



---

# ACM/XCPC Template Sweat Boys

---

浙江工商大学

糖果超甜

October 27, 2021

# Contents

<b>0</b>	<b>text</b>	<b>1</b>
0.1	o2,o3 . . . . .	1
0.2	测试模板 . . . . .	1
0.3	读入输出模板 . . . . .	1
0.4	. . . . .	2
0.5	main . . . . .	3
<b>1</b>	<b>dp (动态规划)</b>	<b>4</b>
1.1	二维费用 . . . . .	4
1.2	LIS 问题 $n \log n$ . . . . .	4
1.3	SOS dp . . . . .	4
1.4	SOSDP . . . . .	5
1.5	SOSDP2 . . . . .	5
1.6	背包问题方案 . . . . .	6
1.7	背包最优方案计数问题 . . . . .	6
1.8	单调队列 dp . . . . .	7
1.9	多重背包 . . . . .	7
1.10	分组背包 . . . . .	8
1.11	混合背包 . . . . .	9
1.12	区间 dp . . . . .	9
1.13	数位 dp . . . . .	10
1.14	四边形优化 . . . . .	10
1.15	完全背包 . . . . .	10
1.16	悬线法 . . . . .	11
1.17	有依赖的背包 (树形 dp) . . . . .	11
1.18	01 背包 . . . . .	12
<b>2</b>	<b>树形结构</b>	<b>13</b>
2.1	lca(倍增) . . . . .	13
2.2	lca (树链剖分) . . . . .	13
2.3	次小生成树 . . . . .	13
2.4	点分治 . . . . .	16
2.5	树的半径 . . . . .	18
2.6	树的直径 . . . . .	19
2.7	树哈希 . . . . .	19
2.8	树链剖分 . . . . .	20
2.9	树上 k-th 祖先 $O(q+n \log n)$ . . . . .	21
2.10	树上差分 (边差分) . . . . .	23
2.11	树上距离 (lca tarjan) . . . . .	23
2.12	树上距离 (lca 倍增) . . . . .	25
2.13	树上启发式合并 . . . . .	26
2.14	k 级祖先 . . . . .	27
<b>3</b>	<b>数据结构</b>	<b>28</b>
3.1	Max XOR(tire) . . . . .	28
3.2	splay(待完善) . . . . .	28
3.3	ST 表 . . . . .	32
3.4	. . . . .	32
3.5	Treap(文艺平衡树) . . . . .	34
3.6	并查集 . . . . .	36
3.7	单调队列 . . . . .	37
3.8	单调栈 . . . . .	38
3.9	笛卡尔树 . . . . .	38
3.10	静态主席树 . . . . .	39
3.11	珂朵莉树 . . . . .	40
3.12	可持久化并查集 . . . . .	42
3.13	可持久化数组 . . . . .	43
3.14	平衡树 (pb <sub>ds</sub> ) . . . . .	45

3.15	扫描线	45
3.16	树状数组	47
3.17	树状数组套主席树	48
3.18	替罪羊树 (平衡树)	51
3.19	线段树 (区间最长连续段)	53
3.20	线段树 (最长子序和)	53
3.21	线段树	55
3.22	线段树 + dfs 序	57
3.23	线段树标记永久化	57
3.24	线段树合并 + 动态开点	58
3.25	Hash 表	61
4	数学	63
4.1	$EX_{CRT}$	63
4.2	$EX_{Lucas}$	63
4.3	Lucas	65
4.4	$Miller_{Rabin}$	66
4.5	Min25 筛	67
4.6	杜教筛	68
4.7	杜教筛	69
4.8	二次剩余	70
4.9	反演相关	71
4.10	斐波那契	72
4.11	光速幂	72
4.12	康托展开	72
4.13	快速幂	73
4.14	莫比乌斯反演	74
4.15	逆元	74
4.16	欧拉函数	75
4.17	欧拉降幂	76
4.18	筛法树	76
4.19	算术基本定理	76
4.20	质数、积性函数	77
4.21	线性筛	78
4.22	线性同余方程	79
4.23	原根	79
4.24	原根表	81
4.25	整除分块	82
4.26	整数拆分	82
4.27	整数分块	83
4.28	BSGS	83
4.29	Simpson 积分	84
4.30	2 维计算几何	84
4.31	3 维计算几何	98
4.32	三分套三分	102
4.33	三维几何	102
4.34	多项式 $\ln_{exp} pow - 1$	108
4.35	二次剩余处理边界不为 1	109
4.36	x 不连续、暴力插值	111
4.37	x 连续、前缀优化	112
4.38	FFT 加速带有通配符字符串匹配	112
4.39	FFT 加速带有通配符字符串匹配 2	114
4.40	FFT 加速朴素字符串匹配	115
4.41	$CDQ_{FFT}$	116
4.42	FFT	118
4.43	FWT	119
4.44	NTT	120
4.45	高斯消元	124
4.46	高斯消元 2	125

4.47	高斯消元异或	126
4.48	矩阵快速幂	127
4.49	矩阵求逆	127
4.50	线性基	128
4.51	线性基	129
4.52	康托展开	130
4.53	模数非质数的组合	133
4.54	普通型母函数	134
4.55	生成函数	135
4.56	斯特林数	136
4.57	五边形数	137
4.58	指数型母函数	137
4.59	组合数学基础	138
4.60	组合数	139
4.61	卡特兰数	140
5	随机化算法	141
5.1	SA2	141
6	图论	143
6.1	DAG 图上建支配树	143
6.2	spfa 判负环 (01 分数规划)	144
6.3	边双连通分量 +e-dcc 缩点	146
6.4	差分约束系统	147
6.5	点双连通分量 +v-dcc 缩点	148
6.6	二分图 HK 算法	149
6.7	二分图 KM 算法	150
6.8	二分图匈牙利算法	152
6.9	割边	152
6.10	割点	153
6.11	克鲁斯卡尔重构树	153
6.12	克鲁斯卡尔重构树上建主席树	154
6.13	欧拉图	157
6.14	强联通分量	158
6.15	限制流量费用流 (Dijkstra)	159
6.16	一般图最大匹配 (带花树)	160
6.17	最大独立集 (一般图)	162
6.18	最大流 (Dinic)	162
6.19	最大团 (一般图)	163
6.20	最短路	164
6.21	最小费用最大流 (SPFA)	165
6.22	最小路径生成树	166
6.23	最小生成树 boruvka	168
6.24	最小生成树 prim	168
7	杂项	170
7.1	DEBUG	170
7.2	MODINT	170
7.3	大数模板	171
7.4	待修莫队	177
7.5	对拍	179
7.6	二进制压缩枚举子集	179
7.7	二维差分	179
7.8	分块	180
7.9	高精度	180
7.10	环形均分	183
7.11	回滚莫队	184
7.12	离散化	186
7.13	莫队 + 树状数组	186

7.14	普通莫队	187
7.15	三维偏序问题 (CDQ 分治)	189
7.16	树上莫队	190
<b>8</b>	<b>字符串</b>	<b>193</b>
8.1	(SAM)k 长子串最大出现次数	195
8.2	(SAM) 第 k 小子串	196
8.3	(SAM) 多串 lcs	198
8.4	ac 自动机	199
8.5	kmp	201
8.6	二维字符串哈希	201
8.7	后缀自动机	202
8.8	回文自动机	204
8.9	拓展 kmp	205
8.10	字典树	206
8.11	字符串哈希	207
8.12	最小表示法	207

## 0 text

### 0.1 o2,o3

```
1 #pragma GCC optimize(3,"Ofast","inline")
2
3 #pragma GCC optimize(2)
```

### 0.2 测试模板

```
1 signed main() {
2     ios_base::sync_with_stdio(false);
3     cin.tie(0);
4     cout.tie(0);
5     #ifdef ACM_LOCAL
6         freopen("in.txt", "r", stdin);
7         //freopen("out.txt", "w", stdout);
8         signed test_index_for_debug = 1;
9         char acm_local_for_debug = 0;
10        do {
11            if (acm_local_for_debug == '$') exit(0);
12            if (test_index_for_debug > 20)
13                throw runtime_error("Check the stdin!!!");
14            auto start_clock_for_debug = clock();
15            solve();
16            auto end_clock_for_debug = clock();
17            cout << "Test " << test_index_for_debug << " successful" << endl;
18            cerr << "Test " << test_index_for_debug++ << " Run Time: "
19                << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC <<
20                "s" << endl;
21            cout << "-----" << endl;
22        } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
23    #else
24        solve();
25    #endif
26    return 0;
27 }
```

### 0.3 读入输出模板

```
1 inline int read(){
2     int s=0,w=1;
3     char ch=getchar();
4     while(ch<'0' || ch>'9'){if(ch=='-')w=-1;ch=getchar();}
5     while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
6     return s*w;
7 }
8
9 -----
10 namespace StandardIO {
11     inline char nc() {
12         static char buf[1000000],*p1=buf,*p2=buf;
13         return p1==p2&&(p2=(p1=buf)+fread(buf,1,1000000,stdin),p1==p2)?EOF:*p1++;
14     }
15     template <typename _Tp> inline void read(_Tp&sum) {
16         char ch=nc();sum=0;
17         while(!(ch>='0'&&ch<='9')) ch=nc();
```

```

18     while(ch>='0'&&ch<='9') sum=(sum<<3)+(sum<<1)+(ch-48),ch=nc();
19 }
20
21 template<typename T>
22 inline void write(T x) {
23     if (x < 0) putchar('-'), x *= -1;
24     if (x >= 10) write(x / 10);
25     putchar(x % 10 + '0');
26 }
27 }
28
29 -----
30 inline __int128 read()
31 {
32     int X=0,w=0; char ch=0;
33     while(!isdigit(ch)) {wl=ch=='-';ch=getchar();}
34     while(isdigit(ch)) X=(X<<3)+(X<<1)+(ch^48),ch=getchar();
35     return w?-X:X;
36 }
37 inline void print(__int128 x)
38 {
39     if(x<0){putchar('-');x=-x;}
40     if(x>9) print(x/10);
41     putchar(x%10+'0');
42 }

```

#### 0.4

```

1  /***  1 1  ***/
2  /*SiberianSquirrel*///CuteKiloFish*/
3  #include <bits/stdc++.h>
4  // #include<bits/extc++.h>
5  #include<ext/rope>
6  #include<ext/pb_ds/assoc_container.hpp>
7  #include<ext/pb_ds/tree_policy.hpp>
8  using namespace std;
9  using namespace __gnu_cxx;
10 using namespace __gnu_pbds;
11 #define Inv(x) quick_pow(x, mod - 2)
12 #define Polynomial vector<int>
13 #define DEBUG(x, y) cout << x << ": " << y << '\n';
14 #define mem(a, x) memset(a, x, sizeof a)
15 #define right_1_pos(x) __builtin_ffs(x)
16 #define left_0_num(x) __builtin_clz(x)
17 #define right_0_num(x) __builtin_ctz(x)
18 #define num_of_1(x) __builtin_popcount(x)
19 #define Pii pair<int, int>
20 #define mp_(x, y) make_pair(x, y)
21 #define all(v) (v).begin(), (v).end()
22 using ld = long double;
23 using ll = long long;
24 //using ill = __int128;
25 using ull = unsigned long long;
26 using i16 = short;
27 const ld pi = acos(-1.0);
28 const ld eps = 1e-8;
29 const ll mod = 998244353, mod_g = 3, img = 86583718;
30 int inv2, inv3;

```

```
31 tree<ll, null_type, less<ll>, rb_tree_tag, tree_order_statistics_node_update> tr;
```

## 0.5 main

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef long long ll;
5  typedef pair<ll, ll> PII;
6  typedef unsigned long long ull;
7  const ll inf = 1e18;
8  const int N = 2e6 + 10;
9  const int M = 1e6 + 10;
10 const double eps = 1e-8;
11 const int mod = 1e9 + 7;
12
13 #define fi first
14 #define se second
15 #define re register
16 #define lowbit (-x&x)
17
18 signed main() {
19     ios_base::sync_with_stdio(false);
20     cin.tie(0);
21     cout.tie(0);
22 #ifdef ACM_LOCAL
23     freopen("input", "r", stdin);
24     freopen("output", "w", stdout);
25 #endif
26
27 #ifdef ACM_LOCAL
28     auto start = clock();
29 #endif
30     int t = 1;
31     // cin >> t;
32     while (t--)
33         solve();
34 #ifdef ACM_LOCAL
35     auto end = clock();
36     cerr << "Run Time: " << double(end - start) / CLOCKS_PER_SEC << "s" << endl;
37 #endif
38     return 0;
39 }
```



## 1 dp (动态规划)

### 1.1 二维费用

```
1 时间复杂度  $O(n^3)$ 
2
3  $f[i][j] = \max(f[i][j], f[i-a]+f[j-b] + w);$ 
```

### 1.2 LIS 问题 $n \log n$

```
1  /*
2
3  void solve() {
4      while (cin >> a[++n]);
5      n--;
6      for (int i = 1; i <= n; i++) {
7          if (a[i] > b[len1]) b[++len1] = a[i];
8          else {
9              int pos = lower_bound(b + 1, b + len1 + 1, a[i]) - b;
10             b[pos] = a[i];
11         }
12     }
13
14     reverse(a+1, a+n+1);
15     for (int i = 1; i <= n; i++) {
16         if (a[i] >= b[len2]) b[++len2] = a[i];
17         else {
18             int pos = upper_bound(b + 1, b + len2 + 1, a[i]) - b;
19             b[pos] = a[i];
20         }
21     }
22     cout << len2 << endl;
23     cout << len1 << endl;
24 }
25
26
27 */
```

### 1.3 SOS dp

```
1
2 //
3 // Created by SANZONG on 2021/7/21.
4 //
5 #include "bits/stdc++.h"
6
7 #define int long long
8 using namespace std;
9 const int N = 1 << 20;
10 const int mod = 998244353;
11 int a[N], b[N];
12 int c[N];
13 int d1[N], d2[N];
14 int d3[N], d4[N];
15
16 signed main() {
17     ios::sync_with_stdio(false);
```

```

18     int T;
19     cin >> T;
20     while (T--) {
21         int n;
22         cin >> n;
23         for (int i = 0; i < n; ++i) {
24             cin >> a[i]; d1[i] = a[i]; d2[i] = a[i];
25         }
26         for (int i = 0; i < n; ++i) {
27             cin >> b[i]; d3[i] = b[i]; d4[i] = b[i];
28         }
29         c[n] = -(1ll << 62);
30         for (int i = n - 1; i >= 0; i--) {           //枚举每个数
31             for (int j = 0; (1 << j) < n; j++) {
32                 if (i & (1 << j)) {                 //属于子集
33                     d1[i ^ (1<<j)] = max(d1[i ^ (1<<j)], d1[i ]);           //求k<=i中k
能对应的最大的a[i], 注意转移
34                     d2[i ^ (1<<j)] = min(d2[i ^ (1<<j)], d2[i]);
35                     d3[i ^ (1<<j)] = max(d3[i ^ (1<<j)], d3[i ]);
36                     d4[i ^ (1<<j)] = min(d4[i ^ (1<<j)], d4[i ]);
37                 }
38             }
39         }
40         int ans = 0;
41         for (int i = n - 1; i >= 0; --i) {
42             c[i] = max({c[i + 1], d2[i] * d4[i], d1[i] * d3[i], d1[i] * d4[i], d2[i] *
d3[i]});
43             ans = (ans + c[i]) % mod;
44         }
45         cout << (ans + mod) % mod << endl;
46     }
47 }

```

#### 1.4 SOSDP

```

1  const int bit = 20;
2  const int mx = 1 << bit | 1;
3  void sosdp(vector<int> &a, vector<int> &dp, int n) {
4      // for(int i = 0; i <= n; ++i) dp[i] = 0;
5      for(int i = n; i >= 0; ++i) {
6          for(int j = 0; j < bit; ++j) {
7              if((i | (1 << j)) > n) continue;
8              dp[i] = (dp[i] + dp[i | (1 << j)]) % mod;
9          }
10     }
11 }

```

#### 1.5 SOSDP2

```

1  const int bit = 19;
2  const int mx = 1 << bit | 1;
3  ll A[mx], F[mx], B[mx];
4  ll maxx[2][mx], minn[2][mx];
5
6  void solve() {
7      int o; cin >> o; while(o--) {
8          int n; cin >> n;

```

```

9     for (int i = 0; i < n; ++ i) {
10         cin >> A[i];
11         maxx[0][i] = minn[0][i] = A[i];
12     }
13     for (int i = 0; i < n; ++ i) {
14         cin >> B[i];
15         maxx[1][i] = minn[1][i] = B[i];
16     }
17
18     for(int j = mx; j >= 0; -- j) {
19         for(int i = 0; i < bit; ++ i) {
20             if((j | (1 << i)) >= n) continue;
21             maxx[0][j] = max(maxx[0][j], maxx[0][j | (1 << i)]);
22             maxx[1][j] = max(maxx[1][j], maxx[1][j | (1 << i)]);
23             minn[0][j] = min(minn[0][j], minn[0][j | (1 << i)]);
24             minn[1][j] = min(minn[1][j], minn[1][j | (1 << i)]);
25         }
26     }
27     ll maxxx = -2e18, res = 0;
28     for(int i = n - 1; i >= 0; -- i) {
29         maxxx = max(maxxx, maxx[0][i] * maxx[1][i]);
30         maxxx = max(maxxx, maxx[0][i] * minn[1][i]);
31         maxxx = max(maxxx, minn[0][i] * maxx[1][i]);
32         maxxx = max(maxxx, minn[0][i] * minn[1][i]);
33         res = (res + maxxx) % mod;
34         while(res < 0) res += mod;
35     }
36     cout << res << '\n';
37 }
38 }

```

## 1.6 背包问题方案

- 1 如果要求字典序最小
- 2
- 3  $n \rightarrow 1$  做一遍普通的背包
- 4
- 5  $1 \rightarrow n$  取一遍方案, 如果  $f[i][cur] = f[i+1][cur-v]+w$ , 那么  $i$  就是方案
- 6
- 7 字典序最大, 反过来即可
- 8
- 9 时间复杂度  $O(nm)$

## 1.7 背包最优方案计数问题

- 1 可以不装满背包, 保证总价值最大
- 2
- 3  $f[N]$  记录价值,  $cnt[N]$  记录个数
- 4
- 5  $f[j] = \max(f[j], f[j-v]+w)$ ;
- 6
- 7 如果当前  $f[j-v] + w > f[j]$ , 说明是  $f[j-v]+w$  更新的答案, 因此  $cnt[j] = cnt[j-v]$ ;
- 8
- 9 如果当前  $f[j-v] + w = f[j]$ , 说明两者都可,  $cnt[j] += cnt[j-v]$ ;
- 10
- 11 别忘了  $cnt$  初始化为 1, 因为背包可以空着也算一种方案
- 12
- 13 时间复杂度  $O(nm)$

## 1.8 单调队列 dp

```

1  //
2  // Created by SANZONG on 2021/9/17.
3  //
4
5  //
6  // Created by SANZONG on 2020/8/10.
7  //
8  //lca模板
9
10 #include "bits/stdc++.h"
11 #define int long long
12 const int maxn = 1e6;
13 using namespace std;
14 int dp[maxn];
15 int sum[maxn];
16 int q[maxn];
17 int a[maxn];
18 signed main() {
19     int n,k;
20     cin >> n >> k;
21     for (int i = 1; i <= n; ++i) {
22         int s;cin >> s;
23         sum[i] = sum[i-1] + s;
24     }
25     int head = 1;
26     int tail = 0;
27     q[++tail] = 0;
28     for (int i = 1; i <= n; ++i) {
29         dp[i] = a[q[head]] + sum[i];
30         a[i] = dp[i-1]-sum[i];
31         while (head <= tail && i - q[head] + 1 > k)
32             head++;
33         while (head <= tail && a[q[tail]] < a[i])
34             tail--;
35         q[++tail] = i;
36     }
37     cout << dp[n] << endl;
38 }

```

## 1.9 多重背包

```

1  1.朴素算法
2
3  时间复杂度 $O(n^3)$ 
4
5  for (int i = 1; i <= n; i++)
6      for (int j = m; j >= v[i]; j--)
7          for (int k = 1; k <= s[i]; k++)
8              f[j] = max(f[j], f[j - k*v[i]] + k*w[i]);
9
10 核心 f[j] = max(f[j], f[j-v] + w, f[j-v*2] + 2*w, f[j-3*v] + 3*w ....)
11
12 2.二进制优化
13
14 时间复杂度 $O(nm*\log(n))$ 
15

```

```

16 例如:  $10 = 1 + 2 + 4 + 3$ 
17
18 将问题转换为01背包问题
19
20 3.单调队列优化
21
22 时间复杂度 $O(nm)$ 
23
24  $f[0] = \max(f[0], f[v] + w, f[2*v] + 2*w, f[3*v] + 3*w...)$ 
25  $f[1] = \max(f[1], f[1+v] + w, f[1+2*v] + 2*w, f[1+3*v] + 3*w...)$ 
26  $f[2] = \max(f[2], f[2+v] + w, f[2+2*v] + 2*w, f[2+3*v] + 3*w...)$ 
27 ...
28  $f[v-1] = \max(f[v-1], f[v-1+v] + w, f[v-1+2*v] + 2*w, f[v-1+3*v] + 3*w...)$ 
29
30 所有的  $m$  都可以转换为  $c + a * x$ , 而且是互不影响的
31
32  $f[j] = f[j]$ 
33  $f[j+v] = \max(f[j+v], f[j] + w)$ 
34  $f[j+2*v] = \max(f[j+2*v], f[j+v] + w, f[j] + 2*w)$ 
35
36 修改一下变成
37  $f[j] = f[j]$ 
38  $f[j+v] = \max(f[j+v] - w, f[j]) + w$ 
39  $f[j+2*v] = \max(f[j+2*v] - 2*w, f[j+v] - w, f[j]) + 2*w$ 
40
41 将  $f[j + k*w] - k*w$  存入单调队列中, 取值时用加上  $k*w$ 
42
43 #include <bits/stdc++.h>
44 using namespace std;
45
46 const int N = 20010;
47
48 int f[N], g[N], n, m;
49 int q[N];
50
51 int main() {
52     cin >> n >> m;
53     for (int i = 1; i <= n; i++) {
54         int v, w, s;
55         cin >> v >> w >> s;
56         memcpy(g, f, sizeof f);
57         for (int j = 0; j < v; j++) {
58             int hh = 1, tt = 0;
59             for (int k = j; k <= m; k += v) {
60                 while (hh <= tt && k - q[hh] > s*v) hh++; //队列中最多只能有s+1个值
61                 if (hh <= tt) f[k] = max(f[k], g[q[hh]] + (k - q[hh]) / v * w);
62                 while (hh <= tt && g[q[tt]] - (q[tt] - j) / v * w <= g[k] - (k - j) / v
63                     * w) tt--;
64                 q[++tt] = k;
65             }
66         }
67         cout << f[m] << endl;
68     }

```

## 1.10 分组背包

1  $v[0 \dots n]$ 代表组

```

2
3 f[j] = max(f[j], f[j-v[0]]+w[0], f[j-v[1]]+w[1], f[j-v[2]]+w[2].....)
4
5 枚举三层循环即可

```

### 1.11 混合背包

```

1 假设存在无限个, 一个, 有限个的物品
2
3 将多重背包用二进制优化转换为01背包
4
5 接着按照完全背包和01背包做就行

```

### 1.12 区间 dp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef unsigned long long ull;
5 #define ACM_LOCAL
6
7 const int N = 200 + 5;
8 const int Mod = 1e9 + 7;
9 int n, s[N], f1[N][N], f2[N][N];
10
11 void solve() {
12     cin >> n;
13     for (int i = 1; i <= n; i++) cin >> s[i], s[i+n] = s[i];
14     for (int i = 1; i <= 2*n; i++) s[i] += s[i-1];
15
16     for (int len = 2; len <= n; len++) {
17         for (int i = 1; i + len - 1 <= 2*n; i++) {
18             int j = i + len - 1;
19             f1[i][j] = 1e8;
20             for (int k = i; k < j; k++) {
21                 f1[i][j] = min(f1[i][j], f1[i][k] + f1[k+1][j] + s[j] - s[i-1]);
22                 f2[i][j] = max(f2[i][j], f2[i][k] + f2[k+1][j] + s[j] - s[i-1]);
23             }
24         }
25     }
26     int Max = 0, Min = 1e8;
27     for (int i = 1; i + n - 1 <= 2*n; i++) {
28         Max = max(Max, f2[i][i+n-1]);
29         Min = min(Min, f1[i][i+n-1]);
30     }
31     cout << Min << endl;
32     cout << Max << endl;
33 }
34
35 signed main() {
36     ios_base::sync_with_stdio(false);
37     cin.tie(0);
38     cout.tie(0);
39 #ifdef ACM_LOCAL
40     freopen("in.txt", "r", stdin);
41     freopen("out.txt", "w", stdout);
42 #endif

```

```

43     solve();
44     return 0;
45 }

```

### 1.13 数位 dp

```

1 ll dfs(int pos, int pre, bool limit, bool lead) {
2     if (pos == -1) return 1;
3     if (!limit && !lead && dp[pos][...][...] != -1) return dp[pos][...][...];
4     int End = limit ? a[pos] : 9;
5     ll ans = 0;
6     for (int i = 0; i <= End; i++) {
7         *****//操作1
8     }
9     if (!limit && !lead) dp[pos][...][...] = ans;
10    return ans;
11 }
12 ll calc(ll x) {
13     int pos = 0;
14     while (x) {
15         a[pos++] = x % 10;
16         x /= 10;
17     }
18     return dfs(pos-1, 0, true, true);
19 }

```

### 1.14 四边形优化

```

1 //
2 // Created by acer on 2020/11/23.
3 //
4
5 //m(i,j)=min{m(i,k-1),m(k,j)}+w(i,j)(i≤k≤j)
6 //满足对于 i≤i'<j≤j', 有 w(i',j)≤w(i,j'), 且对于 i≤i'<j≤j', 有 w(i,j)+w(i',j')≤w(i',j)+w(i,
7 //则m函数具有相同性质, 故可以简单的推出m的最优决策点有s(i,j)≤s(i,j+1)≤s(i+1,j+1)
8 int main() {
9     int f[100][100];
10    int m[100][100];
11    int n, INF = 100;
12    for (int i = 1; i <= n; ++i) {
13        m[i][i] = i;
14    }
15    for (int len = 2; len <= n; ++len) // 枚举区间长度
16        for (int l = 1, r = len; r <= n; ++l, ++r) { // 枚举长度为len的所有区间
17            f[l][r] = INF;
18            for (int k = m[l][r - 1]; k <= m[l + 1][r]; ++k)
19                if (f[l][r] > f[l][k] + f[k + 1][r] + w(l, r)) {
20                    f[l][r] = f[l][k] + f[k + 1][r] + w(l, r); // 更新状态值
21                    m[l][r] = k; // 更新 (最小) 最优决策点
22                }
23        }
24 }

```

### 1.15 完全背包

```

1 f[i][j] = max(f[i-1][j], f[i-1][j-v[i]] + w[i], f[i-1][j-2*v[i]] + 2*w[i], ..... )
2
3 f[i][j - v[i]] + w[i] = max(f[i-1][j-v[i]] + w[i], f[i-1][j-2*v[i]] + 2*w[i], ..... )
4
5 f[i][j] = max(f[i-1][j], f[i-1][j-v[i]]+w[i])
6
7 ==> for (int i = v[i]; i <= m; i++)
8
9 因此 j 使用当前的状态更新自己, 所以正序

```

## 1.16 悬线法

```

1 #include<cstdio>
2
3 #define N 2005
4 #define max(a, b) a>b?a:b
5 #define min(a, b) a<b?a:b
6 using namespace std;
7 int up[N][N], left[N][N], right[N][N], ansa, ansb, a[N][N], m, n;
8
9 int main() {
10     scanf("%d%d", &n, &m);
11     for (int i = 1; i <= n; i++)
12         for (int j = 1; j <= m; j++)
13             up[i][j] = 1, left[i][j] = j, right[i][j] = j, scanf("%d", &a[i][j]); // up
14             初值, 读入, left/right 最初值
15
16     for (int i = 1; i <= n; i++)
17         for (int j = 2; j <= m; j++)
18             if (a[i][j] ^ a[i][j - 1])
19                 left[i][j] = left[i][j - 1];
20     for (int i = 1; i <= n; i++)
21         for (int j = m; j > 1; j--)
22             if (a[i][j] ^ a[i][j - 1])
23                 right[i][j - 1] = right[i][j]; //left/right初值, 即 (i, j) 点向左/右的最大宽
24             度
25
26     for (int i = 1; i <= n; i++)
27         for (int j = 1; j <= m; j++) {
28             if (i > 1 && a[i][j] ^ a[i - 1][j])
29                 up[i][j] = up[i - 1][j] + 1, left[i][j] = max(left[i][j], left[i - 1][j]);
30             right[i][j] = min(right[i][j], right[i - 1][j]);
31             int a = right[i][j] - left[i][j] + 1;
32             int b = min(a, up[i][j]);
33             ansa = max(ansa, b * b);
34             ansb = max(ansb, a * up[i][j]);
35         }
36     printf("%d\n%d", ansa, ansb);
37 }

```

## 1.17 有依赖的背包 (树形 dp)

```

1 类似分组背包的思想
2
3 for (int j = m; j >= v[i])
4     for (int k = 0; k <= j; k++)
5         f[x][j] = max(f[x][j], f[x][j-k] + f[y][k]);

```



```
6
7 for (int i = m; i >= v[x]; i--) f[x][i-v[x]] + w[x];
8 for (int i = 0; i < v[x]; i++) f[x][i] = 0;
9 每个子节点为一个组，组内不同的背包容量转移为组内的不同元素
```

### 1.18 01 背包

```
1 f[i][j] = max(f[i-1][j], f[i-1][j-v[i]] + w[i])
2
3 因为 i 只用到前后的两次状态，因此可以压缩成一维的空间
4
5 f[j] = max(f[j], f[j-v[i]] + w[i])
6
7 并且 for (int j = m; j >= v[i]; j--)
8
9 因为f[j-v[i]] + w[i]要用到i-1维的数据更新当前背包空间，所以j倒序
```

## 2 树形结构

### 2.1 lca(倍增)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1e5 + 10;
4  vector<int> g[N];
5
6  int f[N][30], dep[N];
7
8  void dfs(int u, int fa) {
9      dep[u] = dep[fa] + 1; f[u][0] = fa;
10     for (int i = 1; i <= 19; i++) f[u][i] = f[f[u][i-1]][i-1];
11     for (auto &v : g[u]) {
12         if (v == fa) continue;
13         dfs(v, u);
14     }
15 }
16
17 int lca(int x, int y) {
18     if (dep[x] > dep[y]) swap(x, y);
19     for (int i = 19; i >= 0; i--) if (dep[y] - dep[x] >= (1 << i)) y = f[y][i];
20
21     if (x == y) return x;
22     for (int i = 19; i >= 0; i--) {
23         if (f[y][i] != f[x][i])
24             y = f[y][i], x = f[x][i];
25     }
26     return f[x][0];
27 }

```

### 2.2 lca (树链剖分)

```

1  int lca(int u, int v) {
2      while (top[u] != top[v]) {
3          if (d[top[u]] > d[top[v]])
4              u = fa[top[u]];
5          else
6              v = fa[top[v]];
7      }
8      return d[u] > d[v] ? v : u;
9  }

```

### 2.3 次小生成树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define INF 1e16
4  const int N = 1e5 + 200;
5  const int M = 6 * 1e5 + 300;
6  int head[M], edge[M], Next[M], ver[M], tot, fa[M], n, m, father[N][32], deep[N];
7  long long dp[2][N][32], val1, val2, ans_max, ans;
8
9  struct node
10 {
11     int x, y, z, vis;

```

```

12 } s[M];
13
14 int cmp(node a, node b)
15 {
16     return a.z < b.z;
17 }
18
19 struct Edge {
20     void init2() {
21         memset(head, 0, sizeof(head));
22         tot = 0;
23     }
24     void add_edge(int a, int b, int c) {
25         edge[++tot] = b;
26         ver[tot] = c;
27         Next[tot] = head[a];
28         head[a] = tot;
29     }
30     int find(int x) {
31         return x == fa[x] ? x : fa[x] = find(fa[x]);
32     }
33     void Kruskal() {
34         sort(s + 1, s + 1 + m, cmp);
35         for (int i = 1; i <= m; i++)
36         {
37             int a = find(s[i].x), b = find(s[i].y);
38             if (a == b)
39                 continue;
40             s[i].vis = 1;
41             fa[a] = b;
42             ans += s[i].z;
43             add_edge(s[i].x, s[i].y, s[i].z);
44             add_edge(s[i].y, s[i].x, s[i].z);
45         }
46     }
47     void bfs(int root) {
48         deep[root] = 0;
49         queue<int> q;
50         q.push(root);
51         while (q.size()) {
52             int x = q.front(), len = (int)log2(deep[x] + 1);
53             q.pop();
54             for (int i = head[x]; i; i = Next[i]) {
55                 int y = edge[i];
56                 if (y == father[x][0])
57                     continue;
58                 deep[y] = deep[x] + 1;
59                 father[y][0] = x, dp[0][y][0] = ver[i], dp[1][y][0] = -INF;
60                 q.push(y);
61                 for (int t = 1; t <= len; t++) {
62                     father[y][t] = father[father[y][t - 1]][t - 1];
63                     if (dp[0][y][t - 1] != dp[0][father[y][t - 1]][t - 1]) {
64                         dp[0][y][t] = max(dp[0][y][t - 1], dp[0][father[y][t - 1]][t -
1]);
65                         dp[1][y][t] = min(dp[0][y][t - 1], dp[0][father[y][t - 1]][t -
1]);
66                     }
67                     else {
68                         dp[0][y][t] = dp[0][y][t - 1];

```

```

69         dp[1][y][t] = max(dp[1][y][t - 1], dp[1][father[y][t - 1]][t -
1]);
70     }
71 }
72
73     }
74 }
75 }
76 inline void update2(int x) {
77     if (x > val1)
78         val2 = val1, val1 = x;
79     else if (x > val2 && x != val1)
80         val2 = x;
81 }
82 inline void update(int x, int t) {
83     update2(dp[0][x][t]);
84     update2(dp[1][x][t]);
85 }
86 inline void Lca(int x, int y) {
87     val1 = val2 = -INF;
88     if (deep[x] < deep[y])
89         swap(x, y);
90     while (deep[x] > deep[y]) {
91         int t = (int)log2(deep[x] - deep[y]);
92         update(x, t), x = father[x][t];
93     }
94     if (x == y)
95         return;
96     for (int t = (int)log2(deep[x]); t >= 0; t--) {
97         if (father[x][t] != father[y][t]) {
98             update(x, t), update(y, t);
99             x = father[x][t];
100            y = father[y][t];
101        }
102    }
103    update(x, 0), update(y, 0);
104 }
105 } g1;
106
107 void solve() {
108     scanf("%d%d", &n, &m);
109     g1.init2();
110     for (int i = 1; i <= m; i++) {
111         int a, b, c;
112         scanf("%d%d%d", &a, &b, &c);
113         s[i].x = a, s[i].y = b, s[i].z = c;
114         fa[i] = i;
115     }
116     g1.Kruskal();
117     g1.bfs(1);
118     ans_max = INF;
119     for (int i = 1; i <= m; i++) {
120         if (!s[i].vis) {
121             g1.Lca(s[i].x, s[i].y);
122             if (val1 != s[i].z)
123                 ans_max = min(ans_max, ans - val1 + s[i].z);
124             else
125                 ans_max = min(ans_max, ans - val2 + s[i].z);
126         }

```

```

127     }
128     printf("%lld\n", ans_max);
129 }
130
131 int main() {
132     ios_base::sync_with_stdio(false);
133     cin.tie(nullptr);
134     cout.tie(nullptr);
135     #ifdef ACM_LOCAL
136         freopen("in.txt", "r", stdin);
137         //freopen("out.txt", "w", stdout);
138         signed test_index_for_debug = 1;
139         char acm_local_for_debug = 0;
140         do {
141             if (acm_local_for_debug == '$') exit(0);
142             if (test_index_for_debug > 20)
143                 throw runtime_error("Check the stdin!!!");
144             auto start_clock_for_debug = clock();
145             solve();
146             auto end_clock_for_debug = clock();
147             cout << "Test " << test_index_for_debug << " successful" << endl;
148             cerr << "Test " << test_index_for_debug++ << " Run Time: "
149                  << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC <<
150                  "s" << endl;
151             cout << "-----" << endl;
152         } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
153     #else
154         solve();
155     #endif
156     return 0;
157 }

```

## 2.4 点分治

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  #include <queue>
6  #include <stack>
7  #include <cmath>
8  #include <bitset>
9  #include <map>
10
11 using namespace std;
12 typedef long long ll;
13 const int N = 1e4 + 5, M = 2e5 + 5, INF = 0x3f3f3f3f;
14
15 struct Edge {
16     int to, next, vi;
17 } e[N << 1];
18
19 int h[N], cnt;
20
21 void add(int u, int v, int w) {
22     e[cnt].to = v;
23     e[cnt].vi = w;
24     e[cnt].next = h[u];

```

```

25     h[u] = cnt++;
26 }
27
28 ll ans = 0;
29 int root, sum, n;
30 int sz[N], mx[N], vis[N];
31
32 void getroot(int x, int fa) {
33     sz[x] = 1, mx[x] = 0;
34     for (int i = h[x]; ~i; i = e[i].next) {
35         int y = e[i].to;
36         if (vis[y] || y == fa) continue;
37         getroot(y, x);
38         sz[x] += sz[y];
39         mx[x] = max(mx[x], sz[y]);
40     }
41     mx[x] = max(mx[x], sum - sz[x]);
42     if (mx[x] < mx[root]) root = x;
43 }
44
45 int d[N], dep[N];
46
47 void getd(int x, int fa) {
48     d[++d[0]] = dep[x];
49     for (int i = h[x]; ~i; i = e[i].next) {
50         int y = e[i].to;
51         if (vis[y] || y == fa) continue;
52         dep[y] = dep[x] + e[i].vi;
53         getd(y, x);
54     }
55 }
56
57 int K;
58
59 int cal(int x, int now) {
60     dep[x] = now, d[0] = 0;
61     getd(x, -1);
62     sort(d + 1, d + d[0] + 1);
63     int res = 0, l = 1, r = d[0];
64     while (l < r) {
65         if (d[l] + d[r] > K) r--;
66         else res += r - l, l++;
67     }
68     return res;
69 }
70
71 void work(int x) {
72     ans += cal(x, 0);
73     vis[x] = 1;
74     for (int i = h[x]; ~i; i = e[i].next) {
75         int y = e[i].to;
76         if (vis[y]) continue;
77         ans -= cal(y, e[i].vi);
78         sum = sz[y], root = 0;
79         getroot(y, -1);
80         work(root);
81     }
82 }
83

```

```

84 int main() {
85     while (scanf("%d %d", &n, &K), n) {
86         cnt = 0;
87         memset(h, -1, sizeof h);
88         memset(vis, 0, sizeof vis);
89         for (int i = 1; i <= n - 1; i++) {
90             int u, v, w;
91             scanf("%d %d %d", &u, &v, &w);
92             add(u, v, w), add(v, u, w);
93         }
94         root = 0, ans = 0, sum = n, mx[0] = INF;
95         getroot(1, -1);
96         work(root);
97         printf("%lld\n", ans);
98     }
99 }

```

## 2.5 树的半径

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 5000 + 5;
4  int dis[N], n, m, h[N], cnt, maxd1[N], maxd2[N], mv[N], st, r;
5
6  struct edge {
7      int u, to, next, vi;
8  }e[N<<1];
9
10 void add(int u, int v, int w) {
11     e[cnt].u = u;
12     e[cnt].to = v;
13     e[cnt].vi = w;
14     e[cnt].next = h[u];
15     h[u] = cnt++;
16 }
17
18 void dp(int u, int fa) {
19     maxd1[u] = 0, maxd2[u] = 0;
20     for (int i = h[u]; ~i; i = e[i].next) {
21         int v = e[i].to;
22         if (v == fa) continue;
23         dp(v, u);
24         int tmp = maxd1[v] + e[i].vi;
25         if (tmp > maxd1[u]) maxd2[u] = maxd1[u], maxd1[u] = tmp, mv[u] = v;
26         else if (tmp > maxd2[u]) maxd2[u] = tmp;
27     }
28     st = max(st, maxd1[u] + maxd2[u]);
29 }
30
31 void dfs(int u, int fa, int fr) {
32     r = min(r, max(fr, maxd1[u]));
33     for (int i = h[u]; ~i; i = e[i].next) {
34         int v = e[i].to;
35         if (v == fa) continue;
36         if (mv[u] == v) dfs(v, u, max(fr + e[i].vi, maxd2[u] + e[i].vi));
37         else dfs(v, u, max(maxd1[u] + e[i].vi, fr + e[i].vi));
38     }
39 }

```

```

40
41 int main() {
42     cin >> n;
43     memset(h, -1, sizeof h);
44     for (int i = 1; i <= n - 1; i++) {
45         int x, y, z;
46         cin >> x >> y >> z;
47         add(x, y, z), add(y, x, z);
48     }
49     dp(1, 0);
50     r = 0x3f3f3f3f;
51     dfs(1, 0, 0);
52     cout << r << endl;
53 }

```

## 2.6 树的直径

```

1  //-----树形dp
2  int f[N];
3  int ans;
4  vector<int> g[N];
5  void dp(int u, int fa) {
6      for (auto v : g[u]) {
7          if (v == fa) continue;
8          dp(v, u);
9          ans = max(ans, f[u] + f[v] + 1);
10         f[u] = max(f[u], f[v] + 1);
11     }
12 }
13
14
15 //-----两次dfs
16
17 int dfs(int u, int fa) {
18     int maxpos = u;
19     for (auto v : g[u]) {
20         if (vis[v] || v == fa) continue;
21         dis[v] = dis[u] + 1;
22         pre[v] = u;
23         int z = dfs(v, u);
24         if (dis[z] > dis[maxpos]) maxpos = z;
25     }
26     return maxpos;
27 }

```

## 2.7 树哈希

```

1
2 vector<int> g[N];
3 int prime[N], siz[N], mx[N], cnt, rt, n;
4 bool is_prime[N];
5 ull Hash[N];
6 void get_prime(int n) {
7     memset(is_prime, true, sizeof is_prime);
8     for (int i = 2; i <= n; i++) {
9         if (is_prime[i]) prime[++cnt] = i;
10        for (int j = 1; j <= cnt && prime[j] * i <= n; j++) {

```



```

11         is_prime[prime[j] * i] = false;
12         if (i % prime[j] == 0) break;
13     }
14 }
15 }
16 void getRt(int x, int fa) {
17     siz[x] = 1, mx[x] = 0;
18     for (auto v : g[x]) {
19         if (v == fa) continue;
20         getRt(v, x);
21         siz[x] += siz[v];
22         mx[x] = max(mx[x], siz[v]);
23     }
24     mx[x] = max(mx[x], n - siz[x]);
25     if (mx[x] < mx[rt]) rt = x;
26 }
27 ull getTreeHash(int x, int fa) {
28     siz[x] = 1;
29     ull res = 1;
30     for (auto v : g[x]) {
31         if (v == fa) continue;
32         res += getTreeHash(v, x) * prime[siz[v]];
33         siz[x] += siz[v];
34     }
35     return res;
36 }

```

## 2.8 树链剖分

```

1  const int N = 1e5 + 10;
2  vector<int> g[N];
3  int son[N], siz[N], dep[N], fat[N], dfn[N], rnk[N], top[N], tot;
4  void dfs1(int u, int fa) {
5      son[u] = -1; siz[u] = 1; dep[u] = dep[fa] + 1; fat[u] = fa;
6      for (auto v : g[u]) {
7          if (v == fa) continue;
8          dfs1(v, u);
9          siz[u] += siz[v];
10         if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
11     }
12 }
13
14 void dfs2(int u, int t) {
15     rnk[dfn[u] = ++tot] = u; top[u] = t;
16     if (son[u] == -1) return;
17     dfs2(son[u], t);
18     for (auto v : g[u]) {
19         if (v != son[u] && v != fat[u]) dfs2(v, v);
20     }
21 }
22 // -----结合线段树操作
23 int querymax(int x, int y) {
24     int ans = -1;
25     while (top[x] != top[y]) {
26         if (dep[top[x]] < dep[top[y]]) swap(x, y);
27         ans = max(ans, st.query(1, dfn[top[x]], dfn[x]));
28         x = fat[top[x]];
29     }

```

```

30     if (dfn[x] > dfn[y]) swap(x, y);
31     ans = max(ans, st.query1(1, dfn[x], dfn[y])); //在点权转化成边权时, dfn[x]+1开始
32     return ans;
33 }
34
35 int querysum(int x, int y) {
36     int ans = 0;
37     while (top[x] != top[y]) {
38         if (dep[top[x]] < dep[top[y]]) swap(x, y);
39         ans += st.query2(1, dfn[top[x]], dfn[x]);
40         x = fa[top[x]];
41     }
42     if (dfn[x] > dfn[y]) swap(x, y);
43     ans += st.query2(1, dfn[x], dfn[y]);
44     return ans;
45 }
46
47 void update(int x, int y, int c) {
48     while (top[x] != top[y]) {
49         if (dep[top[x]] < dep[top[y]]) swap(x, y);
50         st.update(1, dfn[top[x]], dfn[x], c);
51         x = fa[top[x]];
52     }
53     if (dfn[x] > dfn[y]) swap(x, y);
54     st.update(1, dfn[x], dfn[y], c);
55 }

```

## 2.9 树上 k-th 祖先 $O(q+n\log n)$

```

1  //
2  // Created by SANZONG on 2021/7/6.
3  //
4
5  #include "bits/stdc++.h"
6
7
8  using namespace std;
9  const int maxn = 6e5;
10 int head[maxn << 1];
11 int cnt;
12 #define ui unsigned int
13 ui s;
14
15 inline ui get(ui x) {
16     x ^= x << 13;
17     x ^= x >> 17;
18     x ^= x << 5;
19     return s = x;
20 }
21
22 #define int long long
23 struct node {
24     int next;
25     int to;
26 } a[maxn << 1];
27
28 void add(int u, int v) {
29     a[++cnt].next = head[u];

