2 = distance(p, l.b);
196         return dis1 < dis2 ? dis1 : dis2;
197     }
198     return fabs(Cross(p - l.b, l.a - l.b) / distance(l.a, l.b));
199 }
200
201 /*
    ------------------------------------------------------------------------------------------------------------
    */
202 // 面积
203
204 double area_triangle(Point p1, Point p2, Point p3) { // 三角形面积（输入三顶点）
205     return fabs(Cross(p1 - p3, p2 - p3)) / 2;
206 }
207
208 double area_triangle(double a, double b, double c) { // 三角形面积（输入三边长）
209     double s = (a + b + c) / 2;
210     return sqrt(s * (s - a) * (s - b) * (s - c));
211 }
```

```
212
213 double area_polygon(int n, Point *p) { // 计算多边形面积，顶点按顺时针或逆时针输入
214     double s1 = 0, s2 = 0;
215     for(int i = 0;i < n; i++) {
216         s1 += p[(i + 1) % n].y * p[i].x;
217         s2 += p[(i + 1) % n].y * p[(i + 2) % n].x;
218     }
219     return fabs(s1 - s2) / 2;
220 }
221
222 /*
    ------------------------------------------------------------------------------------------
    */
223 // 球面
224
225 //计算圆心角 lat 表示纬度,-90<=w<=90,lng 表示经度
226 //返回两点所在大圆劣弧对应圆心角,0<=angle<=pi
227
228 double angle(double lng1, double lat1, double lng2, double lat2) {
229     double dlng = fabs(lng1 - lng2) * pi / 180;
230     while(dlng >= pi + pi) {
231         dlng -= pi + pi;
232     }
233     if(dlng > pi)
234     dlng = pi + pi - dlng;
235     lat1 *= pi / 180;
236     lat2 *= pi / 180;
237     return acos(cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2));
238 }
239
240 // 计算两点距离
241
242 double line_dist(double r, double lng1, double lat1, double lng2, double lat2) {
243     double dlng = fabs(lng1 - lng2) * pi / 180;
244     while(dlng >= pi + pi) {
245         dlng -= pi + pi;
246     }
247     if(dlng > pi)
248     dlng = pi + pi - dlng;
249     lat1 *= pi / 180;
250     lat2 *= pi / 180;
251     return r * sqrt(2 - 2 * (cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2))
    );
252 }
253
254 // 计算球面距离
255
256 inline double sphere_dist(double r, double lng1, double lat1, double lng2, double lat2)
     {
257     return r * angle(lng1, lat1, lng2, lat2);
258 }
259
260 /*
    ------------------------------------------------------------------------------------------
    */
261 // 三角形
262
263 // 外心
264
```

```
265 Point circumcenter(Point a, Point b, Point c) {
266     line u, v;
267     u.a.x = (a.x + b.x) / 2;
268     u.a.y = (a.y + b.y) / 2;
269     u.b.x = u.a.x - a.y + b.y;
270     u.b.y = u.a.y + a.x - b.x;
271     v.a.x = (a.x + c.x) / 2;
272     v.a.y = (a.y + c.y) / 2;
273     v.b.x = v.a.x - a.y + c.y;
274     v.b.y = v.a.y + a.x - c.y;
275     return intersection(u, v);
276 }
277
278 // 内心
279
280 Point incenter(Point a, Point b, Point c) {
281     line u, v;
282     double m, n;
283     u.a = a;
284     m = atan2(b.y - a.y, b.x - a.x);
285     n = atan2(c.y - a.y, c.x - a.x);
286     u.b.x = u.a.x + cos((m + n) / 2);
287     u.b.y = u.a.y + sin((m + n) / 2);
288     v.a = b;
289     m=atan2(a.y - b.y, a.x - b.x);
290     n=atan2(c.y - b.y, c.x - b.x);
291     v.b.x=v.a.x + cos((m + n) / 2);
292     v.b.y=v.a.y + sin((m + n) / 2);
293     return intersection(u, v);
294 }
295
296 // 垂心
297
298 Point perpencenter(Point a, Point b, Point c) {
299     line u, v;
300     u.a = c;
301     u.b.x = u.a.x - a.y + b.y;
302     u.b.y = u.a.y + a.x - b.x;
303     v.a = b;
304     v.b.x = v.a.x - a.y + c.y;
305     v.b.y = v.a.y + a.x - c.x;
306 return intersection(u, v);
307 }
308
309 // 重心
310 //到三角形三顶点距离的平方和最小的点
311 //三角形内到三边距离之积最大的点
312
313 Point barycenter(Point a, Point b, Point c) {
314     line u,v;
315     u.a.x = (a.x + b.x) / 2;
316     u.a.y = (a.y + b.y) / 2;
317     u.b = c;
318     v.a.x = (a.x + c.x) / 2;
319     v.a.y = (a.y + c.y) / 2;
320     v.b = b;
321 return intersection(u, v);
322 }
323
```

```
//费马点
//到三角形三顶点距离之和最小的点
Point fermentpoint(Point a, Point b, Point c) {
    Point u,v;
    double step = fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) + fabs(c.y)
    ;
    int i, j, k;
    u.x = (a.x + b.x + c.x) / 3;
    u.y = (a.y + b.y + c.y) / 3;
    while(step > 1e-10)
    for(k = 0;k < 10; step /= 2, k++)
        for (i = -1;i <= 1; i++)
            for (j = -1;j <= 1; j++){
            v.x = u.x + step * i;
            v.y = u.y + step * j;
            if(distance(u,a) + distance(u,b) + distance(u,c) > distance(v,a) + distance
    (v,b) + distance(v,c))
                u = v;
            }
    return u;
}
```

6.79 三维几何

```
#include <math.h>
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
struct point3{double x,y,z;};
struct line3{point3 a,b;};
struct plane3{point3 a,b,c;};
//计算 cross product U x V
point3 Cross(point3 u,point3 v){
    point3 ret;
    ret.x=u.y*v.z-v.y*u.z;
    ret.y=u.z*v.x-u.x*v.z;
    ret.z=u.x*v.y-u.y*v.x;
    return ret;
}
//计算 dot product U . V
double Dot(point3 u,point3 v){
    return u.x*v.x+u.y*v.y+u.z*v.z;
}
//矢量差 U - V
point3 subt(point3 u,point3 v){
    point3 ret;
    ret.x=u.x-v.x;
    ret.y=u.y-v.y;
    ret.z=u.z-v.z;
    return ret;
}
//取平面法向量
point3 pvec(plane3 s){
    return Cross(subt(s.a,s.b),subt(s.b,s.c));
}
point3 pvec(point3 s1,point3 s2,point3 s3){
    return Cross(subt(s1,s2),subt(s2,s3));
}
//两点距离,单参数取向量大小
```

```
35 double distance(point3 p1,point3 p2){
36     return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)+(p1.z-p2.z)*(p1.z-p2.z)
       );
37 }
38 //向量大小
39 double vlen(point3 p){
40     return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
41 }
42 //判三点共线
43 int dots_inline(point3 p1,point3 p2,point3 p3){
44     return vlen(Cross(subt(p1,p2),subt(p2,p3)))<eps;
45 }
46 //判四点共面
47 int dots_onplane(point3 a,point3 b,point3 c,point3 d){
48     return zero(Dot(pvec(a,b,c),subt(d,a)));
49 }
50 //判点是否在线段上,包括端点和共线
51 int dot_online_in(point3 p,line3 l){
52     return zero(vlen(Cross(subt(p,l.a),subt(p,l.b))))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&
53            (l.a.y-p.y)*(l.b.y-p.y)<eps&&(l.a.z-p.z)*(l.b.z-p.z)<eps;
54 }
55 int dot_online_in(point3 p,point3 l1,point3 l2){
56     return zero(vlen(Cross(subt(p,l1),subt(p,l2))))&&(l1.x-p.x)*(l2.x-p.x)<eps&&
57            (l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
58 }
59 //判点是否在线段上,不包括端点
60 int dot_online_ex(point3 p,line3 l){
61     return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y)||!zero(p.z-l.a.z))&&
62             (!zero(p.x-l.b.x)||!zero(p.y-l.b.y)||!zero(p.z-l.b.z));
63 }
64 int dot_online_ex(point3 p,point3 l1,point3 l2){
65     return dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-l1.z))
       &&
66            (!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
67 }
68 //判点是否在空间三角形上,包括边界,三点共线无意义
69 int dot_inplane_in(point3 p,plane3 s){
70     return zero(vlen(Cross(subt(s.a,s.b),subt(s.a,s.c)))-vlen(Cross(subt(p,s.a),subt(p,
       s.b)))-
71                 vlen(Cross(subt(p,s.b),subt(p,s.c)))-vlen(Cross(subt(p,s.c),subt(p,s.a)
       )));
72 }
73 int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3){
74     return zero(vlen(Cross(subt(s1,s2),subt(s1,s3)))-vlen(Cross(subt(p,s1),subt(p,s2)))
       -
75                 vlen(Cross(subt(p,s2),subt(p,s3)))-vlen(Cross(subt(p,s3),subt(p,s1))));
76 }
77 //判点是否在空间三角形上,不包括边界,三点共线无意义
78 int dot_inplane_ex(point3 p,plane3 s){
79     return dot_inplane_in(p,s)&&vlen(Cross(subt(p,s.a),subt(p,s.b)))>eps&&
80            vlen(Cross(subt(p,s.b),subt(p,s.c)))>eps&&vlen(Cross(subt(p,s.c),subt(p,s.a)
       ))>eps;
81 }
82 int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3){
83     return dot_inplane_in(p,s1,s2,s3)&&vlen(Cross(subt(p,s1),subt(p,s2)))>eps&&
84            vlen(Cross(subt(p,s2),subt(p,s3)))>eps&&vlen(Cross(subt(p,s3),subt(p,s1)))>
       eps;
85 }
86 //判两点在线段同侧,点在线段上返回 0,不共面无意义
```

