

## 一、常用

头文件  
快读  
对拍  
\_\_int128

## 二、字符串

kmp  
manacher  
最小表示法  
Z函数  
AC自动机  
SA(nlogn)

## 三、图论

dinic  
二分图最大匹配  
2—SAT—Tarjan  
SCC kosaraju  
SCC Tarjan  
边双连通分量  
割点  
无向图欧拉图  
有向图欧拉图  
笛卡尔树  
dfs序求lca  
点分治

## 四、数论

exgcd  
整数分块  
欧拉筛（质数）  
欧拉筛(约数个数)  
欧拉筛（最小素因子）  
 $ax-by=1$ 的解  
pollard\_rho

## 五、数据结构

ST表  
树状数组  
并查集  
二维树状数组维护区间查询，修改  
线段树（区间查询最小值，最小值个数）  
线段树（区间修改加法，区间查询）  
线段树（区间修改加与乘，区间查询）  
pds

## 六、简单计算几何

点

## 七、杂项

矩阵快速幂  
组合数

# 一、常用

## 头文件

```
1  /*
2
3  _/      _/      _/      _/      _/      _/      _/_/_/_/_/_/      _/_/      _/
4  _/
5  _/      _/      _/      _/      _/      _/      _/      _/      _/      _/
6  _/_/_/
7  _/_/_/      _/_/_/_/_/_/      _/      _/      _/      _/      _/      _/
8  _/
9  _/      _/      _/      _/      _/      _/      _/_/_/      _/
10 _/
11 */
12 #pragma GCC optimize("unroll-loops")
13 #pragma GCC optimize("Ofast")
14 #include<bits/stdc++.h>
15 using namespace std;
16 typedef long long ll;
17 typedef unsigned long long ull;
18 #define rep(i,a,n) for(int i=a;i<n;i++)
19 #define per(i,a,n) for(int i=n-1;i>=a;i--)
20 #define fastio ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
21 #define multi int _;cin>>_;while(_--)
22 #define debug(x) cerr << #x << " = " << (x) << endl;
23 #define int long long
24 #define pb push_back
25 #define eb emplace_back
26 ll gcd(ll a,ll b){ return b?gcd(b,a%b):a;}
27 mt19937 rand(random_device{}());
28 int rnd(int x){ return rand() % x; }
29 void test() {cerr << "\n";}
30 template<typename T, typename... Args>
31 void test(T x, Args... args) {cerr << x << " ";test(args...);}
32 const ll MOD = 998244353;
33 // const ll MOD = 1e9+7;
34 int ksm(int x,int y){int ans=1;x%=MOD;while(y)
35 {if(y&1)ans=ans*x%MOD;x=x*x%MOD,y/=2;}return ans;}
36
37 const ll P1 = 999971, base1 = 101;
38 const ll P2 = 999973, base2 = 103;
39 const ll N = 200005;
40 //head
41
42 signed main()
```

```

43 {
44 #ifdef localfreopen
45     // freopen("1.in","r",stdin);
46 #endif
47     fastio
48
49     return 0;
50 }

```

## 快读

```

1 inline int read()
2 {
3     int x=0,f=1;char ch=getchar();
4     while (ch<'0' || ch>'9'){if (ch=='-') f=-1;ch=getchar();}
5     while (ch>='0'&&ch<='9'){x=x*10+ch-48;ch=getchar();}
6     return x*f;
7 }

```

## 对拍

```

1 :loop
2 data.exe > 1.in
3 my.exe <1.in >my.out
4 std.exe <1.in >std.out
5 fc my.out std.out
6 if not errorlevel 1 goto loop
7 pause
8 goto loop

```

## \_\_int128

```

1 __int128 read()
2 {
3     __int128 f=1,w=0;
4     char ch=getchar();
5     while(ch<'0' || ch>'9')
6     {
7         if(ch=='-')
8             f=-1;
9         ch=getchar();
10    }
11    while(ch<='9'&&ch>='0')
12    {
13        w=w*10+ch-'0';
14        ch=getchar();
15    }
16    return f*w;
17 }
18
19 void print(__int128 x)

```

```

20 {
21     if(x<0)
22     {
23         putchar('-');
24         x=-x;
25     }
26     if(x>9)print(x/10);
27     putchar(x%10+'0');
28 }

```

## 二、字符串

### kmp

```

1  vector<int> kmp(string s)
2  { //string的形式为 '#' + t1 + '#' + s
3      int n = s.size() - 1;
4      vector<int> nxt(s.size());
5      int j = 0;
6      for(int i = 2 ; i <= n ; i++){
7          while(j && s[j + 1] != s[i]) j = nxt[j];
8          if(s[j + 1] == s[i]) j++;
9          nxt[i] = j;
10     }
11     return nxt;
12 } //从第lent + 2 位 到 lent + lens + 1位为 s

```

### manacher

```

1  vector<int> manacher(string s)
2  { //string为#A#B#C#...#Z#
3      int n = s.size();
4      vector<int> d1(n);
5      for (int i = 0, l = 0, r = -1; i < n; i++)
6      {
7          int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
8          while (0 <= i - k && i + k < n && s[i - k] == s[i + k]) k++;
9          d1[i] = k--;
10         if (i + k > r)
11         {
12             l = i - k;
13             r = i + k;
14         }
15     }
16     return d1;
17 }

```

### 最小表示法

```

1  string minrep(string s)

```

```

2  //s从s[0]开始存
3  int k = 0, i = 0, j = 1, n = s.size();
4  while (k < n && i < n && j < n) {
5      if (s[(i + k) % n] == s[(j + k) % n]) {
6          k++;
7      } else {
8          s[(i + k) % n] > s[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
9          if (i == j) i++;
10         k = 0;
11     }
12 }
13 i = min(i, j);
14 return s.substr(i, N) + s.substr(0, i);
15 }

```

## Z函数

```

1  vector<int> exkmp(string s)
2  {
3      vector<int> p(s.size());
4      int n = s.size() - 1;
5      int L = 1, R = 0;
6      p[1] = 0;
7      for(int i = 2 ; i <= n ; i++)
8      {
9          if(i > R)
10             {
11                 p[i] = 0;
12             }else{
13                 int k = i - L + 1;
14                 p[i] = min(p[k], R - i + 1);
15             }
16             while(i + p[i] <= n && s[p[i] + 1] == s[i + p[i]])
17             {
18                 ++p[i];
19             }
20             if(i + p[i] - 1 > R)
21             {
22                 L = i;
23                 R = i + p[i] - 1;
24             }
25         }
26         return p;
27     } //从lent + 2位到lent + lens + 1位为 s
28     //*****p[1] = 0, 但实际从第一位往后能匹配lent的总长