```

```

30     a[cnt].to = v;
31     head[u] = cnt;
32 }
33
34 int f[maxn][33];
35 int dep[maxn];
36 int son[maxn];
37 int dlen[maxn];    //每个点所在的长链长度，画一下就知道
38 int g[maxn];       //log
39
40 void dfs(int u) {
41     dlen[u] = dep[u] = dep[f[u][0]] + 1;
42     for (int i = 1; i <= 30; ++i) {
43         f[u][i] = f[f[u][i - 1]][i - 1];
44     }
45     for (int i = head[u]; i; i = a[i].next) {
46         int v = a[i].to;
47         if (v == f[u][0]) continue;
48         dlen[v] = dep[v] = dep[u] + 1;    //向下时先等于深度
49         dfs(v);
50         dlen[u] = max(dlen[u], dlen[v]);    //逆推时一直取子树最大深度
51         if (dlen[v] > dlen[son[u]]) son[u] = v;
52     }
53 }
54
55 int U[maxn], D[maxn], id[maxn], cnt_t, top[maxn];
56
57 void dfs_t(int u, int root) {
58     id[u] = ++cnt_t;    //一条相同长链上的id是连续的。
59     D[cnt_t] = u;    //dfs序为cnt_t的点的连顶向下数组。
60     U[cnt_t] = root;    //dfs序为cnt_t的点的连顶向上数组。
61     if (son[u]) {
62         top[son[u]] = top[u];
63         dfs_t(son[u], f[root][0]);
64     }
65     for (int i = head[u]; i; i = a[i].next) {
66         int v = a[i].to;
67         if (v == f[u][0] || v == son[u]) continue;
68         top[v] = v;    //短链
69         dfs_t(v, v);
70     }
71 }
72
73 int getkfa(int x, int k) {
74     if (k == 0) return x;
75     x = f[x][g[k]];    //直接跳到链上
76     k -= 1 << g[k];    //跳到f[x][g[k]]的消耗
77     k -= dep[x] - dep[top[x]];    //x 从x跳到顶点
78     x = top[x];
79     return k >= 0 ? U[id[x]+k] : D[id[x]-k];
80 }
81
82 //int getkfa(int x,int k)
83 //{
84 //    if (k == 0) return x;
85 //    int d = dep[x] - k;
86 //    for (int i = 30; i >= 0; --i) {
87 //        if (dep[f[x][i]] > d) x = f[x][i];
88 //    }

```

```

89 // return f[x][0];
90 //}
91 signed main() {
92     ios::sync_with_stdio(0);
93     // freopen("in.txt", "r", stdin);
94     int n, q, root;;
95     cin >> n >> q >> s;
96     g[0] = -1;
97     for (int i = 1; i <= n; ++i) { g[i] = g[i >> 1] + 1; } //求log
98     for (int i = 1; i <= n; ++i) {
99         cin >> f[i][0];
100         if (f[i][0] == 0) {
101             root = i;
102         } else {
103             add(f[i][0], i);
104             add(i, f[i][0]);
105         }
106     }
107     dep[root] = 1;
108     dfs(root);
109     top[root] = root;
110     dfs_t(root, root);
111     // for (int i = 1; i <= n; ++i) {
112     //     cout << id[i] << endl;
113     // }
114     int ans = 0;
115     int res = 0;
116     for (int i = 1; i <= q; ++i) {
117         int xi = ((get(s) ^ ans) % n) + 1;
118         // cout << (get(s) ^ ans) << endl;
119         int ki = (get(s) ^ ans) % dep[xi];
120         ans = getkfa(xi, ki);
121         // cout << xi << ' ' << ki << ' ' << ans << endl;
122         res ^= (i * ans);
123     }
124     cout << res << endl;
125 }
126 }

```

## 2.10 树上差分 (边差分)

- 1 边差分:  $c[x] += val, c[y] += val, c[lca(x, y)] -= 2*val;$
- 2
- 3 点差分:  $c[x] += val, c[y] += val, c[lca(x, y)] -= val, c[fa[lca(x, y)]] -= val;$

## 2.11 树上距离 (lca tarjan)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1e5 + 10;
4 vector<int> query[N], query_id[N];
5
6 struct edge {
7     int to, next, vi;
8 }e[N*2];
9
10 int T, n, q, h[N], cnt, ans[N], d[N], fa[N], vis[N];

```

```

11
12 void add_query(int u, int v, int id) {
13     query[u].push_back(v), query_id[u].push_back(id);
14     query[v].push_back(u), query_id[v].push_back(id);
15 }
16
17 void add(int u, int v, int w) {
18     e[cnt].to = v;
19     e[cnt].vi = w;
20     e[cnt].next = h[u];
21     h[u] = cnt++;
22 }
23
24 int find(int x) {
25     if (fa[x] == x) return x;
26     else return fa[x] = find(fa[x]);
27 }
28
29 void tarjan(int x) {
30     vis[x] = 1;
31     for (int i = h[x]; ~i; i = e[i].next) {
32         int v = e[i].to;
33         if (vis[v]) continue;
34         d[v] = d[x] + e[i].vi;
35         tarjan(v);
36         fa[v] = x;
37     }
38
39     for (int i = 0; i < query[x].size(); i++) {
40         int y = query[x][i], id = query_id[x][i];
41         if (vis[y] == 2) {
42             int lca = find(y);
43             ans[id] = min(ans[id], d[x] + d[y] - 2 * d[lca]);
44         }
45     }
46     vis[x] = 2;
47 }
48
49 int main() {
50     scanf("%d", &T);
51     while (T--) {
52         scanf("%d %d", &n, &q);
53         for (int i = 1; i <= n; i++) {
54             h[i] = -1;
55             d[i] = 0;
56             query[i].clear(), query_id[i].clear();
57             fa[i] = i;
58             vis[i] = 0;
59         }
60         cnt = 0;
61         for (int i = 1; i <= n - 1; i++) {
62             int x, y, z;
63             scanf("%d %d %d", &x, &y, &z);
64             add(x, y, z), add(y, x, z);
65         }
66         for (int i = 1; i <= q; i++) {
67             int x, y;
68             scanf("%d %d", &x, &y);
69             if (x == y) ans[i] = 0;

```

```

70         else {
71             add_query(x, y, i);
72             ans[i] = 999999999;
73         }
74     }
75     tarjan(1);
76     for (int i = 1; i <= q; i++) {
77         printf("%d\n", ans[i]);
78     }
79 }
80 }

```

## 2.12 树上距离 (lca 倍增)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 4e4 + 10;
4
5  int h[N], dis[N], d[N], f[N][30];
6  int T, n, m, cnt, s;
7
8  struct edge {
9      int to, next, vi;
10 } e[N<<1];
11
12 void add(int u, int v, int vi) {
13     e[cnt].to = v;
14     e[cnt].vi = vi;
15     e[cnt].next = h[u];
16     h[u] = cnt++;
17 }
18
19 void dfs(int x, int fa) {
20     f[x][0] = fa, d[x] = d[fa] + 1;
21     for (int i = 1; i <= 20; i++) f[x][i] = f[f[x][i-1]][i-1];
22     for (int i = h[x]; ~i; i = e[i].next) {
23         int y = e[i].to;
24         if (y != fa) dfs(y, x);
25     }
26 }
27
28 int lca(int x, int y) {
29     if (d[x] > d[y]) swap(x, y);
30     for (int i = 20; i >= 0; i--) if (d[f[y][i]] >= d[x]) y = f[y][i];
31
32     if (x == y) return y;
33     for (int i = 20; i >= 0; i--) {
34         if (f[y][i] != f[x][i]) {
35             y = f[y][i];
36             x = f[x][i];
37         }
38     }
39     return f[x][0];
40 }
41
42 int main() {
43     memset(h, -1, sizeof h);
44     scanf("%d %d %d", &n, &m, &s);

```

```

45     for (int i = 1; i <= n - 1; i++) {
46         int a, b, c;
47         scanf("%d %d", &a, &b);
48         add(a, b, 1), add(b, a, 1);
49     }
50     dfs(s, 0);
51     for (int i = 1; i <= m; i++) {
52         int a, b;
53         scanf("%d %d", &a, &b);
54         printf("%d\n", lca(a, b));
55     }
56 }

```

### 2.13 树上启发式合并

```

1  #include <bits/stdc++.h>
2  #define ACM_LOCAL
3  using namespace std;
4  typedef long long ll;
5  typedef pair<int, int> PII;
6  const int INF = 0x3f3f3f3f;
7  const int N = 2e5 + 10;
8  const int M = 1e6 + 10;
9  const int MOD = 1e9 + 7;
10 int sz[N], son[N], h[N], tot, col[N], cnt[N], vis[N], mx;
11 ll sum, ans[N];
12 struct Edge {
13     int to, next;
14 }e[M];
15
16 void add(int u, int v) {
17     e[tot].to = v;
18     e[tot].next = h[u];
19     h[u] = tot++;
20 }
21
22 void dfs(int u, int fa) {
23     sz[u] = 1;
24     for (int i = h[u]; ~i; i = e[i].next) {
25         int v = e[i].to;
26         if (v == fa) continue;
27         dfs(v, u);
28         sz[u] += sz[v];
29         if (!son[u] || sz[v] > sz[son[u]])
30             son[u] = v;
31     }
32 }
33
34 void count(int u, int fa, int k) {
35     //统计子树信息
36
37     //
38     for (int i = h[u]; ~i; i = e[i].next) {
39         int v = e[i].to;
40         if (v == fa || vis[v]) continue;
41         count(v, u, k);
42     }
43 }

```

```

44 }
45
46 void dsu(int u, int fa, int keep) {
47     for (int i = h[u]; ~i; i = e[i].next) {
48         int v = e[i].to;
49         if (v == fa || son[u] == v) continue;
50         dsu(v, u, 0); // 查询轻儿子
51     }
52     if (son[u]) dsu(son[u], u, 1), vis[son[u]] = 1; // 查询重儿子
53     count(u, fa, 1); // 统计子树信息
54     // 统计答案
55     if (son[u]) vis[son[u]] = 0;
56     if (!keep) count(u, fa, -1), mx = sum = 0; // 撤回信息
57 }

```

## 2.14 k 级祖先

```

1  vector<int> g[N];
2  int dep[N], f[N][30];
3  void dfs(int x, int fa) {
4      f[x][0] = fa; dep[x] = dep[fa] + 1;
5      for (int i = 1; i <= 20; i++) {
6          f[x][i] = f[f[x][i-1]][i-1];
7      }
8      for (auto &v : g[x]) {
9          if (v == fa) continue;
10         dfs(v, x);
11     }
12 }
13 int find_Kth(int x, int k) {
14     int bit = 0;
15     while (k) {
16         if (k & 1) x = f[x][bit];
17         k >>= 1;
18         bit++;
19     }
20     return x;
21 }

```



### 3 数据结构

#### 3.1 Max XOR(tire)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100010;
4  const int M = 3000010;
5  int n, a[N];
6  int trie[M][2];
7  int cnt;
8  void insert(int x){
9      int p = 0;
10     for (int i = 30; ~i; i--){
11         int w = x >> i & 1;
12         if (!trie[p][w]) trie[p][w] = ++cnt;
13         p = trie[p][w];
14     }
15 }
16
17 int query(int x){
18     int p = 0;
19     int res = 0;
20     for (int i = 30; ~i; i--){
21         int w = x >> i & 1;
22
23         if (trie[p][w ^ 1]){
24             res += (1 << i);
25             p = trie[p][w ^ 1];
26         }
27         else p = trie[p][w];
28     }
29     return res;
30 }

```

#### 3.2 splay(待完善)

```

1  #include <bits/stdc++.h>
2
3  #define ACM_LOCAL
4  #define fi first
5  #define se second
6  #define pb push_back
7  using namespace std;
8  typedef long long ll;
9  typedef pair<int, int> PII;
10 const int N = 5e5 + 10, M = 5e5 + 10, INF = 0x3f3f3f3f;
11 const int MOD = 1e9 + 7;
12 int n, m, k, cnt;
13 int a[N];
14
15 struct SplayTree {
16     int sz[N], ch[N][2], cnt[N], fa[N], rev[N], val[N], ad[N];
17     int tot, rt;
18     #define rt ch[anc][0]
19     #define keynode ch[ch[rt][1]][0]
20     const int anc = 0;
21     void clear(int x) {sz[x] = cnt[x] = ch[x][0] = ch[x][1] = val[x] = ad[N] = 0;}

```

```

22 void init() {tot = 0, clear(0);}
23 int get(int x) {return ch[fa[x]][1] == x;}
24
25 int newnode(int k) {
26     int x = ++tot; clear(x);
27     val[x] = k;
28     sz[x] = cnt[x] = 1;
29     return tot;
30 }
31
32 void setc(int x, int y, int d) { //基本点操作: 设置x的d号儿子为y
33     ch[x][d] = y;
34     if (y) fa[y] = x;
35     if (x) push_up(x);
36 }
37
38 void reverse(int x) {
39     if (x == 0) return; //Necessary
40     swap(ch[x][0], ch[x][1]);
41     rev[x] ^= 1;
42 }
43 void add(int x, int k) {
44     if (x == 0) return;
45     val[x] += k;
46     ad[x] += k;
47 }
48
49 void push_up(int x) {
50     sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + cnt[x];
51 }
52
53 void push_down(int x) {
54     if (!x) return;
55     if (rev[x]) {
56         reverse(ch[x][0]);
57         reverse(ch[x][1]);
58         rev[x] = 0;
59     }
60     if (ad[x]) {
61         add(ch[x][0], ad[x]);
62         add(ch[x][1], ad[x]);
63         ad[x] = 0;
64     }
65 }
66 void rotate(int x) {
67     int f = fa[x];
68     int ff = fa[f];
69     bool d = get(x);
70     bool dd = get(f);
71     setc(f, ch[x][!d], d); //第一步: 把与x原父节点同方向的子树取作原父节点f的子树
72     setc(x, f, !d); //第二步: 使与x原父节点同方向的子树连接原父节点f
73     setc(ff, x, dd); //第三步: 使得x替代原父节点f, 变为新父节点ff的子树
74 }
75
76 void splay(int x, int anc = 0) { //双旋
77     if (x == 0) return;
78     while (fa[x] != anc) {
79         push_down(fa[x]);
80         push_down(x);

```

```

81         if (fa[fa[x]] != anc) rotate(get(x) == get(fa[x]) ? fa[x] : x);
82         rotate(x);
83     }
84 }
85 int rank(int k) { //排名
86     int x = rt;
87     while (1) {
88         if (x == 0) return 0;
89         if (k == val[x]) { splay(x); return x; }
90         bool d = k > val[x];
91         x = ch[x][d];
92     }
93     splay(x);
94 }
95
96 void ins(int k) {
97     int x = rank(k);
98     if (x) { cnt[x]++; sz[x]++; return; }
99     int fa = anc; x = rt;
100    bool d = 0;
101    while (x) {
102        fa = x;
103        d = k > val[x];
104        x = ch[x][d];
105    }
106    x = newnode(k);
107    setc(fa, x, d);
108    splay(x);
109 }
110
111 int Min(int x) { //子树中最小节点
112     push_down(x);
113     while (ch[x][0]) x = ch[x][0], push_down(x);
114     return x;
115 }
116 int Max(int x) { //子树中最大节点
117     push_down(x);
118     while (ch[x][1]) x = ch[x][1], push_down(x);
119     return x;
120 }
121 int kth(int k) {
122     ++k; //这里涉及到区间操作, 我们在左右界各添加新节点, 因此进入时要++k
123     int x = rt;
124     while (1) {
125         push_down(x);
126         if (sz[ch[x][0]] >= k) x = ch[x][0];
127         else if (sz[ch[x][0]] + cnt[x] >= k) return x;
128         else
129             {
130                 k -= sz[ch[x][0]] + cnt[x];
131                 x = ch[x][1];
132             }
133     }
134 }
135 void del(int x) {
136     splay(x); //转到根后便不再需要pushdown()
137     if (ch[x][0] == 0) setc(anc, ch[x][1], 0); //如果没有左子树, 则直接把右子树放到树根
138     else
139         {

```

```

140         setc(anc, ch[x][0], 0);           //第一步: 把左子树放到树根
141         splay(Max(ch[x][0]), anc);        //第二步: 把左子树最大节点转到树根
142         setc(rt, ch[x][1], 1);           //第三步: 把右子树接到树根上
143     }
144 }
145 //查找操作: 返回树中[l,r]区间段的根节点
146 int segment(int l, int r) {
147     splay(kth(l - 1), anc);
148     splay(kth(r + 1), rt);
149     return keynode;
150 }
151 //分离操作: 把[l,r]区间段从树中分离
152 int split(int l, int r) {
153     int x = segment(l, r); fa[x] = 0;
154     setc(ch[rt][1], 0, 0);
155     push_up(rt);
156     return x;
157 }
158 //合并操作, 把子树x插入到pos位置的右侧
159 void inspos(int x, int pos) {
160     segment(pos + 1, pos); //使得pos位于根, pos+1位于根的右子树
161     setc(ch[rt][1], x, 0);
162     push_up(rt);
163 }
164 //中序遍历
165 void ldr(int x) {
166     if (ch[x][0]) ldr(ch[x][0]);
167     printf("%d ", x);
168     if (ch[x][1]) ldr(ch[x][1]);
169 }
170 void add(int l, int r, int k) { //区间加
171     int x = segment(l, r);
172     add(x, k);
173     splay(x);
174 }
175
176 void reverse(int l, int r) { //区间翻转
177     int x = segment(l, r);
178     reverse(x);
179     splay(x);
180 }
181
182
183 }spy;
184
185 void solve() {
186     int n, opt, x;
187     scanf("%d", &n);
188     for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
189 }
190
191 int main() {
192     ios_base::sync_with_stdio(false);
193     cin.tie(0);
194     cout.tie(0);
195     #ifdef ACM_LOCAL
196     freopen("input", "r", stdin);

```

```

199     freopen("output", "w", stdout);
200 #endif
201     solve();
202     return 0;
203 }

```

### 3.3 ST 表

```

1 namespace ST {
2     int mi[N][21], ma[N][21], lg[N], a[N], gcd[N][21];
3     int cmp1(int x, int y) {
4         return a[x] < a[y] ? x : y;
5     }
6     int cmp2(int x, int y) {
7         return a[x] > a[y] ? x : y;
8     }
9     void init(int n) {
10         for (int i = 1; i <= n; i++) {
11             cin >> a[i];
12             gcd[i][0] = a[i];
13             ma[i][0] = mi[i][0] = i, lg[i] = log2(i);
14         }
15         for (int i = 1; i <= 20; i++) {
16             for (int j = 1; j + (1 << i) - 1 <= n; j++) {
17                 mi[j][i] = cmp1(mi[j][i - 1], mi[j + (1 << (i - 1))][i - 1]);
18                 ma[j][i] = cmp2(ma[j][i - 1], ma[j + (1 << (i - 1))][i - 1]);
19                 gcd[j][i] = __gcd(gcd[j][i - 1], gcd[j + (1 << (i - 1))][i - 1]);
20             }
21         }
22     }
23     int qry_mi(int l, int r) {
24         int k = lg[r - l + 1];
25         return a[cmp1(mi[l][k], mi[r - (1 << k) + 1][k])];
26     }
27     int qry_ma(int l, int r) {
28         int k = lg[r - l + 1];
29         return a[cmp2(ma[l][k], ma[r - (1 << k) + 1][k])];
30     }
31     int qry_pmi(int l, int r) {
32         int k = lg[r - l + 1];
33         return cmp1(mi[l][k], mi[r - (1 << k) + 1][k]);
34     }
35     int qry_pma(int l, int r) {
36         int k = lg[r - l + 1];
37         return cmp2(ma[l][k], ma[r - (1 << k) + 1][k]);
38     }
39     int qry_gcd(int l, int r) {
40         int k = lg[r - l + 1];
41         return __gcd(gcd[l][k], gcd[r - (1 << k) + 1][k]);
42     }
43 }

```

### 3.4

```

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

```

```

4 typedef long long ll;
5 typedef pair<int, int> PII;
6 const int N = 1e5 + 100, M = 5e5 + 5, INF = 0x3f3f3f3f;
7 mt19937 rnd(233);
8 struct Treap {
9     int lc[N], rc[N], val[N], key[N], sz[N];
10    int cnt, root;
11
12    inline int newnode(int v) {
13        val[++cnt] = v;
14        key[cnt] = rnd();
15        sz[cnt] = 1;
16        return cnt;
17    }
18
19    inline void update(int now) {
20        sz[now] = sz[lc[now]] + sz[rc[now]] + 1;
21    }
22
23    void split(int now, int v, int &x, int &y) {
24        if (!now) x = y = 0;
25        else {
26            if (val[now] <= v) {
27                x = now;
28                split(rc[now], v, rc[now], y);
29            } else {
30                y = now;
31                split(lc[now], v, x, lc[now]);
32            }
33            update(now);
34        }
35    }
36
37    int merge(int x, int y) {
38        if (!x || !y) return x + y;
39        if (key[x] > key[y]) {
40            rc[x] = merge(rc[x], y);
41            update(x);
42            return x;
43        } else {
44            lc[y] = merge(x, lc[y]);
45            update(y);
46            return y;
47        }
48    }
49
50    int x, y, z;
51
52    inline void insert(int val) {
53        split(root, val, x, y);
54        root = merge(merge(x, newnode(val)), y);
55    }
56
57    inline void del(int val) {
58        split(root, val, x, z);
59        split(x, val - 1, x, y);
60        y = merge(lc[y], rc[y]);
61        root = merge(merge(x, y), z);
62    }

```

```

63
64     inline int getrank(int v) { //查询v的排名
65         split(root, v - 1, x, y);
66         int rank = sz[x] + 1;
67         root = merge(x, y);
68         return rank;
69     }
70
71     inline int getnum(int rank) { //查询排名第rank的数
72         int now = root;
73         while (now) {
74             if (sz[lc[now]] + 1 == rank)
75                 break;
76             else if (sz[lc[now]] >= rank)
77                 now = lc[now];
78             else {
79                 rank -= sz[lc[now]] + 1;
80                 now = rc[now];
81             }
82         }
83         return val[now];
84     }
85
86     inline int pre(int v) {
87         split(root, v - 1, x, y);
88         int now = x;
89         while (rc[now])
90             now = rc[now];
91         int pre = val[now];
92         root = merge(x, y);
93         return pre;
94     }
95
96     inline int nxt(int v) {
97         split(root, v, x, y);
98         int now = y;
99         while (lc[now])
100             now = lc[now];
101         int nxt = val[now];
102         root = merge(x, y);
103         return nxt;
104     }
105 }fhq;

```

### 3.5 Treap(文艺平衡树)

```

1  #include <bits/stdc++.h>
2  #define ACM_LOCAL
3  using namespace std;
4  typedef long long ll;
5  typedef pair<int, int> PII;
6  const int N = 1e5 + 10, M = 1e5 + 10, INF = 0x3f3f3f3f;
7  const int MOD = 1e9 + 7;
8  mt19937 rnd(233);
9  struct Treap {
10     int key[N], lc[N], rc[N], val[N], sz[N];
11     bool reverse[N];
12     ll sum[N], tag[N];

```

```

13     int cnt, rt;
14
15     inline int newnode(int v) {
16         val[++cnt] = v;
17         sz[cnt] = 1;
18         key[cnt] = rnd();
19         reverse[cnt] = false;
20         return cnt;
21     }
22     inline void push_up(int now) {
23         sz[now] = sz[lc[now]] + sz[rc[now]] + 1;
24         sum[now] = sum[lc[now]] + sum[rc[now]] + val[now];
25     }
26     inline void push_down(int now) {
27         if (reverse[now]) {
28             swap(lc[now], rc[now]);
29             reverse[lc[now]] ^= 1;
30             reverse[rc[now]] ^= 1;
31             reverse[now] = false;
32         }
33         if (tag[now]) {
34             sum[lc[now]] += sz[lc[now]] * tag[now];
35             val[lc[now]] += tag[now];
36             tag[lc[now]] += tag[now];
37
38             sum[rc[now]] += sz[rc[now]] * tag[now];
39             val[rc[now]] += tag[now];
40             tag[rc[now]] += tag[now];
41         }
42         tag[now] = 0;
43     }
44     void split(int now, int siz, int &x, int &y) {
45         if (!now) x = y = 0;
46         else {
47             push_down(now);
48             if (sz[lc[now]] < siz) {
49                 x = now;
50                 split(rc[now], siz-sz[lc[now]]-1, rc[now], y);
51             } else {
52                 y = now;
53                 split(lc[now], siz, x, lc[now]);
54             }
55             push_up(now);
56         }
57     }
58     int merge(int x, int y) {
59         if (!x || !y) return x + y;
60         if (key[x] < key[y]) {
61             push_down(x);
62             rc[x] = merge(rc[x], y);
63             push_up(x);
64             return x;
65         } else {
66             push_down(y);
67             lc[y] = merge(x, lc[y]);
68             push_up(y);
69             return y;
70         }
71     }

```



```

72 void rev(int l, int r) {
73     int x, y, z;
74     split(rt, l-1, x, y);
75     split(y, r-l+1, y, z);
76     reverse[y] ^= 1;
77     rt = merge(merge(x, y), z);
78 }
79 void ldr(int now) {
80     if (!now) return;
81     push_down(now);
82     ldr(lc[now]);
83     printf("%d ", val[now]);
84     ldr(rc[now]);
85 }
86 void add(int l, int r, int v) {
87     int x, y, z;
88     split(rt, l-1, x, y);
89     split(y, r-l+1, y, z);
90     tag[y] += v;
91     sum[y] += sz[y] * v;
92     val[y] += v;
93     rt = merge(merge(x, y), z);
94 }
95 ll query(int l, int r) {
96     ll ans = 0;
97     int x, y, z;
98     split(rt, l-1, x, y);
99     split(y, r-l+1, y, z);
100    ans = sum[y];
101    rt = merge(merge(x, y), z);
102    return ans;
103 }
104 void insert(int pos, int v) {
105     int x, y, z;
106     split(rt, pos-1, x, y);
107     z = newnode(v);
108     rt = merge(merge(x, z), y);
109 }
110 }fhq;

```

### 3.6 并查集

```

1 //带权并查集
2 const int N = 1e5 + 10;
3 int fa[N], d[N];
4 int find(int x) {
5     int par = fa[x];
6     fa[x] = find(fa[x]);
7     d[x] += d[par];
8 }
9
10 //合并x和y, 并计算路径长度
11 void merge(int x, int y, int w) { //w代表 x->y 路径长
12     int fx = find(x), fy = find(y);
13     if (fx != fy) fa[fx] = fy, d[fx] = d[y] - d[x] + w;
14     else {
15         int dis = d[x] - d[y]; // 如果x和y已经存在关系, dis代表x到y的距离
16     }

```

```

17 }
18
19 //种类并查集
20
21 find(x) --- x的同类
22 find(x+n) ---
23 find(x+n+n....) ---
24
25 //建立虚点
26 void merge(int x, int y) { //正常的合并操作
27     int fx = find(x), fy = find(y);
28     ans[fy] += ans[fx];
29     num[fy] += num[fx];
30     fa[fx] = fy;
31 }
32
33 void move(int x) { //id[x] 代表x目前的编号
34     int fx = find(id[x]);
35     ans[fx] -= x; //原来集合中减去x的贡献
36     num[fx]--;
37     id[x] = ++n; //建立新的节点
38     ans[id[x]] = x;
39     num[id[x]] = 1;
40     fa[id[x]] = id[x];
41 }
42
43 //可撤销并查集
44 struct Undo_dsu {
45     stack<PII> st;
46     int fa[N], siz[N];
47     void init() {
48         while (st.size()) st.pop();
49         for (int i = 1; i <= n; i++) fa[i] = i, siz[i] = 1;
50     }
51     int find(int x) { return fa[x] == x ? x : find(fa[x]); }
52     bool merge(int x, int y) {
53         int fx = find(x), fy = find(y);
54         if (fx == fy) return false;
55         if (siz[fx] > siz[fy]) swap(fx, fy), swap(x, y);
56         siz[fy] += siz[fx], fa[fx] = fy;
57         st.push({fx, fy});
58         return true;
59     }
60     void undo() {
61         PII now = st.top();
62         fa[now.fi] = now.fi;
63         siz[now.se] -= siz[now.fi];
64         st.pop();
65     }
66 }dsu;

```

### 3.7 单调队列

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1000010;
4 int q[N], a[N], n, k;
5 void deque_max() {

```

```

6     int hh = 1, tt = 0;
7     for (int i = 1; i <= n; i++) {
8         while(hh <= tt && q[hh] + k <= i) hh++;
9         while(hh <= tt && a[q[tt]] < a[i]) tt--;
10        q[++tt] = i;
11        if (i >= k) cout << a[q[hh]] << " ";
12    }
13 }
14
15 void deque_min() {
16     int hh = 1, tt = 0;
17     for (int i = 1; i <= n; i++) {
18         while(hh <= tt && q[hh] + k <= i) hh++;
19         while(hh <= tt && a[q[tt]] > a[i]) tt--;
20         q[++tt] = i;
21         if (i >= k) cout << a[q[hh]] << " ";
22     }
23 }

```

### 3.8 单调栈

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 3e6 + 10;
4 int a[N], n, idx[N];
5 stack<int> stk;
6 int main() {
7     cin >> n;
8     for (int i = 1; i <= n; i++) cin >> a[i];
9     for (int i = 1; i <= n; i++) {
10        while (stk.size() && a[stk.top()] < a[i]) {
11            idx[stk.top()] = i;
12            stk.pop();
13        }
14        stk.push(i);
15    }
16    for (int i = 1; i <= n; i++) cout << idx[i] << " ";
17 }

```

### 3.9 笛卡尔树

```

1 #include <bits/stdc++.h>
2 #define ACM_LOCAL
3 using namespace std;
4 typedef long long ll;
5 typedef pair<int, int> PII;
6 const int N = 3e5 + 10, M = 1e5 + 10, INF = 0x3f3f3f3f;
7 const int MOD = 1e9 + 7;
8 int n, m, k;
9 struct Car_tree {
10     int lc[N], rc[N], stk[N], top, val[N];
11     void init(int n) {
12         for (int i = 0; i <= n; i++) lc[i] = rc[i] = 0;
13     }
14     int build(int n) {
15         int rt;
16         for (int i = 1; i <= n; i++) {

```

```

17         scanf("%d", &val[i]);
18         while (top && val[stk[top]] > val[i]) {
19             lc[i] = stk[top], top--;
20         }
21         if (top) rc[stk[top]] = i;
22         stk[++top] = i;
23     }
24     while (top) rt = stk[top--];
25     return rt;
26 }
27 }tr;

```

### 3.10 静态主席树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 2e5 + 5;
5
6  int a[N], b[N], rt[N];
7  int n, m;
8  struct Hash {
9      int b[N], tot;
10     void init() {tot = 0;}
11     void insert(int x) {b[++tot] = x;}
12     void build() {
13         sort(b+1, b+1+tot);
14         tot = unique(b+1, b+tot+1) - (b+1);
15     }
16     int pos(int x) {return lower_bound(b+1, b+tot+1, x) - b;}
17 }ha;
18 struct {
19     int t[N << 5], lc[N << 5], rc[N << 5];
20     int NodeNum = 0;
21     ll sum[N << 5];
22     int build(int l, int r) {
23         int num = ++NodeNum;
24         if (l != r) {
25             int mid = (l + r) >> 1;
26             lc[num] = build(l, mid);
27             rc[num] = build(mid + 1, r);
28         }
29         return num;
30     }
31     int update(int pre, int l, int r, int x) {
32         int num = ++NodeNum;
33         lc[num] = lc[pre], rc[num] = rc[pre], t[num] = t[pre] + 1;
34         if (l != r) {
35             int mid = (l + r) >> 1;
36             if (x <= mid) lc[num] = update(lc[pre], l, mid, x);
37             else rc[num] = update(rc[pre], mid + 1, r, x);
38         }
39         return num;
40     }
41     int Qry_Kth_Num(int u, int v, int l, int r, int k) {
42         if (l == r) return ha.b[l];
43         int mid = (l + r) >> 1, num = t[lc[v]] - t[lc[u]];
44         if (num >= k) return Qry_Kth_Num(lc[u], lc[v], l, mid, k);

```

```

45     else return Qry_Kth_Num(rc[u], rc[v], mid + 1, r, k-num);
46 }
47 ll Qry_Kth_Sum(int u, int v, int l, int r, int k) { //k大数之和
48     if (l == r) return 1ll*ha.b[l] * k;
49     int mid = (l + r) >> 1, num = t[rc[v]] - t[rc[u]];
50     if (num >= k) return Qry_Kth_Sum(rc[u], rc[v], mid+1, r, k);
51     else return sum[rc[v]] - sum[rc[u]] + Qry_Kth_Sum(lc[u], lc[v], l, mid, k-num);
52 }
53 int Binary_Search(int left, int right, int val) { //查找小于等于val的个数
54     int l = 1, r = right - left + 1;
55     while (l <= r) {
56         int mid = l + r >> 1;
57         int num = Qry_Kth_Num(rt[left - 1], rt[right], 1, ha.tot, mid);
58         if (num > val) r = mid - 1;
59         else l = mid + 1;
60     }
61     return r;
62 }
63 }hjt;

```

### 3.11 珂朵莉树

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  #include <stack>
5  #include <queue>
6  #include <map>
7  #include <vector>
8  #include <cstdio>
9  #include <cmath>
10 #include <ctime>
11 #include <bitset>
12 #include <unordered_map>
13 #include <string>
14 #include <set>
15 using namespace std;
16 #define ACM_LOCAL
17 #define IT set<node> ::iterator
18 inline int read(){
19     int s=0,w=1;
20     char ch=getchar();
21     while(ch<'0' || ch>'9'){if(ch=='-')w=-1;ch=getchar();}
22     while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
23     return s*w;
24 }
25 const int N = 3e5 + 10;
26 const int INF = 0x3f3f3f3f;
27 const int MOD = 1e9 + 7;
28 typedef pair<int, int> PII;
29 typedef long long ll;
30 typedef unsigned long long ull;
31 int ans;
32 struct node {
33     int l, r;
34     mutable ll val;
35     node(int L, int R = -1, ll V = 0):l(L), r(R), val(V) {}
36     bool operator < (const node &rhs) const {

```

```

37         return l < rhs.l;
38     }
39 };
40
41 set<node> s;
42 IT split(int pos) {
43     IT it = s.lower_bound(node(pos));
44     if (it != s.end() && it->l == pos) return it;
45     it--;
46     int L = it->l, R = it->r;
47     ll val = it->val;
48     s.erase(it);
49     s.insert(node(L, pos-1, val));
50     return s.insert(node(pos, R, val)).first;
51 }
52
53 void push_down(int l, int r, ll val) {
54     IT itr = split(r+1), itl = split(l);
55     s.erase(itl, itr);
56     s.insert(node(l, r, val));
57 }
58
59 void add(int l, int r, ll val) {
60     IT itr = split(r+1), itl = split(l);
61     for (; itl != itr; ++itl)
62         itl->val += val;
63 }
64
65 ll rank(int l, int r, int k) {
66     vector<pair<ll, int>> vp;
67     IT itr = split(r+1), itl = split(l);
68     vp.clear();
69     for (; itl != itr; ++itl) {
70         vp.push_back(pair<ll, int>(itl->val, itl->r - itl->l + 1));
71     }
72     sort(vp.begin(), vp.end());
73     for (vector<pair<ll, int>>::iterator it = vp.begin(); it != vp.end(); ++it) {
74         k -= it->second;
75         if (k <= 0)
76             return it->first;
77     }
78 }
79
80 ll ksm(ll a, ll b, ll mod) {
81     ll res = 1, ans = a % mod;
82     while (b) {
83         if (b & 1)
84             res = res * ans % mod;
85         ans = ans * ans % mod;
86         b >>= 1;
87     }
88     return res;
89 }
90
91 ll sum(int l, int r, int ex, int mod) {
92     IT itr = split(r+1), itl = split(l);
93     ll res = 0;
94     for (; itl != itr; ++itl)
95         res = (res + (ll)(itl->r - itl->l + 1) * ksm(itl->val, ll(ex), ll(mod))) % mod;

```

```
96     return res;
97 }
```

### 3.12 可持久化并查集

```
1  #include <iostream>
2  #include <cstring>
3  #include <cstdio>
4  #include <cmath>
5  #include <algorithm>
6  #include <queue>
7  #include <stack>
8  #include <string>
9  #include <set>
10 #include <map>
11 #include <bitset>
12 using namespace std;
13 #define ACM_LOCAL
14
15 const int INF = 0x3f3f3f3f;
16 const int N = 2e5 + 5;
17 typedef pair<int, int> PII;
18 typedef long long ll;
19 typedef unsigned long long ull;
20 typedef long double ld;
21
22 struct node {
23     int l, r;
24     int val;
25 }t[N * 40 * 2];
26
27 int cnt, tot, rootdep[N], rootfa[N], n, m;
28
29 void build(int l, int r, int &now) {
30     now = ++cnt;
31     if (l == r) {
32         t[now].val = ++tot;
33         return;
34     }
35     int mid = (l + r) >> 1;
36     build(l, mid, t[now].l);
37     build(mid + 1, r, t[now].r);
38 }
39
40 void update(int l, int r, int ver, int &now, int pos, int val) {
41     t[now = ++cnt] = t[ver];
42     if (l == r) {
43         t[now].val = val;
44         return;
45     }
46     int mid = (l + r) >> 1;
47     if (pos <= mid) update(l, mid, t[ver].l, t[now].l, pos, val);
48     else update(mid + 1, r, t[ver].r, t[now].r, pos, val);
49 }
50
51 int query(int l, int r, int &now, int pos) {
52     if (l == r) return t[now].val;
53     int mid = (l + r) >> 1;
```

```

54     if (pos <= mid) return query(l, mid, t[now].l, pos);
55     else return query(mid + 1, r, t[now].r, pos);
56 }
57
58 int find(int ver, int x) {
59     int fx = query(1, n, rootfa[ver], x);
60     return fx == x ? x : find(ver, fx);
61 }
62
63 void merge(int ver, int x, int y) {
64     x = find(ver-1, x);
65     y = find(ver-1, y);
66     if (x == y) {
67         rootfa[ver] = rootfa[ver-1];
68         rootdep[ver] = rootdep[ver-1];
69     }
70     else {
71         int depx = query(1, n, rootdep[ver-1], x);
72         int depy = query(1, n, rootdep[ver-1], y);
73         if (depx < depy) {
74             update(1, n, rootfa[ver-1], rootfa[ver], x, y);
75             rootdep[ver] = rootdep[ver-1];
76         }
77         else if (depx > depy) {
78             update(1, n, rootfa[ver-1], rootfa[ver], y, x);
79             rootdep[ver] = rootdep[ver-1];
80         }
81         else {
82             update(1, n, rootfa[ver-1], rootfa[ver], x, y);
83             update(1, n, rootdep[ver-1], rootdep[ver], y, depy+1);
84         }
85     }
86 }

```

### 3.13 可持久化数组

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ACM_LOCAL
4  const int N = 1e6 + 5;
5  typedef pair<int, int> PII;
6  typedef long long ll;
7
8  inline int read(){
9      int s=0,w=1;
10     char ch=getchar();
11     while(ch<'0' || ch>'9'){if(ch=='-')w=-1;ch=getchar();}
12     while(ch>='0' && ch<='9') s=s*10+ch-'0',ch=getchar();
13     return s*w;
14 }
15 int NodeNum, n, m, a[N], rt[N];
16
17 struct Node{
18     int t[N << 5], lc[N << 5], rc[N << 5];
19
20     void build(int l, int r, int &now) {
21         now = ++NodeNum;
22         if (l == r) t[now] = a[l];

```



```

23     else {
24         int mid = (l + r) >> 1;
25         build(l, mid, lc[now]);
26         build(mid + 1, r, rc[now]);
27     }
28 }
29 void update(int pre, int l, int r, int &now, int pos, int val) {
30     now = ++NodeNum;
31     lc[now] = lc[pre], rc[now] = rc[pre], t[now] = t[pre];
32     if (l == r) t[now] = val;
33     else {
34         int mid = (l + r) >> 1;
35         if (mid >= pos) update(lc[pre], l, mid, lc[now], pos, val);
36         else update(rc[pre], mid + 1, r, rc[now], pos, val);
37     }
38 }
39 void query(int pre, int l, int r, int &now, int pos) {
40     now = ++NodeNum;
41     lc[now] = lc[pre], rc[now] = rc[pre], t[now] = t[pre];
42     if (l == r) printf("%d\n", t[now]);
43     else {
44         int mid = (l + r) >> 1;
45         if (mid >= pos) query(lc[pre], l, mid, lc[now], pos);
46         else query(rc[pre], mid + 1, r, rc[now], pos);
47     }
48 }
49 }T;
50
51
52 void solve() {
53     n = read(), m = read();
54     for (int i = 1; i <= n; i++) a[i] = read();
55
56     T.build(1, n, rt[0]);
57     for (int i = 1; i <= m; i++) {
58         int opt, id, x, y;
59         id = read(), opt = read();
60         if (opt == 1) {
61             x = read(), y = read();
62             T.update(rt[id], 1, n, rt[i], x, y);
63         }
64         else {
65             x = read();
66             T.query(rt[id], 1, n, rt[i], x);
67         }
68     }
69 }
70 }
71
72 signed main() {
73     ios_base::sync_with_stdio(false);
74     cin.tie(0);
75     cout.tie(0);
76 #ifdef ACM_LOCAL
77     freopen("in.txt", "r", stdin);
78     freopen("out.txt", "w", stdout);
79 #endif
80     solve();
81     return 0;

```

82 }

### 3.14 平衡树 (pb<sub>d</sub>s)

```

1  #include "bits/stdc++.h"
2  #include <ext/pb_ds/tree_policy.hpp>
3  #include <ext/pb_ds/assoc_container.hpp>
4  using namespace std;
5  using namespace __gnu_pbds;
6  #define int long long
7  tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update> t;
8  signed main()
9  {
10     int n;
11     cin >> n;
12     while (n-->0)
13     {
14         int k,s;
15         cin >> k >> s;
16         if (k == 1)
17         {
18             cout << t.order_of_key(s)+1 << endl;           //order_of_key有几个比s小
19         }
20         else if (k == 2){
21             cout << *t.find_by_order(s-1) << endl;         //order是有几个比s小, s-1个比s小
22         } else if (k == 3)
23         {
24             if (t.find_by_order(t.order_of_key(s)-1) != t.end())
25                 cout << *t.find_by_order(t.order_of_key(s)-1) << endl;
26             else
27                 cout << -2147483647 << endl;
28         } else if (k == 4)
29             if (t.upper_bound(s) != t.end())
30                 cout << *t.upper_bound(s) << endl;         //未找到为t.end()
31             else
32                 cout << 2147483647 << endl;
33         else if (k == 5)
34             t.insert(s);
35     }
36 }
```

### 3.15 扫描线

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ACM_LOCAL
4  #define lc u << 1
5  #define rc u << 1 | 1
6  const int N = 2e5 + 5;
7  const int INF = 0x3f3f3f3f;
8  typedef unsigned long long ull;
9  typedef pair<int, int> PII;
10 typedef long long ll;
11
12 int n, m, p;
13 int v[N << 1];
14 struct node {
```

```

15     int l, r;
16     ll len;
17     int cover;
18 }t[N << 3];
19
20 struct L{
21     int x;
22     int y1, y2;
23     int state;
24     bool operator < (L &rhs) {
25         return x < rhs.x;
26     }
27 }line[N << 1];
28
29 void push_up(int u) {
30     if (t[u].cover) t[u].len = t[u].r - t[u].l;
31     else t[u].len = t[lc].len + t[rc].len;
32 }
33
34 void build(int u, int l, int r) {
35     t[u].l = v[l], t[u].r = v[r];
36     if (r - l <= 1) return;
37     int mid = (l + r) >> 1;
38     build(lc, l, mid);
39     build(rc, mid, r);
40     push_up(u);
41 }
42
43 void update(int u, int ql, int qr, int v) {
44     if (ql <= t[u].l && qr >= t[u].r) {
45         t[u].cover += v;
46         push_up(u);
47         return;
48     }
49     if (ql < t[rc].l) update(lc, ql, qr, v);
50     if (qr > t[lc].r) update(rc, ql, qr, v);
51     push_up(u);
52 }
53
54 void solve() {
55     cin >> n;
56     for (int i = 1; i <= n; i++) {
57         int x1, y1, x2, y2;
58         cin >> x1 >> y1 >> x2 >> y2;
59         v[i] = y1, v[i+n] = y2;
60         line[i] = {x1, y1, y2, 1};
61         line[i + n] = {x2, y1, y2, -1};
62     }
63     sort(v+1, v+(n<<1)+1);
64     sort(line+1, line+(n<<1)+1);
65     build(1, 1, n<<1);
66     ll ans = 0;
67     for (int i = 1; i <= n<<1; i++) {
68         ans += 1ll*t[1].len * (line[i].x - line[i-1].x);
69         update(1, line[i].y1, line[i].y2, line[i].state);
70     }
71     cout << ans << endl;
72 }
73

```

```

74 signed main() {
75     ios_base::sync_with_stdio(false);
76     cin.tie(0);
77     cout.tie(0);
78 #ifdef ACM_LOCAL
79     freopen("in.txt", "r", stdin);
80     freopen("out.txt", "w", stdout);
81 #endif
82     solve();
83     return 0;
84 }

```

### 3.16 树状数组

```

1  //-----一维树状数组
2  struct BIT {
3      int c[N];
4      int lowbit(int x) {return x&-x;}
5      void add(int x, int val) {for (int i = x; i < N; i += lowbit(i)) c[i] += val;}
6      int ask(int x) {int res = 0; for (int i = x; i > 0; i -= lowbit(i)) res += c[i];
7      return res;}
8  };
9  //-----二维树状数组
10 int c[N][N];
11
12 //单点修改, 区间查询
13 int lowbit(int x) {return x & -x;}
14 void add(int x, int y, int v) {
15     for (int i = x; i <= N; i += lowbit(i))
16         for (int j = y; j <= N; j += lowbit(j))
17             c[i][j] += v;
18 }
19
20 ll query(int x, int y) {
21     ll res = 0;
22     for (int i = x; i > 0; i -= lowbit(i))
23         for (int j = y; j > 0; j -= lowbit(j))
24             res += c[i][j];
25     return res;
26 }
27
28 //区间修改, 单点查询
29 void add_s(int x1, int y1, int x2, int y2, int v) {
30     add(x2+1, y2+1, v);
31     add(x1, y1, v);
32     add(x1, y2+1, -v);
33     add(x2+1, y1, -v);
34 }
35 //区间修改, 区间查询
36 void add(int x, int y, int v) {
37     for(int i = x; i <= n; i += lowbit(i))
38         for(int j = y; j <= m; j += lowbit(j)) {
39             t1[i][j] += v;
40             t2[i][j] += v*x;
41             t3[i][j] += v*y;
42             t4[i][j] += v*x*y;
43         }

```