```
87  int same_side(point3 p1,point3 p2,line3 l){
88      return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))>eps
        ;
89  }
90  int same_side(point3 p1,point3 p2,point3 l1,point3 l2){
91      return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))>eps;
92  }
93  //判两点在线段异侧,点在线段上返回 0,不共面无意义
94  int opposite_side(point3 p1,point3 p2,line3 l){
95      return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))<-
        eps;
96  }
97  int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2){
98      return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))<-eps;
99  }
100 //判两点在平面同侧,点在平面上返回 0
101 int same_side(point3 p1,point3 p2,plane3 s){
102     return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))>eps;
103 }
104 int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
105     return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))>eps;
106 }
107 //判两点在平面异侧,点在平面上返回 0
108 int opposite_side(point3 p1,point3 p2,plane3 s){
109     return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))<-eps;
110 }
111 int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
112     return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))<-eps;
113 }
114 //判两直线平行
115 int parallel(line3 u,line3 v){
116     return vlen(Cross(subt(u.a,u.b),subt(v.a,v.b)))<eps;
117 }
118 int parallel(point3 u1,point3 u2,point3 v1,point3 v2){
119     return vlen(Cross(subt(u1,u2),subt(v1,v2)))<eps;
120 }
121 //判两平面平行
122 int parallel(plane3 u,plane3 v){
123     return vlen(Cross(pvec(u),pvec(v)))<eps;
124 }
125 int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
126     return vlen(Cross(pvec(u1,u2,u3),pvec(v1,v2,v3)))<eps;
127 }
128 //判直线与平面平行
129 int parallel(line3 l,plane3 s){
130     return zero(Dot(subt(l.a,l.b),pvec(s)));
131 }
132 int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
133     return zero(Dot(subt(l1,l2),pvec(s1,s2,s3)));
134 }
135 //判两直线垂直
136 int perpendicular(line3 u,line3 v){
137     return zero(Dot(subt(u.a,u.b),subt(v.a,v.b)));
138 }
139 int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2){
140     return zero(Dot(subt(u1,u2),subt(v1,v2)));
141 }
142 //判两平面垂直
143 int perpendicular(plane3 u,plane3 v){
```

```
144     return zero(Dot(pvec(u),pvec(v)));
145 }
146 int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
147     return zero(Dot(pvec(u1,u2,u3),pvec(v1,v2,v3)));
148 }
149 //判直线与平面平行
150 int perpendicular(line3 l,plane3 s){
151     return vlen(Cross(subt(l.a,l.b),pvec(s)))<eps;
152 }
153 int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
154     return vlen(Cross(subt(l1,l2),pvec(s1,s2,s3)))<eps;
155 }
156 //判两线段相交,包括端点和部分重合
157 int intersect_in(line3 u,line3 v){
158     if (!dots_onplane(u.a,u.b,v.a,v.b))
159         return 0;
160     if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
161         return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
162     return dot_online_in(u.a,v)||dot_online_in(u.b,v)||dot_online_in(v.a,u)||
    dot_online_in(v.b,u);
163 }
164 int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2){
165     if (!dots_onplane(u1,u2,v1,v2))
166         return 0;
167     if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
168         return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
169     return
170             dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||dot_online_in(v1,u1,u2)||
    dot_online_in(v2,u1,u2);
171 }
172 //判两线段相交,不包括端点和部分重合
173 int intersect_ex(line3 u,line3 v){
174     return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)&&opposite_side(v.a,v
    .b,u);
175 }
176 int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2){
177     return  dots_onplane(u1,u2,v1,v2)&&opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,
    u1,u2);
178   }
179 //判线段与空间三角形相交,包括交于边界和(部分)包含
180 int intersect_in(line3 l,plane3 s){
181     return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
182            !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b,s.b);
183 }
184 int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
185     return !same_side(l1,l2,s1,s2,s3)&&!same_side(s1,s2,l1,l2,s3)&&
186            !same_side(s2,s3,l1,l2,s1)&&!same_side(s3,s1,l1,l2,s2);
187 }
188 //判线段与空间三角形相交,不包括交于边界和(部分)包含
189 int intersect_ex(line3 l,plane3 s){
190     return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,s.c)&&
191            opposite_side(s.b,s.c,l.a,l.b,s.a)&&opposite_side(s.c,s.a,l.a,l.b,s.b);
192 }
193 int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
194     return opposite_side(l1,l2,s1,s2,s3)&&opposite_side(s1,s2,l1,l2,s3)&&
195            opposite_side(s2,s3,l1,l2,s1)&&opposite_side(s3,s1,l1,l2,s2);
196 }
197 //计算两直线交点,注意事先判断直线是否共面和平行!
198 //线段交点请另外判线段相交(同时还是要判断是否平行!)
```

```
199 point3 intersection(line3 u,line3 v){
200     point3 ret=u.a;
201     double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
202              /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
203     ret.x+=(u.b.x-u.a.x)*t;
204     ret.y+=(u.b.y-u.a.y)*t;
205     ret.z+=(u.b.z-u.a.z)*t;
206     return ret;
207 }
208 point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2){
209     point3 ret=u1;
210     double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
211              /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
212     ret.x+=(u2.x-u1.x)*t;
213     ret.y+=(u2.y-u1.y)*t;
214     ret.z+=(u2.z-u1.z)*t;
215     return ret;
216 }
217 //计算直线与平面交点,注意事先判断是否平行,并保证三点不共线!
218 //线段和空间三角形交点请另外判断
219 point3 intersection(line3 l,plane3 s){
220     point3 ret=pvec(s);
221     double t=(ret.x*(s.a.x-l.a.x)+ret.y*(s.a.y-l.a.y)+ret.z*(s.a.z-l.a.z))/
222              (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z));
223     ret.x=l.a.x+(l.b.x-l.a.x)*t;
224     ret.y=l.a.y+(l.b.y-l.a.y)*t;
225     ret.z=l.a.z+(l.b.z-l.a.z)*t;
226     return ret;
227 }
228 point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
229     point3 ret=pvec(s1,s2,s3);
230     double t=(ret.x*(s1.x-l1.x)+ret.y*(s1.y-l1.y)+ret.z*(s1.z-l1.z))/
231              (ret.x*(l2.x-l1.x)+ret.y*(l2.y-l1.y)+ret.z*(l2.z-l1.z));
232     ret.x=l1.x+(l2.x-l1.x)*t;
233     ret.y=l1.y+(l2.y-l1.y)*t;
234     ret.z=l1.z+(l2.z-l1.z)*t;
235     return ret;
236 }
237 //计算两平面交线,注意事先判断是否平行,并保证三点不共线!
238 line3 intersection(plane3 u,plane3 v){
239     line3 ret;
240     ret.a=parallel(v.a,v.b,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(
    v.a,v.b,u.a,u.b,u.
241              c);
242     ret.b=parallel(v.c,v.a,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(
    v.c,v.a,u.a,u.b,u.
243              c);
244     return ret;
245 }
246 line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
247     line3 ret;
248     ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v1,v2,u1,
    u2,u3);
249     ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v3,v1,u1,
    u2,u3);
250     return ret;
251 }
252 //点到直线距离
253 double ptoline(point3 p,line3 l){
```

```
254     return vlen(Cross(subt(p,l.a),subt(l.b,l.a)))/distance(l.a,l.b);
255 }
256 double ptoline(point3 p,point3 l1,point3 l2){
257     return vlen(Cross(subt(p,l1),subt(l2,l1)))/distance(l1,l2);
258 }
259 //点到平面距离
260 double ptoplane(point3 p,plane3 s){
261     return fabs(Dot(pvec(s),subt(p,s.a)))/vlen(pvec(s));
262 }
263 double ptoplane(point3 p,point3 s1,point3 s2,point3 s3){
264     return fabs(Dot(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,s3));
265 }
266 //直线到直线距离
267 double linetoline(line3 u,line3 v){
268     point3 n=Cross(subt(u.a,u.b),subt(v.a,v.b));
269     return fabs(Dot(subt(u.a,v.a),n))/vlen(n);
270 }
271 double linetoline(point3 u1,point3 u2,point3 v1,point3 v2){
272     point3 n=Cross(subt(u1,u2),subt(v1,v2));
273     return fabs(Dot(subt(u1,v1),n))/vlen(n);
274 }
275 //两直线夹角 cos 值
276 double angle_cos(line3 u,line3 v){
277     return Dot(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/vlen(subt(v.a,v.b));
278 }
279 double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2){
280     return Dot(subt(u1,u2),subt(v1,v2))/vlen(subt(u1,u2))/vlen(subt(v1,v2));
281 }
282 //两平面夹角 cos 值
283 double angle_cos(plane3 u,plane3 v){
284     return Dot(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));
285 }
286 double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
287     return Dot(pvec(u1,u2,u3),pvec(v1,v2,v3))/vlen(pvec(u1,u2,u3))/vlen(pvec(v1,v2,v3))
      ;
288 }
289 //直线平面夹角 sin 值
290 double angle_sin(line3 l,plane3 s){
291     return Dot(subt(l.a,l.b),pvec(s))/vlen(subt(l.a,l.b))/vlen(pvec(s));
292 }
293 double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
294     return Dot(subt(l1,l2),pvec(s1,s2,s3))/vlen(subt(l1,l2))/vlen(pvec(s1,s2,s3));
295 }
296
297 // 球体相交
298 double vol_ints(double x1, double y1, double z1, double r1, double x2, double y2,
      double z2, double r2) {
299     double sum = 4.00 / 3.00 * PI * r1 * r1 * r1 + 4.00 / 3.00 * PI * r2 * r2 * r2;
300     double ans = 0;
301     double dis = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) + (z1 - z2) * (z1 -
       z2)); //球心距离
302     if (dis >= r1 + r2) //没有交到的地方
303     {
304         ans = 0;
305     } else if (dis + r1 <= r2)//重合
306     {
307         ans = (4.00 / 3.00) * PI * r1 * r1 * r1;
308     } else if (dis + r2 <= r1) {
309         ans = (4.00 / 3.00) * PI * r2 * r2 * r2;
```

```
310     } else  //相交
311     {
312         double cal = (r1 * r1 + dis * dis - r2 * r2) / (2.00 * dis * r1);
313         double h = r1 * (1 - cal);
314         ans += (1.00 / 3.00) * PI * (3.00 * r1 - h) * h * h;
315         cal = (r2 * r2 + dis * dis - r1 * r1) / (2.00 * dis * r2);
316         h = r2 * (1.00 - cal);
317         ans += (1.00 / 3.00) * PI * (3.00 * r2 - h) * h * h;
318     }
319     return ans;
320 }
```

7 图论

7.1 结论

```
最大匹配数：最大匹配的匹配边的数目

最小点覆盖数：选取最少的点，使任意一条边至少有一个端点被选择

最大独立数：选取最多的点，使任意所选两点均不相连

最小路径覆盖数：对于一个 DAG（有向无环图），选取最少条路径，使得每个顶点属于且仅属于一条路径。路径长可以
    为 0（即单个点）。

定理1：最大匹配数 = 最小点覆盖数（这是 Konig 定理）

定理2：最大匹配数 = 最大独立数

定理3：最小路径覆盖数 = 顶点数 - 最大匹配数
```

7.2 二分图判定

```cpp
//不存在奇环即为二分图

const int N = 1e5 + 10;
bool flag;
int n;
vector <int> v[N], col(N);
void dfs(int x, int y){
  col[x] = y;
  for (int i = 0; i < v[x].size(); i++) {
     if (!col[v[x][i]]) dfs(v[x][i],3 - y);
     if (col[v[x][i]] == col[x]) flag = true;       //产生了冲突
  }
}

void check() {
    memset(col, 0, sizeof col);
    for(int i = 1;i <= n; i++) {
        if(!col[i]) dfs(i, 1); // dfs染色
    }
    if(flag) cout << "no";
    else cout << "yes";
}
```

7.3 hungry

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

const int maxn = 1e3 + 10;
struct Edge {
    int v, next;
}e[maxn << 1];
int head[maxn << 1], cnt;
bool vis[maxn << 1];
int match[maxn << 1];