```

## AC自动机

```

1  int n, idx;
2  struct Node{
3      int fail, nxt[26], end;
4  }trie[150000];
5
6

```

```

7  string ss[155];
8  int cnt[155];
9
10 void add_string(string s, int num)
11 {
12     int p = 0;
13     for(int i = 0 ; i < s.size() ; i++ )
14     {
15         int x = s[i] - 'a';
16         if(!trie[p].nxt[x])
17         {
18             trie[p].nxt[x] = ++idx;
19         }
20         p = trie[p].nxt[x];
21     }
22     trie[p].end = num;
23 }
24
25 void get_fail()
26 {
27     queue<int> q;
28     rep(i, 0, 26)
29     {
30         if(trie[0].nxt[i])
31         {
32             trie[trie[0].nxt[i]].fail = 0;
33             q.push(trie[0].nxt[i]);
34         }
35     }
36     while(!q.empty())
37     {
38         int x = q.front();
39         q.pop();
40         rep(i, 0, 26)
41         {
42             if(trie[x].nxt[i])
43             {
44                 trie[trie[x].nxt[i]].fail = trie[trie[x].fail].nxt[i];
45                 q.push(trie[x].nxt[i]);
46             }else{
47                 trie[x].nxt[i] = trie[trie[x].fail].nxt[i];
48             }
49         }
50     }
51 }
52 void query_string(string s)
53 {
54     int p = 0;
55     for(int i = 0 ; i < s.size() ; i++ )
56     {
57         int x = s[i] - 'a';
58         if(trie[p].nxt[x])
59         {
60             p = trie[p].nxt[x];
61         }else{
62             p = trie[trie[p].fail].nxt[x];

```

```

63     }
64     for(int i = p ; i ; i = trie[i].fail)
65     {
66         cnt[trie[i].end]++;
67     }
68     //cout << p << " \n"[i == s.size() - 1];
69 }
70 }
71
72 signed main()
73 {
74     fastio
75     //freopen("1.in","r",stdin);
76     string s;
77     while(cin >> n)
78     {
79         if(n == 0) break;
80         idx = 0;
81         memset(trie, 0, sizeof(trie));
82         memset(cnt, 0, sizeof(cnt));
83         rep(i, 1, n + 1)
84         {
85             cin >> ss[i];
86             add_string(ss[i], i);
87         }
88         get_fail();
89         cin >> s;
90         query_string(s);
91         ll ans = *max_element(cnt + 1, cnt + n + 1);
92         cout << ans << endl;
93         rep(i, 1, n + 1)
94         {
95             if(cnt[i] == ans) cout << ss[i] << endl;
96         }
97     }

```

## SA(nlogn)

```

1  struct SA{
2      vector<int> sa, rk, oldrk, id, key1, cnt;
3      int i, m = 127, p, w;
4      bool cmp(int x, int y, int w) {
5          return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
6      } // key1[i] = rk[id[i]] (作为基数排序的第一关键字数组)
7      int n;
8      SA(string s)
9      {
10         n = s.size() - 1;
11         oldrk.resize(2 * n + 5);
12         sa.resize(n + 2);
13         rk.resize(n + 2);
14         id.resize(n + 2);
15         key1.resize(n + 2);

```

```

16 cnt.resize(max(n, 130));
17 for (i = 1; i <= n; ++i) ++cnt[rk[i] = s[i]];
18 for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
19 for (i = n; i >= 1; --i) sa[cnt[rk[i]]--] = i;
20 for (w = 1;; w <= 1, m = p) { // m=p 就是优化计数排序值域
21     for (p = 0, i = n; i > n - w; --i) id[++p] = i;
22     for (i = 1; i <= n; ++i)
23         if (sa[i] > w) id[++p] = sa[i] - w;
24     fill(cnt.begin(), cnt.end(), 0);
25     for (i = 1; i <= n; ++i) ++cnt[key1[i] = rk[id[i]]];
26     // 注意这里px[i] != i, 因为rk没有更新, 是上一轮的排名数组
27
28     for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
29     for (i = n; i >= 1; --i) sa[cnt[key1[i]]--] = id[i];
30     for(int i = 1; i <= n; i++)
31     {
32         oldrk[i] = rk[i];
33     }
34     for (p = 0, i = 1; i <= n; ++i)
35         rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
36     if (p == n) {
37         break;
38     }
39 }
40 }
41 }; //传***入的string为1~n

```

## 三、图论

### dinic

```

1  const int v = 1010;
2  const int E = 101000;
3  using ll = long long;
4
5  template<typename T>
6  struct MaxFlow
7  {
8      int s, t, vtot;
9      int head[V], etot;
10     int dis[V], cur[V];
11     struct edge
12     {
13         int v, nxt;
14         T f;
15     }e[E * 2];
16     void addedge(int u, int v, T f)
17     {
18         e[etot] = {v, head[u], f}; head[u] = etot++;
19         e[etot] = {u, head[v], 0}; head[v] = etot++;
20     }
21     bool bfs()

```



```

22     {
23         for(int i = 1 ; i <= vtot ; i++ )
24         {
25             dis[i] = 0;
26             cur[i] = head[i];
27         }
28         queue<int> q;
29         q.push(s); dis[s] = 1;
30         while(!q.empty())
31         {
32             int u = q.front(); q.pop();
33             for(int i = head[u] ; ~i ; i = e[i].nxt)
34             {
35                 if(e[i].f && !dis[e[i].v])
36                 {
37                     int v = e[i].v;
38                     dis[v] = dis[u] + 1;
39                     if(v == t) return true;
40                     q.push(v);
41                 }
42             }
43         }
44         return false;
45     }
46     T dfs(int u, T m)
47     {
48         if(u == t) return m;
49         T flow = 0;
50         for(int i = cur[u]; ~i ; cur[u] = i = e[i].nxt)
51         {
52             if(e[i].f && dis[e[i].v] == dis[u] + 1)
53             {
54                 T f = dfs(e[i].v, min(m, e[i].f));
55                 e[i].f -= f;
56                 e[i ^ 1].f += f;
57                 m -= f;
58                 flow += f;
59                 if(!m) break;
60             }
61         }
62         if(!flow) dis[u] = -1;
63         return flow;
64     }
65     T dinic()
66     {
67         T flow = 0;
68         while(bfs()) flow += dfs(s, numeric_limits<T>::max());
69         return flow;
70     }
71     void init(int s_, int t_, int vtot_)
72     {
73         s = s_;
74         t = t_;
75         vtot = vtot_;
76         etot = 0;
77         for(int i = 1 ; i <= vtot ; i++ )