```

44 }
45 void modify(int x1,int y1,int x2,int y2,int v) {
46     add(x1,y1,v);
47     add(x1,y2+1,-v);
48     add(x2+1,y1,-v);
49     add(x2+1,y2+1,v);
50 }
51 int ask(int x, int y) {
52     int res = 0;
53     for(int i = x;i;i -= lowbit(i))
54         for(int j=y;j;j -= lowbit(j))
55             res += (x+1)*(y+1)*t1[i][j]-(y+1)*t2[i][j]-(x+1)*t3[i][j]+t4[i][j];
56     return res;
57 }
58 int query(int x1,int y1,int x2,int y2) {
59     return ask(x2,y2)-ask(x2,y1-1)-ask(x1-1,y2)+ask(x1-1,y1-1);
60 }

```

### 3.17 树状数组套主席树

```

1 //单点修改动态区间k大
2 #include <bits/stdc++.h>
3 #define lowbit(i) i & -i
4 using namespace std;
5 typedef long long ll;
6 typedef pair<int, int> PII;
7 const ll INF = 1e18;
8 const int N = 1e5 + 10;
9 const int M = 1e6 + 10;
10 const int MOD = 1e9 + 7;
11 struct Segtree {
12     int ls, rs;
13     int sum;
14 }t[N * 400];
15 int tot, n, m, rt[N], a[N], cnt[2], tmp[2][20], b[N];
16 void modify(int &now, int l, int r, int pos, int val) {
17     if (!now) now = ++tot;
18     if (l == r) {
19         t[now].sum += val;
20         return;
21     }
22     int mid = (l + r) >> 1;
23     if (pos <= mid) modify(t[now].ls, l, mid, pos, val);
24     else modify(t[now].rs, mid + 1, r, pos, val);
25     t[now].sum = t[t[now].ls].sum + t[t[now].rs].sum;
26 }
27 void prepare_modify(int x, int val) {
28     for (int i = x; i <= n; i += lowbit(i)) {
29         modify(rt[i], 1, 10, a[x], val); //预处理出修改哪log棵主席树
30     }
31 }
32 int query(int l, int r, int k) {
33     if (l == r) return l;
34     int mid = (l + r) >> 1, sum = 0;
35     for (int i = 1; i <= cnt[1]; i++) sum += t[t[tmp[1][i]].ls].sum;
36     for (int i = 1; i <= cnt[0]; i++) sum -= t[t[tmp[0][i]].ls].sum;
37     if (k <= sum) {
38         for (int i = 1; i <= cnt[1]; i++) tmp[1][i] = t[tmp[1][i]].ls;

```

```

39         for (int i = 1; i <= cnt[0]; i++) tmp[0][i] = t[tmp[0][i]].ls;
40         return query(l, mid, k);
41     } else {
42         for (int i = 1; i <= cnt[1]; i++) tmp[1][i] = t[tmp[1][i]].rs;
43         for (int i = 1; i <= cnt[0]; i++) tmp[0][i] = t[tmp[0][i]].rs;
44         return query(mid + 1, r, k - sum);
45     }
46 }
47 int prepare_query(int l, int r, int k) { //处理出需要进行加减操作的log棵主席树
48     memset(tmp, 0, sizeof tmp);
49     memset(cnt, 0, sizeof cnt);
50     for (int i = r; i > 0; i -= lowbit(i)) tmp[1][++cnt[1]] = rt[i];
51     for (int i = l - 1; i > 0; i -= lowbit(i)) tmp[0][++cnt[0]] = rt[i];
52     return query(1, 10, k);
53 }
54 void solve() {
55     cin >> n >> m;
56     for (int i = 1; i <= n; i++) {
57         cin >> a[i];
58     }
59     for (int i = 1; i <= n; i++) prepare_modify(i, 1);
60     while(m--) {
61         char op; cin >> op;
62         if (op == 'Q') {
63             int l, r, k; cin >> l >> r >> k;
64             cout << prepare_query(l, r, k) << endl;
65         } else {
66             int x, y; cin >> x >> y;
67             prepare_modify(x, -1);
68             a[x] = y;
69             prepare_modify(x, 1);
70         }
71     }
72 }
73
74 //区间修改, 区间K大
75 #include <bits/stdc++.h>
76 #define lowbit(i) i & -i
77 #define Debug(x) cout << x << endl
78 #define fi first
79 #define se second
80 using namespace std;
81 typedef long long ll;
82 typedef pair<int, int> PII;
83 const ll INF = 1e18;
84 const int N = 5e4 + 10;
85 const int M = 1e6 + 10;
86 const int MOD = 1e9 + 7;
87 struct Segment {
88     int ls, rs;
89     ll sum, lazy;
90 } t[N * 400];
91 int rt[N << 2], tot, n, m, b[N], len;
92
93 void push_down(int now, int l, int r) {
94     if (t[now].lazy) {
95         int mid = (l + r) >> 1;
96         if (!t[now].ls) t[now].ls = ++tot;
97         if (!t[now].rs) t[now].rs = ++tot;

```

```

98         t[t[now].ls].lazt += t[now].lazt;
99         t[t[now].rs].lazt += t[now].lazt;
100        t[t[now].ls].sum += 1ll*(mid - l + 1) * t[now].lazt;
101        t[t[now].rs].sum += 1ll*(r - mid) * t[now].lazt;
102        t[now].lazt = 0;
103    }
104 }
105
106 void update(int &now, int ql, int qr, int l, int r) {
107     if (!now) now = ++tot;
108     if (ql <= l && qr >= r) {
109         t[now].sum += r - l + 1;
110         t[now].lazt++;
111         return;
112     }
113     push_down(now, l, r);
114     int mid = (l + r) >> 1;
115     if (ql <= mid) update(t[now].ls, ql, qr, l, mid);
116     if (qr > mid) update(t[now].rs, ql, qr, mid + 1, r);
117     t[now].sum = t[t[now].ls].sum + t[t[now].rs].sum;
118 }
119
120 void add(int u, int ql, int qr, int pos, int l, int r) {
121     update(rt[u], ql, qr, 1, n);
122     if (l == r) return;
123     int mid = (l + r) >> 1;
124     if (pos <= mid) add(u << 1, ql, qr, pos, l, mid);
125     else add(u << 1 | 1, ql, qr, pos, mid + 1, r);
126 }
127
128 ll getsum(int &now, int ql, int qr, int l, int r) {
129     if (!now) return 0;
130     if (ql <= l && qr >= r) return t[now].sum;
131     push_down(now, l, r);
132     int mid = (l + r) >> 1;
133     ll ans = 0;
134     if (ql <= mid) ans += getsum(t[now].ls, ql, qr, l, mid);
135     if (qr > mid) ans += getsum(t[now].rs, ql, qr, mid + 1, r);
136     return ans;
137 }
138
139 int query(int u, int ql, int qr, ll k, int l, int r) {
140     if (l == r) return b[l];
141     int mid = (l + r) >> 1;
142     ll num = getsum(rt[u<<1|1], ql, qr, 1, n);
143     if (k > num) return query(u<<1, ql, qr, k - num, l, mid);
144     else return query(u<<1|1, ql, qr, k, mid + 1, r);
145 }
146
147 struct Query {
148     ll op, l, r, c;
149 } q[N];
150
151 void solve() {
152     cin >> n >> m;
153     for (int i = 1; i <= m; i++) {
154         cin >> q[i].op;
155         if (q[i].op == 1) {
156             cin >> q[i].l >> q[i].r >> q[i].c;

```

```

157         b[++len] = q[i].c;
158     } else {
159         cin >> q[i].l >> q[i].r >> q[i].c;
160     }
161 }
162 sort(b + 1, b + len + 1);
163 len = unique(b + 1, b + len + 1) - b - 1;
164 for (int i = 1; i <= m; i++) {
165     if (q[i].op == 1) {
166         q[i].c = lower_bound(b + 1, b + len + 1, q[i].c) - b;
167         add(1, q[i].l, q[i].r, q[i].c, 1, len);
168     } else {
169         printf("%d\n", query(1, q[i].l, q[i].r, q[i].c, 1, len));
170     }
171 }
172 }

```

### 3.18 替罪羊树 (平衡树)

```

1  #include<bits/stdc++.h>
2  #define lc (son[p][0])
3  #define rc (son[p][1])
4  using namespace std;
5  //#define ACM_LOCAL
6  const int N = 4e5 + 10;
7  struct SGT {
8      const double alpha = 0.75;
9      int st[N], top, tot, rt, son[N][2], fa[N], val[N], siz[N], all[N];
10     bool in[N];
11
12     inline int get() { return top ? st[top--] : ++tot; }
13
14     void del(int t) { st[++top] = t; }
15
16     inline void pushup(int p) { siz[p] = siz[lc] + siz[rc] + in[p], all[p] = all[lc] +
17         all[rc] + 1; }
18
19     inline bool check(int p) { return (all[lc] >= all[p] * alpha) || (all[rc] >= all[p]
20         * alpha); }
21
22     inline int newnode(int v = 0, int pa = 0) {
23         int p = get();
24         lc = rc = 0, val[p] = v, siz[p] = all[p] = 1, in[p] = 1, fa[p] = pa;
25         return p;
26     }
27
28     inline void getpos(int p, vector<int> &v) {
29         if (!p) return;
30         getpos(lc, v);
31         if (in[p]) v.push_back(p);
32         else del(p);
33         getpos(rc, v);
34     }
35
36     inline int build(int l, int r, vector<int> v) {
37         if (l >= r) return 0;
38         int mid = l + r >> 1, p = v[mid];
39         lc = build(l, mid, v), rc = build(mid + 1, r, v), fa[lc] = fa[rc] = p;

```



```

38     return pushup(p), p;
39 }
40
41 inline void rebuild(int &p) {
42     static vector<int> v;
43     v.clear();
44     int pa = fa[p];
45     getpos(p, v);
46     fa[(p = build(0, v.size(), v))] = pa;
47 }
48
49 inline int rank(int v) {
50     int p = rt, ret = 1;
51     while (p) {
52         if (v <= val[p]) p = lc;
53         else ret += siz[lc] + in[p], p = rc;
54     }
55     return ret;
56 }
57
58 inline int kth(int k) {
59     int p = rt;
60     while (p) {
61         if (siz[lc] + 1 == k && in[p]) break;
62         if (siz[lc] >= k) p = lc;
63         else k -= siz[lc] + in[p], p = rc;
64     }
65     return val[p];
66 }
67
68 inline int insert(int &p, int v) {
69     if (!p) return p = newnode(v), 0;
70     ++siz[p], ++all[p];
71     int ret;
72     ret = insert(v <= val[p] ? lc : rc, v), pushup(p);
73     if (check(p)) ret = p;
74     return ret;
75 }
76
77 inline void insert(int v) {
78     int p = insert(rt, v);
79     if (!p) return;
80     if (p == rt) rebuild(rt);
81     else {
82         int f = fa[p];
83         if (p == son[f][0]) rebuild(son[f][0]);
84         else rebuild(son[f][1]);
85     }
86 }
87
88 void erase(int p, int k) {
89     --siz[p];
90     if (in[p] && k == siz[lc] + in[p]) {
91         in[p] = 0;
92         return;
93     }
94     if (k <= siz[lc]) erase(lc, k);
95     else erase(rc, k - siz[lc] - in[p]);
96 }

```

```

97
98     void erase(int v) {
99         erase(rt, rank(v));
100         if (siz[rt] < alpha * all[rt]) rebuild(rt);
101     }
102
103     int pre(int v) { return kth(rank(v) - 1); }
104
105     int suf(int v) { return kth(rank(v) + 1); }
106 } tzy;

```

### 3.19 线段树 (区间最长连续段)

```

1  int rt[N], cnt; //主席树版本
2  struct Tree {
3      int ls, rs;
4      int lsum, rsum, tsum;
5  } hjt[N*50];
6  void push_up(int now, int l, int r) {
7      int mid = (l + r) >> 1;
8      hjt[now].lsum = hjt[hjt[now].ls].lsum;
9      if (hjt[hjt[now].ls].lsum == mid - l + 1) hjt[now].lsum += hjt[hjt[now].rs].lsum;
10     hjt[now].rsum = hjt[hjt[now].rs].rsum;
11     if (hjt[hjt[now].rs].rsum == r - mid) hjt[now].rsum += hjt[hjt[now].ls].rsum;
12     hjt[now].tsum = max(hjt[hjt[now].ls].tsum, hjt[hjt[now].rs].tsum);
13     hjt[now].tsum = max(hjt[now].tsum, hjt[hjt[now].ls].rsum + hjt[hjt[now].rs].lsum);
14 }
15 void modify(int &now, int pre, int l, int r, int pos, int val) {
16     now = ++cnt;
17     hjt[now] = hjt[pre];
18     if (l == r) {
19         hjt[now].lsum = hjt[now].rsum = hjt[now].tsum = 1;
20         return;
21     }
22     int mid = (l + r) >> 1;
23     if (pos <= mid) modify(hjt[now].ls, hjt[pre].ls, l, mid, pos, val);
24     else modify(hjt[now].rs, hjt[pre].rs, mid+1, r, pos, val);
25     push_up(now, l, r);
26 }
27 int query(int now, int l, int r, int ql, int qr) {
28     if (ql <= l && qr >= r) return hjt[now].tsum;
29     int mid = (l + r) >> 1;
30     int ans = 0;
31     if (ql <= mid) ans = max(ans, query(hjt[now].ls, l, mid, ql, qr));
32     if (qr > mid) ans = max(ans, query(hjt[now].rs, mid+1, r, ql, qr));
33     ans = max(ans, min(mid-ql+1, hjt[hjt[now].ls].rsum) + min(qr-mid, hjt[hjt[now].rs].lsum));
34     return ans;
35 }

```

### 3.20 线段树 (最长子序和)

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N=5e5+1000;
4  int n,m,x,y;
5  int a[N];

```

```

6  struct Node {
7      int l,r;
8      int sum;
9      int lmax;
10     int rmax;
11     int tmax;
12 }tr[N<<2];
13
14 void push_up(Node &u,Node &l,Node &r) {
15     u.sum=l.sum+r.sum;
16     u.lmax=max(l.lmax,l.sum+r.lmax);
17     u.rmax=max(r.rmax,r.sum+l.rmax);
18     u.tmax=max(max(l.tmax,r.tmax),l.rmax+r.lmax);
19 }
20
21 void push_up(int u) {
22     push_up(tr[u],tr[u<<1],tr[u<<1|1]);
23 }
24
25 void build(int u,int l,int r) {
26     tr[u].l = l, tr[u].r = r;
27     if(l==r) {
28         tr[u]={l,r,a[r],a[r],a[r],a[r]};
29         return ;
30     }
31     int mid = (l+r)>>1;
32
33     build(u<<1, l, mid);
34     build(u<<1|1, mid+1, r);
35     push_up(u);
36 }
37
38 void update(int u,int x,int v) {
39     if(tr[u].l==tr[u].r) {
40         tr[u]={x,x,v,v,v,v};
41         return ;
42     }
43     int mid = (tr[u].l + tr[u].r)>>1;
44     if(x<=mid) update(u<<1, x, v);
45     else update(u<<1|1, x, v);
46     push_up(u);
47 }
48 Node query(int u,int ql,int qr) {
49     if(ql<=tr[u].l&&qr>=tr[u].r) return tr[u];
50     int mid=tr[u].l+tr[u].r>>1;
51     if(qr<=mid) return query(u<<1,ql,qr);
52     else if(ql>mid) return query(u<<1|1,ql,qr);
53     else
54     {
55         auto left=query(u<<1,ql,qr);
56         auto right=query(u<<1|1,ql,qr);
57         Node res;
58         push_up(res,left,right);
59         return res;
60     }
61 }

```

### 3.21 线段树

```

1 namespace Seg {
2 #define ls (u<<1)
3 #define rs (u<<1|1)
4 #define mid ((l+r)>>1)
5 ll sum[N<<2], tag[N<<2];
6 void push_up(int u) {
7     sum[u] = sum[u<<1] + sum[u<<1|1];
8 }
9 void push_down(int u, int l, int r) {
10     if (tag[u]) {
11         tag[u<<1] += tag[u];
12         tag[u<<1|1] += tag[u];
13         sum[u<<1] += 1ll*(mid-l+1)*tag[u];
14         sum[u<<1|1] += 1ll*(r-mid)*tag[u];
15         tag[u] = 0;
16     }
17 }
18 void modify(int u, int ql, int qr, int l, int r, int val) {
19     if (ql <= l && qr >= r) {
20         sum[u] += 1ll*(r-l+1)*val;
21         tag[u] += val;
22         return;
23     }
24     push_down(u, l, r);
25     if (ql <= mid) modify(ls, ql, qr, l, mid, val);
26     if (qr > mid) modify(rs, ql, qr, mid+1, r, val);
27     push_up(u);
28 }
29
30 #undef mid
31 #undef ls
32 #undef rs
33 }
34
35 一些技巧
36
37 1)找到最左端满足条件的位置:
38
39 1. 二分+线段树 复杂度是nlogn^2
40
41 2. 直接在线段树上找 复杂度是nlogn
42
43 int query(int u, int L, int R, int val) {
44     if (t[u].mi > val) return 2e9;
45     if (t[u].l == t[u].r) return t[u].l;
46     int mid = t[u].l + t[u].r >> 1;
47     if (L <= t[u].l && R >= t[u].r) {
48         if (t[u<<1].mi <= val) return query(u<<1, t[u].l, mid, val);
49         else return query(u<<1|1, mid + 1, t[u].r, val);
50     }
51     else {
52         if (R <= mid) return query(u<<1, L, R, val);
53         else if (L > mid) return query(u<<1|1, L, R, val);
54         else return min(query(u<<1, L, mid, val), query(u<<1|1, mid + 1, R, val));
55     }
56 }
57

```

58 2) 势能线段树(均摊时间复杂度)

59

60 1. 区间与, 单点修改, 区间max

61

```

62 struct node {
63     int l, r;
64     int Or, And, Max, tag;
65 } t[N<<2];
66 int a[N];
67 void push_up(int u) {
68     t[u].Or = t[u<<1].Or | t[u<<1|1].Or;
69     t[u].And = t[u<<1].And & t[u<<1|1].And;
70     t[u].Max = max(t[u<<1].Max, t[u<<1|1].Max);
71 }
72 void push_down(int u) {
73     if (t[u].tag != 0) {
74         t[u<<1].tag += t[u].tag;
75         t[u<<1|1].tag += t[u].tag;
76         t[u<<1].Max += t[u].tag;
77         t[u<<1|1].Max += t[u].tag;
78         t[u<<1].And += t[u].tag;
79         t[u<<1|1].And += t[u].tag;
80         t[u<<1].Or += t[u].tag;
81         t[u<<1|1].Or += t[u].tag;
82         t[u].tag = 0;
83     }
84 }
85 void build(int u, int l, int r) {
86     t[u].l = l, t[u].r = r;
87     if (l == r) {
88         t[u].Or = t[u].And = t[u].Max = a[l];
89         return;
90     }
91     int mid = (l + r) >> 1;
92     build(u<<1, l, mid);
93     build(u<<1|1, mid+1, r);
94     push_up(u);
95 }
96 void modify(int u, int ql, int qr, int val) {
97     if ((t[u].Or & val) == t[u].Or) return;
98     if (ql <= t[u].l && qr >= t[u].r && (t[u].Or & val) - t[u].Or == (t[u].And & val) - t[u].And) {
99         int tmp = (t[u].Or & val) - t[u].Or;
100         t[u].And += tmp;
101         t[u].Or += tmp;
102         t[u].Max += tmp;
103         t[u].tag += tmp;
104         return;
105     }
106     push_down(u);
107     int mid = (t[u].l + t[u].r) >> 1;
108     if (ql <= mid) modify(u<<1, ql, qr, val);
109     if (qr > mid) modify(u<<1|1, ql, qr, val);
110     push_up(u);
111 }
112 void update(int u, int pos, int val) {
113     if (t[u].l == t[u].r) {
114         t[u].Max = t[u].Or = t[u].And = val;
115         return;

```

```

116     }
117     push_down(u);
118     int mid = (t[u].l + t[u].r) >> 1;
119     if (pos <= mid) update(u<<1, pos, val);
120     else update(u<<1|1, pos, val);
121     push_up(u);
122 }
123 int query(int u, int ql, int qr) {
124     if (ql <= t[u].l && qr >= t[u].r) return t[u].Max;
125     push_down(u);
126     int mid = (t[u].l + t[u].r) >> 1;
127     int ans = 0;
128     if (ql <= mid) ans = max(ans, query(u<<1, ql, qr));
129     if (qr > mid) ans = max(ans, query(u<<1|1, ql, qr));
130     return ans;
131 }
132
133 2. 区间开根号, 区间加, 区间和
134
135 势能=max-min
136
137 3. 区间质因子问题

```

### 3.22 线段树 +dfs 序

```

1  int in[N], out[N];
2  vector<int> g[N];
3  void dfs(int u) {
4      in[u] = ++tot;
5      for (auto i : g[u]) {
6          dfs(i);
7      }
8      out[u] = tot;
9  }
10
11 修改子树内信息[in[u], out[u]]

```

### 3.23 线段树标记永久化

```

1  typedef long long ll;
2  const int N = 1e5 + 5, M = 5e5 + 5, INF = 0x3f3f3f3f;
3  const int MOD = 1e9 + 7;
4
5  int n, q, a[N];
6
7  struct Segment_tree {
8      #define ls u << 1
9      #define rs u << 1 | 1
10     int L[N << 2], R[N << 2];
11     ll sum[N << 2], lazy[N << 2];
12
13     void push_up(int u) {
14         sum[u] = sum[ls] + sum[rs] + (R[u] - L[u] + 1) * lazy[u];
15     }
16
17     void build(int u, int l, int r) {
18         L[u] = l, R[u] = r;

```

```

19     if (l == r) {
20         sum[u] = a[l];
21         return;
22     }
23     int m = (l + r) >> 1;
24     build(ls, l, m);
25     build(rs, m + 1, r);
26     push_up(u);
27 }
28
29 void update(int u, int ql, int qr, ll k) {
30     if (ql <= L[u] && R[u] <= qr) {
31         sum[u] += (R[u] - L[u] + 1) * k;
32         lazy[u] += k;
33         return;
34     }
35     int m = (L[u] + R[u]) >> 1;
36     if (ql <= m) update(ls, ql, qr, k);
37     if (qr > m) update(rs, ql, qr, k);
38     push_up(u);
39 }
40
41 ll query(int u, int ql, int qr, ll tg) {
42     if (ql <= L[u] && R[u] <= qr) return sum[u] + (R[u] - L[u] + 1) * tg;
43     int m = (L[u] + R[u]) >> 1;
44     ll ans = 0;
45     if (ql <= m) ans += query(ls, ql, qr, tg + lazy[u]);
46     if (qr > m) ans += query(rs, ql, qr, tg + lazy[u]);
47     return ans;
48 }
49
50 #undef ls
51 #undef rs
52 } tr;

```

### 3.24 线段树合并 + 动态开点

```

1  #pragma GCC optimize (2)
2  #include <bits/stdc++.h>
3  #define ACM_LOCAL
4  using namespace std;
5  const int N = 1e5 + 5;
6  const int M = 6e6 + 5;
7  struct edge {
8      int to, next;
9  } e[2000010];
10
11 struct tree {
12     int l, r;
13     int maxx, id;
14 } t[M];
15
16 struct query {
17     int x, y, z;
18 } q[N];
19
20 int tmp[N], n, m, h[N], cnt, son[N], root[N], siz[N], top[N], fa[N], tot, d[N], idx,
    all, ans[N], k;

```

```

21
22 void add(int u, int v) {
23     e[cnt].to = v;
24     e[cnt].next = h[u];
25     h[u] = cnt++;
26 }
27
28 void dfs1(int u) {
29     son[u] = -1;
30     siz[u] = 1;
31     for (int i = h[u]; ~i; i = e[i].next) {
32         int v = e[i].to;
33         if (!d[v]) {
34             d[v] = d[u] + 1;
35             fa[v] = u;
36             dfs1(v);
37             siz[u] += siz[v];
38             if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
39         }
40     }
41 }
42
43 void dfs2(int u, int t) {
44     top[u] = t;
45     if (son[u] == -1) return;
46     dfs2(son[u], t);
47     for (int i = h[u]; ~i; i = e[i].next) {
48         int v = e[i].to;
49         if (v != son[u] && v != fa[u]) dfs2(v, v);
50     }
51 }
52
53 int lca(int u, int v) {
54     while (top[u] != top[v]) {
55         if (d[top[u]] > d[top[v]])
56             u = fa[top[u]];
57         else
58             v = fa[top[v]];
59     }
60     return d[u] > d[v] ? v : u;
61 }
62
63 void push_up(int u) {
64     int ls = t[u].l, rs = t[u].r;
65     if (t[ls].maxx >= t[rs].maxx) t[u].maxx = t[ls].maxx, t[u].id = t[ls].id;
66     else t[u].maxx = t[rs].maxx, t[u].id = t[rs].id;
67 }
68
69 void update(int &now, int l, int r, int pos, int val) {
70     if (!now) now = ++idx;
71     if (l == r && l == pos){
72         t[now].maxx += val;
73         t[now].id = l;
74         return;
75     }
76     int mid = (l + r) >> 1;
77     if (pos <= mid) update(t[now].l, l, mid, pos, val);
78     else update(t[now].r, mid + 1, r, pos, val);
79     push_up(now);

```



```

80 }
81
82 void merge(int &x, int &y, int l, int r) {
83     if (!x) return;
84     if (!y) { y = x; return; }
85     if (l == r) {
86         t[y].maxx += t[x].maxx;
87         return;
88     }
89     int mid = (l + r) >> 1;
90     merge(t[x].l, t[y].l, l, mid);
91     merge(t[x].r, t[y].r, mid + 1, r);
92     push_up(y);
93 }
94
95 void dfs_merge(int u, int fa) {
96     for (int i = h[u]; ~i; i = e[i].next) {
97         int v = e[i].to;
98         if (v == fa) continue;
99         dfs_merge(v, u);
100         merge(root[v], root[u], 1, k);
101     }
102     if (t[root[u]].maxx == 0) ans[u] = 0;
103     else {
104         ans[u] = tmp[t[root[u]].id];
105     }
106 }
107 void solve() {
108     memset(h, -1, sizeof h);
109     scanf("%d %d", &n, &m);
110     for (int i = 1; i <= n - 1; i++) {
111         int x, y;
112         scanf("%d %d", &x, &y);
113         add(x, y), add(y, x);
114     }
115     for (int i = 1; i <= m; i++) {
116         scanf("%d %d %d", &q[i].x, &q[i].y, &q[i].z);
117         tmp[++k] = q[i].z;
118     }
119     d[1] = 1;
120     dfs1(1);
121     dfs2(1, 1);
122     sort(tmp + 1, tmp + 1 + k);
123     k = unique(tmp + 1, tmp + 1 + k) - (tmp + 1);
124     for (int i = 1; i <= m; i++) {
125         q[i].z = lower_bound(tmp + 1, tmp + 1 + k, q[i].z) - tmp;
126         update(root[q[i].x], 1, k, q[i].z, 1);
127         update(root[q[i].y], 1, k, q[i].z, 1);
128         update(root[lca(q[i].x, q[i].y)], 1, k, q[i].z, -1);
129         if (fa[lca(q[i].x, q[i].y)]) update(root[fa[lca(q[i].x, q[i].y)]], 1, k, q[i].z
130 , -1);
131     }
132     dfs_merge(1, 0);
133     for (int i = 1; i <= n; i++) printf("%d\n", ans[i]);
134 }
135 signed main() {
136     ios_base::sync_with_stdio(false);
137     cin.tie(nullptr);

```

```

138     cout.tie(nullptr);
139 #ifdef ACM_LOCAL
140     freopen("in.txt", "r", stdin);
141     //freopen("out.txt", "w", stdout);
142     signed test_index_for_debug = 1;
143     char acm_local_for_debug = 0;
144     do {
145         if (acm_local_for_debug == '$') exit(0);
146         if (test_index_for_debug > 20)
147             throw runtime_error("Check the stdin!!!");
148         auto start_clock_for_debug = clock();
149         solve();
150         auto end_clock_for_debug = clock();
151         cout << "Test " << test_index_for_debug << " successful" << endl;
152         cerr << "Test " << test_index_for_debug++ << " Run Time: "
153             << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC <<
154             "s" << endl;
155         cout << "-----" << endl;
156     } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
157 #else
158     solve();
159 #endif
160     return 0;
161 }

```

### 3.25 Hash 表

```

1 //手写哈希表
2 struct HashSet {
3     const int mod = 1000009;
4     struct node {
5         int k, v, nex;
6     } buf[N];
7     int h[N], tot;
8     void ins(int x) {
9         int pos = x % mod;
10        for (int i=h[pos]; i; i=buf[i].nex) {
11            if (buf[i].k == x) { buf[i].v++; return ; }
12        }
13        buf[++tot] = (node){x, 1, h[pos]};
14        h[pos] = tot;
15    }
16    int find(int x) {
17        int pos = x % mod;
18        for (int i=h[pos]; i; i=buf[i].nex) {
19            if (buf[i].k == x) return buf[i].v;
20        }
21        return 0;
22    }
23 } H;
24
25 //支持单点修改的hash
26 const int N = 2e6 + 10, M = 1e6 + 10;
27 const int mod = 1e9 + 7;
28 char s[N];
29 int n, m;
30 ll f[N];
31 struct BIT {

```

```

32     ll c[N];
33     int lowbit(int x) {return x&-x;}
34     void add(int x, ll val) {
35         for (int i = x; i <= n; i += lowbit(i)) c[i] = (c[i] + val) % mod;
36     }
37     ll qry(int x) {
38         ll res = 0;
39         for (int i = x; i > 0; i -= lowbit(i)) res = (res + c[i]) % mod;
40         return res;
41     }
42     ll query(int l, int r) {
43         return (qry(r) - qry(l-1) + mod) % mod;
44     }
45     bool comp(int l1, int r1, int l2, int r2) {
46         return query(l1, r1)*f[l2-l1] % mod == query(l2, r2);
47     }
48     bool check(int l1, int r1, int l2, int r2) {
49         if (l1 == r1) return true;
50         int mid1 = (l1 + r1) >> 1;
51         int mid2 = (l2 + r2) >> 1;
52         bool f1 = comp(l1, mid1, l2, mid2);
53         bool f2 = comp(mid1+1, r1, mid2+1, r2);
54         if (!f1 && !f2) return false;
55         if (!f1) return check(l1, mid1, l2, mid2);
56         else return check(mid1+1, r1, mid2+1, r2);
57     }
58 }bit;
59 void init() {
60     f[0] = 1;
61     for (int i = 1; i <= n; i++) f[i] = f[i-1] * 2333 % mod;
62     for (int i = 1; i <= n; i++) bit.add(i, s[i] * f[i] % mod);
63 }
64
65 修改操作: 将x位置的改为c
66 bit.add(x, (c-s[x]+mod)%mod*f[x]%mod);
67 s[x] = c;

```



```

13     while(b) {
14         if(b & 1)
15             ans = ans * a % P;
16         a = a * a % P;
17         b >>= 1;
18     }
19     return ans % P;
20 }
21
22 ll ex_gcd(ll a, ll b, ll &x, ll &y) {
23     ll res, t;
24     if(!b) {
25         x = 1;
26         y = 0;
27         return a;
28     }
29     res = ex_gcd(b, a % b, x, y);
30     t = x;
31     x = y;
32     y = t - (a / b) * y;
33     return res;
34 }
35
36 ll INV(ll a, ll mod) {
37     ll x, y;
38     ll d = ex_gcd(a, mod, x, y);
39     return d ? (x % mod + mod) % mod : -1;
40 }
41
42 ll fac(ll n, ll P, ll pk) { // 阶乘除去质因子后模质数幂 (n / p^a) % pk
43     if(!n) return 1;
44     ll ans = 1;
45     for(int i = 1; i < pk; i++) { // 第三部分: n!与p互质的乘积
46         if(i % P)
47             ans = ans * i % pk;
48     }
49     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
50     for(int i = 1; i <= n % pk; i++) { // 第四部分: 循环过后n!剩下的部分
51         if(i % P) ans = ans * i % pk;
52     }
53     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)!
54 }
55
56 ll C(ll m, ll n, ll P, ll pk) { // 组合数模质数幂
57     if(n < 0 || m < 0 || n > m) return 0;
58     ll f1 = fac(m, P, pk), f2 = fac(n, P, pk), f3 = fac(m - n, P, pk), tmp = 0; // tmp
59     = pk1 - pk2 - pk3
60     for(ll i = m; i ; i /= P) tmp += i / P;
61     for(ll i = n; i ; i /= P) tmp -= i / P;
62     for(ll i = m - n; i ; i /= P) tmp -= i / P;
63     return f1 * INV(f2, pk) % pk * INV(f3, pk) * quick_pow(P, tmp, pk) % pk;
64 }
65 ll p[N], a[N];
66 int cnt;
67
68 ll CRT() {
69     ll M = 1, ans = 0;
70     for(int i = 1; i <= cnt; i++) M *= p[i];

```

```

71     for(int i = 1; i <= cnt; i++) {
72         ll m = M / p[i];
73         ans = (ans + a[i] * m % M * INV(m, p[i]) % M) % M;
74     }
75     return (ans % M + M) % M;
76 }
77
78 ll EX_Lucas(ll m, ll n, ll P) {
79     for(int i = 2; i * i <= P; i++) {
80         if(P % i == 0) {
81             ll tmp = 1;
82             while(P % i == 0) {
83                 tmp *= i;
84                 P /= i;
85             }
86             p[++cnt] = tmp;
87             a[cnt] = C(m, n, i, tmp);
88         }
89     }
90     if(P > 1) {
91         p[++cnt] = P;
92         a[cnt] = C(m, n, P, P);
93     }
94     return CRT();
95 }
96 int main()
97 {
98     ll m, n, P;
99     cin >> m >> n >> P;
100     cnt = 0;
101     cout << EX_Lucas(m, n, P) << endl;
102 }

```

### 4.3 Lucas

```

1  // p一定为质数
2
3  #include <iostream>
4  using namespace std;
5
6  typedef long long ll;
7
8  const int N = 1e7 + 10;
9
10 ll p; // C(n,m) % p
11
12 ll f[N], inv[N], invF[N];
13
14 void Init()
15 {
16     f[0] = f[1] = inv[0] = inv[1] = invF[0] = invF[1] = 1;
17     for(int i = 2; i <= p; i++)
18     {
19         f[i] = f[i - 1] * i % p;
20         inv[i] = (p - (p / i)) * inv[p % i] % p;
21         invF[i] = invF[i - 1] * inv[i] % p;
22     }
23 }

```

```

24
25 ll C(ll m, ll n)
26 {
27     if(m < 0 || n < 0 || n > m)
28         return 0;
29     ll ans = f[m];
30     ans = ans * invF[n] % p;
31     ans = ans * invF[m - n] % p;
32     return ans;
33 }
34
35 ll Lucas(ll m, ll n)
36 {
37     if(n == 0)
38         return 1;
39     return Lucas(m / p, n / p) * C(m % p, n % p) % p; // 进制
40 }
41
42 int main()
43 {
44     ll m, n;
45     cin >> m >> n >> p;
46     Init();
47     cout << Lucas(m, n) << endl;
48 }

```

#### 4.4 Miller<sub>Rabin</sub>

```

1 // 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.
2
3 #include <bits/stdc++.h>
4 using namespace std;
5 typedef long long ll;
6 const int N = 1e5 + 7;
7 const int times = 10;
8
9 ll ksc(ll a, ll b, ll mod) {
10     ll ans = 0;
11     while(b > 0) {
12         if(b & 1) {
13             ans = (ans + a) % mod;
14         }
15         a = (a << 1) % mod;
16         b >>= 1;
17     }
18     return ans;
19 }
20
21 ll quick_pow(ll a, ll b, ll mod) {
22     ll ans = 1, base = a;
23     while(b != 0) {
24         if(b & 1) {
25             ans = ans * base % mod;
26         }
27         base = base * base % mod;
28         b >>= 1;
29     }
30     return ans;

```

```

31 }
32
33 bool Miller_Pabin(ll n)//Miller测试的主体结构
34 {
35     if(n < 2) return false;
36     if(n == 2) return true;
37     if(n & 1 == 0) return false;//对于偶数的优化
38     ll k = 0, u = n - 1;//p为Miller测试的k, u为Miller测试的m
39
40     while(u & 1 == 0){ // 把x拆成u*2^k
41         u >>= 1;
42         k++;
43     }
44     srand(time(NULL));
45
46     ll x, pre; // pre为上次探测的x的值
47
48     for(int i = 1; i <= times; i++) {
49         x = rand() % (n - 1) + 1;
50         x = quick_pow(x, u, n); // 先求出x^u(mod n)
51         pre = x;
52         for(int j = 1; j <= k; j++) {
53             x = ksc(x, x, n);
54             if(x == 1 && pre != 1 && pre != n - 1)
55                 return false;
56             pre = x;
57         }
58         if(x != -1)
59             return false;
60     }
61     return true;
62 }
63
64 int main() {
65     ll n; cin >> n;
66     cout << (Miller_Pabin(n) ? "Prime" : "Not a Prime") << endl;
67 }

```

#### 4.5 Min25 筛

```

1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long ll;
6
7  const int N = 1e5 + 10;
8
9
10 namespace Min25 {
11     int prime[N], id1[N], id2[N], flag[N], ncnt, m;
12
13     ll g[N], sum[N], a[N], T, n;
14
15     inline int ID(ll x) {
16         return x <= T ? id1[x] : id2[n / x];
17     }
18

```



```

19 inline ll calc(ll x) {
20     return x * (x + 1) / 2 - 1;
21 }
22
23 inline ll f(ll x) {
24     return x;
25 }
26
27 inline void init() {
28     ncnt = 0, m = 0;
29     T = sqrt(n + 0.5);
30     for (int i = 2; i <= T; i++) {
31         if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
32         for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {
33             flag[i * prime[j]] = 1;
34             if (i % prime[j] == 0) break;
35         }
36     }
37     for (ll l = 1; l <= n; l = n / (n / l) + 1) {
38         a[++m] = n / l;
39         if (a[m] <= T) id1[a[m]] = m; else id2[n / a[m]] = m;
40         g[m] = calc(a[m]);
41     }
42     for (int i = 1; i <= ncnt; i++)
43         for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j]; j++)
44             g[j] = g[j] - (ll)prime[i] * (g[id1[a[j] / prime[i]]] - sum[i - 1]);
45 }
46
47 inline ll Solve(ll x) {
48     if (x <= 1) return x;
49     return n = x, init(), g[id2(n)];
50 }
51
52 }

```

## 4.6 杜教筛

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 typedef long long ll;
6 const int N = 1e6 + 10;
7
8 unordered_map<int, ll> smu, sphi;
9 bool isPrime[N];
10 int prime[N], num;
11 ll mu[N], phi[N];
12
13 void makeMobiusAndEuler(int siz) {
14     mu[1] = phi[1] = 1;
15     for (int i = 2; i <= siz; i++) {
16         if (!isPrime[i]) prime[++num] = i, mu[i] = -1, phi[i] = i - 1;
17         for (int j = 1; j <= num && i * prime[j] <= siz; j++) {
18             isPrime[i * prime[j]] = 1;
19             if (i % prime[j] == 0) {
20                 mu[i * prime[j]] = 0;
21                 phi[i * prime[j]] = phi[i] * prime[j];

```

```

22         break;
23     }
24     else {
25         phi[i * prime[j]] = phi[prime[j]] * phi[i];
26         mu[i * prime[j]] = -mu[i];
27     }
28 }
29 }
30 for (int i = 1; i <= siz; i++) mu[i] += mu[i - 1], phi[i] += phi[i - 1];
31 }
32
33 ll getSmu(int n) {
34     if (n < N) return mu[n];
35     if (smu[n]) return smu[n];
36     ll res = 1;
37     for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
38         r = n / (n / l);
39         res -= 1ll * (r - l + 1) * getSmu(n / l);
40     }
41     return smu[n] = res;
42 }
43
44 ll getSphi(int n) {
45     if (n < N) return phi[n];
46     if (sphi[n]) return sphi[n];
47     ll res = 1ll * n * (n + 1) / 2;
48     for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
49         r = n / (n / l);
50         res -= 1ll * (r - l + 1) * getSphi(n / l);
51     }
52     return sphi[n] = res;
53 }

```

## 4.7 杜教筛

```

1 //筛大范围前缀和
2 ll prime[N], mu[N], k;
3 ll phi[N];
4 bool is_prime[N];
5 inline void init(int n) {
6     memset(is_prime, true, sizeof is_prime);
7     mu[1] = 1; phi[1] = 1;
8     for (re int i = 2; i < n; ++i) {
9         if (is_prime[i]) prime[++k] = i, mu[i] = -1, phi[i] = i-1;
10        for (re int j = 1; j <= k && i * prime[j] < n; ++j) {
11            is_prime[i * prime[j]] = false;
12            if (i % prime[j] == 0) {
13                phi[i * prime[j]] = phi[i] * prime[j];
14                break;
15            } else {
16                mu[i * prime[j]] = -mu[i];
17                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
18            }
19        }
20    }
21    for (re int i = 1; i < n; ++i) mu[i] += mu[i-1], phi[i] += phi[i-1];
22 }
23 unordered_map<ll, ll> sum_mu, sum_phi;

```

```

24 inline ll GetSum_mu(ll n) {
25     if (n <= 3e7) return mu[n];
26     if (sum_mu[n]) return sum_mu[n];
27     ll ans = 1;
28     for (re ll l = 2, r; l <= n; l = r + 1) {
29         r = min(n, n / (n / l));
30         ans -= (r - l + 1) * GetSum_mu(n / l);
31     }
32     return sum_mu[n] = ans;
33 }
34
35 inline ll GetSum_phi(ll n) {
36     if (n <= 3e7) return phi[n];
37     if (sum_phi[n]) return sum_phi[n];
38     ll ans = n * (n + 1) / 2;
39     for (re ll l = 2, r; l <= n; l = r + 1) {
40         r = min(n, n / (n / l));
41         ans -= (r - l + 1) * GetSum_phi(n / l);
42     }
43     return sum_phi[n] = ans;
44 }

```

#### 4.8 二次剩余

```

1  #include <iostream>
2  #include <ctime>
3
4  using namespace std;
5
6  typedef long long ll;
7
8
9  typedef struct{
10     ll x, y; // 把求出来的w作为虚部, 则为a + bw
11 }num;
12
13 ll quick_pow(ll a, ll b, ll p) {
14     ll ans = 1;
15     while(b) {
16         if(b & 1) ans = ans * a % p;
17         a = a * a % p;
18         b >>= 1;
19     }
20     return ans % p;
21 }
22
23
24 num num_mul(num a, num b, ll w, ll p) { // 复数乘法
25     num ans = {0, 0};
26     ans.x = (a.x * b.x % p + a.y * b.y % p * w % p + p) % p;
27     ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
28     return ans;
29 }
30
31 ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
32     num ans = {1, 0};
33     while(b) {
34         if(b & 1)

```

```

35         ans = num_mul(ans, a, w, p);
36         a = num_mul(a, a, w, p);
37         b >>= 1;
38     }
39     return ans.x % p;
40 }
41
42 ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
43     return quick_pow(a, (p - 1) >> 1, p);
44 }
45
46 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
47     n %= p;
48     if(n == 0)
49         return 0;
50     if(p == 2)
51         return 1;
52     if(legendre(n, p) + 1 == p) // 二次非剩余
53         return -1;
54
55     ll a, w;
56
57     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
58         a = rand() % p;
59         w = ((a * a - n) % p + p) % p;
60         if(legendre(w, p) + 1 == p) // 找到w, 非二次剩余条件
61             break;
62     }
63     num x = {a, 1};
64     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
65 }
66
67 int main()
68 {
69     ll n, p;
70     cin >> n >> p;
71     srand((unsigned)time(NULL));
72     cout << Cipolla(n, p) << endl;
73     return 0;
74 }

```

## 4.9 反演相关

```

1  /*
2  莫比乌斯反演
3   $g[n] = \sum_{d|n} f[d]$ 
4   $f[d] = \sum_{d|n} g[d] * \mu[n/d]$ 
5  二项式反演
6   $g[n] = \sum_{i=1}^n C(n, i) * f[i]$ 
7   $f[n] = \sum_{i=1}^n C(n, i) * g[i] * (-1)^{n-i}$ 
8  子集反演
9   $f(S) = \sum_{T \subseteq S} g(T)$ 
10  $g(S) = \sum_{T \subseteq S} f(T) * (-1)^{|S| - |T|}$ 
11 */

```

#### 4.10 斐波那契

```

1 gcd(f[n], f[m]) = f[gcd(n, m)]
2
3 斐波那契前n项和S[n] = f[n+2] - 1;

```

#### 4.11 光速幂

```

1
2 ll v_pow(ll a, ll b) {
3     ll ans = 1;
4     ll base = 65536, k = 1;
5     while(1) {
6         if((b % (k * base)) / k == 0) break;
7         ans = ans * quick_pow(a, (b % (k * base)) / k) % mod;
8         a = quick_pow(a, base) % mod;
9         k = k * base;
10    }
11    return ans;
12 }

```

#### 4.12 康托展开

```

1 #include <bits/stdc++.h> //康托展开
2 using namespace std;
3 typedef long long ll;
4 const int N = 1e6 + 5;
5 const int Mod = 998244353;
6 int T, n;
7 ll a[N], fac[N], c[N];
8 inline int lowbit(int x) { return x & -x; }
9 void add(int x, int v) {
10     for (int i = x; i <= n; i += lowbit(i)) c[i] += v;
11 }
12 int query(int x) {
13     int res = 0;
14     for (int i = x; i > 0; i -= lowbit(i)) res += c[i];
15     return res;
16 }
17 void get_fac() {
18     fac[1] = 1;
19     for (int i = 2; i <= N; i++) fac[i] = fac[i - 1] * i % Mod;
20 }
21 ll cantor() {
22     for (int i = n; i >= 1; i--) cin >> a[i];
23     ll res = 0;
24     for (int i = 1; i <= n; i++) {
25         res += 1ll * query(a[i]) * fac[i - 1];
26         res %= Mod;
27         add(a[i], 1);
28     }
29     return (res + 1) % Mod;
30 }
31
32 int main() {
33     get_fac();
34     cin >> n;

```