```

```
inline void add(int u, int v) {
    e[++cnt].v = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

bool hungry(int u) {
    for(int i = head[u]; i != -1; i = e[i].next) {
        int v = e[i].v;
        if(!vis[v]) {
            vis[v] = true;
            if(match[v] == -1 || dfs(match[v])) {
                match[v] = u;
                return true;
            }
        }
    }
    return false;
}

void solve() {
    int k, m, n;
    while(scanf("%d",&k) && k) {
        mem(match, -1);
        cnt = 0;
        mem(head, -1);
        scanf("%d%d",&m,&n);
        for(int i = 1;i <= k; i++) {
            int u, v;
            scanf("%d%d",&u,&v);
            add(u, v);
        }
        int ans = 0;
        for(int i = 1;i <= m; i++) {
            mem(vis, false);
            if(dfs(i)) ans++;
        }
        printf("%d\n",ans);
    }
}
```

7.4 KM

```
//https://ac.nowcoder.com/acm/contest/view-submission?submissionId=44654655

struct KM {
#define type int
    //#define inf 0x3f3f3f3f
    static const int N = 505;
    static const int INF = 0x3f3f3f3f;
    int n, mx[N], my[N], prv[N];
    type slk[N], lx[N], ly[N], w[N][N];
    bool vx[N], vy[N];

    void init(int siz) {
        n = siz;
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
```

```
16                  w[i][j] = -505;
17              }
18          }
19      }
20
21      void add_edge(int x, int y, type val) { w[x][y] = val; }
22
23      void match(int y) { while (y) swap(y, mx[my[y] = prv[y]]); }
24
25      void bfs(int x) {
26          int i, y;
27          type d;
28          for (i = 1; i <= n; i++) {
29              vx[i] = vy[i] = 0;
30              slk[i] = INF;
31          }
32          queue<int> q;
33          q.push(x);
34          vx[x] = 1;
35          while (1) {
36              while (!q.empty()) {
37                  x = q.front();
38                  q.pop();
39                  for (y = 1; y <= n; y++) {
40                      d = lx[x] + ly[y] - w[x][y];
41                      if (!vy[y] && d <= slk[y]) {
42                          prv[y] = x;
43                          if (!d) {
44                              if (!my[y]) return match(y);
45                              q.push(my[y]);
46                              vx[my[y]] = 1;
47                              vy[y] = 1;
48                          } else slk[y] = d;
49                      }
50                  }
51              }
52              d = INF + 1;
53              for (i = 1; i <= n; i++) {
54                  if (!vy[i] && slk[i] < d) {
55                      d = slk[i];
56                      y = i;
57                  }
58              }
59              for (i = 1; i <= n; i++) {
60                  if (vx[i]) lx[i] -= d;
61                  if (vy[i]) ly[i] += d;
62                  else slk[i] -= d;
63              }
64              if (!my[y]) return match(y);
65              q.push(my[y]);
66              vx[my[y]] = 1;
67              vy[y] = 1;
68          }
69      }
70
71      type max_match() {
72          int i;
73          type res;
74          for (i = 1; i <= n; i++) {
```

```
            mx[i] = my[i] = ly[i] = 0;
            lx[i] = *max_element(w[i] + 1, w[i] + n + 1);
        }
        for (i = 1; i <= n; i++) bfs(i);
        res = 0;
        for (i = 1; i <= n; i++) res += lx[i] + ly[i];
        return res;
    }

#undef type
};
```

7.5 GaleShapley

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

const int N=4;

void GaleShapley(const int (&man)[MAX][MAX], const int (&woman)[MAX][MAX], int (&match)[MAX]) {
    int wm[MAX][MAX];    // wm[i][j]: rank from girl i to boy j
    int choose[MAX];    // choose[i]: current boyfriend of girl i
    int manIndex[MAX]; //    manIndex[i]: how many girls that have rejected boy i
    int i, j;
    int w, m;
    for (i = 0; i < N; i++) {
        match[i] = -1;
        choose[i] = -1;
        manIndex[i] = 0;
        for (j = 0; j < N; j++)
            wm[i][woman[i][j]] = j;
    }

    bool bSingle = false;
    while (!bSingle) {
        bSingle = true;
        for (i = 0; i < N; i++) {
            if (match[i] != -1) // boy i already have a girlfriend
                continue;
            bSingle = false;
            j = manIndex[i]++; // the jth girl that boy i like most
            w = man[i][j];
            m = choose[w];    // current girl w's boyfriend
            if (m == -1 || wm[w][i] < wm[w][m]) { // if girl w prefer boy i
                match[i] = w;
                choose[w] = i;
                if (m != -1)
                    match[m] = -1;
            }
        }
    }
}


void Print(const int(&match)[MAX], int N) {
    for (int i = 0; i < N; i++)
        cout << i << " " << match[i] << endl;
```

```
}


int main(){
    int man[N][N]={
        {2,3,1,0},
        {2,1,3,0},
        {0,2,3,1},
        {1,3,2,0},
    };
    int woman[N][N]={
        {0,3,2,1},
        {0,1,2,3},
        {0,2,3,1},
        {1,0,3,2},
    };

    int match[N];
    GaleShapley(man,woman,match);
    Print(match,N);

    return 0;
}
```

7.6 Bellman$_{F}ord$

```
struct Edge {
    int u, v, w;
}e[1005];

int dis[1005], n, m, s;
bool flag; // 判断负环

void Init() {
    cin >> n >> m >> s;
    for(int i = 1;i <= n; i++) dis[i] = INF;
    dis[s] = 0;
    for(int i = 1;i <= m; i++) {
        cin >> e[i].u >> e[i].v >> e[i].w;
        if(e[i].u == s)
            dis[e[i].v] = e[i].w;
    }
}

void Bellman_Ford() {
    for(int i = 1;i < n; i++) {
        for(int j = 1;j <= m; i++) {
            if(dis[e[j].v] > dis[e[j].u] + e[j].w)
                dis[e[j].v] = dis[e[j].u] + e[j].w;
        }
    }
    flag = true;
    for(int i = 1;i <= m; i++) {
        if(dis[e[i].v] > dis[e[i].u] + e[i].w) {
            flag = false;
            break;
        }
    }
```

```
}
```

7.7 Dijkatra

```
const int maxn = 2e5 + 10;

struct Edge {
    int v, next;
    ll w;
}e[maxn];
int head[maxn], cnt;

inline void add(int u, int v, ll w) ;

struct node {
    int now;
    ll d;
    bool operator < (const node &rhs) const {
        return d > rhs.d;
    }
};

ll dis[maxn];
bool vis[maxn];
priority_queue<node> q;

void dij(int s) {
    dis[s] = 0;
    q.push({s, 0});
    while(!q.empty()) {
        node p = q.top();
        q.pop();
        int u = p.now;
        if(vis[u]) continue;
        vis[u] = 1;
        for(int i = head[u]; i; i = e[i].next) {
            int v = e[i].v;
            if(dis[v] > dis[u] + e[i].w) {
                dis[v] = dis[u] + e[i].w;
                q.push({v, dis[v]});
            }
        }
    }
}
```

7.8 Floyd

```
//三重循环，暴力遍历
#define INF 336860180
int n, m, d[405][405];
void Floyd() {
    memset(d, 20, sizeof d);
    for(int i = 1;i <= m; i++) {
        int u, v, h; cin >> u >> v >> h;
        d[u][v] = h;
    }
    for(int k = 1;k <= n; k++) {
```

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

7.9 SPFA

```
#define INF 0x3f
#define maxn 5000005
int n, m, s;

struct Edge {
    int v, w, next;
}e[maxn];
int head[maxn], cnt, dis[maxn];
bool vis[maxn];

inline void add(int u,int v,int w) ;

queue<int> q;

void SPFA() {
    for(int i = 1;i <= n; i++) {
        dis[i] = INF;
        vis[i] = false;
    }
    q.push(s); dis[s] = 0; vis[s] = true;
    while(!q.empty()) {
        int u = q.front(); q.pop();
        vis[u] = false;
        for(int i = head[u]; i; i = e[i].next) {
            int v = e[i].v;
            if(dis[v] > dis[u] + e[i].w) {
                dis[v] = dis[u] + e[i].w;
                if(!vis[v]) {
                    vis[v] = true;
                    q.push(v);
                }
            }
        }
    }
}
```

7.10 Kruskal

```
//Kruskal算法求最小生成树（稀疏图）

// 一个无向图，求最小生成树各边的和

struct Kruskal {
    static const int N = 1e6 + 10;
    int n, m, ans;
    struct Edge {
        int u, v, w;
```

```
        bool operator < (const Edge& rhs) const {
            return w < rhs.w;
        }
    }e[N << 1];
    int f[N];

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

    void add(int u, int v, int w) {
        e[++m] = {u, v, w};
    }

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

    void kruskal() {
        int cnt = 0;
        sort(e + 1, e + m + 1);
        for(int i = 1;i <= m; i++) {
            int u = find(e[i].u);
            int v = find(e[i].v);
            if(u == v) continue;
            f[v] = u; ans += e[i].w;
            if(++cnt == n - 1) break;
        }
    }
}krl;
```

7.11 prim

```
//prim算法求最小生成树（稠密图）

struct Prim {
    static const int N = 1e6 + 10;
    int n, m;
    double ans;
    struct Edge {
        int v, next;
        double w;
    }e[N << 1];
    int head[N << 1], cnt;
   struct node {
       int now;
       double d;
       bool operator < (const node& rhs) const {
           return d > rhs.d;
       }
   };

   void init() {
       memset(head, -1, sizeof head);
       cnt = ans = 0;
   }

```

```
    inline void add(int u, int v, double w) {
        e[cnt].v = v;
        e[cnt].w = w;
        e[cnt].next = head[u];
        head[u] = cnt++;
    }

    void prim() {
        vector<int> vis(n + 1);
        priority_queue<node> q; q.push({1, 0});
        int res = 0;
        while(!q.empty() && res <= n) {
            node p = q.top(); q.pop();
            int u = p.now;
            double dis = p.d;
            if(vis[u]) continue;
            vis[u] = 1, res++, ans += dis;
            for(int i = head[u]; ~i; i = e[i].next) {
                int v = e[i].v;
                if(!vis[v]) q.push({v, e[i].w});
            }
        }
    }
}pr;
```

7.12 拓扑排序