```

```

78         {
79             head[i] = -1;
80         }
81     }
82 };
83
84 MaxFlow<ll> g;
85 /***记得每次init,
86

```

## 二分图最大匹配

```

1  int a[N];
2  int v[N], n1, n2;
3  int to[N], b[N];
4  int n;
5  vector<int> e[N];
6  //n1为左边点数量, n2为右边点数量, v为右边的点连向左边哪条边
7  bool find(int x)
8  {
9      b[x] = true;
10     for(auto y : e[x])
11     {
12         if(!v[y] || (!b[v[y]] && find(v[y])))
13         {
14             v[y] = x;
15             return true;
16         }
17     }
18     return false;
19 }
20
21 int match()
22 {
23     int ans = 0;
24     memset(v, 0, sizeof(v));
25     for(int i = 1 ; i <= n1 ; i++ )
26     {
27         memset(b, 0, sizeof(b));
28         if(find(i))
29         {
30             ++ans;
31         }
32     }
33     return ans;
34 }

```

## 2—SAT—Tarjan

```

1  vector<int> e[N];
2  int dfn[N], ins[N], low[N], bel[N], idx, cnt;
3  stack<int> st;
4  vector<vector<int> > scc;
5

```

```

6
7 void dfs(int u)
8 {
9     dfn[u] = low[u] = ++idx;
10    ins[u] = true;
11    st.push(u);
12    for(auto v : e[u])
13    {
14        if(!dfn[v])
15        {
16            dfs(v);
17            low[u] = min(low[u], low[v]);
18        }else{
19            if(ins[v]) low[u] = min(low[u], dfn[v]);
20        }
21    }
22    if(dfn[u] == low[u])
23    {
24        vector<int> c;
25        ++cnt;
26        while(true)
27        {
28            int v = st.top();
29            c.push_back(v);
30            ins[v] = false;
31            bel[v] = cnt;
32            st.pop();
33            if(v == u) break;
34        }
35        sort(c.begin(), c.end());
36        scc.push_back(c);
37    }
38 }
39
40 int main()
41 {
42     fastio
43     //freopen("1.in", "r", stdin);
44     int n, m;
45     cin >> n >> m;
46     for(int i = 1 ; i <= m ; i++ )
47     {
48         int u, ch1, v, ch2;
49         cin >> u >> ch1 >> v >> ch2;
50         u = 2 * u + (ch1 == 0);
51         v = 2 * v + (ch2 == 0);
52         e[u ^ 1].push_back(v);
53         e[v ^ 1].push_back(u);
54     }
55     for(int i = 1 ; i <= 2 * n ; i++ )
56     {
57         if(!dfn[i]) dfs(i);
58     }
59     for(int i = 1 ; i <= n ; i++ )
60     {
61         if(bel[2 * i] == bel[2 * i + 1])

```

```

62     {
63         cout << "IMPOSSIBLE\n";
64         return 0;
65     }
66 }
67 cout << "POSSIBLE\n";
68 for(int i = 1 ; i <= n ; i++ )
69 {
70     cout << (bel[2 * i] < bel[2 * i + 1]) << " ";
71 }
72 cout << endl;
73 return 0;
74 }

```

## SCC hosoraju

```

1  int vis[N], n, m;
2  vector<int> out, c, e[N], erev[N];
3  int sz[N];
4  int bel[N], cnt;
5  vector<vector<int> > scc;
6
7  void dfs1(int u)
8  {
9      vis[u] = 1;
10     for(auto v : e[u])
11     {
12         if(!vis[v]) dfs1(v);
13     }
14     out.push_back(u);
15 }
16
17 void dfs2(int u, int cnt)
18 {
19
20     vis[u] = 1;
21     for(auto v : erev[u])
22     {
23         if(!vis[v]) dfs2(v, cnt);
24     }
25     bel[u] = cnt;
26     sz[cnt]++;
27     c.push_back(u);
28 }
29
30 int main()
31 {
32     fastio
33     //freopen("1.in", "r", stdin);
34     int n, m, x, y;
35     cin >> n >> m;
36     for(int i = 1 ; i <= m ; i++ )
37     {
38         cin >> x >> y;
39         e[x].push_back(y);

```

```

40     erev[y].push_back(x);
41 }
42 memset(vis, 0, sizeof(vis));
43 for(int i = 1 ; i <= n ; i++ )
44 {
45     if(!vis[i])
46     {
47         dfs1(i);
48     }
49 }
50 reverse(out.begin(), out.end());
51 memset(vis, 0, sizeof(vis));
52 for(auto u : out)
53 {
54     if(!vis[u])
55     {
56         c.clear();
57         dfs2(u, ++cnt);
58         sort(c.begin(), c.end());
59         scc.push_back(c);
60     }
61 }
62 }
63 sort(scc.begin(), scc.end());
64 for(auto c : scc)
65 {
66     for(auto x : c)
67     {
68         cout << x << " ";
69     }
70     cout << "\n";
71 }
72 return 0;
73 }

```

## SCC Tarjan

```

1  vector<int> e[N];
2  int dfn[N], ins[N], low[N], bel[N], idx, cnt;
3  stack<int> st;
4  vector<vector<int> > scc;
5
6
7  void dfs(int u)
8  {
9      dfn[u] = low[u] = ++idx;
10     ins[u] = true;
11     st.push(u);
12     for(auto v : e[u])
13     {
14         if(!dfn[v])
15         {
16             dfs(v);
17             low[u] = min(low[u], low[v]);
18         }else{

```

```

19         if(ins[v]) low[u] = min(low[u], dfn[v]);
20     }
21 }
22 if(dfn[u] == low[u])
23 {
24     vector<int> c;
25     ++cnt;
26     while(true)
27     {
28         int v = st.top();
29         c.push_back(v);
30         ins[v] = false;
31         bel[v] = cnt;
32         st.pop();
33         //cout << v << " ";
34         if(v == u) break;
35     }
36     //cout << endl;
37     sort(c.begin(), c.end());
38     scc.push_back(c);
39 }
40
41 }
42
43 int main()
44 {
45     fastio
46     //freopen("1.in", "r", stdin);
47     int n, m;
48     cin >> n >> m;
49     for(int i = 1 ; i <= m ; i++ )
50     {
51         int x, y;
52         cin >> x >> y;
53         e[x].push_back(y);
54     }
55     for(int i = 1 ; i <= n ; i++ )
56     {
57         if(!dfn[i])
58         {
59             dfs(i);
60         }
61     }
62     sort(scc.begin(), scc.end());
63     for(auto c : scc)
64     {
65         for(auto x : c)
66         {
67             cout << x << " ";
68         }
69         cout << "\n";
70     }
71     return 0;
72 }