```

35     cout << cantor() << endl;
36 }
37 /*
38
39 ans = sum(a[i] * (i-1)!.....) + 1
40 a[i] 代表第i个数比i到n的数之中大的个数
41
42 */
43
44 //逆康托展开
45 #include <bits/stdc++.h>
46 using namespace std;
47 typedef long long ll;
48 const int N = 1e6 + 5;
49 const int Mod = 998244353;
50 int T, n, order;
51 ll fac[N];
52 vector<int> a, ans;
53 void get_fac() {
54     fac[1] = fac[0] = 1;
55     for (int i = 2; i <= N; i++) fac[i] = fac[i - 1] * i % Mod;
56 }
57 void decantor(int order, int n) {
58     for (int i = 1; i <= n; i++) a.push_back(i);
59
60     for (int i = n; i >= 1; i--) {
61         int r = order % fac[i - 1];
62         int t = order / fac[i - 1];
63         order = r;
64         ans.push_back(a[t]);
65         a.erase(a.begin() + t);
66     }
67 }
68 int main() {
69     get_fac();
70     cin >> order >> n;
71     order--;
72     for (auto x : ans) cout << x << " ";
73 }

```

#### 4.13 快速幂

```

1 ll mul(ll a, ll b) {
2     ll z = (long double) a / mod * b;
3     ll res = (unsigned long long) a * b - (unsigned long long) z * mod;
4     return (res + mod) % mod;
5 }
6 // O(1) quick_mul, use long double
7 inline ll quick_pow(ll ans, ll p, ll res = 1) {
8     for (; p >= 1, ans = mul(ans, ans) % mod)
9         if(p & 1) res = mul(res, ans) % mod;
10    return res % mod;
11 }
12 double gcd(double a, double b) {
13     if(fabs(b) < eps) return a;
14     if(fabs(a) < eps) return b;
15     return gcd(b, fmod(a,b));
16 }

```

```

17 int gcd(int a, int b) { return __gcd(a, b); }
18 ll gcd(ll a, ll b) { return __gcd(a, b); }

```

#### 4.14 莫比乌斯反演

```

1 ll prime[N], mu[N], k;
2 ll phi[N];
3 bool is_prime[N];
4 inline void init(int n) {
5     memset(is_prime, true, sizeof is_prime);
6     mu[1] = 1; phi[1] = 1;
7     for (int i = 2; i < n; ++i) {
8         if (is_prime[i]) prime[++k] = i, mu[i] = -1, phi[i] = i-1;
9         for (int j = 1; j <= k && i * prime[j] < n; ++j) {
10             is_prime[i * prime[j]] = false;
11             if (i % prime[j] == 0) {
12                 phi[i * prime[j]] = phi[i] * prime[j];
13                 break;
14             } else {
15                 mu[i * prime[j]] = -mu[i];
16                 phi[i * prime[j]] = phi[i] * (prime[j] - 1);
17             }
18         }
19     }
20     for (int i = 1; i < n; ++i) mu[i] += mu[i-1], phi[i] += phi[i-1];
21 }
22
23 ll cal(int n, int m) {
24     ll ans = 0;
25     n /= d, m /= d;
26     int mx = min(n, m);
27     for (ll l = 1, r; l <= mx; l = r + 1) {
28         r = min(n / (n / l), m / (m / l));
29         ans += 1ll * (mu[r] - mu[l-1]) * (n / l) * (m / l);
30     }
31     return ans;
32 }

```

#### 4.15 逆元

```

1 /*** 逆元 ***/
2 namespace INV {
3     typedef long long ll;
4     const int N = 2e5 + 10;
5     const int mod = 1e9 + 7;
6     int inv[N];
7     ll x, y;
8     ll gcd(ll a, ll b) {
9         return b ? gcd(b, a % b) : a;
10    }
11    void get_inv(int n) {
12        inv[0] = inv[1] = 1;
13        for (int i = 2; i <= n; ++i)
14            inv[i] = 1ll * (mod - mod / i) * inv[mod % i] % mod;
15    }
16    ll quick_pow(ll ans, ll p, ll res = 1) {
17        ans %= mod;

```

```

18     p %= mod - 1;
19     for(; p; p >>= 1, ans = ans * ans % mod)
20         if(p & 1) res = res * ans % mod;
21     return res % mod;
22 }
23 ll inv1(ll ans) {
24     return quick_pow(ans, mod - 2);
25 }
26 ll ex_gcd(ll a, ll b, ll &x, ll &y) { // 返回 gcd(a, b) 并求出 x, y 使得 ax + by = gcd(a, b)
27     if(!b) {
28         x = 1, y = 0;
29         return a;
30     }
31     ll r = ex_gcd(b, a % b, y, x);
32     ll t = x;
33     x = y;
34     y = t - a / b * y;
35     return r;
36 }
37 ll inv2(ll a) { // 返回 a 在模 mod 下的逆元，即 a * x ≡ 1 (mod mod)
38     ll d = ex_gcd(a, mod, x, y);
39     return d == 1 ? (x % mod + mod) % mod : -1;
40 }
41 }

```

#### 4.16 欧拉函数

```

1  int prime[N], phi[N], k;
2  bool is_prime[N];
3  void get_phi(int n) {
4      memset(is_prime, true, sizeof is_prime);
5      phi[1] = 1;
6      for (int i = 2; i < n; i++) {
7          if (is_prime[i]) prime[++k] = i, phi[i] = i - 1;
8          for (int j = 1; j <= k && i * prime[j] < n; j++) {
9              is_prime[i * prime[j]] = false;
10             if (i % prime[j] == 0) {
11                 phi[i * prime[j]] = phi[i] * prime[j];
12                 break;
13             }
14             phi[i * prime[j]] = phi[i] * (prime[j] - 1);
15         }
16     }
17 }
18
19 ll init(ll n) {
20     ll m = (int)sqrt(n + 0.5);
21     ll ans = n;
22     for (ll i = 2; i <= m; ++i) {
23         if (n % i == 0) {
24             ans = ans / i * (i - 1);
25             while(n % i == 0) n /= i;
26         }
27     }
28     if (n > 1) ans = ans / n * (n - 1);
29     return ans;
30 }

```



## 4.17 欧拉降幂

```

1 void init(int n) {
2     memset(is_prime, true, sizeof is_prime);
3     phi[1] = 1;
4     for (int i = 2; i < n; i++) {
5         if (is_prime[i]) prime[++k] = i, phi[i] = i - 1;
6         for (int j = 1; j <= k && i * prime[j] < n; j++) {
7             is_prime[i * prime[j]] = false;
8             if (i % prime[j] == 0) {
9                 phi[i * prime[j]] = phi[i] * prime[j];
10                break;
11            }
12            phi[i * prime[j]] = phi[i] * (prime[j] - 1);
13        }
14    }
15 }
16 ll mod(ll a, ll mm) {return a >= mm ? a % mm + mm : a;}
17 ll ksm(ll a, ll b, ll mm) {
18     ll res = 1, base = a;
19     while (b) {
20         if (b & 1) res = mod(res * base, mm);
21         base = mod(base * base, mm);
22         b >>= 1;
23     }
24     return res;
25 }
26 ll calc(ll a, ll b, ll mm) {
27     if (b == 0 || mm == 1) return 1;
28     else return ksm(a, calc(a, b-1, phi[mm]), mm);
29 }
30 ll f(ll a, ll b, ll mm) {
31     if (a == 0) return 0;
32     ll ans = calc(a, b, mm) % mm;
33     return ans;
34 }

```

## 4.18 筛法树

```

1 ll fa_prime[N], fa_prime_edge[N];
2 ll prime[N], cnt;
3 bool is_prime[N];
4 void prime_table(int n) {
5     memset(is_prime, true, sizeof is_prime);
6     cnt = 0;
7     for (int i = 2; i < n; i++) {
8         if (is_prime[i]) prime[++cnt] = i, fa_prime[i] = 1, fa_prime_edge[i] = i;
9         for (int j = 1; j <= cnt && i * prime[j] < n; j++) {
10            is_prime[i * prime[j]] = false;
11            fa_prime[i * prime[j]] = i;
12            fa_prime_edge[i * prime[j]] = prime[j];
13            if (i % prime[j] == 0) break;
14        }
15    }
16 }

```

## 4.19 算术基本定理

## 4.20 质数、积性函数

77

```

20 }
21
22 ll Euler(ll n) {
23     ll res = n;
24     for(int i = 2; i * i <= n; ++i) {
25         if(n % i == 0) res = res / i * (i - 1);
26         while(n % i == 0) n /= i;
27     }
28     if(n > 1) res = res / n * (n - 1);
29     return res;
30 }

```

#### 4.21 线性筛

```

1 -----//筛质数
2 const int N = 1e6 + 5;
3 int prime[N];
4 bool is_prime[N];
5 void get_prime(){
6     int k = 0;
7     memset(is_prime, true, sizeof is_prime);
8     is_prime[0] = is_prime[1] = false;
9     for(int i = 2; i <= N; i++){
10         if (is_prime[i]) prime[++k] = i;
11         for(int j = 1; j <= k && i * prime[j] <= N; j++){
12             is_prime[i * prime[j]] = false;
13             if(i % prime[j] == 0) break;
14         }
15     }
16 }
17 -----//约数和
18 int prime[N], cnt;
19 bool is_prime[N];
20 ll sum[N], e[N];
21
22 void init() {
23     memset(is_prime, true, sizeof is_prime);
24     sum[1] = 1;
25     for (int i = 2; i < N; ++i) {
26         if (is_prime[i]) {
27             prime[++cnt] = i;
28             sum[i] = i + 1;
29             e[i] = 1;
30         }
31         for (int j = 1; j <= cnt && 1ll*i * prime[j] < N; ++j) {
32             is_prime[prime[j] * i] = false;
33             if (i % prime[j] == 0) {
34                 sum[i * prime[j]] = sum[i] * prime[j] + e[i];
35                 e[i * prime[j]] = e[i];
36                 break;
37             }
38             sum[i * prime[j]] = sum[i] * (prime[j] + 1);
39             e[i * prime[j]] = sum[i];
40         }
41     }
42 }

```

## 4.22 线性同余方程

```

1  /** 线性同余方程求解模板 **/
2  void RemainderEquation(int a, int b, int n) {
3      ll X, Y, d, res;
4      ll min_res;
5      d = gcd(a, n);
6      exgcd(a, n, X, Y);
7      if(b % d == 0) {
8          X = X * (b / d) % n; // 求最小正整数解
9          for(int i = 0; i < d; i++) {
10             res = (X + (i * (b/d))) % n;
11             cout << res << '\n'; // 输出所有解
12         }
13         min_res = (X % (n/d) + (n/d)) % (n/d);
14         cout << min_res << endl; // 输出最小正整数解
15     } else {
16         cout << "No Solutions!" << '\n';
17     }
18 }

```

## 4.23 原根

```

1
2  #include <iostream>
3  #include <vector>
4
5  using namespace std;
6
7  typedef long long ll;
8
9  vector<ll> YG;
10 ll p, n; // p是模数, n是p的欧拉函数值
11
12 ll gcd(ll a, ll b) {
13     return b ? gcd(b, a % b) : a;
14 }
15
16 ll quick_pow(ll a, ll b, ll p) ;
17
18 ll phi(ll n) {
19     ll ans = n;
20     for(int i = 2; i * i <= n; i++) {
21         if(n % i == 0) {
22             ans = ans - ans / i;
23             while(n % i == 0) {
24                 n /= i;
25             }
26         }
27     }
28     if(n > 1)
29         ans = ans - ans / n;
30     return ans;
31 }
32
33 vector<ll> PrimeFac(ll n) { // n的素因子
34     vector<ll> fac;
35     fac.clear();

```

```

36     for(ll i = 2; i * i <= n; i++) {
37         if(n % i == 0) {
38             fac.push_back(i);
39             while(n % i == 0)
40                 n /= i;
41         }
42     }
43     if(n > 1)
44         fac.push_back(n);
45     return fac;
46 }
47
48 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)
49     if(p == 2 || p == 4) return true;
50     if(p <= 1 || p % 4 == 0) return false;
51     ll num = 0;
52     while(p % 2 == 0) // 2的倍数先筛掉
53         p /= 2;
54     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
55         if(p % i == 0) {
56             num++;
57             while(p % i == 0)
58                 p /= i;
59         }
60     }
61     if(p > 1) num++;
62     if(num == 1) return true;
63     return false;
64 }
65
66 ll Protogen(ll p) {
67     if(!is_Protogen(p)) // 先判断是否存在原根
68         return -1;
69     n = phi(p);
70     if(p == 2) return 1;
71     if(p == 3) return 2;
72     if(p == 4) return 3;
73     vector<ll> fac = PrimeFac(n); // f(p)的素因子
74     for(int i = 2; i <= p - 1; i++) {
75         if(gcd(i, p) != 1) // n是模p的欧拉函数值, i要和n互质
76             continue;
77         bool flag = true;
78         for(ll j = 0; j < fac.size(); j++) {
79             if(quick_pow(i, n / fac[j], p) == 1)
80                 flag = 0;
81         }
82         if(flag) // i就是原根
83             return i;
84     }
85     return -1;
86 }
87
88 void Sum_Protogen(ll k) { // 找出n的所有原根
89     YG.push_back(k);
90     for(int i = 2; i < n; i++) {
91         if(gcd(i, n) == 1) // i要与f(n)互质
92             YG.push_back(quick_pow(k, i, p));
93     }
94 }

```

```

95
96 int main() {
97     cin >> p;
98     ll k = Protogen(p); // p的原根
99     cout << k << endl;
100     Sum_Protogen(k);
101     for(int i = 0; i < YG.size(); i++) {
102         cout << YG[i] << " ";
103     }
104     cout << endl;
105     return 0;
106 }

```

#### 4.24 原根表

1	mod				原根
2	$r \cdot 2^{k+1}$	r	k	g	
3	3	1	1	2	
4	5	1	2	2	
5	17	1	4	3	
6	97	3	5	5	
7	193	3	6	5	
8	257	1	8	3	
9	7681	15	9	17	
10	12289	3	12	11	
11	40961	5	13	3	
12	65537	1	16	3	
13	786433	3	18	10	
14	5767169	11	19	3	
15	7340033	7	20	3	
16	23068673	11	21	3	
17	104857601	25	22	3	
18	167772161	5	25	3	
19	469762049	7	26	3	
20	998244353	119	23	3	这个数常用
21	1004535809	479	21	3	加起来不会爆int
22	2013265921	15	27	31	
23	2281701377	17	27	3	这个数平方刚好不会爆ll
24	3221225473	3	30	5	
25	75161927681	35	31	3	
26	77309411329	9	33	7	
27	206158430209	3	36	22	
28	2061584302081	15	37	7	
29	2748779069441	5	39	3	
30	6597069766657	3	41	5	
31	39582418599937	9	42	5	
32	79164837199873	9	43	5	
33	263882790666241	15	44	7	
34	1231453023109121	35	45	3	
35	1337006139375617	19	46	3	
36	3799912185593857	27	47	5	
37	4222124650659841	15	48	19	
38	7881299347898369	7	50	6	
39	31525197391593473	7	52	3	
40	180143985094819841	5	55	6	
41	1945555039024054273	27	56	5	
42	4179340454199820289	29	57	3	

## 4.25 整除分块

```

1 int calc(int n, int m) {
2     //  $\sum_{i=1}^m n / i$ 
3     // 向下取整
4     for (int l = 1, r; l <= m; l = r + 1) {
5         if (n / l) r = min(m, n / (n / l));
6         else r = m;
7         // [l, r] 之间的  $n / l$  都相等
8     }
9
10    // 向上取整
11    for (int l = 1, r; l <= m; l = r + 1) {
12        int t = (n + l - 1) / l;
13        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 }

```

## 4.26 整数拆分

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef long long ll;
5
6 // 递归
7 ll PartitionCount(ll n, ll m)
8 {
9     if (n == 1 || m == 1)
10         return 1;
11     else if (n < m)
12         return PartitionCount(n, n);
13     else if (n == m)
14         return PartitionCount(n, n - 1) + 1;
15     else
16         return PartitionCount(n - m, m) + PartitionCount(n, m - 1);
17 }
18
19 // DP
20 ll dp[10005][10005];
21
22 void Partition_DP(ll n, ll m)
23 {
24     for (ll i = 1; i <= n + 1; i++)
25     {
26         for (ll j = 1; j <= m + 1; j++)
27         {
28             if (i == 1 || j == 1)
29                 dp[i][j] = 1;
30             else if (i == j)
31                 dp[i][j] = 1 + dp[i][j - 1];
32             else if (i < j)
33                 dp[i][j] = dp[i][i];
34             else
35                 dp[i][j] = dp[i - j][j] + dp[i][j - 1];
36         }
37     }
38 }

```

```

36     }
37 }
38 }

```

#### 4.27 整数分块

```

1 void cal(ll a) {
2     for (ll l = 1, r; l <= a; l = r + 1) {
3         r = min(a, (a / (a / l)));
4         ans = (ans + (r - l + 1) % MOD * (a / l));
5     }
6 }

```

#### 4.28 BSGS

```

1  /**BSGS**/
2  int A,B,C; //  $A^x = B \pmod C$ 
3  struct Hashmap { // 10 0 10 0 0 map
4      static const int Ha = 999917, maxe = 46340;
5      int E,lnk[Ha],son[maxe+5],nxt[maxe+5],w[maxe+5];
6      int top,stk[maxe+5];
7      void clear() {E=0;while (top) lnk[stk[top--]]=0;}
8      void Add(int x,int y) {son[++E]=y;nxt[E]=lnk[x];w[E]=((1<<30)-1)*2+1;lnk[x]=E;}
9      bool count(int y) {
10         int x=y%Ha;
11         for (int j=lnk[x];j;j=nxt[j])
12             if (y==son[j]) return true;
13         return false;
14     }
15     int& operator [] (int y) {
16         int x=y%Ha;
17         for (int j=lnk[x];j;j=nxt[j])
18             if (y==son[j]) return w[j];
19         Add(x,y);stk[++top]=x;return w[E];
20     }
21 } f;
22 int exgcd(int a,int b,int &x,int &y) {
23     if (!b) { x = 1; y = 0; return a; }
24     int r = exgcd(b,a % b,x,y), t = x; x = y; y = t - a / b * y;
25     return r;
26 }
27 int BSGS(int A,int B,int C) {
28     if (C==1) if (!B) return A!=1; else return -1;
29     if (B==1) if (A) return 0; else return -1;
30     if (A%C==0) if (!B) return 1; else return -1; //%, 0 0 0 0 0
31     int m=ceil(sqrt(C)),D=1,Base=1;f.clear();
32     for (int i=0;i<=m-1;i++) { // 0 0  $A^j$  0 10 0 10
33         f[Base]=min(f[Base],i);
34         Base=((ll)Base*A)%C;
35     }
36     for (int i=0;i<=m-1;i++) {
37         int x,y,r=exgcd(D,C,x,y);
38         x=((ll)x*B%C+C)%C; //  $y$  0  $A^j$ 
39         if (f.count(x)) return i*m+f[x]; // 0 0 % 0
40         D=((ll)D*Base)%C;
41     }
42     return -1;

```



43 }

## 4.29 Simpson 积分

```

1
2 inline double f(double x) {
3     return 0.0;
4 }
5 double simpson(double l, double r) {
6     double mid = (l + r) / 2;
7     return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;
8 }
9 double asr(double l, double r, double ans, double eps) {
10     double mid = (l + r) / 2;
11     double fl = simpson(l, mid), fr = simpson(mid, r);
12     if (abs(fl + fr - ans) <= 15 * eps)
13         return fl + fr + (fl + fr - ans) / 15;
14     return asr(l, mid, fl, eps * 0.5) + asr(mid, r, fr, eps * 0.5);
15 }

```

## 4.30 2 维计算几何

```

1 const double pi = acos(-1.0);
2 // 弧度制转角度制
3 const double eps = 1e-8;
4 const int maxp = 1010;
5 //`Compares a double to zero`
6 int sgn(double x) {
7     if(fabs(x) < eps) return 0;
8     return (x > 0? 1: -1);
9 }
10 //square of a double
11 inline double sqr(double x) { return x * x; }
12 /*
13  * Point
14  * Point() - Empty constructor
15  * Point(double _x,double _y) - constructor
16  * input() - double input
17  * output() - %.2f output
18  * operator == - compares x and y
19  * operator < - compares first by x, then by y
20  * operator - - return new Point after subtracting corresponding x and y
21  * operator ^ - cross product of 2d points
22  * operator * - dot product
23  * len() - gives length from origin
24  * len2() - gives square of length from origin
25  * distance(Point p) - gives distance from p
26  * operator + Point b - returns new Point after adding corresponding x and y
27  * operator * double k - returns new Point after multiplying x and y by k
28  * operator / double k - returns new Point after dividing x and y by k
29  * rad(Point a,Point b)- returns the angle of Point a and Point b from this Point
30  * trunc(double r) - return Point that if truncated the distance from center to r
31  * rotright() - returns 90 degree ccw rotated point
32  * rotright() - returns 90 degree cw rotated point
33  * rotate(Point p,double angle) - returns Point after rotating the Point centering at
    p by angle radian ccw
34 */

```

```

35 struct Point {
36     double x,y;
37     Point() {}
38     Point(double _x,double _y) { x = _x; y = _y; }
39     void input() { cin >> x >> y; }
40     bool operator == (Point b)const {
41         return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
42     }
43     bool operator < (Point b)const {
44         return sgn(x - b.x) == 0? sgn(y - b.y) < 0: x < b.x;
45     }
46     Point operator -(const Point &b)const {
47         return Point(x - b.x, y - b.y);
48     }
49     double operator ^(const Point &b)const { //^2
50         return x * b.y - y * b.x;
51     }
52     double operator *(const Point &b)const { //μ
53         return x * b.x + y * b.y;
54     }
55     double len() { //μ ㄖ
56         return hypot(x, y); //ㄖ
57     }
58     double len2() { //μ ㄖ
59         return x * x + y * y;
60     }
61     double distance(Point p) { //μ }μ ㄖ
62         return hypot(x - p.x, y - p.y);
63     }
64     Point operator +(const Point &b)const {
65         return Point(x + b.x, y + b.y);
66     }
67     Point operator *(const double &k)const {
68         return Point(x * k, y * k);
69     }
70     Point operator /(const double &k)const {
71         return Point(x / k, y / k);
72     }
73     //^ ㄖ ㄖ pa ° pb μ ㄖ
74     double rad(Point a,Point b) {
75         Point p = *this;
76         return fabs(atan2( fabs((a-p)^(b-p)),(a-p)*(b-p) ));
77     }
78     //^ ㄖ ㄖ rμ ㄖ
79     Point trunc(double r) {
80         double l = len();
81         if(!sgn(l)) return *this;
82         r /= l;
83         return Point(x*r,y*r);
84     }
85     //^ ㄖ ㄖ ㄖ 90 ㄖ
86     Point rotleft() { return Point(-y,x); }
87     //^ ㄖ ㄖ ㄖ 90 ㄖ
88     Point rotright() { return Point(y,-x); }
89     //^ ㄖ ㄖ ㄖ μ ㄖ ㄖ angle
90     Point rotate(Point p,double angle) {
91         Point v = (*this) - p;
92         double c = cos(angle), s = sin(angle);

```

```

93         return Point(p.x + v.x * c - v.y * s,
94                       p.y + v.x * s + v.y * c);
95     }
96 };
97 /*
98  * Stores two points
99  * Line() - Empty constructor
100  * Line(Point _s,Point _e) - Line through _s and _e
101  * operator == - checks if two points are same
102  * Line(Point p,double angle) - one end p , another end at angle degree
103  * Line(double a,double b,double c) - Line of equation ax + by + c = 0
104  * input() - inputs s and e
105  * adjust() - orders in such a way that s < e
106  * length() - distance of se
107  * angle() - return 0 <= angle < pi
108  * relation(Point p) - 3 if point is on line
109  * - 1 if point on the left of line
110  * - 2 if point on the right of line
111  * pointonseg(double p) - return true if point on segment
112  * parallel(Line v) - return true if they are parallel
113  * segcrossseg(Line v) - returns 0 if does not intersect
114  * - returns 1 if non-standard intersection
115  * - returns 2 if intersects
116  * linecrossseg(Line v) - line and seg
117  * linecrossline(Line v) - 0 if parallel
118  * - 1 if coincides
119  * - 2 if intersects
120  * crosspoint(Line v) - returns intersection point
121  * dispointtoline(Point p) - distance from point p to the line
122  * dispointtoseg(Point p) - distance from p to the segment
123  * dissegtoseg(Line v) - distance of two segment
124  * lineprog(Point p) - returns projected point p on se line
125  * symmetrypoint(Point p) - returns reflection point of p over se
126  *
127 */
128 struct Line {
129     Point s, e;
130     Line() {}
131     Line(Point _s,Point _e) { s = _s; e = _e; }
132     bool operator ==(Line v) {
133         return (s == v.s) && (e == v.e);
134     }
135     //^, 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151
136     Line(Point p, double angle) {
137         s = p;
138         if(sgn(angle-pi/2) == 0) e = (s + Point(0,1));
139         else e = (s + Point(1, tan(angle)));
140     }
141     //ax + by + c = 0
142     Line(double a, double b, double c) {
143         if(sgn(a) == 0) {
144             s = Point(0,-c/b);
145             e = Point(1,-c/b);
146         } else if(sgn(b) == 0) {
147             s = Point(-c/a,0);
148             e = Point(-c/a,1);
149         } else {
150             s = Point(0,-c/b);
151             e = Point(1,(-c-a)/b);

```



88



```

327     Point a = v.lineprog(p);
328     double d = v.dispointtoline(p);
329     d = sqrt(r * r - d * d);
330     if(sgn(d) == 0) { p1 = p2 = a; return 1; }
331     p1 = a + (v.e-v.s).trunc(d);
332     p2 = a - (v.e-v.s).trunc(d);
333     return 2;
334 }
335 //μδ%1 a,b}μ得° r1μ } , 0 0 `
336 int gercircle(Point a,Point b,double r1,circle &c1,circle &c2) {
337     circle x(a,r1),y(b,r1);
338     int t = x.pointcrosscircle(y,c1.p,c2.p);
339     if(!t) return 0;
340     c1.r = c2.r = r1;
341     return t;
342 }
343 //μδ% 0 0 0 u 0 0 Y-1 μ q, ° r1μ 0 `
344 int getcircle(Line u,Point q,double r1,circle &c1,circle &c2) {
345     double dis = u.dispointtoline(q);
346     if(sgn(dis - r1 * 2) > 0) return 0;
347     if(sgn(dis) == 0) {
348         c1.p = q + ((u.e - u.s).rotleft().trunc(r1));
349         c2.p = q + ((u.e - u.s).rotright().trunc(r1));
350         c1.r = c2.r = r1;
351         return 2;
352     }
353     Line u1 = Line((u.s + (u.e-u.s).rotleft().trunc(r1)),
354                    (u.e + (u.e-u.s).rotleft().trunc(r1)));
355     Line u2 = Line((u.s + (u.e-u.s).rotright().trunc(r1)),
356                    (u.e + (u.e-u.s).rotright().trunc(r1)));
357     circle cc = circle(q,r1);
358     Point p1,p2;
359     if(!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
360     c1 = circle(p1,r1);
361     if(p1 == p2) { c2 = c1; return 1; }
362     c2 = circle(p2,r1);
363     return 2;
364 }
365 //r° 0 0 0 u,v 0 0 Y-° r1μ 0 `
366 int getcircle(Line u,Line v,double r1,circle &c1,circle &c2,circle &c3,circle &c4)
367 {
368     if(u.parallel(v))return 0;//} 0 0 0 0
369     Line u1 = Line(u.s + (u.e-u.s).rotleft().trunc(r1),
370                    u.e + (u.e-u.s).rotleft().trunc(r1));
371     Line u2 = Line(u.s + (u.e-u.s).rotright().trunc(r1),
372                    u.e + (u.e-u.s).rotright().trunc(r1));
373     Line v1 = Line(v.s + (v.e-v.s).rotleft().trunc(r1),
374                    v.e + (v.e-v.s).rotleft().trunc(r1));
375     Line v2 = Line(v.s + (v.e-v.s).rotright().trunc(r1),
376                    v.e + (v.e-v.s).rotright().trunc(r1));
377     c1.r = c2.r = c3.r = c4.r = r1;
378     c1.p = u1.crosspoint(v1);
379     c2.p = u1.crosspoint(v2);
380     c3.p = u2.crosspoint(v1);
381     c4.p = u2.crosspoint(v2);
382     return 4;
383 }
384 //r° 0 0 0 cx,cy 0 0 Y-° r1μ 0 `
385 int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2) {

```

```

385     circle x(cx.p, r1 + cx.r), y(cy.p, r1 + cy.r);
386     int t = x.pointcrosscircle(y, c1.p, c2.p);
387     if(!t) return 0;
388     c1.r = c2.r = r1;
389     return t;
390 }
391
392 // 判断两圆是否相切
393 int tangentline(Point q, Line &u, Line &v) {
394     int x = relation(q);
395     if(x == 2) return 0;
396     if(x == 1) {
397         u = Line(q, q + (q - p).rotleft());
398         v = u;
399         return 1;
400     }
401     double d = p.distance(q);
402     double l = r * r / d;
403     double h = sqrt(r * r - l * l);
404     u = Line(q, p + ((q - p).trunc(l) + (q - p).rotleft().trunc(h)));
405     v = Line(q, p + ((q - p).trunc(l) + (q - p).rotright().trunc(h)));
406     return 2;
407 }
408 // 计算两圆公共部分的面积
409 double areacircle(circle v) {
410     int rel = relationcircle(v);
411     if(rel >= 4) return 0.0;
412     if(rel <= 2) return min(area(), v.area());
413     double d = p.distance(v.p);
414     double hf = (r + v.r + d) / 2.0;
415     double ss = 2 * sqrt(hf * (hf - r) * (hf - v.r) * (hf - d));
416     double a1 = acos((r * r + d * d - v.r * v.r) / (2.0 * r * d));
417     a1 = a1 * r * r;
418     double a2 = acos((v.r * v.r + d * d - r * r) / (2.0 * v.r * d));
419     a2 = a2 * v.r * v.r;
420     return a1 + a2 - ss;
421 }
422 // 计算两圆公共部分的面积 (使用 long double)
423 double areacircle2(circle v) {
424     double a = hypot(p.x - v.p.x, p.y - v.p.y), b = r, c = v.r;
425     double s1 = pi * r * r, s2 = pi * v.r * v.r;
426     if(sgn(a - b - c) >= 0) return 0;
427     if(sgn(a + min(b, c) - max(b, c)) <= 0) return min(s1, s2);
428     else {
429         double cta1 = 2 * acos((a * a + b * b - c * c) / (2 * a * b));
430         double cta2 = 2 * acos((a * a + c * c - b * b) / (2 * a * c));
431         return cta1 / (2 * pi) * s1 - 0.5 * sin(cta1) * b * b +
432             cta2 / (2 * pi) * s2 - 0.5 * sin(cta2) * c * c;
433     }
434 }
435 // 计算三角形的面积
436 double areatriangle(Point a, Point b) {
437     if(sgn((p - a) ^ (p - b)) == 0) return 0.0;
438     Point q[5];
439     int len = 0;
440     q[len++] = a;
441     Line l(a, b);
442     Point p1, p2;
443     if(pointcrossline(l, q[1], q[2]) == 2) {

```



```

444         if(sgn((a - q[1]) * (b - q[1])) < 0) q[len++] = q[1];
445         if(sgn((a - q[2]) * (b - q[2])) < 0) q[len++] = q[2];
446     }
447     q[len++] = b;
448     if(len == 4 && sgn((q[0] - q[1]) * (q[2] - q[1])) > 0) swap(q[1], q[2]);
449     double res = 0;
450     for(int i = 0; i < len - 1; ++i) {
451         if(relation(q[i]) == 0 || relation(q[i + 1]) == 0) {
452             double arg = p.rad(q[i], q[i + 1]);
453             res += r * r * arg / 2.0;
454         } else res += fabs((q[i] - p) ^ (q[i + 1] - p)) / 2.0;
455     }
456     return res;
457 }
458 };
459 /*
460 * n,p Line l for each side
461 * input(int _n)                - inputs _n size polygon
462 * add(Point q)                 - adds a point at end of the list
463 * getline()                   - populates line array
464 * cmp                          - comparision in convex_hull order
465 * norm()                       - sorting in convex_hull order
466 * getconvex(polygon &convex)   - returns convex hull in convex
467 * Graham(polygon &convex)      - returns convex hull in convex
468 * isconvex()                   - checks if convex
469 * relationpoint(Point q)       - returns 3 if q is a vertex
470 *                               2 if on a side
471 *                               1 if inside
472 *                               0 if outside
473 * convexcute(Line u,polygon &po) - left side of u in po
474 * gercircumference()           - returns side length
475 * getarea()                    - returns area
476 * getdir()                     - returns 0 for cw, 1 for ccw
477 * getbarycentre()              - returns barycenter
478 *
479 */
480 struct polygon {
481     int n;
482     Point p[maxp];
483     Line l[maxp];
484     void input(int _n) {
485         n = _n;
486         for(int i = 0; i < n; i++)
487             p[i].input();
488     }
489     void add(Point q) {
490         p[n++] = q;
491     }
492     void getline() {
493         for(int i = 0; i < n; i++) {
494             l[i] = Line(p[i], p[(i+1)%n]);
495         }
496     }
497     struct cmp {
498         Point p;
499         cmp(const Point &p0) { p = p0; }
500         bool operator()(const Point &aa, const Point &bb) {
501             Point a = aa, b = bb;
502             int d = sgn((a-p)^(b-p));

```

```

503         if(d == 0) {
504             return sgn(a.distance(p)-b.distance(p)) < 0;
505         }
506         return d > 0;
507     }
508 };
509 // 求凸包
510 // 输入点集 p[0] ~ p[n-1]
511 // 输出凸包点集 p[0] ~ p[top-1]
512 void norm() {
513     Point mi = p[0];
514     for(int i = 1; i < n; i++) mi = min(mi, p[i]);
515     sort(p, p + n, cmp(mi));
516 }
517 // 求凸包
518 // 输入点集 p[0] ~ p[n-1]
519 // 输出凸包点集 p[0] ~ p[top-1]
520 // 输入点集 p[0] ~ p[n-1]
521 void getconvex(polygon &convex) {
522     sort(p, p+n);
523     convex.n = n;
524     for(int i = 0; i < min(n, 2); i++) {
525         convex.p[i] = p[i];
526     }
527     if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;
528     if(n <= 2)return;
529     int &top = convex.n;
530     top = 1;
531     for(int i = 2; i < n; i++) {
532         while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
533             top--;
534         convex.p[++top] = p[i];
535     }
536     int temp = top;
537     convex.p[++top] = p[n-2];
538     for(int i = n-3; i >= 0; i--) {
539         while(top != temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
540             top--;
541         convex.p[++top] = p[i];
542     }
543     if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;
544     convex.norm();
545 }
546 // 求凸包
547 void Graham(polygon &convex) {
548     norm();
549     int &top = convex.n;
550     top = 0;
551     if(n == 1) {
552         top = 1;
553         convex.p[0] = p[0];
554         return;
555     }
556     if(n == 2) {
557         top = 2;
558         convex.p[0] = p[0];
559         convex.p[1] = p[1];
560         if(convex.p[0] == convex.p[1])top--;
561         return;

```

```

562     convex.p[0] = p[0];
563     convex.p[1] = p[1];
564     top = 2;
565     for(int i = 2; i < n; i++) {
566         while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])^(p[i]-convex.p[top-2])) <= 0 )
567             top--;
568         convex.p[top++] = p[i];
569     }
570     if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;
571 }
572 // 判断点集是否凸
573 bool isconvex() {
574     bool s[3];
575     memset(s,false,sizeof(s));
576     for(int i = 0; i < n; i++) {
577         int j = (i+1)%n;
578         int k = (j+1)%n;
579         s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
580         if(s[0] && s[2])return false;
581     }
582     return true;
583 }
584 // 判断点是否在凸多边形内部
585 int relationpoint(Point q) {
586     for(int i = 0; i < n; i++) {
587         if(p[i] == q)return 3;
588     }
589     getline();
590     for(int i = 0; i < n; i++) {
591         if(l[i].pointonseg(q))return 2;
592     }
593     int cnt = 0;
594     for(int i = 0; i < n; i++) {
595         int j = (i+1)%n;
596         int k = sgn((q-p[j])^(p[i]-p[j]));
597         int u = sgn(p[i].y-q.y);
598         int v = sgn(p[j].y-q.y);
599         if(k > 0 && u < 0 && v >= 0)cnt++;
600         if(k < 0 && v < 0 && u >= 0)cnt--;
601     }
602     return cnt != 0;
603 }
604 // 计算凸包的周长
605 // 计算凸包的面积
606 void convexpolygon(Line u,polygon &po) {
607     int &top = po.n;
608     top = 0;
609     for(int i = 0; i < n; i++) {
610         int d1 = sgn((u.e-u.s)^(p[i]-u.s));
611         int d2 = sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
612         if(d1 >= 0)po.p[top++] = p[i];
613         if(d1*d2 < 0)po.p[top++] = u.crosspoint(Line(p[i],p[(i+1)%n]));
614     }
615 }
616 // 计算凸包的周长
617 double getcircumference() {
618     double sum = 0;
619     for(int i = 0; i < po.n; i++) {
620         sum += dist(po.p[i],po.p[(i+1)%po.n]);
621     }
622     return sum;
623 }

```

95

```

679 //`AB X AC`
680 double cross(Point A,Point B,Point C) {
681     return (B-A)^(C-A);
682 }
683 //`AB*AC`
684 double dot(Point A,Point B,Point C) {
685     return (B-A)*(C-A);
686 }
687 //`0 0 C%0 0 0 0 0 ,^,0 `
688 //` A ±0 0 0 0 0 °0 (¶0 0 0 0 0 0 °0 0 .0 0 )`
689 double minRectangleCover(polygon A) {
690     //`Ç0 0 0 A.n < 3µ0 0 0 `
691     if(A.n < 3)return 0.0;
692     A.p[A.n] = A.p[0];
693     double ans = -1;
694     int r = 1, p = 1, q;
695     for(int i = 0; i < A.n; i++) {
696         //`¿`³0 0 0 A.p[i] - A.p[i+1]0 0 µj0 `
697         while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],A.p[i+1],A.p[r]) )
698             >= 0 )
699             r = (r+1)%A.n;
700         //`¿`³0 A.p[i] - A.p[i+1]:%0 0 0 0 0 0 n0 0 µj0 `
701         while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0 )
702             p = (p+1)%A.n;
703         if(i == 0)q = p;
704         //`¿`³0 A.p[i] - A.p[i+1]:%0 0 0 °0 0 0 µj0 `
705         while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i+1],A.p[q])) <= 0)
706             q = (q+1)%A.n;
707         double d = (A.p[i] - A.p[i+1]).len2();
708         double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
709             (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
710         if(ans < 0 || ans > tmp)ans = tmp;
711     }
712     return ans;
713 }
714 //`0 0 0 0 0 ¶0 0 0 `
715 //`¶0 0 0 0 0 0 °0 0 ¿-0 0 q1q2µ0 0 0 `
716 vector<Point> convexCut(const vector<Point> &ps,Point q1,Point q2) {
717     vector<Point>qs;
718     int n = ps.size();
719     for(int i = 0; i < n; i++) {
720         Point p1 = ps[i], p2 = ps[(i+1)%n];
721         int d1 = sgn((q2-q1)^(p1-q1)), d2 = sgn((q2-q1)^(p2-q1));
722         if(d1 >= 0)
723             qs.push_back(p1);
724         if(d1 * d2 < 0)
725             qs.push_back(Line(p1,p2).crosspoint(Line(q1,q2)));
726     }
727     return qs;
728 }
729 //`°0 0 0 漆`
730 struct halfplane:public Line {
731     double angle;
732     halfplane() {}
733     //`±0 `0 0 -s->e0 0 °0 0 (0 0 0 )µİ0 0 0 `
734     halfplane(Point _s,Point _e) {
735         s = _s;
736         e = _e;

```

```

736     }
737     halfplane(Line v) {
738         s = v.s;
739         e = v.e;
740     }
741     void calcangle() {
742         angle = atan2(e.y-s.y,e.x-s.x);
743     }
744     bool operator <(const halfplane &b)const {
745         return angle < b.angle;
746     }
747 };
748 struct halfplanes {
749     int n;
750     halfplane hp[maxp];
751     Point p[maxp];
752     int que[maxp];
753     int st,ed;
754     void push(halfplane tmp) {
755         hp[n++] = tmp;
756     }
757     //z[]
758     void unique() {
759         int m = 1;
760         for(int i = 1; i < n; i++) {
761             if(sgn(hp[i].angle-hp[i-1].angle) != 0)
762                 hp[m++] = hp[i];
763             else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s) ) > 0)
764                 hp[m-1] = hp[i];
765         }
766         n = m;
767     }
768     bool halfplaneinsert() {
769         for(int i = 0; i < n; i++)hp[i].calcangle();
770         sort(hp, hp+n);
771         unique();
772         que[st=0] = 0;
773         que[ed=1] = 1;
774         p[1] = hp[0].crosspoint(hp[1]);
775         for(int i = 2; i < n; i++) {
776             while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)ed--;
777             while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)st++;
778             que[++ed] = i;
779             if(hp[i].parallel(hp[que[ed-1]]))return false;
780             p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
781         }
782         while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0)ed--;
783         while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s))<0)st
784         ++;
785         if(st+1>=ed)return false;
786         return true;
787     }
788     //μö%  0 0 0 0 漆μö%μ  0 ¶ 0 0 0 `
789     // 0 0 ç0 0 0 0 halfplaneinsert() 0 ¼μ»  true`
790     void getconvex(polygon &con) {
791         p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
792         con.n = ed-st+1;
793         for(int j = st,i = 0; j <= ed; i++,j++)
794             con.p[i] = p[j];

```

```

794 }
795     double minRectangleCover(polygon A) {
796         // 如果 A.n < 3 则无法构成多边形
797         if(A.n < 3) return 0.0;
798         A.p[A.n] = A.p[0];
799         double ans = -1;
800         int r = 1, p = 1, q;
801         for(int i = 0; i < A.n; i++) {
802             // 枚举以 A.p[i] 为起点的边
803             while( sgn( cross(A.p[i], A.p[i+1], A.p[r+1]) - cross(A.p[i], A.p[i+1], A.p[r])
804 ) >= 0 )
805                 r = (r+1)%A.n;
806             // 枚举以 A.p[i] 为起点的边
807             while(sgn( dot(A.p[i], A.p[i+1], A.p[p+1]) - dot(A.p[i], A.p[i+1], A.p[p]) ) >=
808 0 )
809                 p = (p+1)%A.n;
810             // 枚举以 A.p[i] 为起点的边
811             while(sgn(dot(A.p[i], A.p[i+1], A.p[q+1]) - dot(A.p[i], A.p[i+1], A.p[q])) <=
812 0)
813                 q = (q+1)%A.n;
814             double d = (A.p[i] - A.p[i+1]).len2();
815             double tmp = cross(A.p[i], A.p[i+1], A.p[r]) *
816                 (dot(A.p[i], A.p[i+1], A.p[p]) - dot(A.p[i], A.p[i+1], A.p[q]))/d;
817             if(ans < 0 || ans > tmp) ans = tmp;
818         }
819         return ans;
820     }
821     circle minCircleCover(int n, Point p[], Point P = Point(0, 0)) {
822         random_shuffle(p, p + n);
823         double r2 = 0;
824         for(int i = 0; i < n; ++i) {
825             if((p[i] - P).len2() > r2) {
826                 P = p[i], r2 = 0;
827                 for(int j = 0; j < i; ++j) {
828                     if((p[j] - P).len2() > r2) {
829                         P = (p[i] + p[j]) / 2, r2 = (p[j] - P).len2();
830                         for(int k = 0; k < j; ++k) {
831                             if((p[k] - P).len2() > r2) {
832                                 P = circle(p[i], p[j], p[k]).p, r2 = (p[k] - P).len2();
833                             }
834                         }
835                     }
836                 }
837             }
838         }
839         return circle(P, sqrt(r2));
840     }
841 }

```

### 4.31 3 维计算几何

```

1 struct Point3 {
2     double x, y, z;
3     Point3(double xx = 0, double yy = 0, double zz = 0) { x = xx, y = yy, z = zz; }
4     void input() { cin >> x >> y >> z; }
5     void output(void) { cout << fixed << setprecision(3) << x << ' ' << y << ' ' << z
6     << '\n'; }
7     double len(void) { return sqrt(x * x + y * y + z * z); }

```





```

61     deal(p,F[now].b,F[now].a);
62     deal(p,F[now].c,F[now].b);
63     deal(p,F[now].a,F[now].c);
64 }
65 bool same(int s,int t) {
66     Point3 &a=P[F[s].a];
67     Point3 &b=P[F[s].b];
68     Point3 &c=P[F[s].c];
69     int d1=sgn(volume_four(a,b,c,P[F[t].a]));
70     int d2=sgn(volume_four(a,b,c,P[F[t].b]));
71     int d3=sgn(volume_four(a,b,c,P[F[t].c]));
72     return (d1==0)&&(d2==0)&&(d3==0);
73 }
74 //11% 0 0 0 °
75 void create() {
76     num=0;
77     face add;
78     //± 0 j 0 k 依12 0
79     //*****
80     bool flag=true;
81     for(int i=1;i<n;i++) {
82         if(!(P[0]==P[i])) {
83             swap(P[1],P[i]);
84             flag=false;
85             break;
86         }
87     }
88     if(flag) return ;
89     flag=true;
90     for(int i=2;i<n;i++) {
91         if(sgn(((P[1]-P[0])^(P[i]-P[0])).len())>0) {
92             swap(P[2],P[i]);
93             flag=false;
94             break;
95         }
96     }
97     if(flag) return ;
98     flag=true;
99     for(int i=3;i<n;i++) {
100         if(sgn(((P[1]-P[0])^(P[2]-P[0]))*(P[i]-P[0]))!=0) {
101             swap(P[3],P[i]);
102             flag=false;
103             break;
104         }
105     }
106     if(flag) return ;
107     //*****
108     for(int i=0;i<4;i++) {
109         add.a=(i+1)%4;
110         add.b=(i+2)%4;
111         add.c=(i+3)%4;
112         add.ok=true;
113         if(sgn(dblcmp(P[i],add))>0) swap(add.b,add.c);
114         g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=num;
115         F[num++]=add;
116     }
117     for(int i=4;i<n;i++) {
118         for(int j=0;j<num;j++) {
119             if(F[j].ok&&sgn(dblcmp(P[i],F[j]))>0) {