```
const int N = 1e5 + 10;

int deg[N], toppar[N];
struct Edge {
    int u, v, next;
}e[N];

int head[N], cnt, tot;
int n, m;

inline void add(int u, int v) {
    e[++cnt].v = v;
    e[cnt].next = head[u];
    head[u] = cnt;
    deg[v]++;
}

void topsort() {
    queue<int> q;
    for(int i = 1;i <= n; i++) {
        if(deg[i] == 0) q.push(i);
    }
    while(!q.empty()) {
        int x = q.front(); q.pop();
        toppar[++tot] = x;
        for(int i = head[x]; i ; i = e[i].next) {
            int v = e[i].v;
            if(--deg[v] == 0) q.push(v);
        }
    }
}
```

```

int main() {
    cin >> n >> m;
    for(int i = 1;i <= m; i++) {
        int u, v;
        cin >> u >> v;
        add(u, v);
    }
    topsort();
    for(int i = 1;i <= tot; i++) cout << toppar[i] << endl;
}
```

7.13 链式前向星建图

```
const int N = 1e5 + 10;

struct Edge {
    int v, w, next;
}e[N];

int head[N], cnt;

inline void add(int u, int v, int w) {
    e[cnt].v = v;
    e[cnt].w = w;
    e[cnt].next = head[u];
    head[u] = cnt;
}

//在二分匹配的时候进行删边操作优化时间复杂度
//在250 * 50的图上能优化100ms左右
inline int del(int x)
{
    if (x == 0)
        return 0;
    int k = del(e[x].next);
    e[x].next = k;
    if ("表达式说明要删")
        return k;
    else
        return x;
}
// for (int i = 1; i < t; i++)
// {
//     del(head[i]);
// }
```

7.14 同余最短路

```
/*
洛谷P3403: 给定x[0], x[1], x[2], ..., x[n - 1], 对于k <= h, 求有多少个k满足a[0]x[0] + a[1]x
    [1] + ... + a[n - 1]x[n - 1] = k
洛谷P2662: 最大的不能被x[0], x[1], ..., x[n - 1]表示的数(从小到大), 显然如果gcd(x[i]) = x[0],
    无解, 否则跑同余最短路求出max(dis) - x[0]即为答案
解决形如上述类型的题目
 */

```

```
/*
一是体现在建图
二是体现在最短路
*/

typedef long long ll;

const int N = 2e6 + 10;
#define INF 0x3f3f3f3f
struct Edge {
    int v, next;
    ll w;
}e[N];
int head[N], cnt;

void add(int u, int v, int w) {
    e[++cnt].v = v;
    e[cnt].w = w;
    e[cnt].next = head[u];
    head[u] = cnt;
}

struct node {
    int now;
    ll d;
    bool operator < (const node& rhs) const {
        return d > rhs.d;
    }
};

priority_queue<node> q;
ll dis[N];

void Dij(int s, int n) {
    vector<bool> vis(n);
    for(int i = 0;i < n; i++) dis[i] = (INF);
    dis[s] = 1;
    q.push({s, 1});
    while(!q.empty()) {
        node p = q.top(); q.pop();
        int u = p.now;
        if(vis[u]) continue;
        vis[u] = 1;
        for(int i = head[u]; i; i = e[i].next) {
            int v = e[i].v;
            if(dis[v] > dis[u] + e[i].w) {
                dis[v] = dis[u] + e[i].w;
                q.push({v, dis[v]});
            }
        }
    }
}

ll solve(ll *x, int n, ll h) {
    sort(x, x + n);
    if(x[0] == 1) return h;
    for(int i = 0;i < x[0]; i++) {
        for(int j = 1;j < n; j++) {
            add(i, (i + x[j]) % x[0], x[j]);
```

```
66          }
67      }
68      Dij(1, x[0]);
69      ll ans = 0;
70      for(int i = 0;i < x[0]; i++) {
71          if(h >= dis[i]) ans += (h - dis[i]) / x[0] + 1;
72      }
73      return ans;
74  }
75
76  int main() {
77      ll h, n; scanf("%lld", &h);
78      for (int i = 0; i < n; i++) scanf("%lld", &x[i]);
79      printf("%lld\n", solve(x, n, h));
80  }
81
82  /*------------------------------------------------------*/
83
84
85  const int N = 1e6 + 10;
86  struct edge {
87      int v; ll w;
88  };
89
90  struct node {
91      int now; ll d;
92      bool operator < (const node& rhs) const {
93          return d > rhs.d;
94      }
95  };
96
97  vector<edge> g[5];
98  ll dis[5][N], w;
99
100 void Dij() {
101     for(int i = 1;i <= 4; i++) {
102         for(int j = 0;j < w; j++) {
103             dis[i][j] = 1e18;
104         }
105     }
106     dis[2][0] = 0;
107     priority_queue<node> q;
108     q.push({2, 0});
109     while(!q.empty()) {
110         node p = q.top(); q.pop();
111         int u = p.now;
112         if(dis[u][p.d % w] < p.d) continue;
113         for(auto e : g[u]) {
114             int v = e.v;
115             ll len = e.w + p.d;
116             if(dis[v][len % w] > len) {
117                 dis[v][len % w] = len;
118                 q.push({v, len});
119             }
120         }
121     }
122 }
123
124 void solve() {
```

```
    int _; cin >> _;
    while(_--) {
        ll d12, d23, d34, d41, K;
        cin >> K >> d12 >> d23 >> d34 >> d41;
        g[1].emplace_back(edge{2, d12});g[2].emplace_back(edge{1, d12});
        g[2].emplace_back(edge{3, d23});g[3].emplace_back(edge{2, d23});
        g[3].emplace_back(edge{4, d34});g[4].emplace_back(edge{3, d34});
        g[4].emplace_back(edge{1, d41});g[1].emplace_back(edge{4, d41});
        w = 2 * min(d12, d23);
        Dij();
        ll ans = 1e19;
        for(int i = 0;i < w; i++) {
            if(dis[2][i] >= K) ans = min(dis[2][i], ans);
            else ans = min(ans, dis[2][i] + (K - dis[2][i] + w - 1) / w * w);
        }
        cout << ans << endl;
        for(int i = 1;i <= 4; i++) g[i].clear();
    }
}
```

7.15 三元环计数

```
const int N = 1e5 + 10;
int d[N], vis[N];
vector<int> g[N];

int main() {
    int n, m; cin >> n >> m;
    vector<pair<int, int>> a(m + 1);
    for(int i = 1;i <= m; i++) {
        cin >> a[i].first >> a[i].second;
        d[a[i].first]++;
        d[a[i].second]++;
    }
    for(int i = 1;i <= m; i++) {
        int u = a[i].first, v = a[i].second;
        if(d[u] < d[v] || (d[u] == d[u] && u > v)) swap(u, v);
        g[u].push_back(v);
    }
    int ans = 0;
    for(int u = 1;u <= n; u++) {
        for(auto v : g[u]) vis[v] = u;
        for(auto v : g[u]) {
            for(auto z : g[v]) {
                if(vis[z] == u) ans++;
            }
        }
    }
    cout << ans << endl;

}
```

7.16 欧拉图

```

const int N = 1e4 + 10;
struct Edge {
```

```
4       int to, next;
5       int index;
6       int dir;
7       bool flag;
8   } edge[N << 1];
9
10  int head[10], tot;
11
12  void init() {
13      memset(head,-1, sizeof(head));
14      tot = 0;
15  }
16
17  void addedge(int u, int v, int index) {
18      edge[tot].to = v;
19      edge[tot].next = head[u];
20      edge[tot].index = index;
21      edge[tot].dir = 0;
22      edge[tot].flag = false;
23      head[u] = tot++;
24      edge[tot].to = u;
25      edge[tot].next = head[v];
26      edge[tot].index = index;
27      edge[tot].dir = 1;
28      edge[tot].flag = false;
29      head[v] = tot++;
30  }
31
32  int du[10];
33  vector<int> ans;
34
35  void dfs(int u) {
36      for (int i = head[u]; i != -1;
37      i = edge[i].next)
38      if (!edge[i].flag) {
39          edge[i].flag = true;
40          edge[i ^ 1].flag = true;
41          dfs(edge[i].to);
42          ans.push_back(i);
43      }
44  }
45
46  int main() {
47      int n;
48      while (scanf("%d", &n) == 1) {
49          init();
50          int u, v;
51          memset(du, 0, sizeof(du));
52          for (int i = 1; i <= n; i++) {
53              scanf("%d%d", &u, &v);
54              addedge(u, v, i);
55              du[u]++;
56              du[v]++;
57          }
58          int s = -1;
59          int cnt = 0;
60          for (int i = 0; i <= 6; i++) {
61              if (du[i] & 1) {
62                  cnt++;
```

```
                s = i;
            }
            if (du[i] > 0 && s == -1)
            s = i;
        }
        bool ff = true;
        if (cnt != 0 && cnt != 2) {
            printf("No solution\n");
            continue;
        }
        ans.clear()
        dfs(s);
        if (ans.size() != n) {
            printf("No solution\n");
            continue;
        }
        for (int i = 0; i < ans.size(); i++) {
            printf("%d", edge[ans[i]].index);
            if (edge[ans[i]].dir == 0)printf("-\n");
            else printf("+\n");
        }
    }
    return 0;
}
```

7.17 DFS 判连通块

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 5e5 + 10;

int n, m;
vector<int> vec[maxn];
int vis[maxn]; // 标记数组

void DFS(int u)
{
    vis[u] = 1;
    for(int i = 0;i < vec[u].size(); i++) {
        int v = vec[u][i];
        if(vis[v] == 0)
            DFS(v);
    }
}

void solve() {
    cin >> n >> m;
    int u, v;
    for(int i = 1;i <= m; i++) {
        cin >> u >> v;
        vec[u].push_back(v);
        vec[v].push_back(u);
    }
    int ans = 0;
    for(int i = 1;i <= n; i++) {
        if(vis[i] == 0) {
            DFS(i);
            ans++;
```