```

## 边双连通分量

```
1  int head[N], e[N], nxt[N], idx = 1, n, m;
2  int dfn[M], low[M], cnt, b[N], bel[N], ansCnt[M];
3  vector<vector<int>> dcc;
4  void add(int x, int y)
5  {
6      nxt[++idx] = head[x];
7      head[x] = idx;
8      e[idx] = y;
9  }
10 void tarjan(int x, int e_in)
11 {
12     dfn[x] = low[x] = ++cnt;
13     for(int i = head[x] ; i ; i = nxt[i])
14     {
15         int y = e[i];
16         if(!dfn[y])
17         {
18             tarjan(y, i);
19             if(dfn[x] < low[y])
20             {
21                 b[i] = b[i ^ 1] = 1;
22             }
23             low[x] = min(low[x], low[y]);
24         } else if (i != (e_in ^ 1))
25         {
26             low[x] = min(low[x], dfn[y]);
27         }
28     }
29 }
30
31 vector<int> v;
32
33 void dfs(int x, int cnt)
34 {
35     bel[x] = cnt;
36     v.push_back(x);
37     ansCnt[cnt]++;
38     for(int i = head[x] ; i ; i = nxt[i])
39     {
40         int y = e[i];
41         if(bel[y] || b[i]) continue;
42         dfs(y, cnt);
43     }
44 }
45
46 signed main()
47 {
48     fastio
49     //freopen("1.in", "r", stdin);
50     cin >> n >> m;
51     int x, y;
52     for(int i = 1 ; i <= m ; i++)
53     {
```

```

54     cin >> x >> y;
55     if(x == y) continue;
56     add(x, y);
57     add(y, x);
58 }
59 for(int i = 1 ; i <= n ; i++ )
60 {
61     if(!dfn[i]) tarjan(i, 0);
62 }
63 int ans = 0;
64 for(int i = 1 ; i <= n ; i++ )
65 {
66     if(!bel[i])
67     {
68         v.clear();
69         dfs(i, ++ans);
70         dcc.push_back(v);
71     }
72 }
73 int sz = dcc.size();
74 cout << dcc.size() << "\n";
75 for(int i = 0 ; i < sz ; i++ )
76 {
77     auto v = dcc[i];
78     cout << ansct[i + 1] << " ";
79     for(auto x : v)
80     {
81         cout << x << " ";
82     }
83     cout << "\n";
84 }
85 return 0;
86 }
87 }

```

## 割点

```

1  int n, m;
2  int dfn[N], idx, low[N];
3  bool vis[N], cut[N];
4  vector<int> e[N];
5  int cnt;
6
7  void dfs(int u, int root)
8  {
9      vis[u] = 1;
10     dfn[u] = ++idx;
11     low[u] = idx;
12     int child = 0;
13     for(auto v : e[u])
14     {
15         if(!vis[v])
16         {
17             dfs(v, root);
18             low[u] = min(low[u], low[v]);

```



```

19         if(low[v] >= dfn[u] && u != root)
20         {
21             cut[u] = 1;
22         }
23         if(u == root)
24         {
25             child++;
26         }
27     }
28     low[u] = min(low[u], dfn[v]);
29 }
30 if(child >= 2 && u == root)
31 {
32     cut[u] = 1;
33 }
34 }
35
36 int main()
37 {
38     fastio
39     //freopen("1.in", "r", stdin);
40     cin >> n >> m;
41     rep(i, 1, m + 1)
42     {
43         int x, y;
44         cin >> x >> y;
45         e[x].push_back(y);
46         e[y].push_back(x);
47     }
48     rep(i, 1, n + 1)
49     {
50         if(!vis[i])
51         {
52             dfs(i, i);
53         }
54     }
55     cout << accumulate(cut + 1, cut + n + 1, 0ll) << "\n";
56     rep(i, 1, n + 1)
57     {
58         if(cut[i])
59         {
60             cout << i << " ";
61         }
62     }
63     return 0;
64 }

```

## 无向图欧拉图

```

1 vector<pair<int ,int > > e[N];
2 int d[N], n, m;
3 int f[N], b[N], sz[N], ans[N], idxans;
4
5 void dfs(int x)
6 {

```

```

7      //cout << "dfs = " << x << endl;
8      for(; f[x] < sz[x] ; )
9      {
10         int y = e[x][f[x]].first, id = e[x][f[x]].second;
11         if(!b[id])
12         {
13             b[id] = 1;
14             f[x]++;
15             dfs(y);
16             ans[++idxans] = y;
17         }else{
18             f[x]++;
19         }
20     }
21 }
22
23 void Euler()
24 {
25     memset(f, 0, sizeof(f));
26     memset(b, 0, sizeof(b));
27     int cnt = 0, x = 0;
28     for(int i = 1 ; i <= n ; i++ )
29     {
30         if(d[i] & 1)
31         {
32             cnt++;
33             x = i;
34         }
35     }
36     if(!(cnt == 0 || cnt == 2))
37     {
38         cout << "No\n";
39         return;
40     }
41     for(int i = 1 ; i <= n ; i++ )
42     {
43         sz[i] = e[i].size();
44         if(!x)
45             if(d[i])
46             {
47                 x = i;
48             }
49     }
50     dfs(x);
51     ans[++idxans] = x;
52     if(idxans == m + 1)
53     {
54         cout << "Yes\n";
55     }else{
56         cout << "No\n";
57     }
58 }
59 int main()
60 {
61     fastio
62     //freopen("1.in", "r", stdin);

```

```

63     cin >> n >> m;
64     int idx = 0;
65     for(int i = 1 ; i <= m ; i++ )
66     {
67         int x, y;
68         cin >> x >> y;
69         ++idx;
70         ++d[x];
71         ++d[y];
72         e[x].push_back({y, idx});
73         e[y].push_back({x, idx});
74
75     }
76     Euler();
77     return 0;
78 }

```

## 有向图欧拉图

```

1  int n;
2  vector<int> e[N];
3  int ind[N], outd[N], f[N], sz[N], ans[N], idx = 0;
4
5  void dfs(int x)
6  {
7      for(; f[x] < sz[x] ;)
8      {
9          int y = e[x][f[x]];
10         f[x]++;
11         dfs(y);
12         ans[++idx] = y;
13     }
14 }
15 void Euler()
16 {
17     memset(f, 0, sizeof(f));
18     int cntdiff = 0;
19     int cntin = 0;
20     int x = 0;
21     for(int i = 1 ; i <= n ; i++ )
22     {
23         if(ind[i] != outd[i])
24         {
25             cntdiff++;
26         }
27         if(ind[i] + 1 == outd[i])
28         {
29             cntin++;
30             x = i;
31         }
32     }
33     if(!(cntdiff == 2 && cntin == 1 || cntdiff == 0))
34     {
35         cout << "No\n";
36         return;