```

```

120         dfs(i,j);
121         break;
122     }
123 }
124 }
125 int tmp=num;
126 num=0;
127 for(int i=0;i<tmp;i++)
128     if(F[i].ok)
129         F[num++]=F[i];
130 }
131 //±0 0 0 0
132 double area(void) {
133     double res=0;
134     if(n==3) {
135         Point3 p=cross(P[0],P[1],P[2]);
136         return p.len()/2;
137     }
138     for(int i=0;i<num;i++)
139         res+=area_triangle(P[F[i].a],P[F[i].b],P[F[i].c]);
140     return res/2.0;
141 }
142 //0 0 0
143 double volume(void) {
144     double res=0;
145     Point3 tmp=Point3(0,0,0);
146     for(int i=0;i<num;i++)
147         res+=volume_four(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
148     return abs(res/6.0);
149 }
150 //±0 0 0 0 0 0 0 0 0 0 0 0
151 int sum_of_triangle(void) {
152     return num;
153 }
154 //±0 0 0 0 0 0 0 0 0 0
155 int sum_of_polygon(void) {
156     int res=0;
157     for(int i=0;i<num;i++) {
158         bool flag=true;
159         for(int j=0;j<i;j++) {
160             if(same(i,j)) {
161                 flag=false;
162                 break;
163             }
164         }
165         res+=flag;
166     }
167     return res;
168 }
169 //0 0 0 0
170 Point3 bary_center(void) {
171     Point3 ans=Point3(0,0,0);
172     Point3 o=Point3(0,0,0);
173     double all=0;
174     for(int i=0;i<num;i++) {
175         double vol=volume_four(o,P[F[i].a],P[F[i].b],P[F[i].c]);
176         ans=ans+(((o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0)*vol);
177         all+=vol;
178     }

```

```

179     ans=ans/all;
180     return ans;
181 }
182 //μ溫 0 10 0 0
183 double dis_point_to_face(Point3 p,int i) {
184     double tmp1=abs(volume_four(P[F[i].a],P[F[i].b],P[F[i].c],p));
185     double tmp2=((P[F[i].b]-P[F[i].a])^(P[F[i].c]-P[F[i].a])).len();
186     return tmp1/tmp2;
187 }
188 };

```

#### 4.32 三分套三分

```

1  int n; double maxx = -double(inf);
2  double xmax = -10000000, xmin = 10000000, ymax = -10000000, ymin = 10000000;
3
4  double check(Point P) {
5      double minn = double(inf);
6      for(int i = 1; i <= n; ++ i) {
7          minn = min(minn, P.rad(p[i], p[i + 1]));
8      }
9      return minn;
10 }
11
12 double Find(double x) {
13     double yl = ymin, yr = ymax, midl, midr;
14     while(yr - yl > eps) {
15         midl = (yl + yl + yr) / 3.0;
16         midr = (yl + yr + yr) / 3.0;
17         if(check(Point(x, midl)) < check(Point(x, midr))) yl = midl;
18         else yr = midr;
19     }
20     maxx = max(maxx, check(Point(x, yl)));
21     return check(Point(x, yl));
22 }
23
24 inline void solve() {
25     cin >> n; for(int i = 1; i <= n; ++ i) {
26         p[i].input();
27         xmax = max(xmax, p[i].x);
28         xmin = min(xmin, p[i].x);
29         ymax = max(ymax, p[i].y);
30         ymin = min(ymin, p[i].y);
31     } p[n + 1] = p[1];
32
33     double xl = xmin, xr = xmax, midl, midr;
34     while(xr - xl > eps) {
35         midl = (xl + xl + xr) / 3.0;
36         midr = (xl + xr + xr) / 3.0;
37         if(Find(midl) < Find(midr)) xl = midl;
38         else xr = midr;
39     }
40     cout << fixed << setprecision(10) << maxx * 180.0 / pi << endl;
41 }

```

#### 4.33 三维几何

```

1  #include <math.h>
2  #define eps 1e-8
3  #define zero(x) (((x)>0?(x):-<(x))<eps)
4  struct point3{double x,y,z;};
5  struct line3{point3 a,b;};
6  struct plane3{point3 a,b,c;};
7  //计算 cross product U x V
8  point3 Cross(point3 u,point3 v){
9      point3 ret;
10     ret.x=u.y*v.z-v.y*u.z;
11     ret.y=u.z*v.x-u.x*v.z;
12     ret.z=u.x*v.y-u.y*v.x;
13     return ret;
14 }
15 //计算 dot product U . V
16 double Dot(point3 u,point3 v){
17     return u.x*v.x+u.y*v.y+u.z*v.z;
18 }
19 //向量差 U - V
20 point3 subtr(point3 u,point3 v){
21     point3 ret;
22     ret.x=u.x-v.x;
23     ret.y=u.y-v.y;
24     ret.z=u.z-v.z;
25     return ret;
26 }
27 //取平面法向量
28 point3 pvec(plane3 s){
29     return Cross(subtr(s.a,s.b),subtr(s.b,s.c));
30 }
31 point3 pvec(point3 s1,point3 s2,point3 s3){
32     return Cross(subtr(s1,s2),subtr(s2,s3));
33 }
34 //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 }
39 //向量大小
40 double vlen(point3 p){
41     return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
42 }
43 //判三点共线
44 int dots_inline(point3 p1,point3 p2,point3 p3){
45     return vlen(Cross(subtr(p1,p2),subtr(p2,p3)))<eps;
46 }
47 //判四点共面
48 int dots_onplane(point3 a,point3 b,point3 c,point3 d){
49     return zero(Dot(pvec(a,b,c),subtr(d,a)));
50 }
51 //判点是否在线段上,包括端点和共线
52 int dot_online_in(point3 p,line3 l){
53     return zero(vlen(Cross(subtr(p,l.a),subtr(p,l.b))))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&
54         (l.a.y-p.y)*(l.b.y-p.y)<eps&&(l.a.z-p.z)*(l.b.z-p.z)<eps;
55 }
56 int dot_online_in(point3 p,point3 l1,point3 l2){
57     return zero(vlen(Cross(subtr(p,l1),subtr(p,l2))))&&(l1.x-p.x)*(l2.x-p.x)<eps&&
58         (l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
59 }

```

```

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         &&
67         (!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
68 }
69 //判点是否在空间三角形上,包括边界,三点共线无意义
70 int dot_inplane_in(point3 p,plane3 s){
71     return zero(vlen(Cross(subt(s.a,s.b),subt(s.a,s.c)))-vlen(Cross(subt(p,s.a),subt(p,
72         s.b))) -
73         vlen(Cross(subt(p,s.b),subt(p,s.c)))-vlen(Cross(subt(p,s.c),subt(p,s.a)
74         ))));
75 }
76 int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3){
77     return zero(vlen(Cross(subt(s1,s2),subt(s1,s3)))-vlen(Cross(subt(p,s1),subt(p,s2)))
78         -
79         vlen(Cross(subt(p,s2),subt(p,s3)))-vlen(Cross(subt(p,s3),subt(p,s1))));
80 }
81 //判点是否在空间三角形上,不包括边界,三点共线无意义
82 int dot_inplane_ex(point3 p,plane3 s){
83     return dot_inplane_in(p,s)&&vlen(Cross(subt(p,s.a),subt(p,s.b)))>eps&&
84         vlen(Cross(subt(p,s.b),subt(p,s.c)))>eps&&vlen(Cross(subt(p,s.c),subt(p,s.a)
85         ))>eps;
86 }
87 int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3){
88     return dot_inplane_in(p,s1,s2,s3)&&vlen(Cross(subt(p,s1),subt(p,s2)))>eps&&
89         vlen(Cross(subt(p,s2),subt(p,s3)))>eps&&vlen(Cross(subt(p,s3),subt(p,s1)))>
90         eps;
91 }
92 //判两点在线段同侧,点在线段上返回 0,不共面无意义
93 int same_side(point3 p1,point3 p2,line3 l){
94     return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))>eps
95     ;
96 }
97 int same_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 //判两点在线段异侧,点在线段上返回 0,不共面无意义
101 int opposite_side(point3 p1,point3 p2,line3 l){
102     return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))<-
103     eps;
104 }
105 int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2){
106     return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))<-eps;
107 }
108 //判两点在平面同侧,点在平面上返回 0
109 int same_side(point3 p1,point3 p2,plane3 s){
110     return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))>eps;
111 }
112 int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
113     return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))>eps;
114 }
115 //判两点在平面异侧,点在平面上返回 0
116 int opposite_side(point3 p1,point3 p2,plane3 s){
117     return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))<-eps;
118 }

```

```

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 //判直线与平面垂直
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(
241         v.a,v.b,u.a,u.b,u.
242         c);
243     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(
244         v.c,v.a,u.a,u.b,u.
245         c);
246     return ret;
247 }
248 line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
249     line3 ret;
250     ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v1,v2,u1,
251         u2,u3);
252     ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v3,v1,u1,
253         u2,u3);
254     return ret;
255 }
256 //点到直线距离
257 double ptoline(point3 p,line3 l){
258     return vlen(Cross(subt(p,l.a),subt(l.b,l.a)))/distance(l.a,l.b);
259 }
260 double ptoline(point3 p,point3 l1,point3 l2){
261     return vlen(Cross(subt(p,l1),subt(l2,l1)))/distance(l1,l2);
262 }
263 //点到平面距离
264 double ptoplane(point3 p,plane3 s){
265     return fabs(Dot(pvec(s),subt(p,s.a)))/vlen(pvec(s));
266 }
267 double ptoplane(point3 p,point3 s1,point3 s2,point3 s3){
268     return fabs(Dot(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,s3));
269 }
270 //直线到直线距离
271 double linetoline(line3 u,line3 v){
272     point3 n=Cross(subt(u.a,u.b),subt(v.a,v.b));
273     return fabs(Dot(subt(u.a,v.a),n))/vlen(n);
274 }
275 double linetoline(point3 u1,point3 u2,point3 v1,point3 v2){
276     point3 n=Cross(subt(u1,u2),subt(v1,v2));
277     return fabs(Dot(subt(u1,v1),n))/vlen(n);
278 }
279 //两直线夹角 cos 值
280 double angle_cos(line3 u,line3 v){
281     return Dot(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/vlen(subt(v.a,v.b));
282 }

```



```

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 }
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,
299     double z2, double r2) {
300     double sum = 4.00 / 3.00 * PI * r1 * r1 * r1 + 4.00 / 3.00 * PI * r2 * r2 * r2;
301     double ans = 0;
302     double dis = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) + (z1 - z2) * (z1 -
303         z2)); //球心距离
304     if (dis >= r1 + r2) //没有交到的地方
305     {
306         ans = 0;
307     } else if (dis + r1 <= r2)//重合
308     {
309         ans = (4.00 / 3.00) * PI * r1 * r1 * r1;
310     } else if (dis + r2 <= r1) {
311         ans = (4.00 / 3.00) * PI * r2 * r2 * r2;
312     } else //相交
313     {
314         double cal = (r1 * r1 + dis * dis - r2 * r2) / (2.00 * dis * r1);
315         double h = r1 * (1 - cal);
316         ans += (1.00 / 3.00) * PI * (3.00 * r1 - h) * h * h;
317         cal = (r2 * r2 + dis * dis - r1 * r1) / (2.00 * dis * r2);
318         h = r2 * (1.00 - cal);
319         ans += (1.00 / 3.00) * PI * (3.00 * r2 - h) * h * h;
320     }
321     return ans;
322 }

```

#### 4.34 多项式 $\ln_{exp_{pow}}$ 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
26 inline void MTT(ll *x, ll *y, ll *z, int len) ;
27
28 void Get_Inv(ll *f, ll *g, int n) ;
29
30 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; }
31
32 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; }
33
34 void Get_Ln(ll *f, ll *g, int n) ;
35
36 void Get_Exp(ll *f, ll *g, int n) ;
37
38 void Get_Pow(ll *f, ll *g, int n, ll k) ;
39
40 void Get_Sqrt(ll *f, ll *g, int n) {
41     static ll a[N];
42     Get_Ln(f, a, n);
43     for(int i = 0; i < n; i++) a[i] = a[i] * inv2 % mod;
44     Get_Exp(a, g, n);
45     int len = getLen(n);
46     for(int i = n; i < len; i++) g[i] = 0;
47     for(int i = 0; i < len; i++) a[i] = 0;
48 }

```

#### 4.35 二次剩余处理边界不为 1

```

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 = 1e5 + 10;
7
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 *
15         rhs.y + y * rhs.x); }
16     Complex conj() { return Complex(x, -y); }
17 } w[N];
18 ll mod, inv2;
19 int tr[N];
20 ll F[N], G[N];
21
22 ll quick_pow(ll a, ll b) ;
23
24 typedef struct{
25     ll x, y; // 把求出来的w作为虚部, 则为a + bw
26 }num;
27
28 num num_mul(num a, num b, ll w, ll p) { // 复数乘法
29     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;
87     for(int i = 0; i < len; i++) c[i] = invg[i] = 0;
88 }
89
90 int main() {
91     inv2 = quick_pow(2, mod - 2);
92     int n;
93     cin >> n;
94     for(int i = 0; i < n; i++) cin >> F[i];
95     Get_Sqrt(F, G, n);
96     for(int i = 0; i < n; i++) cout << G[i] << " ";
97 }

```

#### 4.36 x 不连续、暴力插值

```

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 ll mod;
9 ll X[N], Y[N];
10
11 ll quick_pow(ll a, ll b) ;
12
13 ll Lagrange(ll *x, ll *y, int n, int k) {
14     ll ans = 0;
15     for(int i = 0; i < n; i++) {
16         ll s1 = 1, s2 = 1;
17         for(int j = 0; j < n; j++) {
18             if(i == j) continue;
19             s1 = s1 * (k - x[j] + mod) % mod;
20             s2 = s2 * (x[i] - x[j] + mod) % mod;
21         }
22         ans = (ans + 1ll * y[i] * s1 % mod * quick_pow(s2, mod - 2) % mod) % mod;
23     }
24     return ans;
25 }
26
27 int main() {

```

```

28     int n, k;
29     cin >> n >> k;
30     for(int i = 0; i < n; i++) cin >> X[i] >> Y[i];
31     cout << Lagrange(X, Y, n, k) << endl;
32 }

```

#### 4.37 x 连续、前缀优化

```

1
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 const int N = 1e5 + 10;
6
7 ll mod;
8 ll F[N];
9 ll pre[N], suf[N];
10 ll fac[N], invf[N];
11
12
13 ll quick_pow(ll a, ll b) ;
14
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 -
31             i] % mod * f[i] % mod + mod) % mod;
32     }
33     return f[k] = ans;
34 }
35
36 int main() {
37     init();
38     int n, k;
39     cin >> n >> k;
40     for(int i = 0; i <= n; i++) cin >> F[i];
41     cout << Lagrange(F, k, n) << endl;
42 }

```

#### 4.38 FFT 加速带有通配符字符串匹配

```

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

```

```

4
5 //  $p[x] = \sum_{i=0}^{m-1} A[i]^3 * B[x-m+i+1] + \sum_{i=0}^{m-1} A[i] * B[x-m+i+1]^3 -$ 
    $2 * \sum_{i=0}^{m-1} A[i]^2 * B[x-m+i+1]^2$ 
6
7 const int N = 1e6 + 1e5;
8
9 ll qpow(ll a, ll b, ll mod) {
10     ll ans = 1;
11     while(b) {
12         if(b & 1) ans = ans * a % mod;
13         a = a * a % mod;
14         b >>= 1;
15     }
16     return ans % mod;
17 }
18
19 const ll G = 3;
20 const ll invG = qpow(G, mod - 2, mod);
21 int tr[N];
22
23 void NTT(ll *A, int len, int type) {
24     for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
25     for (int i = 2; i <= len; i <= 1) {
26         int mid = i / 2;
27         ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
28         for (int k = 0; k < len; k += i) {
29             ll w = 1;
30             for (int l = k; l < k + mid; l++) {
31                 ll t = w * A[l + mid] % mod;
32                 A[l + mid] = (A[l] - t + mod) % mod;
33                 A[l] = (A[l] + t) % mod;
34                 w = w * Wn % mod;
35             }
36         }
37     }
38     if (type == -1) {
39         ll invn = qpow(len, mod - 2, mod);
40         for (int i = 0; i < len; i++)
41             A[i] = A[i] * invn % mod;
42     }
43 }
44
45 void mul(ll *a, ll *b, int n) {
46     int len = 1; while (len <= n) len <<= 1;
47     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);
51 }
52
53 ll a1[N], a2[N], a3[N], b1[N], b2[N], b3[N];
54
55 void solve() {
56     int m, n; cin >> m >> n;
57     string s, t; cin >> t >> s;
58     for(int i = 0; i < m; i++) {
59         if(t[i] == '*') continue;
60         int temp = t[i] - 'a' + 1;
61         a1[i] = temp;

```

```

62     a2[i] = temp * temp;
63     a3[i] = temp * temp * temp;
64 }
65 for(int i = 0; i < n; i++) {
66     if(s[i] == '*') continue;
67     int temp = s[i] - 'a' + 1;
68     b1[i] = temp;
69     b2[i] = temp * temp;
70     b3[i] = temp * temp * temp;
71 }
72 reverse(a1, a1 + m);
73 reverse(a2, a2 + m);
74 reverse(a3, a3 + m);
75 mul(a1, b3, n + m);
76 mul(a2, b2, n + m);
77 mul(a3, b1, n + m);
78 vector<int> ans;
79 for(int x = m - 1; x < n; x++) {
80     ll res = a1[x] + a3[x] - a2[x] * 2;
81     if(!res) ans.push_back(x - m + 2);
82 }
83 cout << ans.size() << endl;
84 for(int i = 0; i < ans.size(); i++) cout << ans[i] << (i == ans.size() - 1 ? endl :
85 " ");
86 }

```

#### 4.39 FFT 加速带有通配符字符串匹配 2

```

1 void solve() {
2     int o; cin >> o; while(o --) {
3         int n, m; cin >> n >> m;
4         vector<int> ans(n + m), res(n + m);
5         string s, t; cin >> s >> t;
6         reverse(t.begin(), t.end());
7         Polynomial A(s.size() + 10), B(t.size() + 10), C;
8
9         for(int cch = 0; cch <= 9; ++ cch) {
10             char now = cch <= 9? char(cch + '0'): '*';
11             for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] =
12             0;
13             for (int i = 0; i < s.size(); ++i) A[i] = s[i] == now;
14             for (int i = 0; i < t.size(); ++i) B[i] = t[i] == now;
15             C = A * B;
16             for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];
17         }
18         // for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
19         for(int i = 0; i < s.size(); ++ i) A[i] = s[i] == '*';
20         for(int i = 0; i < t.size(); ++ i) B[i] = 1;
21         C = A * B;
22         for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];
23
24         // for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
25         for(int i = 0; i < s.size(); ++ i) A[i] = 1;
26         for(int i = 0; i < t.size(); ++ i) B[i] = t[i] == '*';
27         C = A * B;
28         for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];
29     }
30 }

```

```

30
31 //      for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
32      for(int i = 0; i < s.size(); ++ i) A[i] = s[i] == '*';
33      for(int i = 0; i < t.size(); ++ i) B[i] = t[i] == '*';
34      C = A * B;
35      for(int i = m - 1; i < n; ++ i) ans[i - m + 1] -= C[i];
36
37      for(int i = 0; i <= n - m; ++ i) res[m - ans[i]] ++;
38      for(int i = 1; i <= m; ++ i) res[i] += res[i - 1];
39      for(int i = 0; i <= m; ++ i) cout << res[i] << '\n';
40
41  }
42 }

```

#### 4.40 FFT 加速朴素字符串匹配

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 4e5 + 10;
4
5  //  $P[x] = \sum_{i=0}^{m-1} A[i] + \sum_{i=0}^{m-1} B[x - m + i + 1] - 2 * \sum_{i=0}^{m-1} A[i] * B[x - m + i + 1]$ 
6
7  // reverse(a)
8
9  // 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
10
11 ll qpow(ll a, ll b, ll mod) ;
12
13 const ll mod = 998244353;
14 const ll G = 3;
15 const ll invG = qpow(G, mod - 2, mod);
16 int tr[N];
17
18 void NTT(ll *A, int len, int type) {
19     for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
20     for (int i = 2; i <= len; i <= 1) {
21         int mid = i / 2;
22         ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
23         for (int k = 0; k < len; k += i) {
24             ll w = 1;
25             for (int l = k; l < k + mid; l++) {
26                 ll t = w * A[l + mid] % mod;
27                 A[l + mid] = (A[l] - t + mod) % mod;
28                 A[l] = (A[l] + t) % mod;
29                 w = w * Wn % mod;
30             }
31         }
32     }
33     if (type == -1) {
34         ll invn = qpow(len, mod - 2, mod);
35         for (int i = 0; i < len; i++)
36             A[i] = A[i] * invn % mod;
37     }
38 }
39
40 void mul(ll *a, ll *b, int n) {
41     int len = 1; while (len <= n) len <= 1;

```



```

42     for (int i = 0; i < len; i++) tr[i] = (tr[i] >> 1) >> 1) | (i & 1 ? len >> 1 : 0);
43     NTT(a, len, 1), NTT(b, len, 1);
44     for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
45     NTT(a, len, -1);
46 }
47
48 ll a[N], b[N];
49
50 void solve() {
51     string s, t; cin >> s >> t;
52     int n = s.length(), m = t.length();
53     for(int i = 0; i < n; i++) a[i] = s[i] - 'a' + 1;
54     for(int i = 0; i < m; i++) b[i] = t[i] - 'a' + 1;
55     reverse(b, b + m);
56     mul(a, b, n + m - 2);
57     double P = 0;
58     for(int i = 0; i < m; i++) {
59         P += (t[i] - 'a' + 1) * (t[i] - 'a' + 1);
60     }
61     vector<int> f(n + 1);
62     for(int i = 1; i < n; i++) {
63         f[i] = f[i - 1] + (s[i] - 'a' + 1) * (s[i] - 'a' + 1);
64     }
65     for(int x = m - 1; x < n; x++) {
66         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 }

```

#### 4.41 $CDQ_{FFT}$

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define gcd(a,b) __gcd(a,b)
4  #define lcm(a,b) (1ll * a * b / gcd(a, b))
5  #define Polynomial vector<int>
6  #define Inv(x) quick_pow(x, mod - 2)
7  #define DEBUG(x, y) cout << x << ": " << y << '\n';
8  using ld = long double;
9  using ll = long long;
10 using ull = unsigned long long;
11 const ll mod = 100003;
12 const ld pi = acos(-1.0);
13 struct Complex {
14     ld r, i;
15     Complex(ld _r = 0, ld _i = 0) : r(_r), i(_i) {}
16 };
17
18 Complex operator + (const Complex &a, const Complex &b) {
19     return Complex(a.r + b.r, a.i + b.i);
20 }
21 Complex operator - (const Complex &a, const Complex &b) {
22     return Complex(a.r - b.r, a.i - b.i);
23 }
24 Complex operator * (const Complex &a, const Complex &b) {
25     return Complex(a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);

```

```

26 }
27 Complex operator / (const Complex &a, const Complex &b) {
28     return Complex((a.r * b.r + a.i * b.i) / (b.r * b.r + b.i * b.i), (a.i * b.r - a.r
    * b.i) / (b.r * b.r + b.i * b.i));
29 }
30
31 int R[int(1e6 + 10)];
32 Complex x[int(1e6 + 10)], y[int(1e6 + 10)];
33
34 void get_R(int lim) {
35     for (int i = 0; i < lim; i++) {
36         R[i] = (i & 1) * (lim >> 1) + (R[i >> 1] >> 1);
37     }
38 }
39
40 void FFT(Complex *f, int lim, int rev) {
41     for (int i = 0; i < lim; i++) {
42         if (i < R[i]) swap(f[i], f[R[i]]);
43     }
44     for (int mid = 1; mid < lim; mid <= 1) {
45         Complex wn = Complex(cos(pi / mid), rev * sin(pi / mid));
46         for (int len = mid << 1, cur = 0; cur < lim; cur += len) {
47             Complex w = Complex(1, 0);
48             for (int k = 0; k < mid; k++, w = w * wn) {
49                 Complex x = f[cur + k], y = w * f[cur + mid + k];
50                 f[cur + k] = x + y, f[cur + mid + k] = x - y;
51             }
52         }
53     }
54     if (rev == -1) {
55         for (int i = 0; i < lim; i++) {
56             f[i].r /= lim;
57         }
58     }
59 }
60
61 int s[int(1e6 + 10)];
62 vector<int> ans[int(1e6 + 10)];
63
64 int quick_pow(int ans, int p, int res = 1) {
65     for (; p >= 1, ans = 1ll * ans * ans % mod)
66         if (p & 1) res = 1ll * res * ans % mod;
67     return res % mod;
68 }
69 int fac[int(1e6 + 10)] = {1}, ifac[int(1e6 + 10)] = {1};
70 void init(int n) {
71     for (int i = 1; i <= n; ++i)
72         fac[i] = 1ll * fac[i - 1] * i % mod,
73         ifac[i] = Inv(fac[i]);
74 }
75 ll C(int n, int m) {
76     if (n < 0 || m < 0 || m > n) return 0;
77     if (m == 0 || m == n) return 1;
78     return 1ll * fac[n] * ifac[m] % mod * ifac[n - m] % mod;
79 }
80 ll F(ll a, ll n, ll k) {
81     return (1ll * ans[1][k] - C(n, k) + mod) % mod * Inv(a - 1) % mod;
82 }
83 void CDQ_FFT(int rt, int l, int r) {

```

```

84     if (l == r) {
85         ans[rt].push_back(1);
86         ans[rt].push_back(s[l]);
87         return ;
88     }
89     int mid = l + r >> 1;
90     CDQ_FFT(rt << 1, l, mid);
91     CDQ_FFT(rt << 1 | 1, mid + 1, r);
92     int len1 = mid - l + 1, len2 = r - mid;
93     for (int i = 0; i <= len1; i++) x[i] = Complex(ans[rt << 1][i], 0);
94     for (int i = 0; i <= len2; i++) y[i] = Complex(ans[rt << 1 | 1][i], 0);
95     int lim = 1; while (lim <= r - l + 1) lim <= 1;
96     get_R(lim);
97     FFT(x, lim, 1);
98     FFT(y, lim, 1);
99     for (int i = 0; i < lim; i++) x[i] = x[i] * y[i];
100    FFT(x, lim, -1);
101    for (int i = 0; i <= r - l + 1; i++) ans[rt].push_back((x[i].r + 0.5) % mod);
102    for (int i = 0; i < lim; i++) x[i] = y[i] = Complex(0, 0);
103 }
104
105 inline void solve() {
106     init(int(1e6));
107     int n, a, q; cin >> n >> a >> q;
108     for(int i = 1; i <= n; ++ i) {
109         cin >> s[i]; s[i] = quick_pow(a, s[i]);
110     }
111     CDQ_FFT(1, 1, n);
112     while(q --) {
113         int k; cin >> k;
114         cout << F(a, n, k) << '\n';
115     }
116 }

```

#### 4.42 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 *
10         rhs.b + b * rhs.a); }
11     Complex operator + (const Complex &rhs) { return Complex(a + rhs.a, b + rhs.b); }
12     Complex operator - (const Complex &rhs) { return Complex(a - rhs.a, b - rhs.b); }
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(Complex *a, Complex *b, int n) {
34     int len = 1; while (len <= n) len <<= 1;
35     for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
36     FFT(a, len, 1), FFT(b, len, 1);
37     for (int i = 0; i < len; i++) a[i] = a[i] * b[i];
38     FFT(a, len, -1);
39     for (int i = 0; i < len; i++) a[i].a /= len;
40 }
41
42 Complex a[N], b[N];
43
44 void solve() {
45     int n, m;
46     scanf("%d%d", &n, &m);
47     for (int i = 0; i <= n; i++) scanf("%lf", &a[i].a);
48     for (int i = 0; i <= m; i++) scanf("%lf", &b[i].a);
49
50     mul(a, b, n + m);
51     for (int i = 0; i <= n + m; i++)
52         printf("%d ", (int)(a[i].a + 0.5));
53 }

```

#### 4.43 FWT

```

1  inline void OR(ll *f, int n, int x = 1) {
2      for (int o = 2; o <= n; o <<= 1) {
3          for (int i = 0, k = o >> 1; i < n; i += o) {
4              for (int j = 0; j < k; ++j) {
5                  f[i + j + k] = (f[i + j + k] + f[i + j] * x % mod + (x == 1 ? 0 : mod))
6                      % mod;
7              }
8          }
9      }
10 inline void AND(ll *f, int n, int x = 1) {
11     for (int o = 2; o <= n; o <<= 1) {
12         for (int i = 0, k = o >> 1; i < n; i += o) {
13             for (int j = 0; j < k; ++j) {
14                 f[i + j] = (f[i + j] + f[i + j + k] * x + (x == 1 ? 0 : mod)) % mod;
15             }
16         }
17     }
18 }
19 inline void XOR(ll *f, int n, int x = 1) {
20     for (int o = 2; o <= n; o <<= 1) {
21         for (int i = 0, k = o >> 1; i < n; i += o) {
22             for (int j = 0; j < k; ++j) {

```

```

23         f[i + j] += f[i + j + k],
24         f[i + j + k] = f[i + j] - f[i + j + k] - f[i + j + k],
25         f[i + j] *= x, f[i + j + k] *= x;
26         f[i + j] %= mod;
27         f[i + j + k] %= mod;
28         while (f[i + j] < 0) f[i + j] += mod;
29         while (f[i + j + k] < 0) f[i + j + k] += mod;
30     }
31 }
32 }
33 }
34 inline void OR(vector<ll> &f, int n, int x = 1) {
35     for (int o = 2; o <= n; o <= 1) {
36         for (int i = 0, k = o >> 1; i < n; i += o) {
37             for (int j = 0; j < k; ++j) {
38                 f[i + j + k] = (f[i + j + k] + f[i + j] * x % mod + (x == 1 ? 0 : mod))
39                 % mod;
40             }
41         }
42     }
43 inline void AND(vector<ll> &f, int n, int x = 1) {
44     for (int o = 2; o <= n; o <= 1) {
45         for (int i = 0, k = o >> 1; i < n; i += o) {
46             for (int j = 0; j < k; ++j) {
47                 f[i + j] = (f[i + j] + f[i + j + k] * x + (x == 1 ? 0 : mod)) % mod;
48             }
49         }
50     }
51 }
52 inline void XOR(vector<ll> &f, int n, int x = 1) {
53     for (int o = 2; o <= n; o <= 1) {
54         for (int i = 0, k = o >> 1; i < n; i += o) {
55             for (int j = 0; j < k; ++j) {
56                 ll X = f[i + j], Y = f[i + j + k];
57                 f[i + j] = (X + Y) % mod;
58                 f[i + j + k] = ((X - Y) % mod + mod) % mod;
59                 // mod is a prime
60                 if (x != 1) {
61                     f[i + j] = f[i + j] * inv2 % mod;
62                     f[i + j + k] = f[i + j + k] * inv2 % mod;
63                 }
64             }
65         }
66     }
67     // mod is not a prime, let mod = mod * (1 << m), n = 1 << m;
68     //     if (x != 1) for (int i = 0; i < n; ++i) {
69     //         f[i] /= n;
70     //     }
71 }

```

#### 4.44 NTT

```

1 Polynomial R;
2 inline int Binary_Rounding(const int &n, int len = 1) { // 0 0 0 0 0 0 0 0 0 0
3     f->NTT±任±, if
4     while (len < n) len <= 1;
5     return len;

```

```

5 }
6 //0 10 0 R0 0 0 養0 ±,±任f-0 0 y'0 NTO 0 j0 0 0 0 %Çµ0 0 ô'0 0 jf
7 inline int Prepare_Transformation(int n) {
8     int L = 0, len;
9     for (len = 1; len < n; len <= 1) L++;
10    R.clear();
11    R.resize(len);
12    for (int i = 0; i < len; ++i)
13        R[i] = (R[i >> 1] >> 1) | ((i & 1) << (L - 1));
14    return len;
15 }
16 inline void NTT(Polynomial &a, int f) {
17     int n = a.size();
18     for (int i = 0; i < n; ++i)
19         if (i < R[i]) swap(a[i], a[R[i]]);
20     for (int i = 1; i < n; i <= 1)
21         for (int j = 0, gn = quick_pow(mod_g, (mod - 1) / (i << 1)); j < n; j += (i <<
22             1))
23             for (int k = 0, g = 1, x, y; k < i; ++k, g = 1ll * g * gn % mod)
24                 x = a[j + k], y = 1ll * g * a[i + j + k] % mod,
25                 a[j + k] = (x + y) % mod, a[i + j + k] = (x - y + mod) % mod;
26     if (f == -1) {
27         reverse(a.begin() + 1, a.end());
28         int inv = Inv(n);
29         for (int i = 0; i < n; ++i) a[i] = 1ll * a[i] * inv % mod;
30     }
31 }
32 inline Polynomial operator+(const Polynomial &a, const int &b) {
33     int sizea = a.size();
34     Polynomial ret = a;
35     ret.resize(sizea);
36     for (int i = 0; i < sizea; ++i) ret[i] = (1ll * a[i] + b + mod) % mod;
37     return ret;
38 }
39 inline Polynomial operator-(const Polynomial &a, const int &b) {
40     int sizea = a.size();
41     Polynomial ret = a;
42     ret.resize(sizea);
43     for (int i = 0; i < sizea; ++i) ret[i] = (1ll * a[i] - b + mod) % mod;
44     return ret;
45 }
46 inline Polynomial operator*(const Polynomial &a, const int &b) {
47     int sizea = a.size();
48     Polynomial ret = a;
49     ret.resize(sizea);
50     for (int i = 0; i < sizea; ++i) ret[i] = (1ll * a[i] * b % mod + mod) % mod;
51     return ret;
52 }
53 inline Polynomial operator+(const Polynomial &a, const Polynomial &b) {
54     int sizea = a.size(), sizeb = b.size(), size = max(sizea, sizeb);
55     Polynomial ret = a;
56     ret.resize(size);
57     for (int i = 0; i < sizeb; ++i) ret[i] = (1ll * ret[i] + b[i]) % mod;
58     return ret;
59 }
60 inline Polynomial operator-(const Polynomial &a, const Polynomial &b) {
61     int sizea = a.size(), sizeb = b.size(), size = max(sizea, sizeb);
62     Polynomial ret = a;
63     ret.resize(size);

```

```

63     for (int i = 0; i < sizeb; ++i) ret[i] = (1ll * ret[i] - b[i] + mod) % mod;
64     return ret;
65 }
66 inline Polynomial Inverse(const Polynomial &a) {
67     Polynomial ret, inv_a;
68     ret.resize(1);
69     ret[0] = Inv(a[0]);
70     int ed = a.size();
71     for (int len = 2; len <= ed; len <= 1) {
72         int n = Prepare_Transformation(len << 1);
73         inv_a = a;
74         inv_a.resize(n);
75         ret.resize(n);
76         for (int i = len; i < n; ++i) inv_a[i] = 0;
77         NTT(inv_a, 1);
78         NTT(ret, 1);
79         for (int i = 0; i < n; ++i)
80             ret[i] = 1ll * (2ll - 1ll * inv_a[i] * ret[i] % mod + mod) % mod * ret[i] %
mod;
81         NTT(ret, -1);
82         for (int i = len; i < n; ++i) ret[i] = 0;
83     }
84     ret.resize(ed);
85     return ret;
86 }
87
88
89 inline Polynomial operator*(const Polynomial &a, const Polynomial &b) {
90     Polynomial lsa = a, lsb = b, ret;
91     int n = lsa.size(), m = lsb.size();
92     n = Prepare_Transformation(n + m);
93     lsa.resize(n);
94     lsb.resize(n);
95     ret.resize(n);
96     NTT(lsa, 1);
97     NTT(lsb, 1);
98     for (int i = 0; i < n; ++i) ret[i] = 1ll * lsa[i] * lsb[i] % mod;
99     NTT(ret, -1);
100    return ret;
101 }
102 inline Polynomial operator/(const Polynomial &a, const Polynomial &b) {
103     Polynomial ret = a, ls = b;
104     reverse(ret.begin(), ret.end());
105     reverse(ls.begin(), ls.end());
106     ls.resize(Binary_Rounding(a.size() + b.size()));
107     ls = Inverse(ls);
108     ls.resize(a.size() + b.size());
109     ret = ret * ls;
110     ret.resize(a.size() - b.size() + 1);
111     reverse(ret.begin(), ret.end());
112     return ret;
113 }
114 inline Polynomial operator%(const Polynomial &a, const Polynomial &b) {
115     Polynomial ret = a / b;
116     ret = ret * b;
117     ret.resize(a.size() + b.size());
118     ret = a - ret;
119     ret.resize(a.size() + b.size());
120     return ret;

```

```

121 }
122 inline Polynomial Derivation(const Polynomial &a) {
123     int size = a.size();
124     Polynomial ret;
125     ret.resize(size);
126     for (int i = 1; i < size; ++i) ret[i - 1] = 1ll * i * a[i] % mod;
127     ret[size - 1] = 0;
128     return ret;
129 }
130 inline Polynomial Integral(const Polynomial &a) {
131     int size = a.size();
132     Polynomial ret;
133     ret.resize(size);
134     for (int i = 1; i < size; ++i) ret[i] = 1ll * Inv(i) * a[i - 1] % mod;
135     ret[0] = 0;
136     return ret;
137 }
138 inline Polynomial Composition_Inverse(const Polynomial &a) {
139     int n = a.size();
140     Polynomial ret, Cinv = a, Pow;
141     Cinv.resize(n);
142     ret.resize(n);
143     Pow.resize(n);
144     Pow[0] = 1;
145     for (int i = 0; i < n - 1; ++i) Cinv[i] = Cinv[i + 1];
146     Cinv[n - 1] = 0;
147     Cinv = Inverse(Cinv);
148     for (int i = 1; i < n; ++i) {
149         Pow = Pow * Cinv;
150         Pow.resize(n);
151         ret[i] = 1ll * Pow[i - 1] * Inv(i) % mod;
152     }
153     return ret;
154 }
155 inline Polynomial Logarithmic(const Polynomial &a) {
156     Polynomial ln_a = Derivation(a) * Inverse(a);
157     ln_a.resize(a.size());
158     return Integral(ln_a);
159 }
160 inline Polynomial Exponential(const Polynomial &a, int Constant = 1) {
161     Polynomial ret, D;
162     int ed = a.size();
163     ret.resize(1);
164     ret[0] = Constant;
165     for (int len = 2; len <= ed; len <= 1) {
166         D = Logarithmic(ret);
167         D.resize(len);
168         D[0] = (1ll * a[0] + 1ll - D[0] + mod) % mod;
169         for (int i = 1; i < len; ++i) D[i] = (1ll * a[i] - D[i] + mod) % mod;
170         int n = Prepare_Transformation(len << 1);
171         ret.resize(n);
172         D.resize(n);
173         NTT(ret, 1);
174         NTT(D, 1);
175         for (int i = 0; i < n; ++i) ret[i] = 1ll * ret[i] * D[i] % mod;
176         NTT(ret, -1);
177         for (int i = len; i < (len << 1); ++i) ret[i] = D[i] = 0;
178     }
179     ret.resize(ed);

```



```

180     return ret;
181 }

```

#### 4.45 高斯消元

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N=210;
4  int a[N][N]; //增广矩阵
5  int x[N]; //解集
6  int freeX[N]; //自由变元
7  // equ:方程个数 var:变量个数
8  int Gauss(int equ, int var){ //返回自由变元个数
9      /*初始化*/
10     for(int i = 0; i <= var; i++){
11         x[i] = 0;
12         freeX[i] = 0;
13     }
14
15     /*转换为阶梯阵*/
16     int col = 0; //当前处理的列
17     int num = 0; //自由变元的序号
18     int k; //当前处理的行
19     for(k = 0; k < equ && col < var; k++, col++){ //枚举当前处理的行
20         int maxr = k; //当前列绝对值最大的行
21         for(int i = k + 1; i < equ; i++){ //寻找当前列绝对值最大的行
22             if(a[i][col] > a[maxr][col]){
23                 maxr = i;
24                 swap(a[k], a[maxr]); //与第k行交换
25                 break;
26             }
27         }
28         if(a[k][col] == 0){ //col列第k行以下全是0, 处理当前行的下一列
29             freeX[num++] = col; //记录自由变元
30             k--;
31             continue;
32         }
33
34         for(int i = k + 1; i < equ; i++){
35             if(a[i][col] != 0){
36                 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
37                     a[i][j] ^= a[k][j];
38                 }
39             }
40         }
41     }
42
43     /*求解*/
44     //无解: 化简的增广阵中存在(0, 0, ..., a)这样的行, 且a!=0
45     for(int i = k; i < equ; i++){
46         if(a[i][col] != 0)
47             return -1;
48
49     //无穷解: 在var*(var+1)的增广阵中出现(0, 0, ..., 0)这样的行
50     if(k < var) //返回自由变元数
51         return var - k; //自由变元有var-k个
52
53     //唯一解: 在var*(var+1)的增广阵中形成严格的上三角阵

```

```

54     for(int i = var - 1; i >= 0; i--) { // 计算解集
55         x[i] = a[i][var];
56         for(int j = i + 1; j < var; j++)
57             x[i] ^= (a[i][j] && x[j]);
58     }
59     return 0;
60 }

```

#### 4.46 高斯消元 2

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define MAX_SIZE 1048
4  int Matrix[MAX_SIZE][MAX_SIZE];
5  int Free_x[MAX_SIZE]; // 0 0 0 0
6  int X_Ans[MAX_SIZE]; // %I
7  int Free_num=0; // 0 0 0 0 0 0
8  int gcd(int a,int b) { return b==0? a : gcd(b,a%b); }
9  int lcm(int a,int b) { return a/gcd(a,b)*b; }
10
11 int Gauss(int Row,int Column) { // 0 0 0 %0 0 0 0 0 Y=0 0
12     int row=0,col=0,max_r;
13     for(row=0;row<Row&&col<Column;row++,col++) {
14         max_r=row;
15         for(int i=row+1;i<Row;i++) // 0 x0 j0 e0 0 0 0 0
16             if(abs(Matrix[i][col])>abs(Matrix[max_r][col]))
17                 max_r=i;
18         if(Matrix[max_r][col]==0) { // 0 0 0 0 0 0f-μ0 0 0 0 0 0 0 f-μ0 %
19             row--;
20             Free_x[++Free_num]=col+1;
21             continue;
22         }
23         if(max_r!=row) // %«0 0 0 0 »μμ±j0 0
24             for(int i=col;i<Column+1;i++)
25                 swap(Matrix[row][i],Matrix[max_r][i]);
26         for(int i=row+1;i<Row;i++) { // 0 0 0
27             if(Matrix[i][col]!=0) {
28                 int LCM=lcm(abs(Matrix[i][col]),abs(Matrix[row][col]));
29                 int ta=LCM/abs(Matrix[i][col]);
30                 int tb=LCM/abs(Matrix[row][col]);
31                 if(Matrix[i][col]*Matrix[row][col]<0) // 0 0 0 0 r0 0 0
32                     tb=-tb;
33                 for(int j=col;j<Column+1;j++)
34                     Matrix[i][j]=Matrix[i][j]*ta-Matrix[row][j]*tb;
35             }
36         }
37     }
38     // row0 0 0 0 ±0 ' %0 0 0 0 0 0 0 0 0
39
40     for(int i=row;i<Row;i++) // 0 0 0
41         if(Matrix[i][Column]!=0)
42             return -1;
43
44     if(row<Column) // 0 0 0 0 0 0 .μ»0 0 0 0 0 0 0 0
45         return Column-row;
46
47
48     for(int i=Column-1;i>=0;i--) { // %h%0

```

```

49     int temp=Matrix[i][Column];
50     for(int j=i+1;j<Column;j++)
51         if(Matrix[i][j]!=0)
52             temp-=Matrix[i][j]*X_Ans[j];
53     X_Ans[i]=temp/Matrix[i][i];
54 }
55 return 0;
56 }

```

#### 4.47 高斯消元异或

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define MAX_SIZE 350
4  #define ll long long
5  ll Matrix[MAX_SIZE][MAX_SIZE];
6  ll Free_x[MAX_SIZE]; //0 0 0 0 0
7  ll X_Ans[MAX_SIZE]; //0
8  ll Free_num=0; //0 0 0 0 0
9
10 ll Guass(ll Row,ll Column) { //0 0 0 0 0 0 0 0 0
11     ll row=0,col=0,max_r;
12     for(row=0;row<Row&&col<Column;row++,col++) {
13         max_r=row;
14         for(ll i=row+1;i<Row;i++) //0 x 0 0 0 0 0
15             if(abs(Matrix[i][col])>abs(Matrix[max_r][col]))
16                 max_r=i;
17         if(Matrix[max_r][col]==0) { //0 0 0 0 0 0
18             row--;
19             Free_x[Free_num++]=col+1;
20             continue;
21         }
22         if(max_r!=row) //0 0 0 0 0
23             for(ll i=col;i<Column+1;i++)
24                 swap(Matrix[row][i],Matrix[max_r][i]);
25         for(ll i=row+1;i<Row;i++) { //0 0 0
26             if(Matrix[i][col]!=0) {
27                 for(ll j=col;j<Column+1;j++)
28                     Matrix[i][j]^=Matrix[row][j];
29             }
30         }
31     }
32     for(ll i=row;i<Row;i++) //0 0 0
33         if(Matrix[i][Column]!=0)
34             return -1;
35
36     if(row<Column) //0 0 0 0 0
37         return Column-row;
38
39     //0 0 0
40     for(ll i=Column-1;i>=0;i--) {
41         X_Ans[i]=Matrix[i][Column];
42         for(ll j=i+1;j<Column;j++)
43             X_Ans[i]^=(Matrix[i][j]&&X_Ans[j]);
44     }
45     return 0;
46 }

```

## 4.48 矩阵快速幂