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

7.18 并查集判连通块

```
const int maxn = 5e5 + 10;

int n, m, k;
int pre[maxn], cnt[maxn];

int find(int x) {
    if(pre[x] == x)
        return x;
    int y = find(pre[x]);
    pre[x] = y;
    return y;
}

void merge(int u, int v) {
    int x = find(u);
    int y = find(v);
    if(x != y) pre[x] = y;
}

void init(int n) {
    for(int i = 1;i <= n; i++) {
        pre[i] = i;
        cnt[i] = 0;
    }
}

void solve() {
    cin >> n >> m;
    init(n);
    int u, v;
    for(int i = 1;i <= m; i++) {
        cin >> u >> v;
        merge(u, v);
    }

    int ans = 0;
    // first
    for(int i = 1;i <= n; i++) {
        cnt[find(i)]++;
    }
    for(int i = 1;i <= n; i++) {
        if(cnt[i])
            ans++;
    }

    //second
    for(int i = 1;i <= n; i++) {
        int s = find(i);
        if(cnt[s] == 0) {
            cnt[s]++;
            ans++;
```

```
52          }
53      }
54
55      //third
56      for(int i = 1;i <= n; i++) {
57          if(pre[i] == i)
58              ans++;
59      }
60      cout << ans << endl;
61  }
```

7.19 Tarjan

```
1
2   #include <bits/stdc++.h>
3
4   using namespace std;
5
6   const int N = 2e5 + 10;
7
8   struct Edge{
9       int u, v, next;
10  }e[N * 2];
11
12  int head[N], cnt;
13
14  int dfn[N], low[N]; // 第一次访问到的节点的编号和最早访问的节点的编号（时间戳）
15  bool vis[N]; // 标记节点是否访问过并且还在栈中
16  int Time = 1;
17  stack<int> s;
18
19  void add(int u, int v) {
20      e[++cnt].v = v;
21      e[cnt].next = head[u];
22      head[u] = cnt;
23  }
24
25  void DFS(int u) {
26      s.push(u);
27      dfn[u] = low[u] = Time++;
28      vis[u] = 1;
29
30      for(int i = head[u]; i; i = e[i].next) {
31          int v = e[i].v;
32          if(!dfn[v]) {
33              DFS(v);
34              low[u] = min(low[u], low[v]); // 更新时间戳
35          }
36          else if(vis[v]) {
37              low[u] = min(low[u], low[v]); // 更新时间戳
38          }
39
40      }
41
42      if(dfn[u] == low[u]) { // 找到了一个强连通分量
43          int x;
44          do {
45              x = s.top();
```

```
            cout << x << " ";
            vis[x] = 0;
            s.pop();
        }while(x != u);
        cout << endl;
    }
}

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

    for(int i = 1;i <= m; i++) {
        int u, v;
        cin >> u >> v;
        add(u, v);
    }

    for(int i = 1;i <= n; i++) {
        if(!dfn[i])
            DFS(i);
    }
}
```

7.20 割点和桥

```
const int N = 2e5 + 10;

struct Edge{
    int u, v, next;
}e[N * 2];

int head[N], cnt;

int dfn[N], low[N]; // 第一次访问到的节点的编号和最早访问的节点的编号（时间戳）
bool vis[N]; // 标记节点是否访问过并且还在栈中
int fa[N]; // 父亲节点
int Time = 1;
set<int> s1;
set<pair<int, int> > s2;

void add(int u, int v) {
    e[++cnt].v = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

void DFS(int u) {
    dfn[u] = low[u] = Time++;
    int child = 0;

    for(int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;

        if(!dfn[v]) {
            child++;
            fa[v] = u;
            DFS(v);
```

```
33              if(fa[u] == -1 && child >= 2) // u是根节点，并且有两个儿子----是割点
34                  s1.insert(u);
35              else if(fa[u] != -1 && low[v] >= dfn[u]) // u不是根节点，并且有儿子，并且low[v
    ]>=low[u]----是割点
36                  s1.insert(u);
37              else if(low[v] > dfn[u])
38                  s2.insert({u, v});
39              low[u] = min(low[u], low[v]);
40          }
41          else if(v != fa[u]) // 这一步说明可以不经过父亲而回到祖先，可以尝试更新(条件可以不写，不影
    响)
42              low[u] = min(low[u], dfn[v]);
43      }
44  }
45
46  void Tarjan() {
47      int n, m;
48      cin >> n >> m;
49      memset(fa, -1, sizeof(fa));
50
51      for(int i = 1;i <= m; i++) {
52          int u, v;
53          cin >> u >> v;
54          add(u, v);
55          add(v, u);
56      }
57
58      for(int i = 1;i <= n; i++) {
59          if(!dfn[i])
60              DFS(i);
61      }
62
63      cout << s1.size() << endl;
64      for(auto x : s1) cout << x << endl;
65
66      cout << s2.size() << endl;
67      for(auto x : s2) cout << x.first << "---" << x.second << endl;
68  }
```

7.21 差分约束

```
1
2   /*
3   差分约束是解决这样一类问题
4   给出n个形如x[j] - x[i] <= k的式子，求x[n] - x[1]的最大/最小值
5   最大值—>把所有式子整理为x[j] - x[i] <= k，从i向j连一条边权为k的边，跑最短路
6   最小值—>把所有式子整理为x[j] - x[i] >= k，从i向j连一条边权为k的边，跑最长路
7   注意初始化 有时候需要超级源点0
8   */
9
10  bool spfa(int u) {//dfs跑差分约束最短路
11      vis[u] = 1;
12      for (int i = head[u], v; i; i = e[i].nxt)
13          if (dis[u] + e[i].w < dis[v = e[i].to]) {
14              if (vis[v]) return false;
15              else {
16                  dis[v] = dis[u] + e[i].w;
17                  if (!spfa(v)) return false;
```

```
            }
        }
    vis[u] = 0;
    return true;
}
```

7.22 AHU 算法

```
//用来判断两棵树是否同构
// AHU :判断两棵树是否是同构
//同构:在更换节点的标号之后两棵树能完全相同

const int N = 1e5 + 5;
const int maxn = N << 1;

int n;
struct Edge {
    int v, nxt;
} e[maxn << 1];
int head[maxn], sz[maxn], f[maxn], maxv[maxn], tag[maxn], tot, Max;
vector<int> center[2], L[maxn], subtree_tags[maxn];
void addedge(int u, int v) {
    e[tot].v = v;
    e[tot].nxt = head[u];
    head[u] = tot++;
    e[tot].v = u;
    e[tot].nxt = head[v];
    head[v] = tot++;
}

void dfs_size(int u, int fa) {
    sz[u] = 1;
    maxv[u] = 0;
    for (int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].v;
        if (v == fa) continue;
        dfs_size(v, u);
        sz[u] += sz[v];
        maxv[u] = max(maxv[u], sz[v]);
    }
}

void dfs_center(int rt, int u, int fa, int id) {
    maxv[u] = max(maxv[u], sz[rt] - sz[u]);
    if (Max > maxv[u]) {
        center[id].clear();
        Max = maxv[u];
    }
    if (Max == maxv[u]) center[id].push_back(u);
    for (int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].v;
        if (v == fa) continue;
        dfs_center(rt, v, u, id);
    }
}

int dfs_height(int u, int fa, int depth) {
    L[depth].push_back(u);
```

```
51      f[u] = fa;
52      int h = 0;
53      for (int i = head[u]; i; i = e[i].nxt) {
54          int v = e[i].v;
55          if (v == fa) continue;
56          h = max(h, dfs_height(v, u, depth + 1));
57      }
58      return h + 1;
59  }
60
61  void init(int n) {
62      for (int i = 1; i <= 2 * n; i++) head[i] = 0;
63      tot = 1;
64      center[0].clear();
65      center[1].clear();
66
67      int u, v;
68      for (int i = 1; i <= n - 1; i++) {//在这里输入第一棵树的边
69          scanf("%d %d", &u, &v);
70          addedge(u, v);
71      }
72      dfs_size(1, -1);
73      Max = n;
74      dfs_center(1, 1, -1, 0);
75
76      for (int i = 1; i <= n - 1; i++) {//在这里输入第二棵树的边
77          scanf("%d %d", &u, &v);
78          addedge(u + n, v + n);
79      }
80      dfs_size(1 + n, -1);
81      Max = n;
82      dfs_center(1 + n, 1 + n, -1, 1);
83  }
84
85  bool cmp(int u, int v) { return subtree_tags[u] < subtree_tags[v]; }
86
87  bool rootedTreeIsomorphism(int rt1, int rt2) {
88      for (int i = 0; i <= 2 * n + 1; i++) L[i].clear(), subtree_tags[i].clear();
89      int h1 = dfs_height(rt1, -1, 0);
90      int h2 = dfs_height(rt2, -1, 0);
91      if (h1 != h2) return false;
92      int h = h1 - 1;
93      for (int j = 0; j < (int)L[h].size(); j++) tag[L[h][j]] = 0;
94      for (int i = h - 1; i >= 0; i--) {
95          for (int j = 0; j < (int)L[i + 1].size(); j++) {
96              int v = L[i + 1][j];
97              subtree_tags[f[v]].push_back(tag[v]);
98          }
99
100         sort(L[i].begin(), L[i].end(), cmp);
101
102         for (int j = 0, cnt = 0; j < (int)L[i].size(); j++) {
103             if (j && subtree_tags[L[i][j]] != subtree_tags[L[i][j - 1]]) ++cnt;
104             tag[L[i][j]] = cnt;
105         }
106     }
107     return subtree_tags[rt1] == subtree_tags[rt2];
108 }
109
```

```
bool treeIsomorphism() {
    if (center[0].size() == center[1].size()) {
        if (rootedTreeIsomorphism(center[0][0], center[1][0])) return true;
        if (center[0].size() > 1)
            return rootedTreeIsomorphism(center[0][0], center[1][1]);
    }
    return false;
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        scanf("%d", &n);
        init(n);
        puts(treeIsomorphism() ? "YES" : "NO");
    }
    return 0;
}
```