```

```

37     }
38     for(int i = 1 ; i <= n ; i++ )
39     {
40         sz[i] = e[i].size();
41         //cout << e[i].size();
42         if(!x)
43         {
44             if(ind[i])
45             {
46                 x = i;
47             }
48         }
49     }
50     dfs(x);
51     ans[++idx]= x;
52     if(idx == n + 1)
53     {
54         cout << "Yes\n";
55     }else{
56         cout << "No\n";
57     }
58     for(int i = idx ; i > 0 ; i--)
59     {
60         cout << ans[i] << " ";
61     }
62 }

```

## 笛卡尔树

```

1  //每个父节点都小于其所有子节点
2
3  int a[N], n, l[N], r[N];
4  int root = 0;
5
6  void build()
7  {
8      stack<int> st;
9      for(int i = 1 ; i <= n ; i++ )
10     {
11         int last = 0;
12         while(!st.empty() && a[st.top()] > a[i])
13         {
14             last = st.top();
15             st.pop();
16         }
17         if(!st.empty())
18         {
19             r[st.top()] = i;
20         }else{
21             root = i;
22         }
23         l[i] = last;
24         st.push(i);

```

```

25     }
26 }

```

## dfs序求lca

```

1  int main()
2  {
3      int idx = 0;
4      vector<int> dfn(n + 5);
5      vector st(__lg(n) + 2, vector<int> (n + 5)); //****不能改成23****
6      function<int(int,int)> get = [&](int x, int y)
7      {
8          return dfn[x] < dfn[y] ? x : y;
9      };
10     function<void(int,int)> dfs = [&](int x, int fa)
11     {
12         st[0][dfn[x] = ++idx] = fa;
13         for(int y : adj[x]) if(y != fa) dfs(y, x);
14     };
15     function<int(int,int)> lca = [&](int u, int v)
16     {
17         if(u == v) return u;
18         if((u = dfn[u]) > (v = dfn[v])) swap(u, v);
19         int d = __lg(v - u++);
20         return get(st[d][u], st[d][v - (1 << d) + 1]);
21     };
22     dfs(s, 0);
23     for(int i = 1 ; i <= __lg(n) ; i++ ) //****不能改成23****
24     {
25         for(int j = 1 ; j + (1 << i - 1) <= n ; j++ ) // ****注意边界****
26         {
27             st[i][j] = get(st[i - 1][j], st[i - 1][j + (1 << i - 1)]);
28         }
29     }
30     /// lca(u, v);
31 }

```

## 点分治

```

1  signed main()
2  {
3      fastio
4      int n, k, ans = 0;
5      cin >> n >> k;
6      ans = n + 1;
7      vector<vector<pair<int,int>>> adj(n + 1);
8      vector<int> sz(n + 1, 0), maxsz(n + 1, 0), del(n + 1, 0);
9      vector<int> mark(k + 1, 0), c(k + 1, 0);
10     int T = 1;
11     int u, v, w;
12     for(int i = 1 ; i < n ; i++ )
13     {
14         cin >> u >> v >> w;
15         u++;

```

```

16     v++;
17     adj[u].emplace_back(v, w);
18     adj[v].emplace_back(u, w);
19 }
20 function<void(int, int)> solve = [&](int x, int s)
21 {
22     T++;
23     int mxs = s + 1, root = -1;
24     function<void(int, int)> dfs1 = [&](int x, int fx)
25     {
26         sz[x] = 1;
27         maxsz[x] = 0;
28         for(auto [y, w] : adj[x])
29         {
30             if(del[y] || y == fx) continue;
31             dfs1(y, x);
32             sz[x] += sz[y];
33             maxsz[x] = max(maxsz[x], sz[y]);
34         }
35         maxsz[x] = max(maxsz[x], s - sz[x]);
36         if(maxsz[x] < mxs)
37         {
38             mxs = maxsz[x], root = x;
39         }
40     };
41     dfs1(x, -1);
42     ///////////////////////////////////
43     mark[0] = T;
44     c[0] = 0;
45     for(auto [y, w] : adj[root])
46     {
47         if(del[y]) continue;
48         vector<pair<int, int>> self;
49         function<void(int, int, int, int)> dfs2 = [&](int x, int fx, int
dis, int dep)
50         {
51             self.emplace_back(dis, dep);
52             for(auto [y, w] : adj[x])
53             {
54                 if(del[y] || y == fx) continue;
55                 dfs2(y, x, dis + w, dep + 1);
56             }
57         };
58         dfs2(y, root, w, 1);
59         for(auto [dis, dep] : self)
60         {
61             if(k - dis >= 0 && mark[k - dis] == T)
62             {
63                 ans = min(ans, c[k - dis] + dep);
64             }
65         }
66         for(auto [dis, dep] : self)
67         {
68             if(dis > k) continue;
69             if(mark[dis] == T)
70             {

```

```

71         c[dis] = min(c[dis], dep);
72     }else{
73         c[dis] = dep;
74         mark[dis] = T;
75     }
76 }
77 }
78 ///////////////////////////////////////////////////
79 del[root] = 1;
80 for(auto [y, w] : adj[root])
81 {
82     if(del[y]) continue;
83     solve(y, sz[y]);
84 }
85 };
86 solve(1, n);
87 cout << (ans > n ? -1 : ans) << "\n";
88 return 0;
89 }

```

## 四、数论

### exgcd

```

1  int exgcd(int a, int b, int &x, int &y)
2  {
3      if(b == 0)
4      {
5          x = 1;
6          y = 0;
7          return a;
8      }
9      int d = exgcd(b, a % b, y, x);
10     y -= (a / b) * x;
11     return d;
12 }

```

### 整数分块

```

1  for(ll l = 1 ; l <= n ; l++ )
2      {
3          ll d = n / l, r = n / d;
4          cout << l << " : " << r << " = " << d << endl;
5          l = r;
6      }