```

1  /**% 0 0 0 0 0 0 0 0 ***/
2
3  struct Matrix {
4      static const int M = 2;
5      double mx[M][M];
6
7      Matrix() { memset(mx, 0, sizeof mx); }
8
9      void Out() {
10         for (auto &i : mx) {
11             for (auto j : i)
12                 cout << j << ' ';
13             cout << endl;
14         }
15     }
16
17     void Tranfer_E() {
18         memset(mx, 0, sizeof mx);
19         for (int i = 0; i < M; ++i) mx[i][i] = 1;
20     }
21
22     Matrix operator*(const struct Matrix a) const {
23         Matrix x;
24         memset(x.mx, 0, sizeof x.mx);
25         for (int i = 0; i < M; ++i)
26             for (int j = 0; j < M; ++j)
27                 for (int k = 0; k < M; ++k)
28                     x.mx[i][j] = (x.mx[i][j] + mx[i][k] * a.mx[k][j]);
29         return x;
30     }
31 };
32 Matrix mat_pow(Matrix mx, int p) {
33     Matrix res;
34     res.Tranfer_E();
35     for (; p; p >>= 1, mx = mx * mx)
36         if (p & 1) res = res * mx;
37     return res;
38 }

```

## 4.49 矩阵求逆

```

1  /**% 0 0 0 0 0 0 0 0 ***/
2  void Gauss_jordan() {
3      /**% 0 el'»»»&% 0 0 0 0 0 0 0 0 ***/
4      for(int i = 0, r; i < n; ++ i) { //0 0 0 !0 0 0 0 i0 0
5          r = i;
6          for(int j = i + 1; j < n; ++j)
7              if(fabs(a[j][i]) > fabs(a[r][i])) r = j;
8          if(fabs(a[r][i]) < eps) {
9              puts("No Solution");
10             return;
11         }
12         if(i!=r) swap(a[i],a[r]);
13
14         for(int k = 0; k < n; ++ k) {
15             //yh %' i0 0

```



```

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

```

#### 4.51 线性基

```

1 struct LinerBase {
2     static const int MAX_BIT = 61;
3     bool isZero; ll num[N], tmp[N];
4     //判断线性基中是否存在0
5     LinerBase() {

```

```

6      memset(num, 0, sizeof num);
7      memset(tmp, 0, sizeof tmp);
8  }
9  void insert(ll x) {
10     for (int i = MAX_BIT; i >= 0; i--) {
11         if (!(x & (1ll << i))) continue;
12         if (num[i]) {
13             x ^= num[i];
14         } else {
15             num[i] = x;
16             return;
17         }
18     }
19     isZero = true;
20 }
21 bool find(ll x) {
22     for (int i = MAX_BIT; i >= 0; i--) {
23         if (x & (1ll << i)) {
24             if (!num[i]) return false;
25             x ^= num[i];
26         }
27     }
28     return true;
29 }
30 int countNumber() {
31     int cnt = 0;
32     for (int i = 0; i <= MAX_BIT; i++) if (num[i] & (1ll << i)) cnt++;
33     return cnt;
34 }
35 void rebuild() {
36     int cnt = 0;
37     for (int i = MAX_BIT; i >= 0; i--)
38         for (int j = i-1; j >= 0; j--)
39             if (num[i] & (1ll << j)) num[i] ^= num[j];
40
41     for (int i = 0; i <= MAX_BIT; i++) {
42         if (num[i]) tmp[cnt++] = num[i];
43     }
44 }
45 ll qryKth(ll k) {
46     if (isZero) k--;
47     if (!k) return 0;
48     int cnt = countNumber(); ll ans = 0;
49     if (k >= (1ll << cnt)) return -1;
50     for (int i = 0; i < cnt; i++) {
51         if (k & (1ll << i)) ans ^= tmp[i];
52     }
53     return ans;
54 }
55 }lb;

```

#### 4.52 康托展开

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;

```

```

6
7 typedef long long ll;
8 const int mod = 1e9 + 7;
9 const int N = 1e5 + 10;
10
11 ll fac[N];
12 int a[N]; // 排列, 康托展开求解
13 int n;
14 ll x; // 逆康托展开求解
15
16 void Get_F() {
17     fac[0] = 1;
18     for(int i = 1; i < N; i++)
19         fac[i] = fac[i - 1] * i % mod;
20 }
21
22 ll CanTor() {
23     ll ans = 0;
24     for(int i = 1; i <= n; i++) {
25         ll smaller = 0;
26         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;
86     if(l == r) {
87         t[u].sum = 0;
88         return ;
89     }
90     if(k <= m) update(lc, l, m, k);
91     else update(rc, m + 1, r, k);
92     push_up(u);
93 }
94
95 ll query(int u, int l, int r, int ql, int qr) {
96     if(ql > qr) return 0;
97     if(ql == l && qr == r) {
98         return t[u].sum;
99     }
100     if(qr <= m) return query(lc, l, m, ql, qr) % mod;
101     else if(ql > m) return query(rc, m + 1, r, ql, qr) % mod;
102     else return (query(lc, l, m, ql, m) + query(rc, m + 1, r, m + 1, qr)) % mod;
103 }
104
105 void Get_F() {
106     fac[0] = 1;
107     for(int i = 1; i < N; i++)
108         fac[i] = fac[i - 1] * i % mod;
109 }
110
111 void solve()
112 {
113     Get_F();
114     cin >> n;
115     build(root, 1, n);
116     ll ans = 0;
117     for(int i = 1; i <= n; i++) {
118         cin >> a[i];
119         update(root, 1, n, a[i]);
120         ans = (ans + query(root, 1, n, 1, a[i] - 1) * fac[n - i]) % mod;
121     }
122     cout << (ans + 1) % mod << endl;
123 }

```

## 4.53 模数非质数的组合

```

1 // 模数非质数情况下的组合问题
2 // one way, use CRT merge ans
3 // https://ac.nowcoder.com/discuss/655940?type=101&order=0&pos=2&page=1&channel=-1&
  source_id=discuss_tag_nctrack
4 // another way
5 // https://ac.nowcoder.com/acm/contest/view-submission?submissionId=47754622
6
7 #include <bits/stdc++.h>
8
9 using namespace std;
10 typedef long long ll;
11 const int N = 1e6 + 10;
12
13 ll qpow(ll a, ll b, ll mod) {
14     ll res = 1;
15     while (b) {
16         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;
76     }
77     return ans;
78 }
79
80 ll r[20], m[20];
81
82 int main() {
83     #ifdef ACM_LOCAL
84         freopen("input.in", "r", stdin);
85         freopen("output.out", "w", stdout);
86     #endif
87     int n, p;
88     scanf("%d%d", &n, &p);
89     int num = 0;
90     for (int i = 2; i * i <= p; i++)
91         if (p % i == 0) {
92             int k = 0;
93             m[++num] = 1;
94             while (p % i == 0) p /= i, k++, m[num] *= i;
95             r[num] = calc(n, i, k);
96         }
97     if (p > 1) {
98         m[++num] = p;
99         r[num] = calc(n, p, 1);
100     }
101     printf("%lld\n", china(r, m, num));
102
103     return 0;
104 }

```

#### 4.54 普通型母函数

```

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

```

```

8
9 #include <bits/stdc++.h>
10 using namespace std;
11
12 typedef long long ll;
13
14 // 求解硬币等普通问题
15
16 const int N = 1e5 + 10;
17
18 int a[N]; // 权重为i的组合数, a[P]为答案
19 int b[N]; // 辅助数组
20 int P; // 需要被分解的数
21 int k; // 物品个数
22 int v[N]; // 每个物品的权重
23 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
24 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
25
26 // 模板一(标准)
27
28 void Calc1() {
29     memset(a, 0, sizeof(a));
30     a[0] = 1;
31
32     for(int i = 1; i <= k; i++) { // 枚举每个物品因子
33         memset(b, 0, sizeof(b));
34         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
35             for(int m = 0; m + j * v[i] <= P; m++) { // 循环a的每个项
36                 b[m + j * v[i]] += a[m]; // 把结果加到对应项里,有点dp的味道
37             }
38         }
39         memcpy(a, b, sizeof(b));
40     }
41 }
42
43 // 模板二 (数据量大的时候可以用, 快速)
44
45 void Calc2() {
46     memset(a, 0, sizeof(a));
47     a[0] = 1;
48     int last = 0;
49     for(int i = 1; i <= k; i++) {
50         int last2 = min(last + n2[i] * v[i], P); // 计算下一次的last
51         memset(b, 0, sizeof(int) * (last2 + 1)); // 只清空b[0..last2]
52         for(int j = n1[i]; j <= n2[i] && j * v[i] <= last2; j++) // last2
53             for(int m = 0; m <= last && m + j * v[i] <= last2; m++) // 一个是last, 一个是
last2
54                 b[m + j * v[i]] += a[m];
55         memcpy(a, b, sizeof(int) * (last2 + 1)); // b赋值给a, 只赋值0..last2
56         last = last2; // 更新last
57     }
58 }

```

#### 4.55 生成函数

```

1
2 // O(n*sqrt(n))

```

```

3
4 #include <bits/stdc++.h>
5
6 using namespace std;
7
8 typedef long long ll;
9
10 const int N = 1e5 + 5;
11 const ll mod = 1e9 + 7;
12 int f1[270], f2[270];
13
14 ll ans[N];
15
16 void init() {
17     for (int i = 1; ; i++) {
18         f1[i] = (3 * i * i - i) >> 1;
19         if (f1[i] > 100000) break;
20         f2[i] = (3 * i * i + i) >> 1;
21         if (f2[i] > 100000) break;
22     }
23     ans[0] = 1;
24     for (int i = 1; i <= 100000; i++) {
25         for (int j = 1; ; j++) {
26             if (f1[j] <= i) ans[i] += j & 1 ? ans[i-f1[j]] : -ans[i-f1[j]];
27             else break;
28             if (f2[j] <= i) ans[i] += j & 1 ? ans[i-f2[j]] : -ans[i-f2[j]];
29             else break;
30         }
31         ans[i] = (ans[i] % mod + mod) % mod;
32     }
33 }

```

#### 4.56 斯特林数

```

1 /**r  0 0 0 0 0 ***/
2 1
3 1 1
4 2 3 1
5 6 11 6 1
6 24 50 35 10 1
7 120 274 225 85 15 1
8 720 1764 1624 735 175 21 1
9 namespace STIRLING {
10     typedef long long ll;
11     const int N = 21;
12     const int mod = 1e9 + 7;
13     ll Stirling1[N][N], fac[N] = {1};
14     void get_stirling1() {
15         for(int i = 1; i < N; i++)
16             fac[i] = 1ll * fac[i - 1] * i % mod;
17         Stirling1[0][0] = 0;
18         Stirling1[1][1] = 1;
19         for(int i = 2; i < N; i++) {
20             for(int j = 1; j <= i; j++) {
21                 Stirling1[i][j] = (Stirling1[i - 1][j - 1] + (i - 1) * Stirling1[i -
22                 1][j] % mod) % mod;
23             }
24         }
25     }
26 }

```

```

24     }
25 1
26 1 1
27 1 3 1
28 1 7 6 1
29 1 15 25 10 1
30 1 31 90 65 15 1
31 1 63 301 350 140 21 1
32 1 127 966 1701 1050 266 28 1
33     ll Stirling2[N][N];
34     void get_stirling2() {
35         Stirling2[0][0] = 0;
36         Stirling2[1][1] = 1;
37         for(int i = 2; i < N; i++) {
38             for(int j = 1; j <= i; j++) {
39                 Stirling2[i][j] = (Stirling2[i - 1][j - 1] + j * Stirling2[i - 1][j] %
mod) % mod;
40             }
41         }
42     }
43 }

```

#### 4.57 五边形数

```

1  /**[] [] [] [] [] */
2  namespace WUBIANXING {
3  /*1,5,12,22,35...n^2+n(n-1)/2*/
4      const int N = 2e5 + 10;
5      const int mod = 1e5 + 10;
6      int f[N], p[N];
7      void init() {
8          for (int i = 1; i < N; i++) {
9              f[i + i - 1] = (1ll * i * (i + i + i - 1) >> 1) % mod;
10             f[i + i] = (1ll * i * (i + i + i + 1) >> 1) % mod;
11         }
12         for (int i = p[0] = 1; i < N; i++) {
13             p[i] = 0;
14             for (int j = 1, k = -1; i >= f[j]; j++) {
15                 if (j & 1) k *= -1;
16                 p[i] = (p[i] + 1ll * k * p[i - f[j]]) % mod;
17             }
18             p[i] = (p[i] + mod) % mod;
19         }
20     }
21 }

```

#### 4.58 指数型母函数

```

1
2  // 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  $k_1, k_2, \dots, k_n$  个，求
   从中选出 $m$ 件物品的排列数。
3
4  // 对 $n$ 个元素全排列，方案数为 $n!/(n_1!n_2!\dots n_k!)$ ，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 $G(x)=(1+x/1!+x^2/2!+\dots+x^{k_1}/k_1!)(1+x/1!+x^2/2!+\dots+x^{k_2}/k_2!)\dots(1+x/1!+x^{k_n}/n!+\dots+x^{k_n}/k_n!)$ 
5
6  // 化简得 $G(x)=a_0 + a_1*x+a_2*x^2/2!+\dots+a_p*x^p/p!$  ( $p = k_1+k_2+k_3+\dots$ )  $a_i$ 为选出 $i$ 个物品的排列
   方案数

```

```

7
8 // 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( $x^2/2! + x^4/4! + \dots + x^{ki}/ki!$ )
9
10 #include <bits/stdc++.h>
11 using namespace std;
12
13 typedef long double ld;
14
15 double num[15]; // 每种物品的数量, 第i个物品有num[i]个
16
17 double a[15], b[15];
18
19 double f[120]; // 阶乘
20
21 void fac()
22 {
23     f[0] = 1;
24     for(int i = 1; i <= 105; i++)
25         f[i] = f[i - 1] * i;
26 }
27
28 void Calc() {
29     int n, m;
30     cin >> n >> m;
31     for(int i = 1; i <= n; i++)
32         cin >> num[i];
33
34     memset(a, 0, sizeof(a));
35     memset(b, 0, sizeof(b));
36
37     for(int i = 0; i <= num[1]; i++) {
38         a[i] = 1.0 / f[i];
39     }
40
41     for(int i = 2; i <= n; i++) {
42         for(int j = 0; j <= m; j++) {
43             for(int k = 0; k <= num[i] && j + k <= m; k++) {
44                 b[j + k] += a[j] / f[k];
45             }
46         }
47
48         for(int j = 0; j <= m; j++) {
49             a[j] = b[j];
50             b[j] = 0;
51         }
52     }
53
54     cout << a[m] * f[m] << endl;
55 }

```

#### 4.59 组合数学基础

```

1 /** 组合数学基础 A 1 */
2 namespace COMB {
3     typedef long long ll;
4     const int mod = 1e9 + 7;
5     const int N = 5e5 + 10;
6     int fac[N];

```

```

7   void get_fac(int n, int i = 1) {
8       for (fac[0] = 1; i <= n; i++) fac[i] = 1ll * fac[i - 1] * i % mod;
9   }
10  ll quick_pow(ll ans, ll p, ll res = 1) {
11      ans %= mod, p %= mod - 1;
12      for (; p; p >>= 1, ans = 1ll * ans * ans % mod) {
13          if (p & 1) res = 1ll * res * ans % mod;
14      }
15      return res % mod;
16  }
17  ll inv(ll ans) {
18      return quick_pow(ans, mod - 2);
19  }
20  ll C(ll n, ll m) {
21      if (m == 0 || n == m) return 1;
22      if (n < m || n < 0 || m < 0) return 0;
23      return 1ll * fac[n] % mod * inv(1ll * fac[m] * fac[n - m] % mod) % mod;
24  }
25  ll Lucas(ll n, ll m) {
26      if (m == 0) return 1;
27      return 1ll * (C(n % mod, m % mod) * Lucas(n / mod, m / mod)) % mod;
28  }
29  }

```

#### 4.60 组合数

```

1  ll f[N];
2  ll ksm(ll a, ll b) {
3      ll res = 1, base = a;
4      while (b) {
5          if (b & 1) res = res * base % MOD;
6          base = base * base % MOD;
7          b >>= 1;
8      }
9      return res;
10 }
11 void init() {
12     f[0] = 1;
13     for (ll i = 1; i < N; i++) f[i] = f[i-1] * i % MOD;
14 }
15 ll C(ll a, ll b) {
16     if (a < 0 || b < 0 || b > a) return 0;
17     return f[a] * ksm(f[a-b], MOD-2) % MOD * ksm(f[b], MOD-2) % MOD;
18 }
19 -----//卢卡斯定理
20 ll f[N];
21 ll ksm(ll a, ll b, ll mm) {
22     ll res = 1, base = a;
23     while (b) {
24         if (b & 1) res = res * base % mm;
25         base = base * base % mm;
26         b >>= 1;
27     }
28     return res;
29 }
30 void init(ll mm) {
31     f[0] = 1;
32     for (ll i = 1; i < N; i++) f[i] = f[i-1] * i % mm;

```



```
33 }
34 ll C(ll a, ll b, ll mm) {
35     if (a < 0 || b < 0 || b > a) return 0;
36     return f[a] * ksm(f[a-b], mm-2, mm) % mm * ksm(f[b], mm-2, mm) % mm;
37 }
38
39 ll lucas(ll a, ll b, ll mm) {
40     if (b == 0) return 1;
41     return C(a % mm, b % mm, mm) * lucas(a / mm, b / mm, mm) % mm;
42 }
```

#### 4.61 卡特兰数

```
1 namespace KART {
2     /*1,1,2,5,14,42,132,429,1430,4862,16796...*/
3     /*h(n) = h(n-1)(4n+2)/(n+1) = C(2n,n)/(n+1) = C(2n,n)-C(2n,n-1) (n=0,1,2,...)*/
4     typedef long long ll;
5     const int N = 1e5 + 10;
6     const int mod = 1e9 + 7;
7     int cart[N], inv[N];
8     void get_inv() {
9         inv[0] = inv[1] = 1;
10        for(int i = 2; i < N; ++i)
11            inv[i] = 1ll * (mod - mod / i) * inv[mod % i] % mod;
12    }
13    void get_cart() {
14        cart[0] = 1;
15        for(int i = 1; i < N; ++i)
16            cart[i] = 1ll * cart[i - 1] * ((4 * i % mod - 2 + mod) % mod) % mod * inv[i
+ 1] % mod;
17    }
18 }
```

## 5 随机化算法

```

1  int n, m, r;
2  circle c[20];
3  Point p[1010], ansp;
4  double ans, now, nxt;
5
6  double Rand() { return (double)rand() / RAND_MAX; }
7  double calc(Point P) {
8      double minn = r;
9      rep(i, 1, n) minn = min(minn, P.distance(c[i].p) - c[i].r);
10     double res = 0;
11     rep(i, 1, m) if(P.distance(p[i]) <= minn) res ++;
12     return res;
13 }
14
15 void SA() {
16     double x = ansp.x, y = ansp.y;
17     double t = 3722;
18     while(t > 9e-16) {
19         double X = x + ((rand() * 2) - RAND_MAX) * t;
20         double Y = y + ((rand() * 2) - RAND_MAX) * t;
21         now = calc(Point(X, Y));
22         double Delta = now - ans;
23         if(Delta > 0) {
24             ansp = Point(X, Y);
25             x = X, y = Y;
26             ans=now;
27         } else if(exp(-Delta/t) * RAND_MAX < rand()) x = X, y = Y;
28         t *= 0.997577;
29     }
30 }
31 inline void solve() {
32     srand(time(0));
33     cin >> n >> m >> r;
34     rep(i, 1, n) c[i].input();
35     rep(i, 1, m) {
36         p[i].input();
37         ansp = ansp + p[i];
38     }
39     ansp = ansp / m;
40     ans = calc(ansp);
41     while ((double)clock()/CLOCKS_PER_SEC < 0.5) SA();
42     cout << ans << '\n';
43 }

```

### 5.1 SA2

```

1  circle c[4];
2  Point p;
3
4  double dx[4]={0, 1, 0, -1},
5         dy[4]={1, 0, -1, 0};
6
7  double calc(Point a) {
8      double dif = 0, d[3];
9      for(int i = 0; i < 3; ++ i)
10         d[i] = a.distance(c[i].p) / c[i].r;

```

```
11     for(int i = 0; i < 3; ++ i)
12         for(int j = i + 1; j < 3; ++ j)
13             dif += (d[i] - d[j]) * (d[i] - d[j]);
14     return dif;
15 }
16 void solve() {
17     double dis = 1.0, now, t;
18     for(int i = 0; i < 3; ++ i) {
19         c[i].input();
20         p = p + c[i].p;
21     }
22     p = p / 3.0;
23     while(dis > eps) {
24         now = calc(p);
25         int best = -1;
26         for(int i = 0; i < 4; ++ i) {
27             t = calc(p + Point(dis * dx[i], dis * dy[i]));
28             if(now > t) {
29                 now = t;
30                 best = i;
31             }
32         }
33         if(best == -1) dis = 0.7 * dis;
34         else p = p + Point(dis * dx[best], dis * dy[best]);
35     }
36 }
37 if(calc(p) < eps) p.output();
38 }
```

## 6 图论

```

1  /*
2
3       $x = w/l \rightarrow w - l * x = 0$ 
4       $f(x) = w - l * x;$ 
5      将边权更改为  $w - l * x$  来求生成树
6
7      因为  $f(x)$  是个单调递减函数, 随着  $x$  的增大而减少
8      对于任意一个生成树如果
9           $f(x) > 0$       则  $l$  需要增大
10          $f(x) < 0$       否则  $l$  需要减小
11
12     若要满足  $f(x) == 0$  恒成立
13     1. 若要  $x$  取最大值, 则不能存在任意一个生成树  $f(x) > 0$ , 否则  $x$  还能继续增大, 即任意生成树  $f(x) \leq 0$ 
14        若存在一个生成树  $f(x) > 0$ , 则那个生成树的比率一定大于当前  $x, w/l > x \rightarrow w - l * x > 0$ 
15     2. 若要  $x$  取最小值, 则不能存在任意一个生成树  $f(x) < 0$ , 否则  $x$  还能继续减小, 即任意生成树  $f(x) \geq 0$ 
16        若存在一个生成树  $f(x) < 0$ , 则那个生成树的比率一定小于当前  $x, w/l < x \rightarrow w - l * x < 0$ 
17
18     若要满足  $f(x) > 0$  恒成立, 则最小生成树  $> 0$ 
19     若要满足  $f(x) < 0$  恒成立, 则最大生成树  $< 0$ 
20
21     此题目求解最小的  $x$  值, 也就是检查是否所有的生成树  $f(x) \geq 0$ , 即最小生成树  $\geq 0$ 
22
23     如果最小生成树大于  $0$ , 所有的生成树都满足  $f(x) > 0$ , 尝试增加  $x$  得到  $f(x) = 0$ 
24     否则, 有生成树不满足这个条件, 那么  $x$  一定要减少来使所有  $f(x) \geq 0$ 
25
26  */

```

### 6.1 DAG 图上建支配树

```

1  支配树概念: 比如说我们要到  $a$  节点必须经过  $b$  节点, 那么  $b$  节点就是  $a$  的支配点, 在一棵支配树下, 所有节点都是子树节
   点的支配点。
2  vector<int> G1[N], G2[N];
3  int in[N], n, m, q[N], idx;
4  int f[N][25], dep[N];
5  void tupo() {
6      idx = 0; queue<int> que;
7      for (int i = 1; i <= n; i++) {
8          if (!in[i]) que.push(i);
9          if (!G1[i].size()) G1[i].push_back(0), in[0]++;
10     }
11     while (que.size()) {
12         int now = que.front();
13         que.pop();
14         q[++idx] = now;
15         for (auto v : G1[now]) {
16             in[v]--;
17             if (!in[v]) que.push(v);
18         }
19     }
20 }
21 int lca(int x, int y) {
22     if (dep[x] > dep[y]) swap(x, y);
23     for (int i = 20; i >= 0; i--) if (dep[f[y][i]] >= dep[x]) y = f[y][i];
24
25     if (x == y) return x;
26     for (int i = 20; i >= 0; i--) {

```

```

27         if (f[y][i] != f[x][i])
28             y = f[y][i], x = f[x][i];
29     }
30     return f[x][0];
31 }
32
33 void solve() {
34     int T; cin >> T; while (T--) {
35         cin >> n >> m;
36         for (int i = 0; i <= n; i++) G1[i].clear(), G2[i].clear(), dep[i] = 0;
37         for (int i = 1; i <= m; i++) {
38             int u, v; cin >> u >> v;
39             G1[u].push_back(v);
40             in[v]++;
41         }
42         tupo();
43         for (int i = n; i >= 1; i--) {
44             int now = q[i];
45             if (!G1[now].size()) continue;
46             int _lca = G1[now][0];
47             for (int j = 1; j < G1[now].size(); j++) _lca = lca(_lca, G1[now][j]);
48             f[now][0] = _lca;
49             dep[now] = dep[_lca] + 1;
50             for (int j = 1; j <= 20; j++) f[now][j] = f[f[now][j-1]][j-1];
51         }
52         int qry; cin >> qry; while (qry--) {
53             int x, y; cin >> x >> y;
54             cout << dep[x] + dep[y] - dep[lca(x, y)] << endl;
55         }
56     }
57 }

```

## 6.2 spfa 判负环 (01 分数规划)

```

1
2 //bfs版
3 #include <bits/stdc++.h>
4 using namespace std;
5 double eps = 1e-4;
6 const int N = 1e3 + 5;
7 const int M = 5e3 + 5;
8 struct edge {
9     int to, next;
10    double vi;
11 }e[M * 2];
12
13 int vis[N], n, cnt, m, h[N], times[N];
14 double dis[N];
15 int ve[N];
16
17 void add(int u, int v, double w) {
18     e[cnt].to = v;
19     e[cnt].vi = w;
20     e[cnt].next = h[u];
21     h[u] = cnt++;
22 }
23 bool spfa(double mid) {
24     queue<int> q;

```

```

25     for (int i = 1; i <= n; i++) {
26         q.push(i);
27         vis[i] = 1, dis[i] = 0, times[i] = 0;
28     }
29     while (q.size()) {
30         int u = q.front();
31         q.pop();
32         vis[u] = 0;
33         for (int i = h[u]; ~i; i = e[i].next) {
34             int v = e[i].to;
35             double va = mid * e[i].vi - (double)ve[u];
36             if (dis[v] > dis[u] + va) {
37                 dis[v] = dis[u] + va;
38                 times[v] = times[u] + 1;
39                 if (times[v] >= n) return true;
40                 if (!vis[v]) {
41                     q.push(v);
42                     vis[v] = 1;
43                 }
44             }
45         }
46     }
47     return false;
48 }
49
50 //dfs版
51 #include <bits/stdc++.h>
52 using namespace std;
53 double eps = 1e-9;
54 const int N = 3e3 + 5;
55 const int M = 1e4 + 5;
56 struct edge {
57     int to, next;
58     double vi;
59 }e[M << 1];
60
61 int vis[N], n, cnt, m, h[N], times[N], flag;
62 double dis[N];
63
64 void add(int u, int v, double w) {
65     e[cnt].to = v;
66     e[cnt].vi = w;
67     e[cnt].next = h[u];
68     h[u] = cnt++;
69 }
70 void spfa(int u, double mid) {
71     vis[u] = 1;
72     for (int i = h[u]; ~i; i = e[i].next) {
73         int v = e[i].to;
74         double tmp = dis[u] + e[i].vi - mid;
75         if (dis[v] > tmp) {
76             dis[v] = tmp;
77             if (vis[v]) {
78                 flag = 1;
79                 return;
80             }
81             spfa(v, mid);
82         }
83     }

```

```

84     vis[u] = 0;
85 }
86
87 bool check(double mid) {
88     memset(vis, 0, sizeof vis);
89     memset(times, 0, sizeof times);
90     for (int i = 1; i <= n; i++) dis[i] = 0;
91     for (int i = 1; i <= n; i++) {
92         flag = 0;
93         spfa(i, mid);
94         if (flag) return true;
95     }
96     return false;
97 }
98
99 int main() {
100     scanf("%d %d", &n, &m);
101     memset(h, -1, sizeof h);
102
103     for (int i = 1; i <= m; i++) {
104         int x, y;
105         double z;
106         scanf("%d %d %lf", &x, &y, &z);
107         add(x, y, z);
108     }
109
110     double l = -1e5, r = 1e5, mid;
111     while (r - l > eps) {
112         mid = (l + r) * 0.5;
113         if (check(mid)) r = mid;
114         else l = mid;
115     }
116     printf("%.8lf", mid);
117 }

```

### 6.3 边双连通分量 + e-dcc 缩点

```

1  const int N = 5e5 + 10, M = 5e5 + 10, INF = 0x3f3f3f3f;
2  const int MOD = 1e9 + 7;
3  int in[N];
4  int ans;
5  struct E_DCC {
6      int dfn[N], low[N], bridge[M], h1[N], h2[N], belong[N];
7      int cnt1, cnt2, idx, scc, num;
8      struct Edge {
9          int to, next;
10     } e1[N << 1], e2[N << 1];
11
12     void init(int n, int m) {
13         idx = cnt1 = cnt2 = 0;
14         for (int i = 0; i <= n; i++) h1[i] = h2[i] = -1;
15         for (int i = 0; i <= n; i++) belong[i] = low[i] = dfn[i] = 0;
16         for (int i = 0; i <= 2*m; i++) bridge[i] = 0;
17     }
18
19     void add(int u, int v) {
20         e1[cnt1].to = v;
21         e1[cnt1].next = h1[u];

```

```

22     h1[u] = cnt1++;
23 }
24
25 void add_cc(int u, int v) {
26     e2[cnt2].to = v;
27     e2[cnt2].next = h2[u];
28     h2[u] = cnt2++;
29 }
30
31 void tarjan(int u, int in_edge) {
32     dfn[u] = low[u] = ++idx;
33     for (int i = h1[u]; ~i; i = e1[i].next) {
34         int v = e1[i].to;
35         if (!dfn[v]) {
36             tarjan(v, i);
37             low[u] = min(low[u], low[v]);
38             if (low[v] > dfn[u]) num++, bridge[i] = bridge[i ^ 1] = 1;
39         } else if (i != (in_edge ^ 1))
40             low[u] = min(low[u], dfn[v]);
41     }
42 }
43
44 void rebuild(int x) {
45     belong[x] = scc;
46     for (int i = h1[x]; ~i; i = e1[i].next) {
47         int v = e1[i].to;
48         if (belong[v] || bridge[i]) continue;
49         rebuild(v);
50     }
51 }
52 }cc;
53 void solve() {
54     cc.init(n, m);
55     for (int i = 1; i <= m; i++) {
56         int u, v; cin >> u >> v;
57         cc.add(u, v), cc.add(v, u);
58     }
59     for (int i = 1; i <= n; i++) if (!cc.dfn[i]) cc.tarjan(i, 0);
60     for (int i = 1; i <= n; i++) {
61         if (!cc.belong[i]) {
62             ++cc.scc; cc.rebuild(i);
63         }
64     }
65     for (int i = 0; i < cc.cnt1; i+=2) {
66         int u = cc.e1[i].to, v = cc.e1[i^1].to;
67         if (cc.belong[u] == cc.belong[v]) continue;
68         cc.add_cc(cc.belong[u], cc.belong[v]), cc.add_cc(cc.belong[v], cc.belong[u]);
69         in[cc.belong[u]]++, in[cc.belong[v]]++;
70     }
71 }

```

## 6.4 差分约束系统

```

1  /*
2  1. 求 $s[n] - s[1]$ 的最小值时, 转换为1到n的最长路模型
3
4  2. 求 $s[n] - s[1]$ 的最大值时, 转换为1到n的最短路模型
5

```



```

6 ep:
7 求最短路时
8 1.  $x_i + T \geq x_j \rightarrow i$  向  $j$  建一条长度为  $T$  的边
9 2.  $x_i + T > x_j \rightarrow x_i + T - 1 \geq x_j$   $i$  向  $j$  建一条长度为  $T - 1$  的边
10 3.  $x_i + T == x_j \rightarrow x_i + T \geq x_j \ \&\& \ x_i + T \leq x_j$  (同(1))
11
12 求最长路时, 将  $\geq$  换成  $\leq$  即可
13 */

```

## 6.5 点双连通分量 + v-dcc 缩点

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 100010;
4 int dfn[N], low[N], idx, cut[N], cnt, stk[N], tp;
5 vector<int> g[N], dcc[N];
6 int n, m;
7 void tarjan(int u, int root) {
8     dfn[u] = low[u] = ++idx;
9     stk[++tp] = u;
10    if (u == root && g[u].empty()) {
11        dcc[++cnt].push_back(u);
12        return;
13    }
14    int child = 0;
15    for (auto v : g[u]) {
16        if (!dfn[v]) {
17            tarjan(v, root);
18            low[u] = min(low[u], low[v]);
19            if (low[v] >= dfn[u]) {
20                child++;
21                if (child > 1 || u != root) cut[u] = 1;
22                cnt++;
23                int y;
24                while (y = stk[tp--]) {
25                    dcc[cnt].push_back(y);
26                    if (y == v) break;
27                }
28                dcc[cnt].push_back(u);
29            }
30        }
31        else
32            low[u] = min(low[u], dfn[v]);
33    }
34 }
35
36 int main() {
37     cin >> n >> m;
38     for (int i = 0; i < m; i++) {
39         int x, y;
40         cin >> x >> y;
41         g[x].push_back(y);
42         g[y].push_back(x);
43     }
44     for (int i = 1; i <= n; i++) if (!dfn[i]) tarjan(i, i);
45     for (int i = 1; i <= n; i++) {
46         for (auto x : dcc[i]) cout << x << " ";
47         cout << endl;
48     }
49 }

```

```

48     }
49 }
50
51 /*
52 9 11
53 1 2
54 1 5
55 2 3
56 2 5
57 3 4
58 4 5
59 1 6
60 6 7
61 9 8
62 6 9
63 6 8
64 */

```

## 6.6 二分图 HK 算法

```

1  #include <iostream>
2  #include <cstring>
3  #include <cstdio>
4  #include <cmath>
5  #include <algorithm>
6  #include <queue>
7  #include <stack>
8  #include <string>
9  #include <set>
10 #include <map>
11 #include <bitset>
12 using namespace std;
13 #define ACM_LOCAL
14
15 const int N = 3e3 + 5;
16 const int INF = 0x3f3f3f3f;
17 typedef pair<int, int> PII;
18 typedef long long ll;
19 typedef unsigned long long ull;
20 typedef long double ld;
21
22 int h[N], cnt, n; //存图
23 int dx[N], dy[N], dis; //左边部分距离, 右边部分距离, 记录右边部分没有被匹配过的
                        //点的最大距离
24 int machx[N], machy[N]; //左边部分点匹配右边点, 左边部分的点个数, 右边部分点匹配左边的
                        //点
25 bool vist[N];
26
27 struct edge {
28     int to, next;
29 }e[N * N];
30
31 void add(int u, int v) {
32     e[cnt].to = v;
33     e[cnt].next = h[u];
34     h[u] = cnt++;
35 }
36

```

```

37 bool searchpath() { //找有没有增广路
38     queue<int> q;
39     dis = INF;
40     memset(dx, -1, sizeof(dx));
41     memset(dy, -1, sizeof(dy));
42
43     for (int i = 1; i <= n; i++)
44         if (machx[i] == -1)
45             q.push(i), dx[i] = 0;
46
47     while (!q.empty()) {
48         int u = q.front(); q.pop();
49         if (dx[u] > dis)
50             break;
51         for (int i = h[u]; i != -1; i = e[i].next) {
52             int v = e[i].to;
53             if (dy[v] == -1) {
54                 dy[v] = dx[u] + 1;
55                 if (machy[v] == -1)
56                     dis = dy[v];
57                 else {
58                     dx[machy[v]] = dy[v] + 1;
59                     q.push(machy[v]);
60                 }
61             }
62         }
63     }
64     return dis != INF;
65 }
66 bool findroad(int u) {
67     for (int i = h[u]; i != -1; i = e[i].next) {
68         int v = e[i].to;
69         if (!vist[v] && dy[v] == dx[u] + 1) {
70             vist[v] = 1;
71             if (machy[v] != -1 && dy[v] == dis)
72                 continue;
73             if (machy[v] == -1 || findroad(machy[v])) {
74                 machy[v] = u; machx[u] = v; return true;
75             }
76         }
77     }
78     return false;
79 }
80 int MaxMatch() {
81     int ans = 0;
82     memset(machx, -1, sizeof(machx));
83     memset(machy, -1, sizeof(machy));
84     while (searchpath()) {
85         memset(vist, 0, sizeof(vist));
86         for (int i = 1; i <= n; i++)
87             if (machx[i] == -1)
88                 ans += findroad(i);
89     }
90     return ans;
91 }

```

## 6.7 二分图 KM 算法

```

1  #include <iostream>
2  #include <cstring>
3  #include <cstdio>
4  #include <cmath>
5  #include <algorithm>
6  #include <queue>
7  #include <stack>
8  #include <string>
9  #include <set>
10 #include <map>
11 #include <bitset>
12 using namespace std;
13 #define ACM_LOCAL
14
15 const int INF = 0x3f3f3f3f;
16 typedef pair<int, int> PII;
17 typedef long long ll;
18 typedef unsigned long long ull;
19 typedef long double ld;
20 const int N = 310;
21 int n, m, match[N], pre[N];
22 bool vis[N];
23 ll g[N][N];
24 ll val1[N], val2[N], slack[N];
25
26 void bfs(int p){
27     memset(pre, 0, sizeof pre);
28     memset(slack, 0x3f, sizeof slack);
29     match[0] = p;
30     int x = 0, nex = 0;
31     do{
32         vis[x] = true;
33         int y = match[x];
34         ll d = INF;
35         for (int i = 1; i <= m; i++){
36             if (!vis[i]){
37                 if (slack[i] > val1[y] + val2[i] - g[y][i])
38                     slack[i] = val1[y] + val2[i] - g[y][i], pre[i] = x;
39                 if (slack[i] < d)
40                     d = slack[i], nex = i;
41             }
42         }
43         for (int i = 0; i <= m; i++){
44             if (vis[i])
45                 val1[match[i]] -= d, val2[i] += d;
46             else
47                 slack[i] -= d;
48         }
49         x = nex;
50     } while (match[x]);
51     while (x){
52         match[x] = match[pre[x]];
53         x = pre[x];
54     }
55 }
56
57 ll KM(){
58     memset(match, 0, sizeof match);
59     memset(val1, 0, sizeof val1);
60     memset(val2, 0, sizeof val2);

```

```

60     for (int i = 1; i <= n; i++){
61         memset(vis, false, sizeof vis);
62         bfs(i);
63     }
64     ll res = 0;
65     for (int i = 1; i <= m; i++) res += g[match[i]][i];
66     return res;
67 }

```

## 6.8 二分图匈牙利算法

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 10010;
4  int match[N], vis[N];
5  int n, m, ans;
6  vector<int> g[N];
7
8  bool dfs(int x) {
9      for (auto v : g[x]) {
10         if (!vis[v]) {
11             vis[v] = 1;
12             if (!match[v] || dfs(match[v])) {
13                 match[v] = x;
14                 return true;
15             }
16         }
17     }
18     return false;
19 }

```

## 6.9 割边

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N = 100010;
4  int dfn[N], low[N], idx;
5  int n, m, ans;
6  vector<int> g[N];
7  void tarjan(int u, int fa) {
8      dfn[u] = low[u] = ++idx;
9      for (auto v : g[u]) {
10         if(v == fa) continue;
11         if (!dfn[v]) {
12             tarjan(v, u);
13             low[u] = min(low[u], low[v]);
14             if (low[v] > dfn[u]) ans++;
15         }
16         else
17             low[u] = min(low[u], dfn[v]);
18     }
19 }
20
21 int main() {
22     cin >> n >> m;
23     for (int i = 0; i < m; i++) {
24         int x, y;

```

```

25         cin >> x >> y;
26         g[x].push_back(y);
27         g[y].push_back(x);
28     }
29
30     tarjan(1, 0);
31     cout << m - ans;
32 }

```

## 6.10 割点

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100010;
4  int dfn[N], low[N], idx, cut[N], siz[N], n, m;
5  vector<int> g[N];
6  int n, m;
7  void tarjan(int u, int root) {
8      dfn[u] = low[u] = ++idx;
9      int child = 0;
10     for (auto v : g[u]) {
11         if (!dfn[v]) {
12             tarjan(v, root);
13             siz[u] += siz[v];
14             low[u] = min(low[u], low[v]);
15             if (low[v] >= dfn[u]) {
16                 child++;
17                 if (child > 1 || u != root) cut[u] = 1;
18             }
19         }
20         else
21             low[u] = min(low[u], dfn[v]);
22     }
23 }

```

## 6.11 克鲁斯卡尔重构树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1e5 + 10;
4  const int M = 2e5 + 10;
5  struct Edge {
6      int u, v, w;
7      bool operator < (const Edge &rhs) const {
8          return w > rhs.w;
9          // > 最大边中的最小值
10         // < 最小边中的最大值
11     }
12 } e[M];
13 int n, m, son[N], dep[N], pre[N], siz[N], top[N], tot, q, rnk[N], dfn[N], fat[N];
14 int cnt, val[N], fa[N];
15 int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
16
17 vector<int> g[N];
18
19 void dfs1(int u, int fa) {
20     son[u] = -1; siz[u] = 1; dep[u] = dep[fa] + 1; fat[u] = fa;

```

```

21     for (auto v : g[u]) {
22         if (v == fa) continue;
23         dfs1(v, u);
24         siz[u] += siz[v];
25         if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
26     }
27 }
28
29 void dfs2(int u, int t) {
30     rnk[dfn[u] = ++tot] = u; top[u] = t;
31     if (son[u] == -1) return;
32     dfs2(son[u], t);
33     for (auto v : g[u]) {
34         if (v != son[u] && v != fat[u]) dfs2(v, v);
35     }
36 }
37
38 int lca(int u, int v) {
39     while (top[u] != top[v]) {
40         if (dep[top[u]] > dep[top[v]])
41             u = pre[top[u]];
42         else
43             v = pre[top[v]];
44     }
45     return dep[u] > dep[v] ? v : u;
46 }
47
48 void exKruskal() {
49     cnt = n; for (int i = 1; i < (n << 1); i++) fa[i] = i;
50     sort(e + 1, e + 1 + m);
51     for (int i = 1; i <= m; i++) {
52         int u = find(e[i].u), v = find(e[i].v);
53         if (u == v) continue;
54         val[++cnt] = e[i].w;
55         fa[u] = fa[v] = cnt;
56         g[u].push_back(cnt), g[cnt].push_back(u);
57         g[v].push_back(cnt), g[cnt].push_back(v);
58         if (cnt == (n << 1) - 1) break; //最多2N-1个点
59     }
60     for (int i = 1; i <= cnt; i++)
61         if (!siz[i]) { //未访问过
62             int rt = find(i); //下树剖lca
63             dfs1(rt, 0); dfs2(rt, rt);
64         }
65 }

```

## 6.12 克鲁斯卡尔重构树上建主席树

```

1  #include <bits/stdc++.h>
2  #define ACM_LOCAL
3  using namespace std;
4  typedef long long ll;
5  const int INF = 0x3f3f3f3f;
6  const int N = 2e5 + 10;
7  const int M = 5e5 + 10;
8  const int MOD = 1e9 + 7;
9
10 struct Hash {

```

```

11     int b[N], tot;
12     void init() { tot = 0; }
13     void insert(int x) { b[++tot] = x; }
14     void build() {
15         sort(b + 1, b + 1 + tot);
16         tot = unique(b + 1, b + 1 + tot) - (b + 1);
17     }
18     int pos(int x) { return lower_bound(b + 1, b + 1 + tot, x) - b; }
19 } ha;
20 struct Edge {
21     int u, v, w;
22     bool operator < (const Edge &rhs) const {
23         return w < rhs.w;
24         // > 最大边中的最小值
25         // < 最小边中的最大值
26     }
27 } e[M];
28
29 int n, m, d[N], dfn[N], t, f[N][20], q, idx, rnk[N], in[N], out[N];
30 int cnt, val[N], fa[N], a[N];
31 int rt[N], NodeNum;
32 int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
33
34 vector<int> g[N];
35
36 void dfs(int x, int fa) {
37     rnk[dfn[x] = ++idx] = x, f[x][0] = fa, d[x] = d[fa] + 1;
38     for (int i = 1; i <= 19; i++) f[x][i] = f[f[x][i-1]][i-1];
39     in[x] = idx;
40     for (auto &y : g[x])
41         if (y != fa) dfs(y, x);
42     out[x] = idx;
43 }
44
45 void exKruskal() {
46     cnt = n; for (int i = 1; i < (n << 1); i++) fa[i] = i;
47     sort(e + 1, e + 1 + m);
48     for (int i = 1; i <= m; i++) {
49         int u = find(e[i].u), v = find(e[i].v);
50         if (u == v) continue;
51         val[++cnt] = e[i].w;
52         fa[u] = fa[v] = cnt;
53         g[u].push_back(cnt), g[cnt].push_back(u);
54         g[v].push_back(cnt), g[cnt].push_back(v);
55         if (cnt == (n << 1) - 1) break; //最多2N-1个点
56     }
57     for (int i = 1; i <= cnt; i++)
58         if (!d[i]) { //未访问过
59             int rt = find(i); //下树剖lca
60             dfs(rt, 0);
61         }
62 }
63 int FindPoint (int x, int p) {
64     for (int i = 20; i >= 0; i--)
65         if (d[x] > (1 << i) && val[f[x][i]] <= p) x = f[x][i];
66     return x;
67 }
68
69 struct {

```