7.23 Astar

```
#include "bits/stdc++.h"
using namespace std;
//A*
//用来计算点A到点B的第k短的路径

const int MAXN = 55;
const int MAXM = MAXN * MAXN;

int dis[MAXN];
int n, m, k, a, b, cnt;
bool hav = false;

namespace G1{//反图
    int to[MAXM], val[MAXM], head[MAXN], nxt[MAXM], cnt;
    bool vis[MAXN];

    void AddEdge(int u, int v, int w) {
        cnt++;
        to[cnt] = v;
        val[cnt] = w;
        nxt[cnt] = head[u];
        head[u] = cnt;
    }

    void Spfa(int s, int t) {//SPFA+SLF跑最短路
        memset(dis, 0x7f, sizeof(dis)); dis[s] = 0;
        deque<int> q; q.push_back(s); vis[s] = true;
        while (!q.empty()) {
            int u = q.front(); q.pop_front(); vis[u] = false;
            for (int i = head[u]; i; i = nxt[i]) {
                int v = to[i];
                if (dis[v] > dis[u] + val[i]) {
                    dis[v] = dis[u] + val[i];
                    if (!vis[v]) {
                        vis[v] = true;
                        if (!q.empty() && dis[v] < dis[q.front()]) {
```

```
                            q.push_front(v);
                        } else {
                            q.push_back(v);
                        }
                    }
                }
            }
        }
    }
}

namespace G2{//原图
    int to[MAXM], val[MAXM], nxt[MAXM], head[MAXN], cnt;

    void AddEdge(int u, int v, int w) {
        cnt++;
        to[cnt] = v;
        val[cnt] = w;
        nxt[cnt] = head[u];
        head[u] = cnt;
    }

    struct Data{//当前位置，走过的距离，s->now->t总距离，走的步骤
        int now, pas, val;
        vector<int> route;
        /*
        bool operator < (const Data &b) const {return val > b.val;}
        */
        bool operator < (const Data &b) const {//重载
            if (val != b.val) return val > b.val;
            int sz = min(route.size(), b.route.size());
            for (int i = 0; i < sz; i++) {
                if (route[i] != b.route[i]) return route[i] > b.route[i];
            }
            return route.size() > b.route.size();
        }
    };

    void Astar(int s, int t) {//A*
        priority_queue<Data> q;
        Data st;
        st.now = s; st.pas = 0; st.val = dis[s]; st.route = vector<int>{s};
        q.push(st);
        vector<int> vec;
        while (!q.empty()) {
            Data u = q.top(); q.pop();
            if (u.now == t) {//更新路径数
                :: cnt++;
                if (:: cnt == k) {//最终答案
                    cout << u.route[0];
                    for (int i = 1, sz = u.route.size(); i < sz; i++)
                        cout << '-' << u.route[i];
                    hav = true;
                    return;
                }
            }
            for (int i = head[u.now]; i; i = nxt[i]) {//广搜
                int v = to[i];
                vec = u.route;
```

```
                bool visit = false;
                for (int j = 0, sz = vec.size(); j < sz; j++) {//记录是否重复经过
                    if (vec[j] == v) {
                        visit = true;
                        break;
                    }
                }
                if (visit) continue;
                Data nx = u;
                nx.now = v;
                nx.pas = u.pas + val[i];
                nx.val = dis[v] + nx.pas;
                nx.route.push_back(v);
                q.push(nx);
            }
        }
    }
}

int main() {
    cin >> n >> m >> k >> a >> b;
    for (int i = 1; i <= m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        G1 :: AddEdge(v, u, w);
        G2 :: AddEdge(u, v, w);
    }
    G1 :: Spfa(b, a);
    G2 :: Astar(a, b);
    if (!hav) cout << "No" << endl;
    return 0;
}
```

7.24 Dinic

```

struct Dinic {
    static const int N = 1e3 + 10, M = 2e5 + 10, INF = 0x3f3f3f;
    int n, m, s, t;
    int maxflow;
    int deep[N], cur[N];

    struct Edge {
        int v, next;
        int cap;
    }e[M << 1];
    int head[M << 1], cnt;

    void init() {
        memset(head, -1, sizeof head);
        cnt = maxflow = 0;
    }

    void add(int u, int v, int cap) {
        e[cnt].v = v;
        e[cnt].cap = cap;
        e[cnt].next = head[u];
        head[u] = cnt++;
```

```
24
25          e[cnt].v = u;
26          e[cnt].cap = 0;
27          e[cnt].next = head[v];
28          head[v] = cnt++;
29      }
30
31      bool bfs() {
32          for(int i = 0;i <= t; i++) {
33              deep[i] = -1, cur[i] = head[i];
34          }
35          queue<int> q;
36          q.push(s); deep[s] = 0;
37          while(!q.empty()) {
38              int u = q.front(); q.pop();
39                  for(int i = head[u]; ~i; i = e[i].next) {
40                  int v = e[i].v;
41                  if(deep[v] == -1 && e[i].cap) {
42                      deep[v] = deep[u] + 1;
43                      q.push(v);
44                  }
45              }
46          }
47          if(deep[t] >= 0) return true;
48          else return false;
49      }
50
51      int dfs(int u, int mx) {
52          int a;
53          if(u == t) return mx;
54          for(int i = cur[u]; ~i; i = e[i].next) {
55              cur[u] = i;
56              int v = e[i].v;
57              if(e[i].cap && deep[v] == deep[u] + 1 && (a = dfs(v, min(mx, e[i].cap)))) {
58                  e[i].cap -= a;
59                  e[i ^ 1].cap += a;
60                  return a;
61              }
62          }
63          return 0;
64      }
65
66      void dinic() {
67          while(bfs()) {
68              while(1) {
69                  int res = dfs(s, INF);
70                  if(!res) break;
71                  maxflow += res;
72              }
73          }
74      }
75  }mf;
```

7.25 ISAP

```
1   struct ISAP {
2       const static int N = ...;//node size
3       struct Edge {
```

```
4           int from, to, cap, flow;
5           bool operator < (const Edge &rhs) const {
6               return from < rhs.from || (from == rhs.from && to < rhs.to);
7           }
8       };
9       int n, m, s, t;
10      vector<Edge> edges;
11      vector<int> g[N];
12      bool vis[N];
13      int dep[N], cur[N], p[N], num[N];
14
15      void addEdge(int from, int to, int cap) {
16          edges.push_back(Edge{from, to, cap, 0});
17          edges.push_back(Edge{to, from, 0, 0});
18          m = edges.size();
19          g[from].push_back(m - 2);
20          g[to].push_back(m - 1);
21      }
22
23      bool bfs() {
24          memset(vis, 0, sizeof(vis));
25          queue<int> q; q.push(t); vis[t] = 1, dep[t] = 0;
26          while (!q.empty()) {
27              int u = q.front(); q.pop();
28              for (auto &v: g[u]) {
29                  Edge &e = edges[v ^ 1];
30                  if (!vis[e.from] && e.cap > e.flow) {
31                      dep[e.from] = dep[u] + (vis[e.from] = 1);
32                      q.push(e.from);
33                  }
34              }
35          }
36          return vis[s];
37      }
38
39      void init(int siz) {
40          n = siz;
41          for (int i = 0; i < siz; i++) g[i].clear();
42          edges.clear();
43      }
44
45      int augment() {
46          int u = t, a = INF;
47          while (u != s) {
48              Edge &e = edges[p[u]];
49              a = min(a, e.cap - e.flow);
50              u = edges[p[u]].from;
51          }
52          u = t;
53          while (u != s) {
54              edges[p[u]].flow += a;
55              edges[p[u] ^ 1].flow -= a;
56              u = edges[p[u]].from;
57          }
58          return a;
59      }
60
61      int maxFlow(int S, int T) {
62          s = S, t = T;
```

```
63          int flow = 0; bfs();
64          memset(num, 0, sizeof(num));
65          for (int i = 0; i < n; i++) num[dep[i]]++;
66          int u = S;
67          memset(cur, 0, sizeof(cur));
68          while (dep[S] < n) {
69              if (u == T) {
70                  flow += augment();
71                  u = S;
72              }
73              int ok = 0;
74              for (int i = cur[u]; i < g[u].size(); i++) {
75                  Edge &e = edges[g[u][i]];
76                  if (e.cap > e.flow && dep[u] == dep[e.to] + 1) {
77                      ok = 1;
78                      p[e.to] = g[u][i];
79                      cur[u] = i;
80                      u = e.to;
81                      break;
82                  }
83              }
84              if (!ok) {
85                  int mn = n - 1;
86                  for (int i = 0; i < g[u].size(); i++) {
87                      Edge &e = edges[g[u][i]];
88                      if (e.cap > e.flow) mn = min(mn, dep[e.to]);
89                  }
90                  if (--num[dep[u]] == 0) break;
91                  num[dep[u] = mn + 1]++;
92                  cur[u] = 0;
93                  if (u != S) u = edges[p[u]].from;
94              }
95          }
96          return flow;
97      }
98
99  } flow;
```

7.26 MCMF

```
1   const int maxn = 1005;      //点数
2
3   struct Edge {
4       int from, to, cap, flow, cost;
5
6       Edge(int u, int v, int c, int f, int cc)
7               : from(u), to(v), cap(c), flow(f), cost(cc) {}
8   };
9
10  struct MCMF {
11      int n, m;
12      vector<Edge> edges;
13      vector<int> G[maxn];
14      int inq[maxn];  //是否在队列中
15      int d[maxn];    //bellmanford
16      int p[maxn];    //上一条弧
17      int a[maxn];    //可改进量
18      void init(int n) {
```

```
19          this->n = n;
20          for (int i = 0; i <= n; ++i) G[i].clear();
21          edges.clear();
22      }
23
24      void add(int from, int to, int cap, int cost) {
25          edges.emplace_back(Edge(from, to, cap, 0, cost));
26          edges.emplace_back(Edge(to, from, 0, 0, -cost));
27          m = int(edges.size());
28          G[from].emplace_back(m - 2);
29          G[to].emplace_back(m - 1);
30      }
31
32      bool spfa(int s, int t, int &flow, int &cost) {
33          for (int i = 1; i <= n; ++i) d[i] = INF;
34          memset(inq, 0, sizeof(inq));
35          d[s] = 0;
36          inq[s] = 1;
37          p[s] = 0;
38          queue<int> q;
39          a[s] = INF;
40          q.push(s);
41          while (!q.empty()) {
42              int u = q.front();
43              q.pop();
44              inq[u] = 0;
45              for (int i = 0; i < int(G[u].size()); ++i) {
46                  Edge &e = edges[G[u][i]];
47                  if (e.cap > e.flow && d[e.to] > d[u] + e.cost) {
48                      d[e.to] = d[u] + e.cost;
49                      p[e.to] = G[u][i];
50                      a[e.to] = min(a[u], e.cap - e.flow);
51                      if (!inq[e.to]) {
52                          q.push(e.to);
53                          inq[e.to] = 1;
54                      }
55                  }
56              }
57          }
58          if (d[t] == INF) return false;
59          flow += a[t];
60          cost += d[t] * a[t];
61          for (int u = t; u != s; u = edges[p[u]].from) {
62              edges[p[u]].flow += a[t];
63              edges[p[u] ^ 1].flow -= a[t];
64          }
65          return true;
66      }
67
68      int MincostMaxflow(int s, int t, int &cost) {
69          int flow = 0;
70          cost = 0;
71          while (spfa(s, t, flow, cost));
72          return flow;
73      }
74  } mcmf;
```