```

### 欧拉筛（质数）

```

1  const ll MAXN = 1e6 + 5;
2  ll prime[MAXN], idxprime = 0;
3  bool isprime[MAXN];
4
5  void prime_build()
6  {
7      for(int i = 2 ; i < MAXN ; i++ )
8      {
9          if(isprime[i] == 0)
10         {
11             prime[++idxprime] = i;
12         }
13         for(int j = 1 ; j <= idxprime && i * prime[j] < MAXN ; j++ )
14         {
15             isprime[i * prime[j]] = 1;
16             if(i % prime[j] == 0) break;
17         }
18     }
19 }

```

## 欧拉筛(约数个数)

```

1  ll prim[50000005], sum[50000005], d[50000005], len;
2  bool vis[50000005];
3
4  inline void sieve(int x) {
5      for(int i = 2; i <= x; i++) {
6          if(! vis[i]) {
7              prim[++ len] = i;
8              d[i] = 2;
9              sum[i] = 1;
10         }
11         for(int j = 1; j <= len && i * prim[j] <= x; j++) {
12             vis[i * prim[j]] = 1;
13             if(i % prim[j] == 0) {
14                 sum[i * prim[j]] = sum[i] + 1;
15                 d[i * prim[j]] = d[i] / (sum[i] + 1) * (sum[i] + 2);
16                 break;
17             }
18             sum[i * prim[j]] = 1;
19             d[i * prim[j]] = d[i] * 2;
20         }
21     }
22 }
23

```

## 欧拉筛 (最小素因子)

```

1  int MAXN = 50;
2  int p[N], pr[N], idx;
3
4  void build()
5  {
6      for(int i = 2 ; i < MAXN ; i++ )

```



```

7      {
8          if(!p[i])
9          {
10             p[i] = i;
11             pr[++idx] = i;
12         }
13         for(int j = 1 ; j <= idx && pr[j] * i < MAXN ; j++ )
14         {
15             p[i * pr[j]] = pr[j];
16             if(p[i] == pr[j]) break;
17         }
18     }
19 }

```

## ax-by=1的解

```

1  ll exgcd(ll a, ll b, ll &x, ll &y)
2  {
3      if(b == 0)
4      {
5          x = 1;
6          y = 0;
7          return a;
8      }
9      int d = exgcd(b, a % b, y, x);
10     y -= (a / b) * x;
11     return d;
12 }
13
14 void solve()
15 {
16     ll a, b;
17     cin >> a >> b;
18     ll x, y;
19     ll d = exgcd(a, b, x, y);
20     y = -y;
21     while(x < 0 || y < 0)
22     {
23         x += b/d;
24         y += a/d;
25     }
26     while(x >= b/d && y >= a/d)
27     {
28         x -= b/d;
29         y -= a/d;
30     }
31     cout << x << " " << y << "\n";
32 }

```

## pollard\_rho

```

1  using i64 = long long;
2  using i128 = __int128;
3  i64 power(i64 a, i64 b, i64 m) {

```

```

4      i64 res = 1;
5      for (; b; b >>= 1, a = i128(a) * a % m) {
6          if (b & 1) {
7              res = i128(res) * a % m;
8          }
9      }
10     return res;
11 }
12
13 bool isprime(i64 p) {
14     if (p < 2) {
15         return 0;
16     }
17     i64 d = p - 1, r = 0;
18     while (!(d & 1)) {
19         r++;
20         d >>= 1;
21     }
22     int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
23     for (auto a : prime) {
24         if (p == a) {
25             return true;
26         }
27         i64 x = power(a, d, p);
28         if (x == 1 || x == p - 1) {
29             continue;
30         }
31         for (int i = 0; i < r - 1; i++) {
32             x = i128(x) * x % p;
33             if (x == p - 1) {
34                 break;
35             }
36         }
37         if (x != p - 1) {
38             return false;
39         }
40     }
41     return true;
42 }
43
44 mt19937 rng((unsigned int)
45 chrono::steady_clock::now().time_since_epoch().count());
46
47 i64 pollard_rho(i64 x) {
48     i64 s = 0, t = 0;
49     i64 c = i64(rng()) % (x - 1) + 1;
50     i64 val = 1;
51     for (int goal = 1; ; goal <= 1, s = t, val = 1) {
52         for (int step = 1; step <= goal; step++) {
53             t = (i128(t) * t + c) % x;
54             val = i128(val) * abs(t - s) % x;
55             if (step % 127 == 0) {
56                 i64 g = gcd(val, x);
57                 if (g > 1) {
58                     return g;
59                 }

```

```

59         }
60     }
61     i64 g = gcd(val, x);
62     if (g > 1) {
63         return g;
64     }
65 }
66 }
67
68 unordered_map<i64, int> getprimes(i64 x) {
69     unordered_map<i64, int> p;
70     function<void(i64)> get = [&](i64 x) {
71         if (x < 2) {
72             return;
73         }
74         if (isprime(x)) {
75             p[x]++;
76             return;
77         }
78         i64 mx = pollard_rho(x);
79         get(x / mx);
80         get(mx);
81     };
82     get(x);
83     return p;
84 }
85

```

## 五、数据结构

### ST表

```

1  for(int i = 1 ; i <= n ; i++ )
2  {
3      a[i] = read();
4      f[0][i] = a[i];
5  }
6  for(int i = 1 ; i <= 22 ; i++ )
7  {
8      for(int j = 1 ; j + (1 << i) - 1 <= n ; j++ )
9      {
10         f[i][j] = max(f[i-1][j], f[i-1][j + (1 << i - 1)]);
11     }
12 }
13 for(int i = 1 ; i <= m ; i++ )
14 {
15     int l = read(), r = read();
16     int len = __lg(r - l + 1);
17     printf("%d\n", max(f[len][l], f[len][r - (1 << len) + 1]));
18 }

```

## 树状数组

```
1  template<class T>
2  struct BIT{
3      T c[N];
4      void change(int x, T y)
5      {
6          for(; x < N ; x += x & (-x))
7              {
8                  c[x] += y;
9              }
10     }
11     T query(int x)
12     {
13         T s = 0;
14         for(; x ; x -= x & (-x))
15             {
16                 s += c[x];
17             }
18         return s;
19     }
20 };
```

## 并查集

```
1  struct DSU {
2      std::vector<int> f, siz;
3      DSU(int n) : f(n), siz(n, 1) { std::iota(f.begin(), f.end(), 0); }
4      int leader(int x) {
5          while (x != f[x]) x = f[x] = f[f[x]];
6          return x;
7      }
8      bool same(int x, int y) { return leader(x) == leader(y); }
9      bool merge(int x, int y) {
10         x = leader(x);
11         y = leader(y);
12         if (x == y) return false;
13         siz[x] += siz[y];
14         f[y] = x;
15         return true;
16     }
17     int size(int x) { return siz[leader(x)]; }
18 };
```