```

70     int t[N << 5], lc[N << 5], rc[N << 5];
71     int update(int pre, int l, int r, int x) {
72         int num = ++NodeNum;
73         lc[num] = lc[pre], rc[num] = rc[pre], t[num] = t[pre] + 1;
74         if (l != r) {
75             int mid = (l + r) >> 1;
76             if (x <= mid) lc[num] = update(lc[pre], l, mid, x);
77             else rc[num] = update(rc[pre], mid + 1, r, x);
78         }
79         return num;
80     }
81     int query(int pre, int now, int l, int r, int k) {
82         if (t[now] - t[pre] < k) return -1; //如果之间的数少于k个, 返回-1
83         k = t[now] - t[pre] - k + 1; //这里主席树写的是第k小, 转换一下变成第k大
84         while (l < r) {
85             int sum = t[lc[now]] - t[lc[pre]];
86             if (k <= sum) {
87                 now = lc[now], pre = lc[pre];
88                 r = l + r >> 1;
89             }
90             else {
91                 now = rc[now], pre = rc[pre];
92                 l = (l + r >> 1) + 1;
93                 k -= sum;
94             }
95         }
96         return ha.b[l];
97     }
98 }hjt;
99
100 int ask(int x, int p, int k) {
101     int pos = FindPoint(x, p);
102     return hjt.query(rt[in[pos]-1], rt[out[pos]], 1, ha.tot, k);
103 }
104
105 void solve() {
106     cin >> n >> m >> q;
107     for (int i = 1; i <= n; i++) cin >> a[i], ha.insert(a[i]);
108     ha.build();
109     for (int i = 1; i <= m; i++)
110         cin >> e[i].u >> e[i].v >> e[i].w;
111     exKruskal();
112     for (int i = 1; i <= idx; i++) {
113         rt[i] = rt[i-1];
114         if (rnk[i] <= n)
115             rt[i] = hjt.update(rt[i-1], 1, ha.tot, ha.pos(a[rnk[i]]));
116     }
117
118     while (q--) {
119         int v, x, k;
120         cin >> v >> x >> k;
121         printf("%d\n", ask(v, x, k));
122     }
123 }
124
125 int main() {
126     ios_base::sync_with_stdio(false);
127     cin.tie(0);
128     cout.tie(0);

```

```

129 #ifdef ACM_LOCAL
130     freopen("input", "r", stdin);
131     freopen("output", "w", stdout);
132 #endif
133     solve();
134 }

```

### 6.13 欧拉图

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1e5 + 10;
4  const int M = 2e5 + 10;
5
6  struct edge {
7      int to, next;
8  }e[M<<1];
9
10 int n, m, t, stk[N], tp, cnt, h[N], vis[M<<1], ans[N];
11
12 void add(int u, int v) {
13     e[cnt].to = v;
14     e[cnt].next = h[u];
15     h[u] = cnt++;
16 }
17
18 void euler() {
19     stk[++tp] = 1;
20     while (tp > 0) {
21         int x = stk[tp], i = h[x];
22         while (i && vis[i]) i = e[i].next;
23         if (i) {
24             stk[++tp] = e[i].to;
25             vis[i] = vis[i ^ 1] = 1;
26             h[x] = e[i].next;
27         }
28         else {
29             tp--;
30             ans[++t] = x;
31         }
32     }
33 }
34
35 int main() {
36     cin >> n >> m;
37     for (int i = 0; i < m; i++) {
38         int x, y;
39         cin >> x >> y;
40         add(x, y), add(y, x);
41     }
42     euler();
43     for (int i = t; i >= 1; i--) cout << ans[i] << " ";
44 }
45
46 /*
47 7 9
48 1 2
49 1 3

```

```

50 1 4
51 1 5
52 2 3
53 4 5
54 5 6
55 6 7
56 5 7
57 */

```

## 6.14 强联通分量

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 10010;
4  int h[N], cnt, n, m, scc;
5  int dfn[N], low[N], idx, tp, in_stk[N], vis[N], sd[N];
6  set<int> sc[N];
7  struct edge{
8      int to, next;
9  }e[N*2];
10
11 void add(int u, int v) {
12     e[cnt].to = v;
13     e[cnt].next = h[u];
14     h[u] = cnt++;
15 }
16
17 void tarjan(int x) {
18     low[x] = dfn[x] = ++idx;
19     vis[x] = 1;
20     in_stk[++tp] = x;
21     for (int i = h[x]; ~i; i = e[i].next) {
22         int v = e[i].to;
23         if (!dfn[v]) {
24             tarjan(v);
25             low[x] = min(low[x], low[v]);
26         }
27         else if (vis[v]) {
28             low[x] = min(low[x], dfn[v]);
29         }
30     }
31     if (low[x] == dfn[x]) {
32         int y;
33         ++scc;
34         while(y = in_stk[tp--]) {
35             sd[y] = scc;
36             vis[y] = 0;
37             sc[scc].insert(y);
38             if (x == y) break;
39         }
40     }
41 }
42
43 int main() {
44     cin >> n >> m;
45     memset(h, -1, sizeof h);
46     for (int i = 0; i < m; i++) {
47         int x, y;

```

```

48     cin >> x >> y;
49     add(x, y);
50 }
51 for (int i = 1; i <= n; i++) if (!dfn[i]) tarjan(i);
52
53 for (int i = 1; i <= scc; i++) {
54     cout << "#" << i << ":";
55     for (auto x : sc[i]) {
56         cout << x << " ";
57     }
58     cout << endl;
59 }
60 }

```

### 6.15 限制流量费用流 (Dijkstra)

```

1
2 struct Edge {
3     int to;
4     int cap, dis; //容量、费用
5     int rev;      //(u,v)的反向弧中,v在u的位置
6     Edge() {}
7     Edge(int to, int cap, int dis, int rev): to(to), cap(cap), dis(dis), rev(rev) {}
8 };
9 vector<Edge> G[N];
10 struct Pre { //记录前驱
11     int node; //前驱结点
12     int edge; //对应的边
13 } pre[N];
14 int h[N]; //势能函数
15 ll dis[N]; //费用
16 void addEdge(int x, int y, int cap, int cost) {
17     G[x].push_back(Edge(y, cap, cost, (int)G[y].size())); //正向边
18     G[y].push_back(Edge(x, 0, -cost, (int)(G[x].size() - 1))); //反向边
19 }
20 bool Dijkstra(int S, int T) {
21     memset(dis, 0x3f3f3f3f, sizeof dis);
22     dis[S] = 0;
23
24     priority_queue<PII, vector<PII>, greater<PII>> Q;
25     Q.push(PII(dis[S], S));
26     while (!Q.empty()) {
27         PII now = Q.top();
28         Q.pop();
29
30         int u = now.second;
31         if (dis[u] < now.first)
32             continue;
33
34         for (int i = 0; i < G[u].size(); i++) {
35             int v = G[u][i].to;
36             int cap = G[u][i].cap;
37             int w = G[u][i].dis;
38             if (cap && dis[v] > w + dis[u] + h[u] - h[v]) {
39                 dis[v] = w + dis[u] + h[u] - h[v]; //进行松弛
40                 pre[v].node = u; //记录前驱点
41                 pre[v].edge = i; //记录前驱边
42                 Q.push(PII(dis[v], v));

```

```

43     }
44 }
45 }
46 if (dis[T] == INF)
47     return false;
48 else {
49     for (int i = 0; i <= T + 1; i++) //对于势能函数, 每次加上当前轮的dis
50         h[i] += dis[i];
51     return true;
52 }
53 }
54 void maxFlow(int S, int T, int &flow, ll &cost) { //flow 代表最大总流量
55     memset(h, 0, sizeof(h));
56     memset(pre, 0, sizeof(pre));
57
58     int newFlow = 0; //增广流量
59     while (flow && Dijkstra(S, T)) { //当无法增广时, 即找到答案
60         int minn = INF;
61         for (int i = T; i != S; i = pre[i].node) {
62             int node = pre[i].node;
63             int edge = pre[i].edge;
64             minn = min(minn, G[node][edge].cap);
65         }
66
67         flow -= minn; //原流量
68         newFlow += minn; //增广流量
69         cost += 1ll * h[T] * minn; //增广流花销
70
71         for (int i = T; i != S; i = pre[i].node) {
72             int node = pre[i].node;
73             int edge = pre[i].edge;
74             int rev = G[node][edge].rev;
75             G[node][edge].cap -= minn;
76             G[i][rev].cap += minn;
77         }
78     }
79 }

```

## 6.16 一般图最大匹配 (带花树)

```

1 struct Edge {
2     int to, next;
3 } e[M];
4 int cnt, n, m, nn, h[N], fa[N], match[N], pre[N], tic[N], tim, ty[N];
5 queue<int> que;
6 void add(int u, int v) {
7     e[cnt].to = v;
8     e[cnt].next = h[u];
9     h[u] = cnt++;
10 }
11
12 int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
13
14 int lca(int x, int y) {
15     for (tim++; swap(x, y))
16         if (x) {
17             x = find(x);
18             if (tic[x] == tim) return x;

```

```

19         tic[x] = tim;
20         x = pre[match[x]];
21     }
22 }
23
24 void shrink(int x, int y, int p) {
25     while (find(x) != p) {
26         pre[x] = y;
27         y = match[x];
28         if (ty[y] == 2) ty[y] = 1, que.push(y);
29         if (find(x) == x) fa[x] = p;
30         if (find(y) == y) fa[y] = p;
31         x = pre[y];
32     }
33 }
34
35 bool aug(int s) {
36     for (int i = 1; i <= n; i++) fa[i] = i, ty[i] = pre[i] = 0;
37     while (!que.empty()) que.pop();
38     que.push(s); ty[s] = 1;
39     while (!que.empty()) {
40         int x = que.front();
41         que.pop();
42         for (int i = h[x], y = e[i].to; ~i; i = e[i].next, y = e[i].to) {
43             if (find(x) == find(y) || ty[y] == 2) continue;
44             if (!ty[y]) {
45                 ty[y] = 2;
46                 pre[y] = x;
47                 if (!match[y]) {
48                     for (int tmp; y; y = tmp, x = pre[y])
49                         tmp = match[x], match[x] = y, match[y] = x;
50                     return 1;
51                 } else ty[match[y]] = 1, que.push(match[y]);
52             } else if (ty[y] == 1) {
53                 int p = lca(x, y);
54                 shrink(x, y, p);
55                 shrink(y, x, p);
56             }
57         }
58     }
59     return 0;
60 }
61 int calc() {
62     int ans = 0;
63     for (int i = 1; i <= n; i++) {
64         if (!match[i] && aug(i)) ans++;
65     }
66     return ans;
67 }
68 void init() {
69     for (int i = 1; i <= n; i++) {
70         match[i] = tic[i] = 0;
71         h[i] = -1;
72     }
73     cnt = tim = 0;
74 }

```

## 6.17 最大独立集 (一般图)

```

1  const int N = 200 + 5;
2  int G[N][N];
3  int ans[N];
4  int vis[N];
5  int res, n;
6  void dfs(int x, int cnt) {
7      if (x > n) {
8          if (cnt > res) {
9              res = cnt;
10             for (int i = 1; i <= n; i++)
11                 ans[i] = vis[i];
12             }
13             return;
14         }
15         if (cnt + n - x + 1 < res) return;
16         int pd = 0;
17         for (int i = 1; i < x; i++)
18             if (vis[i] && G[i][x]) {
19                 pd = 1;
20                 break;
21             }
22         if (!pd) {
23             vis[x] = 1;
24             dfs(x+1, cnt+1);
25             vis[x] = 0;
26         }
27         dfs(x+1, cnt);
28     }

```

## 6.18 最大流 (Dinic)

```

1  typedef long long ll;
2  typedef pair<int, int> PII;
3  const int N = 1e3 + 10, M = 2e5 + 10, INF = 0x3f3f3f3f;
4  const int MOD = 1e9 + 7;
5  int state[N][11], n, p, cap[N], st, ed;
6  struct Maxflow {
7      int h[N], cnt, maxflow, deep[N], cur[N];
8      struct Edge {
9          int to, next;
10         ll cap;
11     } e[M<<1];
12
13     void init() {
14         memset(h, -1, sizeof h);
15         cnt = maxflow = 0;
16     }
17     void add(int u, int v, int cap) {
18         e[cnt].to = v;
19         e[cnt].cap = cap;
20         e[cnt].next = h[u];
21         h[u] = cnt++;
22
23         e[cnt].to = u;
24         e[cnt].cap = 0;
25         e[cnt].next = h[v];

```

```

26     h[v] = cnt++;
27
28 }
29
30 bool bfs() {
31     for (int i = 0; i <= ed; i++) deep[i] = -1, cur[i] = h[i];
32     queue<int> q; q.push(st); deep[st] = 0;
33     while (q.size()) {
34         int u = q.front();
35         q.pop();
36         for (int i = h[u]; ~i; i = e[i].next) {
37             int v = e[i].to;
38             if (e[i].cap && deep[v] == -1) {
39                 deep[v] = deep[u] + 1;
40                 q.push(v);
41             }
42         }
43     }
44     if (deep[ed] >= 0) return true;
45     else return false;
46 }
47
48 ll dfs(int u, ll mx) {
49     ll a;
50     if (u == ed) return mx;
51     for (int i = cur[u]; ~i; i = e[i].next) {
52         cur[u] = i; //优化
53         int v = e[i].to;
54         if (e[i].cap && deep[v] == deep[u] + 1 && (a = dfs(v, min(e[i].cap, mx))))
55             {
56                 e[i].cap -= a;
57                 e[i ^ 1].cap += a;
58                 return a;
59             }
60     }
61     return 0;
62 }
63 void dinic() {
64     ll res;
65     while (bfs()) {
66         while (1) {
67             res = dfs(st, INF);
68             if (!res) break;
69             maxflow += res;
70         }
71     }
72 }
73 }mf;

```

### 6.19 最大团 (一般图)

```

1  const int N = 200 + 5;
2  int n;
3  int G[N][N];
4  int cnt[N]; //cnt[i]为>=i的最大团点数
5  int group[N]; //最大团的点
6  int vis[N]; //记录点的位置

```



```

7  int res;//最大团的数目
8  bool dfs(int pos, int num) { //num为已取的点数
9      for (int i = pos + 1; i <= n; i++) {
10         if (cnt[i] + num <= res) //剪枝, 若取i但cnt[i]+已经取了的点数仍<ans
11             return false;
12
13         if (G[pos][i]) { //与当前团中元素比较, 取Non-N(i)
14             int j;
15             for (j = 0; j < num; j++)
16                 if (!G[i][vis[j]])
17                     break;
18             if (j == num) { //若为空, 则皆与i相邻, 则此时将i加入到最大团中
19                 vis[num] = i;
20                 if (dfs(i, num + 1))
21                     return true;
22             }
23         }
24     }
25
26     if (num > res) { //每添加一个点最多使最大团数+1, 后面的搜索就没有意义了
27         res = num; //最大团中点的数目
28         for (int i = 1; i <= num; i++) //最大团的元素
29             group[i] = vis[i - 1];
30         return true;
31     }
32     return false;
33 }
34
35 void maxClique() {
36     res = -1;
37     for (int i = n; i >= 1; i--) { //枚举所有点
38         vis[0] = i;
39         dfs(i, 1);
40         cnt[i] = res;
41     }
42 }

```

## 6.20 最短路

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1e5 + 10;
4  const int M = 5e5 + 10;
5  typedef long long ll;
6  struct Node {
7      struct Edge {
8          int to, next, w;
9      } e[M<<1];
10
11     int h[N], cnt, vis[N], count[N];
12     ll dis[N];
13     void init() {
14         memset(h, -1, sizeof h);
15         cnt = 0;
16     }
17     void add(int u, int v, int w) {
18         e[cnt].to = v;
19         e[cnt].w = w;

```

```

20     e[cnt].next = h[u];
21     h[u] = cnt++;
22 }
23
24 struct node {
25     int now; ll d;
26     bool operator < (const node &rhs) const {
27         return d > rhs.d;
28     }
29 };
30
31 void dij(int st) {
32     memset(dis, 0x3f, sizeof dis);
33     memset(vis, 0, sizeof vis);
34     dis[st] = 0; priority_queue<node> q;
35     q.push({st, dis[st]});
36     while (q.size()) {
37         int u = q.top().now;
38         q.pop();
39         if (vis[u]) continue;
40         vis[u] = 1;
41         for (int i = h[u]; ~i; i = e[i].next) {
42             int v = e[i].to;
43             if (dis[v] > dis[u] + e[i].w) {
44                 dis[v] = dis[u] + e[i].w;
45                 if (!vis[v]) {
46                     q.push({v, dis[v]});
47                 }
48             }
49             else if (dis[v] == dis[u] + e[i].w) {
50                 count[v]++;
51             }
52         }
53     }
54 }
55 }Dij;

```

## 6.21 最小费用最大流 (SPFA)

```

1  typedef long long ll;
2  typedef pair<int, int> PII;
3  const int N = 1e4 + 10, M = 2e5 + 10, INF = 0x3f3f3f3f;
4  const int MOD = 1e9 + 7;
5  int st, ed;
6  struct node {
7      int maxflow, mincost, cnt;
8      int vis[N], dis[N], pre[N], last[N], h[N], flow[N];
9      struct edge {
10         int to, next, cap, cos;
11     } e[M << 1];
12
13     void add(int u, int v, int cap, int cos) {
14         e[cnt].to = v;
15         e[cnt].cap = cap;
16         e[cnt].cos = cos;
17         e[cnt].next = h[u];
18         h[u] = cnt++;
19     }

```

```

20     e[cnt].to = u;
21     e[cnt].cap = 0;
22     e[cnt].cos = -cos;
23     e[cnt].next = h[v];
24     h[v] = cnt++;
25 }
26
27 void init() {
28     memset(h, -1, sizeof h);
29     cnt = 0;
30     mincost = maxflow = 0;
31 }
32 bool spfa() {
33     queue<int> q;
34     for (int i = 0; i <= ed; i++) dis[i] = INF, vis[i] = 0;
35     vis[st] = 1, dis[st] = 0, flow[st] = INF;
36     q.push(st);
37     while (q.size()) {
38         int u = q.front();
39         q.pop(); vis[u] = 0;
40         for (int i = h[u]; ~i; i = e[i].next) {
41             int v = e[i].to;
42             if (e[i].cap && dis[v] > dis[u] + e[i].cos) {
43                 dis[v] = e[i].cos + dis[u];
44                 flow[v] = min(flow[u], e[i].cap);
45                 pre[v] = u;
46                 last[v] = i;
47                 if (!vis[v]) {
48                     vis[v] = 1;
49                     q.push(v);
50                 }
51             }
52         }
53     }
54     if (dis[ed] != INF) return true;
55     else return false;
56 }
57
58 void MCMF() {
59     while (spfa()) {
60         int now = ed;
61         maxflow += flow[ed];
62         mincost += flow[ed] * dis[ed];
63
64         while (st != now) {
65             e[last[now]].cap -= flow[ed];
66             e[last[now] ^ 1].cap += flow[ed];
67             now = pre[now];
68         }
69     }
70 }
71 }mcmf;

```

## 6.22 最小路径生成树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 100010;

```

```

4  typedef long long ll;
5  typedef pair<int, int> PII;
6
7  struct Edge {
8      int to, next, w;
9  }e[N<<1];
10
11  int h[N], cnt;
12  void add(int u, int v, int w) {
13      e[cnt].to = v;
14      e[cnt].w = w;
15      e[cnt].next = h[u];
16      h[u] = cnt++;
17  }
18  int n, m, k;
19  struct Node {
20      vector<PII> g[N];
21      int vis[N];
22      ll dis[N];
23      void init() {
24          for (int i = 1; i <= n; i++) g[i].clear();
25      }
26
27      struct node {
28          int now; ll d;
29          bool operator < (const node &rhs) const {
30              return d > rhs.d;
31          }
32      };
33
34      void dij(int st) {
35          memset(dis, 0x3f, sizeof dis);
36          memset(vis, 0, sizeof vis);
37          dis[st] = 0; priority_queue<node> q;
38          q.push({st, dis[st]});
39          while (q.size()) {
40              int u = q.top().now;
41              q.pop();
42              if (vis[u]) continue;
43              vis[u] = 1;
44              for (auto item : g[u]) {
45                  int v = item.first;
46                  int w = item.second;
47                  if (dis[v] > dis[u] + w) {
48                      dis[v] = dis[u] + w;
49                      if (!vis[v]) {
50                          q.push({v, dis[v]});
51                      }
52                  }
53              }
54          }
55      }
56  }Dij;
57  int tap[N];
58  void build_tree(int x) {
59      tap[x] = 1;
60      for (auto item : Dij.g[x]) {
61          int v = item.first;
62          int w = item.second;

```

```

63     if (tap[v]) continue;
64     if (Dij.dis[v] == Dij.dis[x] + w) {
65         add(x, v, w), add(v, x, w);
66         build_tree(v);
67     }
68 }
69 }
70
71 int main() {
72     scanf("%d %d %d", &n, &m, &k);
73     Dij.init();
74     for (int i = 1; i <= m; i++) {
75         int u, v, w; scanf("%d %d %d", &u, &v, &w);
76         Dij.g[u].push_back(PII{v, w});
77         Dij.g[v].push_back(PII{u, w});
78     }
79     Dij.dij(1);
80     for (int i = 1; i <= n; i++)
81         sort(Dij.g[i].begin(), Dij.g[i].end());
82     memset(h, -1, sizeof h);
83     build_tree(1);
84 }

```

### 6.23 最小生成树 boruvka

```

1  ll boruvka() {
2     for (int i = 1; i <= n; i++) fa[i] = i;
3     ll ans = 0, num = 0;
4     while (num < n-1) {
5         int tmp = 0;
6         for (int i = 1; i <= n; i++) E[i] = PII{INF, INF};
7         for (int i = 1; i <= m; i++) {
8             int fx = find(e[i].u);
9             int fy = find(e[i].v);
10            if (fx == fy) continue;
11            tmp++;
12            E[fx] = min(E[fx], PII{e[i].w, i});
13            E[fy] = min(E[fy], PII{e[i].w, i});
14        }
15        if (tmp == 0) break;
16        for (int i = 1; i <= m; i++) {
17            int fx = find(e[i].u);
18            int fy = find(e[i].v);
19            if (fx == fy) continue;
20            if (E[fx] == PII{e[i].w, i} || E[fy] == PII{e[i].w, i}) {
21                ans += e[i].w;
22                num++;
23                fa[fx] = fy;
24            }
25        }
26    }
27    if (num < n-1) return -1;
28    else return ans;
29 }

```

### 6.24 最小生成树 prim

```
1 int prim() {
2     memset(vis, 0, sizeof vis);
3     memset(dis, 0x3f, sizeof dis);
4     int ans = 0;
5     for (int i = h[1]; ~i; i = e[i].next) {
6         dis[e[i].to] = min(dis[e[i].to], e[i].vi);
7     }
8     dis[1] = 0, vis[1] = 1;
9     for (int i = 2; i <= n; i++) {
10         int minn = INF;
11         int k;
12         for (int j = 1; j <= n; j++) {
13             if (!vis[j] && dis[j] < minn) {
14                 minn = dis[j];
15                 k = j;
16             }
17         }
18         if (minn == INF) return -1;
19         ans += minn;
20         vis[k] = 1;
21         for (int j = h[k]; ~j; j = e[j].next) {
22             dis[e[j].to] = min(dis[e[j].to], e[j].vi);
23         }
24     }
25     return ans;
26 }
```

## 7 杂项

```

1 vector<int> TreeToPrufer(int n) { // 1~n-2个数
2     vector<int> pru(n+1);
3     vector<int> in(n+1);
4     vector<int> fa(n+1);
5     for (int i = 1; i < n; i++) read(fa[i]), ++in[fa[i]];
6     for (int i = 1, j = 1; i <= n - 2; i++, j++) {
7         while (in[j]) ++j; pru[i] = fa[j];
8         while (i <= n - 2 && !--in[pru[i]] && pru[i] < j) pru[i+1] = fa[pru[i]], ++i;
9     }
10    return pru;
11 }
12
13 vector<int> PruferToTree(int n) { // 1~n-1个数
14     vector<int> fa(n+1);
15     vector<int> in(n+1);
16     vector<int> pru(n+1);
17     for (int i = 1; i <= n - 2; i++) read(pru[i]), ++in[pru[i]]; pru[n-1] = n;
18     for (int i = 1, j = 1; i < n; i++, j++) {
19         while (in[j]) ++j; fa[j] = pru[i];
20         while (i < n && !--in[pru[i]] && pru[i] < j) fa[pru[i]] = pru[i+1], ++i;
21     }
22     return fa;
23 }

```

### 7.1 DEBUG

```

1 #define debug(a...) cout << "(" << (#a) << ")" << " = (" ; DEBUG(a)
2 template<typename T> void DEBUG(T value) {
3     cout << value << ")" << endl;
4 }
5 template<typename T1, typename... T2>
6 void DEBUG(T1 now, T2... other) {
7     cout << now << ", ", DEBUG(other...);
8 }

```

### 7.2 MODINT

```

1 namespace MODINT {
2     template<unsigned M_> struct ModInt {
3         static constexpr unsigned M = M_;
4         unsigned x;
5         constexpr ModInt() : x(0U) {}
6         constexpr ModInt(unsigned x_) : x(x_ % M) {}
7         constexpr ModInt(unsigned long long x_) : x(x_ % M) {}
8         constexpr ModInt(int x_) : x(((x_ %= static_cast<int>(M)) < 0) ? (x_ +
9         static_cast<int>(M)) : x_) {}
10        constexpr ModInt(long long x_) : x(
11            ((x_ %= static_cast<long long>(M)) < 0) ? (x_ + static_cast<long long>(M))
12            : x_) {}
13        ModInt &operator+=(const ModInt &a) {
14            x = ((x += a.x) >= M) ? (x - M) : x;
15            return *this;
16        }
17        ModInt &operator-=(const ModInt &a) {
18            x = ((x -= a.x) >= M) ? (x + M) : x;
19        }
20    };
21 }

```

```

17         return *this;
18     }
19     ModInt &operator*=(const ModInt &a) {
20         x = (static_cast<unsigned long long>(x) * a.x) % M;
21         return *this;
22     }
23     ModInt &operator/=(const ModInt &a) { return (*this *= a.inv()); }
24     ModInt quick_pow(long long e) const {
25         if (e < 0) return inv().quick_pow(-e);
26         ModInt a = *this, b = 1U;
27         for (; e; e >>= 1) { if (e & 1) b *= a; a *= a; }
28         return b;
29     }
30     ModInt inv() const {
31         unsigned a = M, b = x;
32         int y = 0, z = 1;
33         for (; b;) {
34             const unsigned q = a / b;
35             const unsigned c = a - q * b;
36             a = b; b = c;
37             const int w = y - static_cast<int>(q) * z;
38             y = z; z = w;
39         }
40         assert(a == 1U);
41         return ModInt(y);
42     }
43     ModInt operator+() const { return *this; }
44     ModInt operator-() const { ModInt a; a.x = x ? (M - x) : 0U; return a; }
45     ModInt operator+(const ModInt &a) const { return (ModInt(*this) += a); }
46     ModInt operator-(const ModInt &a) const { return (ModInt(*this) -= a); }
47     ModInt operator*(const ModInt &a) const { return (ModInt(*this) *= a); }
48     ModInt operator/(const ModInt &a) const { return (ModInt(*this) /= a); }
49     template<class T> friend ModInt operator+(T a, const ModInt &b) { return (
ModInt(a) += b); }
50     template<class T> friend ModInt operator-(T a, const ModInt &b) { return (
ModInt(a) -= b); }
51     template<class T> friend ModInt operator*(T a, const ModInt &b) { return (
ModInt(a) *= b); }
52     template<class T> friend ModInt operator/(T a, const ModInt &b) { return (
ModInt(a) /= b); }
53     explicit operator bool() const { return x; }
54     bool operator==(const ModInt &a) const { return (x == a.x); }
55     bool operator!=(const ModInt &a) const { return (x != a.x); }
56     friend std::ostream &operator<<(std::ostream &os, const ModInt &a) { return os
<< a.x; }
57 };
58 constexpr unsigned M0 = 1000000007;
59 // constexpr unsigned M0 = 998244353;
60 using Mint = ModInt<M0>;
61 }

```

### 7.3 大数模板

```

1  constexpr int base = 1000000000;
2  constexpr int base_digits = 9;
3
4  struct bigint {
5      // value == 0 is represented by empty z

```



```

6     vector<int> z; // digits
7
8     // sign == 1 <==> value >= 0
9     // sign == -1 <==> value < 0
10    int sign;
11
12    bigint() : sign(1) {}
13
14    bigint(ll v) { *this = v; }
15
16    bigint &operator=(ll v) {
17        sign = v < 0 ? -1 : 1;
18        v *= sign;
19        z.clear();
20        for (; v > 0; v = v / base) z.push_back((int) (v % base));
21        return *this;
22    }
23
24    bigint(const string &s) { read(s); }
25
26    bigint &operator+=(const bigint &other) {
27        if (sign == other.sign) {
28            for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
29                if (i == z.size())
30                    z.push_back(0);
31                z[i] += carry + (i < other.z.size() ? other.z[i] : 0);
32                carry = z[i] >= base;
33                if (carry)
34                    z[i] -= base;
35            }
36        } else if (other != 0 /* prevent infinite loop */) {
37            *this -= -other;
38        }
39        return *this;
40    }
41
42    friend bigint operator+(bigint a, const bigint &b) { return a += b; }
43
44    bigint &operator-=(const bigint &other) {
45        if (sign == other.sign) {
46            if (sign == 1 && *this >= other || sign == -1 && *this <= other) {
47                for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
48                    z[i] -= carry + (i < other.z.size() ? other.z[i] : 0);
49                    carry = z[i] < 0;
50                    if (carry)
51                        z[i] += base;
52                }
53                trim();
54            } else {
55                *this = other - *this;
56                this->sign = -this->sign;
57            }
58        } else {
59            *this += -other;
60        }
61        return *this;
62    }
63
64    friend bigint operator-(bigint a, const bigint &b) {

```

```

65     return a -= b;
66 }
67
68 bigint &operator*=(int v) {
69     if (v < 0) sign = -sign, v = -v;
70     for (int i = 0, carry = 0; i < z.size() || carry; ++i) {
71         if (i == z.size()) z.push_back(0);
72         ll cur = (ll) z[i] * v + carry;
73         carry = (int) (cur / base);
74         z[i] = (int) (cur % base);
75     }
76     trim();
77     return *this;
78 }
79
80 bigint operator*(int v) const { return bigint(*this) *= v; }
81
82 friend pair<bigint, bigint> divmod(const bigint &a1, const bigint &b1) {
83     int norm = base / (b1.z.back() + 1);
84     bigint a = a1.abs() * norm;
85     bigint b = b1.abs() * norm;
86     bigint q, r;
87     q.z.resize(a.z.size());
88
89     for (int i = (int) a.z.size() - 1; i >= 0; i--) {
90         r *= base;
91         r += a.z[i];
92         int s1 = b.z.size() < r.z.size() ? r.z[b.z.size()] : 0;
93         int s2 = b.z.size() - 1 < r.z.size() ? r.z[b.z.size() - 1] : 0;
94         int d = (int) (((ll) s1 * base + s2) / b.z.back());
95         r -= b * d;
96         while (r < 0) r += b, --d;
97         q.z[i] = d;
98     }
99
100     q.sign = a1.sign * b1.sign;
101     r.sign = a1.sign;
102     q.trim();
103     r.trim();
104     return {q, r / norm};
105 }
106
107 friend bigint sqrt(const bigint &a1) {
108     bigint a = a1;
109     while (a.z.empty() || a.z.size() % 2 == 1) a.z.push_back(0);
110
111     int n = a.z.size();
112
113     int firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
114     int norm = base / (firstDigit + 1);
115     a *= norm;
116     a *= norm;
117     while (a.z.empty() || a.z.size() % 2 == 1) a.z.push_back(0);
118
119     bigint r = (ll) a.z[n - 1] * base + a.z[n - 2];
120     firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
121     int q = firstDigit;
122     bigint res;
123

```

```

124     for (int j = n / 2 - 1; j >= 0; j--) {
125         for (;;) --q) {
126             bigint r1 = (r - (res * 2 * base + q) * q) * base * base +
127                 (j > 0 ? (ll) a.z[2 * j - 1] * base + a.z[2 * j - 2] : 0);
128             if (r1 >= 0) {
129                 r = r1;
130                 break;
131             }
132         }
133         (res *= base) += q;
134
135         if (j > 0) {
136             int d1 = res.z.size() + 2 < r.z.size() ? r.z[res.z.size() + 2] : 0;
137             int d2 = res.z.size() + 1 < r.z.size() ? r.z[res.z.size() + 1] : 0;
138             int d3 = res.z.size() < r.z.size() ? r.z[res.z.size()] : 0;
139             q = (int) (((ll) d1 * base * base + (ll) d2 * base + d3) / (firstDigit
140 * 2));
141         }
142     }
143     res.trim();
144     return res / norm;
145 }
146
147 bigint operator/(const bigint &v) const {
148     return divmod(*this, v).first;
149 }
150
151 bigint operator%(const bigint &v) const {
152     return divmod(*this, v).second;
153 }
154
155 bigint &operator/=(int v) {
156     if (v < 0) sign = -sign, v = -v;
157     for (int i = (int) z.size() - 1, rem = 0; i >= 0; --i) {
158         ll cur = z[i] + rem * (ll) base;
159         z[i] = (int) (cur / v);
160         rem = (int) (cur % v);
161     }
162     trim();
163     return *this;
164 }
165
166 bigint operator/(int v) const {
167     return bigint(*this) /= v;
168 }
169
170 int operator%(int v) const {
171     if (v < 0) v = -v;
172     int m = 0;
173     for (int i = (int) z.size() - 1; i >= 0; --i)
174         m = (int) ((z[i] + m * (ll) base) % v);
175     return m * sign;
176 }
177
178 bigint &operator*=(const bigint &v) {
179     return *this = *this * v;
180 }
181

```

```

182     bigint &operator/=(const bigint &v) {
183         return *this = *this / v;
184     }
185
186     bool operator<(const bigint &v) const {
187         if (sign != v.sign)
188             return sign < v.sign;
189         if (z.size() != v.z.size())
190             return z.size() * sign < v.z.size() * v.sign;
191         for (int i = (int) z.size() - 1; i >= 0; i--)
192             if (z[i] != v.z[i])
193                 return z[i] * sign < v.z[i] * sign;
194         return false;
195     }
196
197     bool operator>(const bigint &v) const { return v < *this; }
198
199     bool operator<=(const bigint &v) const { return !(v < *this); }
200
201     bool operator>=(const bigint &v) const { return !(*this < v); }
202
203     bool operator==(const bigint &v) const { return !(*this < v) && !(v < *this); }
204
205     bool operator!=(const bigint &v) const { return *this < v || v < *this; }
206
207     void trim() {
208         while (!z.empty() && z.back() == 0) z.pop_back();
209         if (z.empty()) sign = 1;
210     }
211
212     bool isZero() const {
213         return z.empty();
214     }
215
216     friend bigint operator-(bigint v) {
217         if (!v.z.empty()) v.sign = -v.sign;
218         return v;
219     }
220
221     bigint abs() const {
222         return sign == 1 ? *this : -*this;
223     }
224
225     ll longValue() const {
226         ll res = 0;
227         for (int i = (int) z.size() - 1; i >= 0; i--)
228             res = res * base + z[i];
229         return res * sign;
230     }
231
232     friend bigint gcd(const bigint &a, const bigint &b) {
233         return b.isZero() ? a : gcd(b, a % b);
234     }
235
236     friend bigint lcm(const bigint &a, const bigint &b) {
237         return a / gcd(a, b) * b;
238     }
239
240     void read(const string &s) {

```

```

241     sign = 1;
242     z.clear();
243     int pos = 0;
244     while (pos < s.size() && (s[pos] == '-' || s[pos] == '+')) {
245         if (s[pos] == '-') sign = -sign;
246         ++pos;
247     }
248     for (int i = (int) s.size() - 1; i >= pos; i -= base_digits) {
249         int x = 0;
250         for (int j = max(pos, i - base_digits + 1); j <= i; j++)
251             x = x * 10 + s[j] - '0';
252         z.push_back(x);
253     }
254     trim();
255 }
256
257 friend istream &operator>>(istream &stream, bigint &v) {
258     string s;
259     stream >> s;
260     v.read(s);
261     return stream;
262 }
263
264 friend ostream &operator<<(ostream &stream, const bigint &v) {
265     if (v.sign == -1)
266         stream << '-';
267     stream << (v.z.empty() ? 0 : v.z.back());
268     for (int i = (int) v.z.size() - 2; i >= 0; --i)
269         stream << setw(base_digits) << setfill('0') << v.z[i];
270     return stream;
271 }
272
273 static vector<int> convert_base(const vector<int> &a, int old_digits, int
new_digits) {
274     vector<ll> p(max(old_digits, new_digits) + 1);
275     p[0] = 1;
276     for (int i = 1; i < p.size(); i++)
277         p[i] = p[i - 1] * 10;
278     vector<int> res;
279     ll cur = 0;
280     int cur_digits = 0;
281     for (int v : a) {
282         cur += v * p[cur_digits];
283         cur_digits += old_digits;
284         while (cur_digits >= new_digits) {
285             res.push_back(int(cur % p[new_digits]));
286             cur /= p[new_digits];
287             cur_digits -= new_digits;
288         }
289     }
290     res.push_back((int) cur);
291     while (!res.empty() && res.back() == 0) res.pop_back();
292     return res;
293 }
294
295 typedef vector<ll> vll;
296
297 static vll karatsubaMultiply(const vll &a, const vll &b) {
298     int n = a.size();

```

```

299     vll res(n + n);
300     if (n <= 32) {
301         for (int i = 0; i < n; i++)
302             for (int j = 0; j < n; j++)
303                 res[i + j] += a[i] * b[j];
304         return res;
305     }
306
307     int k = n >> 1;
308     vll a1(a.begin(), a.begin() + k);
309     vll a2(a.begin() + k, a.end());
310     vll b1(b.begin(), b.begin() + k);
311     vll b2(b.begin() + k, b.end());
312
313     vll a1b1 = karatsubaMultiply(a1, b1);
314     vll a2b2 = karatsubaMultiply(a2, b2);
315
316     for (int i = 0; i < k; i++) a2[i] += a1[i];
317     for (int i = 0; i < k; i++) b2[i] += b1[i];
318
319     vll r = karatsubaMultiply(a2, b2);
320     for (int i = 0; i < a1b1.size(); i++) r[i] -= a1b1[i];
321     for (int i = 0; i < a2b2.size(); i++) r[i] -= a2b2[i];
322
323     for (int i = 0; i < r.size(); i++) res[i + k] += r[i];
324     for (int i = 0; i < a1b1.size(); i++) res[i] += a1b1[i];
325     for (int i = 0; i < a2b2.size(); i++) res[i + n] += a2b2[i];
326     return res;
327 }
328
329 bigint operator*(const bigint &v) const {
330     vector<int> a6 = convert_base(this->z, base_digits, 6);
331     vector<int> b6 = convert_base(v.z, base_digits, 6);
332     vll a(a6.begin(), a6.end());
333     vll b(b6.begin(), b6.end());
334     while (a.size() < b.size()) a.push_back(0);
335     while (b.size() < a.size()) b.push_back(0);
336     while (a.size() & (a.size() - 1)) a.push_back(0), b.push_back(0);
337     vll c = karatsubaMultiply(a, b);
338     bigint res;
339     res.sign = sign * v.sign;
340     for (int i = 0, carry = 0; i < c.size(); i++) {
341         ll cur = c[i] + carry;
342         res.z.push_back((int) (cur % 1000000));
343         carry = (int) (cur / 1000000);
344     }
345     res.z = convert_base(res.z, 6, base_digits);
346     res.trim();
347     return res;
348 }
349 };

```

## 7.4 待修莫队

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ACM_LOCAL
4  const int N = 2e6 + 10;

```

```

5  int n, m, k, a[N], cnt[N], qnum, cnum, pos[N];
6  int Ans, ans[N], x, y;
7  char op;
8  inline int read(){
9      int s=0,w=1;
10     char ch=getchar();
11     while(ch<'0' || ch>'9'){if(ch=='-')w=-1;ch=getchar();}
12     while(ch>='0' && ch<='9') s=s*10+ch-'0',ch=getchar();
13     return s*w;
14 }
15
16 struct node {
17     int l, r, pre, id;
18 }q[N];
19
20 struct node2 {
21     int val, pos;
22 }c[N];
23
24 bool cmp(node x, node y) {
25     if (x.l != y.l) return pos[x.l] < pos[y.l];
26     if (x.r != y.r) return pos[x.r] < pos[y.r];
27     return x.pre < y.pre;
28 }
29
30 inline void add(int x) {if (++cnt[a[x]] == 1) ++Ans;}
31
32 inline void sub(int x) {if (--cnt[a[x]] == 0) --Ans;}
33
34
35 inline void work(int now, int i) {
36     if (c[now].pos >= q[i].l && c[now].pos <= q[i].r) {
37         if (--cnt[a[c[now].pos]] == 0) --Ans;
38         if (++cnt[c[now].val] == 1) ++Ans;
39     }
40     swap(c[now].val, a[c[now].pos]); // 浜ゆ嶲鎷栧瑰彨鑽勯€煎拃鍒€鐕€濳鑽勯€煎纭紉緇冨笗究鏁堢爲€€鑽勯€煎祿鍊
41     櫳 鍒€?
42 }
43
44 void solve() {
45     n = read(), m = read();
46     int siz = pow(n, 2.0 / 3.0);
47     for (int i = 1; i <= n; ++i) a[i] = read(), pos[i] = (i - 1) / siz + 1;
48
49     while (m--) {
50         op = getchar();
51         if (op == 'Q') {
52             q[++qnum].l = read();
53             q[qnum].r = read();
54             q[qnum].id = qnum;
55             q[qnum].pre = cnum;
56         }
57         else {
58             c[++cnum].pos = read();
59             c[cnum].val = read();
60         }
61     }
62     sort(q+1, q+qnum+1, cmp);

```

```

63     int l = 1, r = 0, now = 0;
64     for (int i = 1; i <= qnum; ++i) {
65         while (q[i].l < l) add(--l);
66         while (q[i].r > r) add(++r);
67         while (q[i].l > l) sub(l++);
68         while (q[i].r < r) sub(r--);
69         while (now < q[i].pre) work(++now, i);
70         while (now > q[i].pre) work(now--, i);
71         ans[q[i].id] = Ans;
72     }
73
74     for (int i = 1; i <= qnum; ++i) printf("%d\n", ans[i]);
75 }
76
77 signed main() {
78     //ios_base::sync_with_stdio(false);
79     //cin.tie(0);
80     //cout.tie(0);
81     #ifdef ACM_LOCAL
82         freopen("in.txt", "r", stdin);
83         freopen("out.txt", "w", stdout);
84     //#else
85         solve();
86     #endif
87     return 0;
88 }

```

## 7.5 对拍

```

1  /**\0 \0 \0 */
2  //\0 \0 \0 \0 \0 f°test.cpp biaoda.cpp data.cpp input.txt duipai.bat:
3  duipai.bat:
4  :again
5  data > input.txt
6  biaoda < input.txt > biaoda_out.txt
7  test < input.txt > test_out.txt
8  fc biaoda_out.txt test_out.txt
9  if not errorlevel 1 goto again
10 pause

```

## 7.6 二进制压缩枚举子集

```

1  for (int sub = S; sub; sub = (sub - 1) & S) {
2      // sub 为 S 的子集
3  }
4
5  a ^ b = c    -->    c ^ b = a

```

## 7.7 二维差分

```

1  void Insert(int x1, int y1, int x2, int y2, int v) {
2      c[x1][y1] += v;
3      c[x1][y2+1] -= v;
4      c[x2+1][y1] -= v;
5      c[x2+1][y2+1] += v;
6  }

```



## 7.8 分块

```

1 int block = sqrt(n);
2 int num = n / block + (n % block ? 1 : 0);
3 for (int i = 1; i <= num; i++) {
4     l[i] = (i - 1) * block + 1;
5     r[i] = i * block;
6 }
7 r[num] = n;
8 for (int i = 1; i <= n; i++) belong[i] = (i - 1) / block + 1;

```

## 7.9 高精度

```

1 #include<cstdio>
2 #include <cassert>
3 #include<cstring>
4 #include<algorithm>
5 using namespace std;
6 const int MOD=10000;
7 const int B=10000;
8 const int SIZEN=505;
9 const int L=505;
10 struct Mat{
11     int num[40][40];
12     void init(int n){
13         for(int i=0;i<n;i++)
14             for(int j=0;j<n;j++)
15                 num[i][j]=i*n+j;
16     }
17     void change(int n){
18         int t_num[40][40];
19         for(int i=0;i<n;i++)
20             for(int j=0;j<n;j++) t_num[j][n-i-1]=num[i][j];
21         for(int i=0;i<n;i++)
22             for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
23     }
24     void change1(int n){
25         int t_num[40][40];
26         for(int i=0;i<n;i++)
27             for(int j=0;j<n;j++) t_num[i][n-j-1]=num[i][j];
28         for(int i=0;i<n;i++)
29             for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
30     }
31     void change2(int n){
32         int t_num[40][40];
33         for(int i=0;i<n;i++)
34             for(int j=0;j<n;j++) t_num[n-i-1][j]=num[i][j];
35         for(int i=0;i<n;i++)
36             for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
37     }
38     void change3(int n){
39         int t_num[40][40];
40         for(int i=0;i<n;i++)
41             for(int j=0;j<n;j++) t_num[j][i]=num[i][j];
42         for(int i=0;i<n;i++)
43             for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
44     }
45     void change4(int n){

```

```

46     int t_num[40][40];
47     for(int i=0;i<n;i++)
48         for(int j=0;j<n;j++) t_num[n-1-j][n-1-i]=num[i][j];
49     for(int i=0;i<n;i++)
50         for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
51 }
52 void output(int n){
53     for(int i=0;i<n;i++){
54         for(int j=0;j<n;j++) printf("%d ",num[i][j]);
55         printf("\n");
56     }
57 }
58 };
59 struct BigInteger {
60     BigInteger(int number = 0) : length(!!number) {
61         assert(0 <= number && number < B);
62         memset(digit, 0, sizeof(digit));
63         digit[0] = number;
64     }
65
66     BigInteger normalize() {
67         while (length && !digit[length - 1]) {
68             length --;
69         }
70         return *this;
71     }
72
73     int operator[](int index) const {
74         return digit[index];
75     }
76
77     int& operator[](int index) {
78         return digit[index];
79     }
80
81     void output(){
82         printf("%d",digit[length-1]);
83         for(int i=length-2;i>=0;i--) printf("%04d",digit[i]);
84         printf("\n");
85     }
86
87     int length, digit[L];
88 };
89
90 bool operator < (const BigInteger &a, const BigInteger &b)
91 {
92     if (a.length != b.length) {
93         return a.length < b.length;
94     }
95     for (int i = 0; i < a.length; ++ i) {
96         if (a[i] != b[i]) {
97             return a[i] < b[i];
98         }
99     }
100     return false;
101 }
102
103 BigInteger operator + (const BigInteger &a, const BigInteger &b)
104 {