8 其他

8.1 普通莫队

```cpp
// O(n*sqrt(n))
const int N = 1e5 + 10;

int a[N], cnt[N], ans[N];
int belong[N];

struct Q {
    int l, r, id;
}q[N];

int Size, bnum;

bool cmp(Q a, Q b) {
    return (belong[a.l] ^ belong[b.l]) ? belong[a.l] < belong[b.l] : belong[a.l] & 1 ? a.r < b.r : a.r > b.r;
}

int now = 0;

inline void add(int pos) {
    if(!cnt[a[pos]]) now++;
    ++cnt[a[pos]];
}

inline void del(int pos) {
    --cnt[a[pos]];
    if(!cnt[a[pos]]) --now;
}

int main() {
    int n, m;
    cin >> n >> m;
    Size = sqrt(n);
    bnum = ceil((double)n / Size);
    for(int i = 1;i <= bnum; i++) {
        for(int j = (i - 1) * Size + 1;j <= i * Size; j++) {
            belong[j] = i;
        }
    }
    for(int i = 1;i <= n; i++) cin >> a[i];
    for(int i = 1;i <= m; i++) {
        cin >> q[i].l >> q[i].r;
        q[i].id = i;
    }
    sort(q + 1, q + m + 1, cmp);
    int l = 1, r = 0;
    for(int i = 1;i <= m; i++) {
        int ql = q[i].l, qr = q[i].r;
        while(r < qr) add(++r);
        while(r > qr) del(r--);
        while(l < ql) del(l++);
        while(l > ql) add(--l);
        ans[q[i].id] = now;
    }
    for(int i = 1;i <= m; i++) cout << ans[i] << endl;
```

```
}
```

8.2 带修莫队

```
//带修莫队模板题
//查询[ql, qr]间不同颜色数量，带修改

// O(n^{5/3})

int n, m;
int c[N], cnt[N];
int belong[N], size, totq, totm;
int ans[N], res;

struct Query {
    int l, r, t, id;
    bool operator < (const Query &rhs) const {
        return belong[l] ^ belong[rhs.l] ? belong[l] < belong[rhs.l] :
               (belong[r] ^ belong[rhs.r] ? belong[r] < belong[rhs.r] : t < rhs.t);
    }
} q[N];

struct Modify {
    int pos, val;
} modify[N];

void add(int x) {
    if (!cnt[c[x]]) res++;
    cnt[c[x]]++;
}

void del(int x) {
    cnt[c[x]]--;
    if (!cnt[c[x]]) res--;
}

void upd(int x, int ql, int qr) {
    int pos = modify[x].pos;
    if (ql <= pos && pos <= qr) {
        cnt[c[pos]]--; if (!cnt[c[pos]]) res--;
        if (!cnt[modify[x].val]) res++; cnt[modify[x].val]++;
    }
    swap(modify[x].val, c[pos]);//□ □ □ ĵ□ □ □ □ ´□ □ Ļ□ ẓ
}

int main() {
#ifdef ACM_LOCAL
    freopen("input.in", "r", stdin);
    freopen("output.out", "w", stdout);
#endif
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) scanf("%d", &c[i]);
    for (int i = 1; i <= m; i++) {
        char op[10]; scanf("%s", op);
        if (op[0] == 'Q') {
            int ql, qr; scanf("%d%d", &ql, &qr); totq++;
            q[totq] = Query{ql, qr, totm, totq};
        }
```

```
55          else {
56              int pos, val; scanf("%d%d", &pos, &val); totm++;
57              modify[totm] = Modify{pos, val};
58          }
59      }
60
61      //size = N ^ (2 / 3), (N * totm) ^ (1 / 3)
62      size = ceil(pow(n, (long double)2.0 / 3)); int num = ceil((long double)n / size);
63      for (int i = 1, j = 1; i <= num; i++)
64          while (j <= i * size && j <= n)
65              belong[j++] = i;
66
67      sort(q + 1, q + 1 + totq);
68
69      int l = 1, r = 0, t = 0;
70      for (int i = 1; i <= totq; i++) {
71          int ql = q[i].l, qr = q[i].r, qt = q[i].t;
72          while (l < ql) del(l++);
73          while (l > ql) add(--l);
74          while (r < qr) add(++r);
75          while (r > qr) del(r--);
76          while (t < qt) upd(++t, ql, qr);
77          while (t > qt) upd(t--, ql, qr);
78          ans[q[i].id] = res;
79      }
80
81      for (int i = 1; i <= totq; i++) printf("%d\n", ans[i]);
82
83      return 0;
84  }
```

8.3 回滚莫队

```
1   //问题可以莫队（询问可以离线，不带修改）
2   //区间伸长的时候很好维护信息
3   //区间缩短的时候不太好维护信息（如最大值，删除以后不知道次大值是多少）
4   // O(nsqrt(n))
5
6   struct Hash {
7       int b[N], tot;
8       void init() { tot = 0; }
9       void insert(int x) { b[++tot] = x; }
10      void build() {
11          sort(b + 1, b + 1 + tot);
12          tot = unique(b + 1, b + 1 + tot) - (b + 1);
13      }
14      int pos(int x) { return lower_bound(b + 1, b + 1 + tot, x) - b; }
15  } ha;
16
17  int n, m;
18  int c[N], pos[N], cnt[N], cntt[N];
19  int belong[N], sizz;
20  ll ans[N], res;
21
22  struct Query {
23      int l, r, id;
24      bool operator < (const Query &rhs) const {
25          return belong[l] ^ belong[rhs.l] ? belong[l] < belong[rhs.l] : r < rhs.r;
```

```
    }
} q[N];

ll bruteForce(int ql, int qr) {
    ll result = 0;
    for (int i = ql; i <= qr; i++) {
        cntt[pos[i]]++;
        result = max(result, 1ll * c[i] * cntt[pos[i]]);
    }
    for (int i = ql; i <= qr; i++) cntt[pos[i]]--;
    return result;
}

void add(int x) {
    cnt[pos[x]]++;
    res = max(res, 1ll * c[x] * cnt[pos[x]]);
}

void del(int x) {
    cnt[pos[x]]--;
}

int main() {


    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) scanf("%d", &c[i]), ha.insert(c[i]);
    ha.build();
    for (int i = 1; i <= n; i++) pos[i] = ha.pos(c[i]);

    sizz = sqrt(n); int num = ceil((long double)n / sizz);
    for (int i = 1, j = 1; i <= num; i++)
        while (j <= i * sizz && j <= n)
            belong[j++] = i;

    for (int i = 1; i <= m; i++) scanf("%d%d", &q[i].l, &q[i].r), q[i].id = i;
    sort(q + 1, q + 1 + m);

    for (int i = 1, j = 1; i <= num; i++) {
        memset(cnt, 0, sizeof(cnt));
        int right = min(i * sizz, n);
        res = 0;
        for (int l = right + 1, r = right; j <= m && belong[q[j].l] == i; j++, l =
    right + 1) {
            int ql = q[j].l, qr = q[j].r;
            if (qr - ql + 1 <= sizz) {
                ans[q[j].id] = bruteForce(ql, qr);
                continue;
            }
            while (r < qr) add(++r);
            ll tmp = res;
            while (l > ql) add(--l);
            ans[q[j].id] = res;
            res = tmp;
            while (l < right + 1) del(l++);
        }
    }

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

```
84
85      return 0;
86  }
```

8.4 树上莫队

```
1   const int N = 1e5 + 10;
2
3   struct Hash {
4       int b[N], tot;
5       void init() { tot = 0; }
6       void insert(int x) { b[++tot] = x; }
7       void build() {
8           sort(b + 1, b + 1 + tot);
9           tot = unique(b + 1, b + 1 + tot) - (b + 1);
10      }
11      int pos(int x) { return lower_bound(b + 1, b + 1 + tot, x) - b; }
12  } ha;
13
14  int n, m;
15  int c[N], cnt[N];
16  vector<int> g[N];
17  int st[N], ed[N], dfnt, nodeOf[N << 1], tag[N];
18  int belong[N], sizz;
19  int ans[N], res;
20
21  struct Query {
22      int l, r, id, k;
23      bool operator < (const Query &rhs) const {
24          return belong[l] ^ belong[rhs.l] ? belong[l] < belong[rhs.l] : r < rhs.r;
25      }
26  } q[N];
27
28  int son[N], siz[N], top[N], fa[N], dep[N];
29  void dfs(int u, int par) {
30      dep[u] = dep[fa[u] = par] + (siz[u] = 1);
31      int max_son = -1; nodeOf[st[u] = ++dfnt] = u;
32      for (auto &v: g[u])
33          if (v != par) {
34              dfs(v, u);
35              siz[u] += siz[v];
36              if (max_son < siz[v])
37                  son[u] = v, max_son = siz[v];
38          }
39      nodeOf[ed[u] = ++dfnt] = u;
40  }
41  void dfs2(int u, int topf) {
42      top[u] = topf;
43      if (!son[u]) return;
44      dfs2(son[u], topf);
45      for (auto &v: g[u])
46          if (v != fa[u] && v != son[u]) dfs2(v, v);
47  }
48  int lca(int x, int y) {
49      while (top[x] != top[y]) {
50          if (dep[top[x]] < dep[top[y]]) swap(x, y);
51          x = fa[top[x]];
52      }
```

```
53      return dep[x] < dep[y] ? x : y;
54  }
55
56  void upd(int x) {
57      x = nodeOf[x];
58      if (tag[x]) {
59          cnt[c[x]]--;
60          if (!cnt[c[x]]) res--;
61      }
62      else {
63          if (!cnt[c[x]]) res++;
64          cnt[c[x]]++;
65      }
66      tag[x] ^= 1;
67  }
68
69  int main() {
70  #ifdef ACM_LOCAL
71      freopen("input.in", "r", stdin);
72      freopen("output.out", "w", stdout);
73  #endif
74      scanf("%d%d", &n, &m);
75      for (int i = 1; i <= n; i++) scanf("%d", &c[i]), ha.insert(c[i]);
76      ha.build();
77      for (int i = 1; i <= n; i++) c[i] = ha.pos(c[i]);
78
79      for (int i = 1; i < n; i++) {
80          int u, v; scanf("%d%d", &u, &v);
81          g[u].push_back(v); g[v].push_back(u);
82      }
83      int rt = 1; dfs(rt, 0); dfs2(rt, rt);
84
85      sizz = sqrt(dfnt); int num = ceil((long double)dfnt / sizz);
86      for (int i = 1, j = 1; i <= num; i++)
87          while (j <= i * sizz && j <= dfnt)
88              belong[j++] = i;
89      for (int i = 1; i <= m; i++) {
90          int u, v; scanf("%d%d", &u, &v);
91          int tlca = lca(u, v);
92          if (st[u] > st[v]) swap(u, v);
93          if (u == tlca) q[i] = Query{st[u], st[v], i, 0};
94          else q[i] = Query{ed[u], st[v], i, tlca};
95
96      }
97      sort(q + 1, q + 1 + m);
98
99      int l = 1, r = 0;
100     for (int i = 1; i <= m; i++) {
101         int ql = q[i].l, qr = q[i].r;
102         while (l < ql) upd(l++);
103         while (l > ql) upd(--l);
104         while (r < qr) upd(++r);
105         while (r > qr) upd(r--);
106         ans[q[i].id] = res + (q[i].k ? (cnt[c[q[i].k]] == 0) : 0);
107     }
108
109     for (int i = 1; i <= m; i++) printf("%d\n", ans[i]);
110
111     return 0;
```