## 二维树状数组维护区间查询，修改

```
1  ll c1[N][N], c2[N][N], c3[N][N], c4[N][N];
2
3  int n, m, k, q;
4
5  int lowbit(int x)
6  {
7      return x & (-x);
```

```

8  }
9
10 void add(ll x, ll y, ll d)
11 {
12     for(int i = x ; i <= n ; i += lowbit(i))
13     {
14         for(int j = y ; j <= m ; j += lowbit(j))
15         {
16             //cout << "test" << endl;
17             c1[i][j] += d;
18             c2[i][j] += d * x;
19             c3[i][j] += d * y;
20             c4[i][j] += d * x * y;
21         }
22     }
23 }
24
25 void modify(int x1, int y1, int x2, int y2, int d)
26 {
27     add(x1, y1, d);
28     add(x1, y2 + 1, -d);
29     add(x2 + 1, y1, -d);
30     add(x2 + 1, y2 + 1, d);
31 }
32
33 ll sum(ll x, ll y)
34 {
35     ll ans = 0;
36     for(int i = x ; i ; i -= lowbit(i))
37     {
38         for(int j = y ; j ; j -= lowbit(j))
39         {
40             ans += (x + 1) * (y + 1) * c1[i][j];
41             ans -= (y + 1) * c2[i][j];
42             ans -= (x + 1) * c3[i][j];
43             ans += c4[i][j];
44         }
45     }
46     return ans;
47 }
48 ll query(int x1, int y1, int x2, int y2)
49 {
50     return (sum(x2, y2) - sum(x1 - 1, y2) - sum(x2, y1 - 1) + sum(x1 - 1, y1
51 - 1));
52 }
53 int h[100005];
54 int main()
55 {
56     fastio
57     //freopen("1.in", "r", stdin);
58     cin >> n >> m >> k >> q;
59     for(int i = 1 ; i <= k ; i++ )
60     {
61         cin >> h[i];
62     }
63     for(int i = 1 ; i <= q ; i++ )

```

```

63     {
64         int op;
65         cin >> op;
66         if(op == 1)
67         {
68             int a, b, c, d, id;
69             cin >> a >> b >> c >> d >> id;
70             modify(a, b, c, d, h[id]);
71         }else{
72             int a, b, c, d;
73             cin >> a >> b >> c >> d;
74             cout << query(a, b, c, d) << "\n";
75         }
76     }
77     return 0;
78 }
79

```

## 线段树（区间查询最小值，最小值个数）

```

1  struct Node{
2      int minx, cntminx;
3  };
4
5  ll a[N];
6
7  Node tr[4 * N];
8
9  void pushup(int u, int L, int R)
10 {
11     if(tr[u << 1].minx < tr[u << 1 | 1].minx)
12     {
13         tr[u].minx = tr[u << 1].minx;
14         tr[u].cntminx = tr[u << 1].cntminx;
15     }
16     if(tr[u << 1].minx > tr[u << 1 | 1].minx)
17     {
18         tr[u].minx = tr[u << 1 | 1].minx;
19         tr[u].cntminx = tr[u << 1 | 1].cntminx;
20     }
21     if(tr[u << 1].minx == tr[u << 1 | 1].minx)
22     {
23         tr[u].minx = tr[u << 1 | 1].minx;
24         tr[u].cntminx = tr[u << 1].cntminx + tr[u << 1 | 1].cntminx;
25     }
26 }
27
28
29 void build(int u, int L, int R)
30 {
31     int mid = L + R >> 1;
32     if(L == R)
33     {
34         tr[u].minx = a[L];
35         tr[u].cntminx = 1;

```

```

36         return;
37     }
38     build(u << 1, L, mid);
39     build(u << 1 | 1, mid + 1, R);
40     pushup(u, L, R);
41
42 }
43
44 void change(int u, int L, int R, int x, int y)
45 {
46     int mid = L + R >> 1;
47     if(L == R)
48     {
49         tr[u].minx = y;
50         return;
51     }
52     if(x <= mid)
53     {
54         change(u << 1, L, mid, x, y);
55     }
56     if(x > mid)
57     {
58         change(u << 1 | 1, mid + 1, R, x, y);
59     }
60     pushup(u, L, R);
61 }
62
63 pair<int, int> query(int u, int L, int R, int l, int r)
64 {
65     int mid = L + R >> 1;
66     if(l <= L && R <= r)
67     {
68         return {tr[u].minx, tr[u].cntminx};
69     }
70     if(r <= mid)
71     {
72         return query(u << 1, L, mid, l, r);
73     }
74     if(l >= mid + 1)
75     {
76         return query(u << 1 | 1, mid + 1, R, l, r);
77     }
78     auto s1 = query(u << 1, L, mid, l, r);
79     auto s2 = query(u << 1 | 1, mid + 1, R, l, r);
80     if(s1.first < s2.first)
81     {
82         return s1;
83     }
84     if(s1.first > s2.first)
85     {
86         return s2;
87     }
88     return {s1.first, s1.second + s2.second};
89 }
90
91 int main()

```

```

92  {
93      fastio
94      //freopen("1.in","r",stdin);
95      int n, m;
96      cin >> n >> m;
97      for(int i = 1 ; i <= n ; i++ )
98      {
99          cin >> a[i];
100     }
101     build(1, 1, n);
102     for(int i = 1 ; i <= m ; i++ )
103     {
104         int op, x, y;
105         cin >> op >> x >> y;
106         if(op == 1)
107         {
108             change(1, 1, n, x, y);
109         }else{
110             auto [_,__] = query(1, 1, n, x, y);
111             cout << _ << " " << __ << "\n";
112         }
113     }
114     return 0;
115 }

```

## 线段树（区间修改加法，区间查询）

```

1  struct Node{
2      ll sum, lazy, size;
3  };
4  Node tr[N * 4];
5  ll a[N];
6
7  void pushup(int u, int L, int R)
8  {
9      tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
10 }
11
12
13 void build(int u, int L, int R)
14 {
15     int mid = L + R >> 1;
16     tr[u].size = R - L + 1;
17     tr[u].sum = tr[u].lazy = 0;
18     if(L == R)
19     {
20         tr[u].sum = a[L];
21         return;
22     }
23     build(u << 1, L, mid);
24     build(u << 1 | 1, mid + 1, R);
25     pushup(u, L, R);
26
27 }
28