```

```

105     BigInteger c;
106     c.length = std::max(a.length, b.length) + 1;
107     for (int i = 0, delta = 0; i < c.length; ++ i) {
108         delta += a[i] + b[i];
109         c[i] = delta % B;
110         delta /= B;
111     }
112     return c.normalize();
113 }
114
115 BigInteger operator - (const BigInteger &a, int b)
116 {
117     assert(0 <= b && b < B);
118     BigInteger c;
119     c.length = a.length;
120     for (int i = 0, delta = -b; i < a.length; ++ i) {
121         delta += a[i];
122         c[i] = delta;
123         delta = 0;
124         if (c[i] < 0) {
125             c[i] += B;
126             delta = -1;
127         }
128     }
129     return c.normalize();
130 }
131
132 BigInteger operator * (const BigInteger &a, const BigInteger &b)
133 {
134     BigInteger c;
135     c.length = a.length + b.length;
136     for (int i = 0; i < a.length; ++ i) {
137         for (int j = 0, delta = 0; j <= b.length; ++ j) {
138             delta += a[i] * b[j] + c[i + j];
139             c[i + j] = delta % B;
140             delta /= B;
141         }
142     }
143     return c.normalize();
144 }
145
146 BigInteger operator / (const BigInteger &a, int b)
147 {
148     assert(0 <= b && b < B);
149     BigInteger c;
150     c.length = a.length;
151     for (int i = c.length - 1, delta = 0; i >= 0; -- i) {
152         delta = delta * B + a[i];
153         c[i] = delta / b;
154         delta %= b;
155     }
156     return c.normalize();
157 }
158 BigInteger operator ^(const BigInteger &a, int b){
159     BigInteger ret, ta;
160     ret=1;ta=a;
161     while(b){
162         if(b&1) ret=ret*ta;
163         ta=ta*ta;

```

```

164         b>>=1;
165     }
166     return ret;
167 }
168 Mat mat;
169 BigInteger ret,tmp;
170 void solve(int n,int c){
171     ret=0;
172     ret.normalize();
173     tmp=c;
174     tmp.normalize();
175     if(n%2==0){
176         ret=ret+(tmp^(n*n));
177         ret=ret+(tmp^(n*n/4));
178         ret=ret+(tmp^(n*n/2));
179         ret=ret+(tmp^(n*n/4));
180         ret=ret+(tmp^(n*n/2))*2;
181         ret=ret+(tmp^((n*n-n)/2+n))*2;
182     }
183     else{
184         ret=ret+(tmp^(n*n));
185         ret=ret+(tmp^(n*n-1)/4+1);
186         ret=ret+(tmp^(n*n-1)/2+1);
187         ret=ret+(tmp^(n*n-1)/4+1);
188         ret=ret+(tmp^((n*n-n)/2+n))*2;
189         ret=ret+(tmp^((n*n-n)/2+n))*2;
190     }
191     ret=ret/8;
192     ret.output();
193 }
194 int main()
195 {
196     int n,c;
197     while(scanf("%d%d",&n,&c)!=EOF)
198         solve(n,c);
199 }

```

## 7.10 环形均分

```

1  //
2  // Created by SANZONG on 2021/7/8.
3  //
4  #include "bits/stdc++.h"
5
6  #define int long long
7  using namespace std;
8  const int maxn = 3000000;
9  int arr[maxn];
10 int sum[maxn];
11 signed main() {
12     // freopen("in.txt","r",stdin);
13     //前缀和为0说明前一段内部可以自己解决, 其内部的前缀和绝对值相加即为前一段的子段转移花费
14     int n;
15     cin >> n;
16     int ave = 0;
17     for (int i = 1; i <= n; ++i) {
18         cin >> arr[i];
19         ave += arr[i];

```

```

20     }
21     ave /= n;
22     for (int i = 1; i <= n; ++i) {
23         arr[i] -= ave;
24         sum[i] = sum[i-1] + arr[i];
25     }
26     sort(sum+1, sum+1+n);
27     int mid = (n+1)/2;
28     int ans = 0;
29     for (int i = 1; i <= n; ++i) {
30         ans += abs(sum[i] - sum[mid]);    //做前缀和相加模拟出了传递时的花费
31     }
32     cout << ans << endl;
33 }

```

### 7.11 回滚莫队

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  // #define ACM_LOCAL
4  typedef long long ll;
5  const int N = 2e5 + 10;
6  int n, m, k, a[N], pos[N], xl[N], xr[N], b[N], _cnt[N], st[N], ed[N], clear[N];
7  int Max, temp, ans[N], num;
8
9  inline int read(){
10     int s=0, w=1;
11     char ch=getchar();
12     while(ch<'0' || ch>'9'){if(ch=='-')w=-1; ch=getchar();}
13     while(ch>='0' && ch<='9') s=s*10+ch-'0', ch=getchar();
14     return s*w;
15 }
16
17 struct node {
18     int l, r, id;
19     bool operator < (node xx) const {
20         if(pos[l] == pos[xx.l]) return r < xx.r;
21         else return pos[l] < pos[xx.l];
22     }
23 } q[N];
24
25 void solve() {
26     n = read();
27     for (int i = 1; i <= n; i++) a[i] = read(), b[i] = a[i];
28
29     sort(b+1, b+n+1);
30     int tot = unique(b+1, b+n+1) - b - 1;
31     for (int i = 1; i <= n; i++) a[i] = lower_bound(b+1, b+tot+1, a[i]) - b;
32
33     m = read();
34     for (int i = 1; i <= m; i++) q[i].l = read(), q[i].r = read(), q[i].id = i;
35
36     int siz = sqrt(n);
37     for (int i = 1; i <= n; i++){
38         pos[i] = i / siz;
39         xl[pos[i]] = (xl[pos[i]] == 0 || xl[pos[i]] > i) ? i : xl[pos[i]];
40         xr[pos[i]] = (xr[pos[i]] < i) ? i : xr[pos[i]];
41     }

```

```

42
43     sort(q+1, q+m+1);
44
45     int l = 1, r = 0, lastblock = -1;
46
47     for (int i = 1; i <= m; i++) {
48         if (pos[q[i].l] == pos[q[i].r]) {
49             int temp = 0;
50             for (int j = q[i].l; j <= q[i].r; j++) {
51                 if (!_cnt[a[j]]) _cnt[a[j]] = j;
52                 else temp = max(temp, j - _cnt[a[j]]);
53             }
54             for (int j = q[i].l; j <= q[i].r; j++) _cnt[a[j]] = 0;
55             ans[q[i].id] = temp;
56         }
57         else {
58             if (lastblock != pos[q[i].l]) {
59                 l = xr[pos[q[i].l]] + 1;
60                 r = l - 1;
61                 for (int j = 1; j <= num; j++) st[clear[j]] = ed[clear[j]] = 0;
62                 num = 0;
63                 Max = 0, lastblock = pos[q[i].l];
64             }
65             while (r < q[i].r) {
66                 r++;
67                 ed[a[r]] = r;
68                 clear[++num] = a[r];
69                 if (!st[a[r]]) st[a[r]] = r;
70                 Max = max(Max, r - st[a[r]]);
71             }
72             temp = Max;
73             while (l > q[i].l) {
74                 l--;
75                 if (ed[a[l]]) temp = max(temp, ed[a[l]] - l);
76                 else ed[a[l]] = l;
77             }
78             while (l < xr[pos[q[i].l]] + 1) {
79                 if (ed[a[l]] == l) ed[a[l]] = st[a[l]] = 0;
80                 l++;
81             }
82             ans[q[i].id] = temp;
83         }
84     }
85
86     for (int i = 1; i <= m; i++) printf("%d\n", ans[i]);
87 }
88
89 signed main() {
90     //ios_base::sync_with_stdio(false);
91     //cin.tie(0);
92     //cout.tie(0);
93 #ifdef ACM_LOCAL
94     freopen("in.txt", "r", stdin);
95     freopen("out.txt", "w", stdout);
96 #endif
97     solve();
98     return 0;
99 }

```

## 7.12 离散化

```

1 struct Hash {
2     int b[N], tot;
3     void init() {tot = 0;}
4     void insert(int x) {b[++tot] = x;}
5     void build() {
6         sort(b+1, b+1+tot);
7         tot = unique(b+1, b+tot+1) - (b+1);
8     }
9     int pos(int x) {return lower_bound(b+1, b+tot+1, x) - b;}
10 }ha;

```

## 7.13 莫队 + 树状数组

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ACM_LOCAL
4 const int N = 1e5 + 10;
5 int n, m, k, a[N], pos[N], b[N], upa[N], prea[N], sum[N], Ans, ans[N];
6
7 inline int read(){
8     int s=0,w=1;
9     char ch=getchar();
10    while(ch<'0' || ch>'9'){if(ch=='-')w=-1;ch=getchar();}
11    while(ch>='0' && ch<='9') s=s*10+ch-'0',ch=getchar();
12    return s*w;
13 }
14
15 struct Q{
16     int l, r, id;
17 }p[N];
18
19 bool cmp(Q x, Q y) {
20     if (pos[x.l] == pos[y.l]) return x.r < y.r;
21     else return pos[x.l] < pos[y.l];
22 }
23
24 inline int lowbit(int x) {return x & -x;}
25
26 void add(int p, int v) {for (int i = p; i <= 3*n+10; i += lowbit(i)) sum[i] += v;}
27
28 int query(int p) {
29     int res = 0;
30     for (int i = p; i > 0; i -= lowbit(i)) res += sum[i];
31     return res;
32 }
33 int query_(int l, int r) {return query(r) - query(l-1);}
34
35 void solve() {
36     n = read(), m = read(), k = read();
37     int tot = 0;
38     int siz = sqrt(n);
39     for (int i = 1; i <= n; i++) a[i] = read();
40
41     for (int i = 1; i <= n; i++) {
42         upa[i] = a[i] + k, prea[i] = a[i] - k;
43         b[++tot] = a[i], b[++tot] = upa[i], b[++tot] = prea[i];

```

```

44     }
45     sort(b + 1, b + tot + 1);
46     int cnt = unique(b + 1, b + tot + 1) - b - 1;
47     for (int i = 1; i <= n; i++) {
48         a[i] = lower_bound(b+1, b+cnt+1, a[i]) - b;
49         upa[i] = lower_bound(b+1, b+cnt+1, upa[i]) - b;
50         prea[i] = lower_bound(b+1, b+cnt+1, prea[i]) - b;
51     }
52
53     for (int i = 1; i <= m; i++) {
54         p[i].l = read(), p[i].r = read();
55         p[i].id = i;
56     }
57     sort(p + 1, p + m + 1, cmp);
58     int l = 1, r = 0;
59     for (int i = 1; i <= m; i++) {
60         while (p[i].l < l) {
61             l--;
62             Ans += query_(prea[l], upa[l]);
63             add(a[l], 1);
64         }
65         while (p[i].l > l) {
66             add(a[l], -1);
67             Ans -= query_(prea[l], upa[l]);
68             l++;
69         }
70         while (p[i].r < r) {
71             add(a[r], -1);
72             Ans -= query_(prea[r], upa[r]);
73             r--;
74         }
75         while (p[i].r > r) {
76             r++;
77             Ans += query_(prea[r], upa[r]);
78             add(a[r], 1);
79         }
80         ans[p[i].id] = Ans;
81     }
82     for (int i = 1; i <= m; i++) printf("%d\n", ans[i]);
83 }
84
85 signed main() {
86     ios_base::sync_with_stdio(false);
87     cin.tie(0);
88     cout.tie(0);
89     #ifdef ACM_LOCAL
90         freopen("in.txt", "r", stdin);
91         freopen("out.txt", "w", stdout);
92     //#else
93         solve();
94     #endif
95     return 0;
96 }

```

## 7.14 普通莫队

```

1  #include <bits/stdc++.h>
2  //#define ACM_LOCAL

```



```
3 using namespace std;
4 const int N = 5e4 + 10;
5
6 inline int read() {
7     int s = 0, w = 1;
8     char ch = getchar();
9     while (ch < '0' || ch > '9') { if (ch == '-') w = -1; ch = getchar(); }
10    while (ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch = getchar();
11    return s * w;
12 }
13
14 int pos[N], a[N], vis[2 * N];
15 int n, q, num, k;
16 long long res, ans[N];
17
18 struct Q {
19     int l, r, id;
20 } p[N];
21
22 bool cmp(Q x, Q y) {
23     if (pos[x.l] == pos[y.l]) return x.r < y.r;
24     else return pos[x.l] < pos[y.l];
25 }
26
27 void add(int x) {
28
29 }
30
31 void sub(int x) {
32
33 }
34
35 void solve() {
36     n = read(), q = read(), k = read();
37     for (int i = 1; i <= n; i++) a[i] = read();
38
39     int siz = sqrt(n);
40     for (int i = 1; i <= n; i++) pos[i] = i / siz;
41
42     for (int i = 1; i <= q; i++) {
43         p[i].l = read(), p[i].r = read();
44         p[i].id = i;
45     }
46     sort(p + 1, p + q + 1, cmp);
47
48     int l = 1, r = 0;
49     for (int i = 1; i <= q; i++) {
50         while (p[i].l < l) add(--l);
51         while (p[i].r > r) add(++r);
52         while (p[i].l > l) sub(l++);
53         while (p[i].r < r) sub(r--);
54         ans[p[i].id] = res;
55     }
56     for (int i = 1; i <= q; i++) {
57         printf("%lld\n", ans[i]);
58     }
59 }
```

## 7.15 三维偏序问题 (CDQ 分治)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define fi first
4  #define se second
5  #define re register
6  typedef long long ll;
7  typedef pair<int, int> PII;
8  typedef unsigned long long ull;
9  const int N = 5e5 + 10, M = 1e6 + 5, INF = 1e9;
10 const int MOD = 1e9 + 7;
11 int n, k, cnt[N], ans[N], f[N];
12 struct node {
13     int a, b, c, cnt, id;
14     bool operator < (const node &rhs) {
15         if (a == rhs.a) {
16             if (b == rhs.b) return c < rhs.c;
17             return b < rhs.b;
18         }
19         return a < rhs.a;
20     }
21 };
22 node p[N], q[N], tmp[N];
23 int lowbit(int x) {return x&-x;}
24 void add(int x, int c) {
25     for (int i = x; i <= k; i += lowbit(i)) cnt[i] += c;
26 }
27 int query(int x) {
28     int res = 0;
29     for (int i = x; i > 0; i -= lowbit(i)) res += cnt[i];
30     return res;
31 }
32 void clear(int x) {
33     for (int i = x; i <= k; i += lowbit(i)) {
34         if (cnt[i]) cnt[i] = 0;
35         else break;
36     }
37 }
38 void CDQ(int l, int r) {
39     if (l == r) return;
40     int mid = (l + r) >> 1;
41     CDQ(l, mid);
42     CDQ(mid+1, r);
43     int t1 = l, t2 = mid+1;
44     for (int i = l; i <= r; i++) {
45         if ((t1 <= mid && q[t1].b <= q[t2].b) || t2 > r) {
46             add(q[t1].c, q[t1].cnt);
47             tmp[i] = q[t1++];
48         } else {
49             ans[q[t2].id] += query(q[t2].c);
50             tmp[i] = q[t2++];
51         }
52     }
53     for (int i = l; i <= r; i++) q[i] = tmp[i], clear(q[i].c);
54 }
55 void solve() {
56     cin >> n >> k;
57     for (int i = 1; i <= n; i++) cin >> p[i].a >> p[i].b >> p[i].c;

```

```

58     sort(p+1, p+n+1);
59     int tot = 0;
60     q[++tot] = {p[1].a, p[1].b, p[1].c, 1, 1};
61     for (int i = 2; i <= n; i++) {
62         if (p[i].a == p[i-1].a && p[i].b == p[i-1].b && p[i].c == p[i-1].c) {
63             q[tot].cnt++;
64         } else {
65             q[++tot] = {p[i].a, p[i].b, p[i].c, 1, tot};
66         }
67     }
68     CDQ(1, tot);
69     for (int i = 1; i <= tot; i++) {
70         f[ans[q[i].id] + q[i].cnt - 1] += q[i].cnt;
71     }
72     for (int i = 0; i < n; i++) printf("%d\n", f[i]);
73 }
74
75 signed main() {
76     ios_base::sync_with_stdio(false);
77     cin.tie(0);
78     cout.tie(0);
79 #ifdef ACM_LOCAL
80     freopen("input", "r", stdin);
81     freopen("output", "w", stdout);
82 #endif
83     solve();
84     return 0;
85 }

```

## 7.16 树上莫队

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1e5 + 5;
4  inline int read() {
5      int s = 0, w = 1;
6      char ch = getchar();
7      while (ch < '0' || ch > '9') { if (ch == '-') w = -1; ch = getchar(); }
8      while (ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch = getchar();
9      return s * w;
10 }
11 int n, m;
12 struct edge {
13     int to, next;
14 } e[N<<1];
15
16 struct Q {
17     int l, r, id, lca;
18 } p[N];
19
20 int h[N], cnt, tot, in[N], out[N], a[N], b[N], pos[N], son[N], rnk[N], siz[N], top[N],
    d[N], fa[N], res, used[N], ans[N], vis[N];
21
22 void add(int u, int v) {
23     e[cnt].to = v;
24     e[cnt].next = h[u];
25     h[u] = cnt++;
26 }

```

```

27
28 bool cmp(Q x, Q y) {
29     if (pos[x.l] == pos[y.l]) return x.r < y.r;
30     else return pos[x.l] < pos[y.l];
31 }
32
33 void discrete() {
34     sort(b + 1, b + n + 1);
35     int num = unique(b + 1, b + n + 1) - b - 1;
36     for (int i = 1; i <= n; i++) a[i] = lower_bound(b + 1, b + num + 1, a[i]) - b;
37 }
38
39 void dfs1(int u) {
40     son[u] = -1;
41     siz[u] = 1;
42     in[u] = ++tot, rnk[tot] = u;
43     for (int i = h[u]; ~i; i = e[i].next) {
44         int v = e[i].to;
45         if (!d[v]) {
46             d[v] = d[u] + 1;
47             fa[v] = u;
48             dfs1(v);
49             siz[u] += siz[v];
50             if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
51         }
52     }
53     out[u] = ++tot, rnk[tot] = u;
54 }
55
56 void dfs2(int u, int t) {
57     top[u] = t;
58     if (son[u] == -1) return;
59     dfs2(son[u], t);
60     for (int i = h[u]; ~i; i = e[i].next) {
61         int v = e[i].to;
62         if (v != son[u] && v != fa[u]) dfs2(v, v);
63     }
64 }
65
66 int lca(int u, int v) {
67     while (top[u] != top[v]) {
68         if (d[top[u]] > d[top[v]])
69             u = fa[top[u]];
70         else
71             v = fa[top[v]];
72     }
73     return d[u] > d[v] ? v : u;
74 }
75
76 void Add(int x) {
77     if (++vis[a[x]] == 1) ++res;
78 }
79
80 void Sub(int x) {
81     if (--vis[a[x]] == 0) --res;
82 }
83
84 void ADD(int x) {
85     used[x] ? Sub(x) : Add(x);

```

```
86     used[x] ^= 1;
87 }
88
89 int main() {
90     n = read(), m = read();
91     memset(h, -1, sizeof h);
92     int siz = sqrt(n);
93     for (int i = 1; i <= n; i++) a[i] = read(), b[i] = a[i];
94     for (int i = 1; i <= 2 * n; i++) pos[i] = i / siz;
95     discrete();
96     for (int i = 1; i <= n - 1; i++) {
97         int x, y;
98         x = read(), y = read();
99         add(x, y), add(y, x);
100     }
101     d[1] = 1;
102     dfs1(1);
103     dfs2(1, 1);
104     for (int i = 1; i <= m; i++) {
105         int x, y;
106         x = read(), y = read();
107         if (in[x] > in[y]) swap(x, y);
108         int lca_ = lca(x, y);
109         p[i].id = i;
110         if (lca_ == x) p[i].l = in[x], p[i].r = in[y];
111         else p[i].l = out[x], p[i].r = in[y], p[i].lca = lca_;
112     }
113     sort(p + 1, p + m + 1, cmp);
114
115     int l = 1, r = 0;
116     for (int i = 1; i <= m; i++) {
117         while (p[i].l < l) ADD(rnk[--l]);
118         while (p[i].r > r) ADD(rnk[++r]);
119         while (p[i].l > l) ADD(rnk[l++]);
120         while (p[i].r < r) ADD(rnk[r--]);
121         if (p[i].lca) ADD(p[i].lca);
122         ans[p[i].id] = res;
123         if (p[i].lca) ADD(p[i].lca);
124     }
125     for (int i = 1; i <= m; i++) printf("%d\n", ans[i]);
126 }
```

## 8 字符串

```

1  //
2  // Created by acer on 2021/2/16.
3  //
4  //判断子串, 不同子串个数, 所有子串字典序第i大, 最长公共子串
5
6  #include <cstring>
7  #include <iostream>
8  #include "string"
9
10
11 #define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 1e6 + 10;
14 int id[MAXN<<1];
15 int visit[MAXN<<1];
16 int endpos[MAXN << 1];
17 int siz[MAXN << 1];
18 int len[MAXN << 1];
19 int ch[MAXN << 1][27];
20 int fa[MAXN << 1];
21 int last = 1;
22 int tot = 1;
23 int p;
24 int len1;
25 void add(int c) {
26     p = last;
27     last = ++tot;
28     int np = last;
29     siz[np] = 1;
30     len[np] = len[p] + 1;
31     for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
32     if (!p) fa[np] = 1;
33     else {
34         int q = ch[p][c];
35         if (len[q] == len[p] + 1) fa[np] = q;
36         else {
37             int nq = ++tot;
38             memcpy(ch[nq], ch[q], sizeof(ch[nq]));
39             fa[nq] = fa[q];
40             len[nq] = len[p] + 1;
41             fa[np] = fa[q] = nq;
42             for (; p && ch[p][c] == q; p = fa[p]) {
43                 ch[p][c] = nq;
44             }
45         }
46     }
47 }
48
49 }
50 void getTuopu()
51 {
52     for (int k = 1; k <= tot; ++k) {          //给每个点赋权值
53         endpos[len[k]] ++;
54     }
55     for (int m = 1; m <= tot; ++m) {          //获得长度大于等于m的点的总个数, 所以必然是不会相同的,
56         //故可以用来标记。
57         endpos[m] += endpos[m-1];
58     }
59 }

```

```

57     }
58     for (int n = 1; n <= tot; ++n) { //根据点出现顺序获得拓扑序
59         id[endpos[len[n]]--] = n;
60     }
61 }
62 char s[MAXN], s1[MAXN];
63
64 void doit()
65 {
66     for (int i = tot; i >= 1; --i) {
67         int p = id[i];
68         siz[fa[p]] += siz[p];
69     }
70 }
71
72 int solve(int w) {
73     int p = 1;
74     int tmp = 0;
75     int sum = 0;
76     for (int i = 1; i <= len1+len1; ++i) {
77         int v = s1[i]-'a';
78         if (ch[p][v])
79             p = ch[p][v], tmp++;
80         else{
81             while (p && !ch[p][v]) p = fa[p];
82             if (!p) p = 1, tmp = 0;
83             else
84                 tmp = len[p]+1, p = ch[p][v];
85         }
86         ///siz[p]是[i-l+1,i]的出现次数,我们需要的是[i-|t|+1,i]的
87         ///直到parent树上的祖先是孩子的后缀,我们一直往上fa
88         ///直到父亲的longest小于|t|
89
90         ///fa即删前缀字符, ch加后缀字符, 因为在同一个endpos, 所以跳完之后即使长度不等, endpos所包含的个数也是
          一样的
91         if(tmp >= len1)
92         {
93             while( len[fa[p]]>=len1 ) p = fa[p]; //fa的longest严格小于len1, 保证可以加
94             if( visit[p]!=w )
95                 sum+=siz[p], visit[p] = w;
96
97             tmp = len1;
98         }
99     }
100     return sum;
101 }
102
103 int main() {
104     scanf("%s", s + 1);
105     int l = strlen(s + 1);
106     for (int i = 1; i <= l; ++i) {
107         add(s[i] - 'a');
108     }
109     getTuopu();
110     doit();
111     int n;
112     cin >> n;
113     for (int j = 1; j <= n; ++j)
114     {

```

```

115
116     scanf("%s", s1+1);
117     len1 = strlen(s1+1);
118     for (int i = 1; i <= len1; ++i) {
119         s1[i+len1] = s1[i];
120     }
121
122     cout << solve(j) << endl;
123 }
124 }

```

## 8.1 (SAM)k 长子串最大出现次数

```

1 //
2 // Created by acer on 2021/2/16.
3 //
4 //判断子串, 不同子串个数, 所有子串字典序第i大, 最长公共子串
5
6 #include <cstring>
7 #include <iostream>
8 #include "string"
9
10
11 #define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 3e5 + 10;
14
15 int len[MAXN << 1]; int ch[MAXN << 1][27]; int fa[MAXN << 1]; int weig[MAXN << 1];
16 int last = 1; int tot = 1; int p; int size[MAXN << 1];
17 void add(int c) {
18     p = last;
19     last = ++tot;
20     int np = last;
21     size[np] = 1; //np节点表示的后缀出现过几次
22     len[np] = len[p] + 1;
23     for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
24     if (!p) fa[np] = 1;
25     else {
26         int q = ch[p][c];
27         if (len[q] == len[p] + 1) fa[np] = q;
28         else {
29             int nq = ++tot;
30             memcpy(ch[nq], ch[q], sizeof(ch[nq]));
31             fa[nq] = fa[q];
32             len[nq] = len[p] + 1;
33             fa[np] = fa[q] = nq;
34             for (; p && ch[p][c] == q; p = fa[p]) {
35                 ch[p][c] = nq;
36             }
37         }
38     }
39 }
40 }
41 char s[MAXN << 1]; int id[MAXN << 1]; int ans[MAXN << 1];
42 void getTuopu()
43 {
44     for (int k = 1; k <= tot; ++k) { //给每个点赋权值
45         weig[len[k]] ++;

```



```

46     }
47     for (int m = 1; m <= tot; ++m) { //获得长度小于等于m的点的总个数, 所以必然是不会相同的,
    故可以用来标记。
48         weig[m] += weig[m-1];
49     }
50     for (int n = 1; n <= tot; ++n) { //根据点出现顺序获得拓扑序
51         id[weig[len[n]]--] = n;
52     }
53 }
54 int main() {
55     scanf("%s", s + 1);
56     int l = strlen(s+1);
57     for (int i = 1; i <= l; ++i) {
58         add(s[i]-'a');
59     }
60     getTuopu();
61     for (int i = tot; i >= 1; --i) {
62         size[fa[id[i]]] += size[id[i]];
63         ans[len[id[i]]] = max(ans[len[id[i]]], size[id[i]]);
64     }
65     for(int i = tot; i >= 1; --i) ans[i] = max(ans[i], ans[i + 1]);
66     // for(int i = 1; i <= l; ++i) printf("%d\n", ans[i]);
67 }

```

## 8.2 (SAM) 第 k 小子串

```

1 //
2 // Created by acer on 2021/2/16.
3 //
4 //判断子串, 不同子串个数, 所有子串字典序第i大, 最长公共子串
5
6 #include <cstring>
7 #include <iostream>
8 #include "string"
9
10
11 #define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 3e5 + 10;
14
15 int len[MAXN << 1];
16 int ch[MAXN << 1][27];
17 int fa[MAXN << 1];
18 int last = 1;
19 int tot = 1;
20 int p;
21
22 void add(int c) {
23     p = last;
24     last = ++tot;
25     int np = last;
26     len[np] = len[p] + 1;
27     for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
28     if (!p) fa[np] = 1;
29     else {
30         int q = ch[p][c];
31         if (len[q] == len[p] + 1) fa[np] = q;
32         else {

```

```

33         int nq = ++tot;
34         memcpy(ch[nq], ch[q], sizeof(ch[nq]));
35         fa[nq] = fa[q];
36         len[nq] = len[p] + 1;
37         fa[np] = fa[q] = nq;
38         for (; p && ch[p][c] == q; p = fa[p]) {
39             ch[p][c] = nq;
40         }
41     }
42 }
43 }
44 }
45
46 char s[MAXN];
47 int count[MAXN<<1];
48 void dfs(int k)
49 {
50     if (count[k]) return;
51     count[k] = 1;
52     for (int i = 0; i <= 26; ++i) {
53         if (!ch[k][i]) continue;
54         dfs(ch[k][i]);
55         count[k] += count[ch[k][i]];
56     }
57 }
58 int main() {
59     scanf("%s", s + 1);
60     int l = strlen(s + 1);
61     for (int i = 1; i <= l; ++i) {
62         add(s[i] - 'a');
63     }
64     int T;
65     dfs(1);
66     scanf("%d", &T);
67     while (T--) {
68         int k;
69         int now = 1;
70         scanf("%d", &k);
71         while (k)
72         {
73             if (!k) break;
74             for (int i = 0; i <= 26; ++i) {
75                 if (ch[now][i]) {
76                     if (count[ch[now][i]] >= k) {
77                         putchar(i + 'a');
78                         --k;
79                         now = ch[now][i];
80                         break;
81                     } else
82                         k -= count[ch[now][i]];
83                 }
84             }
85         }
86         puts("");
87     }
88 }

```

//此处是实现算法的关键，通过从a遍历到z来计算字典序

//很明显，count的意义就是字典序为k的子串的首字母

//因为不能往前找，所以找到的第一个大于k的就肯定是属于答案的子串中

//这个字母包含的不够多

## 8.3 (SAM) 多串 lcs

```

1  //
2  // Created by acer on 2021/2/16.
3  //
4  //判断子串, 不同子串个数, 所有子串字典序第i大, 最长公共子串
5
6  #include <cstring>
7  #include <iostream>
8  #include "string"
9
10
11 #define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 1e5 + 10;
14 int mxlen[MAXN<<1];
15 int anslen[MAXN<<1];
16 int len[MAXN << 1];
17 int ch[MAXN << 1][27];
18 int fa[MAXN << 1];
19 int id[MAXN << 1];
20 int last = 1;
21 int tot = 1;
22
23 int p;
24 int visit[MAXN<<1];
25 void add(int c) {
26     p = last;
27     last = ++tot;
28     int np = last;
29     len[np] = len[p] + 1;
30     for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
31     if (!p) fa[np] = 1;
32     else {
33         int q = ch[p][c];
34         if (len[q] == len[p] + 1) fa[np] = q;
35         else {
36             int nq = ++tot;
37             memcpy(ch[nq], ch[q], sizeof(ch[nq]));
38             fa[nq] = fa[q];
39             len[nq] = len[p] + 1;
40             fa[np] = fa[q] = nq;
41             for (; p && ch[p][c] == q; p = fa[p]) {
42                 ch[p][c] = nq;
43             }
44         }
45     }
46 }
47
48 }
49 char s[MAXN], s1[MAXN];
50 int mx = 0;
51 void solve() {
52     p = 1;
53     int len1 = strlen(s1 + 1);
54     int tmp = 0;
55     for (int i = 1; i <= len1; ++i) {
56         int c = s1[i] - 'a';
57         if (ch[p][c]) {p = ch[p][c], tmp++;}

```

```

58     else {
59         while (p && !(ch[p][c])) p = fa[p];
60         if (!p) p = 1, tmp = 0;
61         else {tmp = len[p] + 1; p = ch[p][c];}
62     }
63     visit[p]++;
64     mxlen[p] = max(mxlen[p], tmp);
65 }
66 for (int j = tot; j >= 1; --j) {
67     int k = id[j];
68     anslen[k] = min(anslen[k], mxlen[k]);
69     if (anslen[k] && fa[k]) mxlen[fa[k]] = max(mxlen[fa[k]], len[fa[k]]);
70     mxlen[k] = 0;
71 }
72 }
73 }
74
75 int main() {
76     scanf("%s", s + 1);
77     int l = strlen(s + 1);
78     for (int i = 1; i <= l; ++i) {
79         add(s[i] - 'a');
80     }
81     for (int k = 1; k <= tot; ++k) {           //给每个点赋权值
82         anslen[len[k]] ++;
83     }
84     for (int m = 1; m <= tot; ++m) {           //获得长度小于等于m的点的总个数，所以必然是不会相同的，
85         //故可以用来标记。
86         anslen[m] += anslen[m-1];
87     }
88     for (int n = 1; n <= tot; ++n) {           //根据点出现顺序获得拓扑序
89         id[anslen[len[n]]--] = n;
90     }
91     while (scanf("%s", s1 + 1) != EOF)
92     {
93         solve();
94     }
95     for (int j = 1; j <= tot; ++j) {
96         mx = max(mx, anslen[j]);
97     }
98     printf("%d", mx);
99 }
100 }

```

## 8.4 ac 自动机

```

1
2 //多模式串匹配
3
4 #include <queue>
5 #include <cstdlib>
6 #include <cmath>
7 #include <cstdio>
8 #include <string>
9 #include <cstring>
10 #include <iostream>
11 #include <algorithm>

```

```

12 using namespace std;
13 const int maxn = 500000+9;
14 int trie[maxn][26]; //字典树
15 int cntword[maxn]; //记录该单词出现次数
16 int fail[maxn]; //失败时的回溯指针
17 int cnt = 0;
18 void insertWords(string s){
19     int root = 0;
20     for(int i=0;i<s.size();i++){
21         int next = s[i] - 'a';
22         if(!trie[root][next])
23             trie[root][next] = ++cnt;
24         root = trie[root][next];
25     }
26     cntword[root]++;
27 }
28 void getFail(){
29     queue <int>q;
30     for(int i=0;i<26;i++){ //将第二层所有出现了的字母扔进队列
31         if(trie[0][i]){
32             fail[trie[0][i]] = 0;
33             q.push(trie[0][i]);
34         }
35     }
36     while(!q.empty()){
37         int now = q.front();
38         q.pop();
39         for(int i=0;i<26;i++){
40             if(trie[now][i]){
41                 fail[trie[now][i]] = trie[fail[now]][i];
42                 q.push(trie[now][i]);
43             }
44             else trie[now][i] = trie[fail[now]][i];
45         }
46     }
47 }
48
49 int query(string s){
50     int now = 0,ans = 0;
51     for(int i=0;i<s.size();i++){
52         now = trie[now][s[i]-'a'];
53         for(int j=now;j && cntword[j]!=-1;j=fail[j]){
54             ans += cntword[j];
55             cntword[j] = -1;
56         }
57     }
58     return ans;
59 }
60 void init()
61 {
62     for(int i = 0 ; i<= cnt ; i++)
63     {
64         memset(trie[i], 0, sizeof(trie[i]));
65     }
66     memset(cntword,0,sizeof(cntword));
67     cnt = 0;
68 }

```

## 8.5 kmp

```
1 //
2 // Created by acer on 2021/2/8.
3 //
4 #define int long long
5 const int maxn = 1000000;
6 int nxt[maxn];
7 void search(string s)
8 {
9     int k=-1;
10    nxt[0]=-1;
11    int j=0;
12    while(j<s.length())
13    {
14        if(k<0 || s[k]==s[j])
15        {
16            j++;
17            k++;
18            nxt[j] = k;
19        }
20        else {
21            k=nxt[k];
22        }
23    }
24 }
25 }
```

## 8.6 二维字符串哈希

```
1 #include <iostream>
2 #include <algorithm>
3 #include <unordered_map>
4
5 using namespace std;
6
7 typedef unsigned long long ULL;
8
9 const int N = 1010, M = N * N, P = 131;
10
11 int n, m, a, b;
12 ULL hashv[N][N], p[M];
13 char str[N];
14
15 ULL calc(ULL f[], int l, int r)
16 {
17     return f[r] - f[l - 1] * p[r - l + 1];
18 }
19
20 int main()
21 {
22     scanf("%d%d%d%d", &n, &m, &a, &b);
23
24     p[0] = 1;
25     for (int i = 1; i <= n * m; i ++ ) p[i] = p[i - 1] * P;
26
27     for (int i = 1; i <= n; i ++ )
28     {
```

```

29     scanf("%s", str + 1);
30     for (int j = 1; j <= m; j ++ ) hashv[i][j] = hashv[i][j - 1] * P + str[j] - '0'
31     ;
32 }
33 unordered_map<ULL, int> S;
34 for (int i = b; i <= m; i ++ )
35 {
36     ULL s = 0;
37     int l = i - b + 1, r = i;
38     for (int j = 1; j <= n; j ++ )
39     {
40         s = s * p[b] + calc(hashv[j], l, r);
41         if (j - a > 0) s -= calc(hashv[j - a], l, r) * p[a * b];
42         if (j >= a) S[s] = 1;
43     }
44 }
45
46 int Q;
47 scanf("%d", &Q);
48 while (Q -- )
49 {
50     ULL s = 0;
51     for (int i = 0; i < a; i ++ )
52     {
53         scanf("%s", str);
54         for (int j = 0; j < b; j ++ ) s = s * P + str[j] - '0';
55     }
56     if (S[s]) puts("1");
57     else puts("0");
58 }
59
60 return 0;
61 }

```

## 8.7 后缀自动机

```

1  //
2  // Created by acer on 2021/2/16.
3  //
4  //判断子串, 不同子串个数, 所有子串字典序第i大, 最长公共子串
5
6  #include <cstring>
7  #include <iostream>
8  #include "string"
9
10
11 #define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 3e5 + 10;
14
15 int len[MAXN << 1];
16 int ch[MAXN << 1][27];
17 int fa[MAXN << 1];
18 int last = 1;
19 int tot = 1;
20 int p;
21

```

```

22 void add(int c) {
23     p = last;
24     last = ++tot;
25     int np = last;
26     len[np] = len[p] + 1;
27     for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
28     if (!p) fa[np] = 1;
29     else {
30         int q = ch[p][c];
31         if (len[q] == len[p] + 1) fa[np] = q;
32         else {
33             int nq = ++tot;
34             memcpy(ch[nq], ch[q], sizeof(ch[nq]));
35             fa[nq] = fa[q];
36             len[nq] = len[p] + 1;
37             fa[np] = fa[q] = nq;
38             for (; p && ch[p][c] == q; p = fa[p]) {
39                 ch[p][c] = nq;
40             }
41         }
42     }
43 }
44
45 }
46
47 char s[MAXN], s1[MAXN];
48
49 int solve() {
50     int len1 = strlen(s1 + 1);
51     int ans = 0; int tmp = 0;
52     for (int i = 1; i <= len1; ++i) {
53
54         int c = s1[i] - 'a';
55         if (ch[p][c]) p = ch[p][c], tmp++;
56         else {
57             while (p && !(ch[p][c]))
58                 p = fa[p];
59             if (!p) p = 1, tmp = 0;
60             else {
61                 tmp = len[p] + 1;
62                 p = ch[p][c];
63             }
64         }
65         ans = max(ans, tmp);
66     }
67     return ans;
68 }
69 int id[MAXN << 1];
70 int weig[MAXN << 1];
71 void getTuopu()
72 {
73     for (int k = 1; k <= tot; ++k) { //给每个点赋权值
74         weig[len[k]] ++;
75     }
76     for (int m = 1; m <= tot; ++m) { //获得长度大于等于m的点的总个数，所以必然是不会相同的，
77         weig[m] += weig[m-1]; //故可以用来标记。
78     }
79     for (int n = 1; n <= tot; ++n) { //根据点出现顺序获得拓扑序

```



```

80         id[weig[len[n]]--] = n;
81     //         cout << weig[len[n]]+1 << ' ' << id[weig[len[n]]+1] << endl;
82     }
83     //     for (int j = 1; j <= tot; ++j) {
84     //         cout << id[j] << endl;
85     //     }
86     }
87 int main() {
88     //     freopen("in.txt", "r", stdin);
89     //     freopen("out.txt", "w", stdout);
90     scanf("%s", s + 1);
91     int l = strlen(s + 1);
92     for (int i = 1; i <= l; ++i) {
93         add(s[i] - 'a');
94     }
95     scanf("%s", s1 + 1);
96     cout << solve();
97 }

```

## 8.8 回文自动机

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int MAXN = 500000, MAXK = 26;
4  char str[ MAXN + 5 ]; long long weight[MAXN+5];
5  struct Palindrome_Automaton{
6      int Size //总节点
7      , Last , Root0 , Root1 , Trans[ MAXN + 5 ][ MAXK + 5 ] , Fail[ MAXN + 5 ];
8      long long Len[ MAXN + 5 ];
9
10     Palindrome_Automaton( ) {
11         Root0 = Size ++ , Root1 = Size; Last = Root1;
12         Len[ Root0 ] = 0 , Fail[ Root0 ] = Root1;
13         Len[ Root1 ] = -1 , Fail[ Root1 ] = Root1;
14     }
15     void Extend( int ch , int dex ) {
16         int u = Last;
17         while (str[ dex - Len[ u ] - 1 ] != str[ dex ] ) u = Fail[ u ]; //找到合格的后缀
18         if( !Trans[ u ][ ch ] ) { //无现成的边
19             int Newnode = ++ Size , v = Fail[ u ]; //防止取掉整串
20             Len[ Newnode ] = Len[ u ] + 2;
21             while (str[ dex - Len[ v ] - 1 ] != str[ dex ] ) v = Fail[ v ]; //给他找
22             //一个fail指针
23             Fail[ Newnode ] = Trans[ v ][ ch ] , Trans[ u ][ ch ] = Newnode;
24         }
25         Last = Trans[ u ][ ch ];
26         weight[Last]++;
27     }
28     void Build( char *str ) {
29         int len = strlen( str );
30         for( int i = 0 ; i < len ; i ++ ) {
31             Extend( str[ i ] - 'a' + 1 , i );
32         }
33     }
34
35     void getWeight()
36     {

```

```

37     long long ma = 0 ;
38     for (int k = Size; k >= 0; --k) {
39         weight[Fail[k]] += weight[k];
40     }
41     for (int j = 0; j <= Size; ++j) {
42         ma = max(ma, Len[j]*weight[j]);
43     }
44     cout << ma << endl;
45 }
46
47 }PAM;
48
49 signed main( ) {
50     scanf("%s", str );
51     PAM.Build( str );
52     PAM.getWeight();
53     return 0;
54 }

```

## 8.9 拓展 kmp

```

1  #include<bits/stdc++.h>
2  #include "ext/pb_ds/assoc_container.hpp"
3  //using namespace __gnu_pbds;
4  using namespace std;
5  const int N = 1e4 + 10;
6  //typedef long long ll;
7  #define int long long
8  using namespace std;
9  const int MAXN = 2e7 + 5;
10 char p[MAXN], s[MAXN];
11 int pl, sl, z[MAXN], ext[MAXN];
12 //z[i]:    B串  i ~ strlen (B) -1      部分与 B自身的最长相同前缀
13 //
14 //ext[i]:   A串  i ~ strlen (A) -1      部分与B的最长相同前缀,也就是我们要求的東西。
15 void getZ() {
16     z[0] = pl; //从0号位置开始, LCP就是全部字符串
17     //从1开始, 先暴力算
18     int now = 0;
19     while (now + 1 < pl && p[now] == p[now + 1]) now++;
20     z[1] = now;
21     int p0 = 1;
22     //p0是最远情况的起点
23     for (int i = 2; i < pl; ++i) {
24         //p0+z[p0]是此时最远处
25         //i-p0对应着主串的i, 加i就是能到的距离
26         if (i + z[i - p0] < p0 + z[p0]) {
27             z[i] = z[i - p0]; //第一种情况
28         } else {
29             now = p0 + z[p0] - i;
30             now = max(now, 0ll);
31             while (now + i < pl && p[now] == p[now + i]) now++;
32             z[i] = now;
33             p0 = i;
34         }
35     }
36 }
37

```

```

38 void exkmp() {
39     getZ();
40     //先暴力算ext[0]
41     int now = 0;
42     while (now < p1 && now < s1 && p[now] == s[now]) now++;
43     ext[0] = now;
44     int p0 = 0;
45     for (int i = 1; i < s1; ++i) {
46         if (i + z[i - p0] < p0 + ext[p0]) {
47             ext[i] = z[i - p0];
48         } else {
49             now = p0 + ext[p0] - i;
50             now = max(now, 0ll); //防止i太大
51             while (now < p1 && now + i < s1 && p[now] == s[now + i]) now++;
52             ext[i] = now;
53             p0 = i;
54         }
55     }
56 }
57
58 signed main() {
59     scanf("%s%s", s, p);
60     p1 = strlen(p);
61     s1 = strlen(s);
62     exkmp();
63     int ans1 = 0, ans2 = 0;
64     for (int i = 0; i < p1; ++i) {
65         ans1 ^= 1LL * (i + 1) * (z[i] + 1);
66         // cout << z[i] << ' ';
67     }
68     // cout << '\n';
69     for (int i = 0; i < s1; ++i) {
70         ans2 ^= 1LL * (i + 1) * (ext[i] + 1);
71         // cout << ext[i] << ' ';
72     }
73
74     printf("%lld\n%lld\n", ans1, ans2);
75     return 0;
76 }

```

## 8.10 字典树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 100010;
4  int tire[26*N][26], n, m;
5  int cnt, level[N*26];
6
7  void insert(string s) {
8      int p = 0;
9      for (int i = 0; i < s.length(); i++) {
10         int v = s[i] - 'a';
11         if (!tire[p][v]) tire[p][v] = ++cnt;
12         p = tire[p][v];
13     }
14     level[p]++;
15 }
16

```

```
17 int query(string s) {
18     int p = 0;
19     int res = 0;
20     for (int i = 0; i < s.length(); i++) {
21         p = tire[p][s[i]-'a'];
22         if (!p) break;
23         res += level[p];
24     }
25     return res;
26 }
```

### 8.11 字符串哈希

```
1
2 char s[N];
3 unsigned long long f[N], p[N];
4
5 unsigned long long get(int l, int r) { //获取哈希值
6     return f[r] - f[l-1] * p[r-l+1];
7 }
8
9 f[i] = f[i-1] * base + s[i];
10 p[i] = p[i-1] * base;
11
12
13 /*
14  $H(T) = H(S + T) - H(S) - p^{(length(T))}$ 
15 */
16
17 */
```

### 8.12 最小表示法

```
1 //
2 // Created by acer on 2021/2/1.
3 //
4 //求循环字符串的长度为k的最小子串
5 int MinimumRepresentation(int *s, int l)
6 {
7     int i, j, k;
8     i=0; j=1; k=0;
9     while(i<l&&j<l)
10     {
11         k=0;
12         while(s[i+k]==s[j+k]&&k<l) k++;
13         if(k==l) return i;
14         if(s[i+k]>s[j+k])
15             if(i+k+1>j) i=i+k+1;
16             else i=j+1;
17         else if(j+k+1>i) j=j+k+1;
18         else j=i+1;
19     }
20     if(i<l) return i;
21     else return j;
22 }
```