```
}
```

8.5 快读

```
namespace FastIO {
    char gc(void) {
        const int S = 1 << 17;
        static char buf[S], *s = buf, *t = buf;
        if (s == t) t = buf + fread(s = buf, 1, S, stdin);
        if (s == t) return EOF;
        return *s++;
    }

    int read(void) {
        int a = 0, b = 1, c = gc();
        for (; !isdigit(c); c = gc()) b ^= (c == '-');
        for (; isdigit(c); c = gc()) a = a * 10 + c - '0';
        return b ? a : -a;
    }
}
using namespace FastIO;
```

8.6 模拟退火

```
const double DOWN = 0.996;
const double START_T = 5000;
double ansx, ansy, ansz, anse;

void initAns() {
    //初始化一个答案点(可以选任意点)
}

double getEnergy(double x, double y, double z) {
    //具体分析题目
}

void SA() {
    double T = START_T;
    while (T > 1e-15) {
        double newx = ansx + (rand() * 2 - RAND_MAX) * T;
        double newy = ansy + (rand() * 2 - RAND_MAX) * T;
        double newz = ansz + (rand() * 2 - RAND_MAX) * T;
        double newe = getEnergy(newx, newy, newz);
        double delta = newe - anse;
        if (delta < 0) ansx = newx, ansy = newy, ansz = newz, anse = newe;
        else if (exp(-delta / T) * RAND_MAX > rand())
            ansx = newx, ansy = newy, ansz = newz;
        T *= DOWN;
    }
}

void solve() {
    initAns();
    while ((double) clock() / CLOCKS_PER_SEC < 2.0) SA();
}
```

8.7 Cantor

```

//主要应用为将N维的排列状态压缩成数字id
//然后需要知道具体状态时用逆Cantor得到

int N;
int a[MAX], c[MAX];

void upd(int p, int k) { for (; p <= N; p += lowbit(p)) c[p] += k; }
int query(int p) {
    int res = 0;
    for (; p; p -= lowbit(p)) res += c[p];
    return res;
}

int cantor() {
    //ans = 1 + \sum_{i = 1} ^ {N} fac[N - i] * (\sum_{j = i + 1} ^ {N} x[i] > x[j])
    int res = 0, fac = 1;
    for (int i = N; i >= 1; i--) {
        upd(a[i], 1);
        res = (res + 1ll * fac * query(a[i] - 1) % mod) % mod;
        fac = 1ll * fac * (N - i + 1) % mod;
    }
    return res + 1;
}

//逆Cantor
#define lc  u<<1
#define rc  u<<1|1
#define mid (l+r)/2
int sum[MAX << 4];
void push_up(int u) { sum[u] = sum[lc] + sum[rc]; }
void build(int u, int l, int r) {
    if (l == r) {
        sum[u] = 1;
        return;
    }
    build(lc, l, mid); build(rc, mid + 1, r);
    push_up(u);
}
int query(int u, int l, int r, int k) {//查找第k大并且删除该数
    sum[u]--;
    if (l == r) return l;
    if (k <= sum[lc]) return query(lc, l, mid, k);
    else return query(rc, mid + 1, r, k - sum[lc]);
}

vector<int> inCantor(int x, int n) {
    x--;
    vector<int> res;
    ll fac = 1;
    build(1, 1, n);
    for (int i = 1; i <= n; i++) fac = fac * i;
    for (int i = 1; i <= n; i++) {
        fac = fac / (n - i + 1);
        int k = x / fac + 1;//比当前这位大的有x / fac位
        res.push_back(query(1, 1, n, k));//找到没被选的第k大
        x %= fac;
```

```
    }
    return res;
}
```

8.8 BigInteger

```
struct BigInteger {
    typedef unsigned long long LL;

    static const int BASE = 100000000;
    static const int WIDTH = 8;
    vector<int> s;

    BigInteger& clean() { while (!s.back() && s.size()>1)s.pop_back(); return *this; }
    BigInteger(LL num = 0) { *this = num; }
    BigInteger(string s) { *this = s; }
    BigInteger& operator = (long long num) {
        s.clear();
        do {
            s.push_back(num % BASE);
            num /= BASE;
        } while (num > 0);
        return *this;
    }
    BigInteger& operator = (const string& str) {
        s.clear();
        int x, len = (str.length() - 1) / WIDTH + 1;
        for (int i = 0; i < len; i++) {
            int end = str.length() - i*WIDTH;
            int start = max(0, end - WIDTH);
            sscanf(str.substr(start, end - start).c_str(), "%d", &x);
            s.push_back(x);
        }
        return (*this).clean();
    }

    BigInteger operator + (const BigInteger& b) const {
        BigInteger c; c.s.clear();
        for (int i = 0, g = 0; ; i++) {
            if (g == 0 && i >= s.size() && i >= b.s.size()) break;
            int x = g;
            if (i < s.size()) x += s[i];
            if (i < b.s.size()) x += b.s[i];
            c.s.push_back(x % BASE);
            g = x / BASE;
        }
        return c;
    }
    BigInteger operator - (const BigInteger& b) const {
        assert(b <= *this); // 减数不能大于被减数
        BigInteger c; c.s.clear();
        for (int i = 0, g = 0; ; i++) {
            if (g == 0 && i >= s.size() && i >= b.s.size()) break;
            int x = s[i] + g;
            if (i < b.s.size()) x -= b.s[i];
            if (x < 0) { g = -1; x += BASE; }
            else g = 0;
            c.s.push_back(x);
```

```
53          }
54          return c.clean();
55      }
56      BigInteger operator * (const BigInteger& b) const {
57          int i, j; LL g;
58          vector<LL> v(s.size() + b.s.size(), 0);
59          BigInteger c; c.s.clear();
60          for (i = 0; i<s.size(); i++) for (j = 0; j<b.s.size(); j++) v[i + j] += LL(s[i
        ])*b.s[j];
61          for (i = 0, g = 0; ; i++) {
62              if (g == 0 && i >= v.size()) break;
63              LL x = v[i] + g;
64              c.s.push_back(x % BASE);
65              g = x / BASE;
66          }
67          return c.clean();
68      }
69      BigInteger operator / (const BigInteger& b) const {
70          assert(b > 0);  // 除数必须大于0
71          BigInteger c = *this;       // 商:主要是让c.s和(*this).s的vector一样大
72          BigInteger m;               // 余数:初始化为0
73          for (int i = s.size() - 1; i >= 0; i--) {
74              m = m*BASE + s[i];
75              c.s[i] = bsearch(b, m);
76              m -= b*c.s[i];
77          }
78          return c.clean();
79      }
80      BigInteger operator % (const BigInteger& b) const { //方法与除法相同
81          BigInteger c = *this;
82          BigInteger m;
83          for (int i = s.size() - 1; i >= 0; i--) {
84              m = m*BASE + s[i];
85              c.s[i] = bsearch(b, m);
86              m -= b*c.s[i];
87          }
88          return m;
89      }
90      // 二分法找出满足bx<=m的最大的x
91      int bsearch(const BigInteger& b, const BigInteger& m) const {
92          int L = 0, R = BASE - 1, x;
93          while (1) {
94              x = (L + R) >> 1;
95              if (b*x <= m) { if (b*(x + 1)>m) return x; else L = x; }
96              else R = x;
97          }
98      }
99      BigInteger& operator += (const BigInteger& b) { *this = *this + b; return *this; }
100     BigInteger& operator -= (const BigInteger& b) { *this = *this - b; return *this; }
101     BigInteger& operator *= (const BigInteger& b) { *this = *this * b; return *this; }
102     BigInteger& operator /= (const BigInteger& b) { *this = *this / b; return *this; }
103     BigInteger& operator %= (const BigInteger& b) { *this = *this % b; return *this; }
104
105     bool operator < (const BigInteger& b) const {
106         if (s.size() != b.s.size()) return s.size() < b.s.size();
107         for (int i = s.size() - 1; i >= 0; i--)
108             if (s[i] != b.s[i]) return s[i] < b.s[i];
109         return false;
110     }
```

```
111     bool operator >(const BigInteger& b) const { return b < *this; }
112     bool operator<=(const BigInteger& b) const { return !(b < *this); }
113     bool operator>=(const BigInteger& b) const { return !(*this < b); }
114     bool operator!=(const BigInteger& b) const { return b < *this || *this < b; }
115     bool operator==(const BigInteger& b) const { return !(b < *this) && !(b > *this); }
116 };
117
118 ostream& operator << (ostream& out, const BigInteger& x) {
119     out << x.s.back();
120     for (int i = x.s.size() - 2; i >= 0; i--) {
121         char buf[20];
122         sprintf(buf, "%08d", x.s[i]);
123         for (int j = 0; j < strlen(buf); j++) out << buf[j];
124     }
125     return out;
126 }
127
128 istream& operator >> (istream& in, BigInteger& x) {
129     string s;
130     if (!(in >> s)) return in;
131     x = s;
132     return in;
133 }
134
135 int main()
136 {
137     int t;
138     scanf("%d", &t);
139     while (t--)
140     {
141         BigInteger a, b;
142         cin >> a >> b;
143         cout << a + b << endl;
144     }
145 }
```

8.9 牛顿迭代法手动开根

```
1  import java.math.*;
2  import java.util.*;
3
4  public class Main {
5
6      public static void main(String[] args) {
7          Scanner in = new Scanner(System.in);
8
9          BigInteger n, x;
10         String s;
11
12         //定义"two"为BigInteger类，值为2
13         BigInteger two = new BigInteger("2");
14
15         n = in.nextBigInteger();
16
17         //toString()方法返回该对象的字符串表示,保存在s中
18         s = n.toString();
19
20         //x为BigInteger类，值为 截取字符串开头到s.length()/2 + 1位置的子串 并转化成BigInteger类
```

```
21          x = new BigInteger(s.substring(0, s.length()/2 + 1));
22
23          //当x * x > n时,
24          //x = (x + (n/x)) / 2
25          while(x.multiply(x).compareTo(n) > 0)
26              x = x.add(n.divide(x)).divide(two);
27      }
28  }
```