```



```

29 void pushdown(int u)
30 {
31     auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
32     if(root.lazy)
33     {
34         left.sum += root.lazy * left.size;
35         left.lazy += root.lazy;
36         right.sum += root.lazy * right.size;
37         right.lazy += root.lazy;
38         root.lazy = 0;
39     }
40 }
41
42 void pushup(int u)
43 {
44     tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
45 }
46
47 ll query(int u, int L, int R, int l, int r)
48 {
49     int mid = L + R >> 1;
50     if(l <= L && R <= r)
51     {
52         return tr[u].sum;
53     }
54     ll ans = 0;
55     pushdown(u);
56     if(l <= mid)
57     {
58         ans += query(u << 1, L, mid, l, r);
59     }
60     if(r > mid)
61     {
62         ans += query(u << 1 | 1, mid + 1, R, l, r);
63     }
64     return ans;
65 }
66
67 void modify(int u, int L, int R, int l, int r, int x)
68 {
69     int mid = L + R >> 1;
70     if(l <= L && R <= r)
71     {
72         tr[u].lazy += x;
73         tr[u].sum += x * tr[u].size;
74         return;
75     }
76     pushdown(u);
77     if(l <= mid)
78     {
79         modify(u << 1, L, mid, l, r, x);
80     }
81     if(r > mid)
82     {
83         modify(u << 1 | 1, mid + 1, R, l, r, x);
84     }

```

```

85     pushup(u);
86 }

```

## 线段树（区间修改加与乘，区间查询）

```

1  struct Node{
2      ll sum, mul, add, size;
3  } tr[4 * N];
4  ll a[N];
5
6  void pushup(int u)
7  {
8      tr[u].sum = (tr[u << 1].sum % P + tr[u << 1 | 1].sum % P) % P;
9  }
10
11 void pushdown(int u)
12 {
13     auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
14     root.mul %= P, root.add %= P;
15     left.sum  *= root.mul;          left.sum  %= P;
16     left.sum  += root.add * left.size; left.sum  %= P;
17     right.sum *= root.mul;          right.sum %= P;
18     right.sum += root.add * right.size; right.sum %= P;
19     left.add  *= root.mul;          left.add  %= P;
20     left.mul  *= root.mul;          left.mul  %= P;
21     right.add *= root.mul;          right.add %= P;
22     right.mul *= root.mul;          right.mul %= P;
23     left.add  += root.add;          left.add  %= P;
24     right.add += root.add;          right.add %= P;
25     root.mul  = 1;
26     root.add  = 0;
27 }
28
29 void build(int u, int L, int R)
30 {
31     int mid = L + R >> 1;
32     tr[u].size = R - L + 1;
33     tr[u].mul = 1;
34     tr[u].add = 0;
35     if(L == R)
36     {
37         tr[u].sum = a[L] % P;
38         return;
39     }
40     build(u << 1, L, mid);
41     build(u << 1 | 1, mid + 1, R);
42     pushup(u);
43 }
44
45 void modify_add(int u, int L, int R, int l, int r, int x)
46 {
47     int mid = L + R >> 1;
48     if(l <= L && R <= r)
49     {
50         tr[u].sum += tr[u].size * x;    tr[u].sum %= P;

```

```

51
52         tr[u].add += x;                tr[u].add %= P;
53         return;
54     }
55     pushdown(u);
56     if(l <= mid)
57     {
58         modify_add(u << 1, L, mid, l, r, x);
59     }
60     if(r >= mid + 1)
61     {
62         modify_add(u << 1 | 1, mid + 1, R, l, r, x);
63     }
64     pushup(u);
65 }
66
67 void modify_mul(int u, int L, int R, int l, int r, int x)
68 {
69     int mid = L + R >> 1;
70     if(l <= L && R <= r)
71     {
72         tr[u].sum *= x; tr[u].sum %= P;
73         tr[u].add *= x; tr[u].add %= P;
74         tr[u].mul *= x; tr[u].mul %= P;
75         return;
76     }
77     pushdown(u);
78     if(l <= mid)
79     {
80         modify_mul(u << 1, L, mid, l, r, x);
81     }
82     if(r >= mid + 1)
83     {
84         modify_mul(u << 1 | 1, mid + 1, R, l, r, x);
85     }
86     pushup(u);
87 }
88
89 ll query(int u, int L, int R, int l, int r)
90 {
91     if(l <= L && R <= r)
92     {
93         return tr[u].sum % P;
94     }
95     pushdown(u);
96     ll ans = 0;
97     int mid = L + R >> 1;
98     if(l <= mid)
99     {
100         ans += query(u << 1, L, mid, l, r);
101         ans %= P;
102     }
103     if(r >= mid + 1)
104     {
105         ans += query(u << 1 | 1, mid + 1, R, l, r);
106         ans %= P;

```

```

107     }
108     pushup(u);
109     return ans % P;
110 }

```

## pbds

```

1  #include<ext/pb_ds/tree_policy.hpp>
2  #include<ext/pb_ds/assoc_container.hpp>
3
4  using namespace __gnu_pbds;
5  __gnu_pbds::tree<ll, null_type, less<ll>, rb_tree_tag,
   tree_order_statistics_node_update> T;
6
7  if(op == 1)
8  {
9      T.insert({x, i});
10 }else if (op == 2)
11 {
12     T.erase(T.lower_bound({x, 0}));
13 }else if (op == 3)
14 {
15     cout << T.order_of_key({x, 0}) + 1 << "\n";
16 }else if (op == 4)
17 {
18     cout << T.find_by_order(x - 1)->first << "\n";
19 }else if (op == 5)
20 {
21     cout << prev(T.lower_bound({x, 0}))->first << "\n";
22 }else if (op == 6)
23 {
24     cout << T.lower_bound({x + 1, 0})->first << "\n";
25 }

```

## 六、简单计算几何

### 点

```

1  using i64 = long long;
2
3  using T = double;
4  struct Point {
5      T x;
6      T y;
7      Point(T x = 0, T y = 0) : x(x), y(y) {}
8
9      Point &operator+=(const Point &p) {
10         x += p.x, y += p.y;
11         return *this;
12     }
13     Point &operator-=(const Point &p) {

```

```

14     x -= p.x, y -= p.y;
15     return *this;
16 }
17 Point &operator*=(const T &v) {
18     x *= v, y *= v;
19     return *this;
20 }
21 friend Point operator-(const Point &p) {
22     return Point(-p.x, -p.y);
23 }
24 friend Point operator+(Point lhs, const Point &rhs) {
25     return lhs += rhs;
26 }
27 friend Point operator-(Point lhs, const Point &rhs) {
28     return lhs -= rhs;
29 }
30 friend Point operator*(Point lhs, const T &rhs) {
31     return lhs *= rhs;
32 }
33 };
34
35 T dot(const Point &a, const Point &b) {
36     return a.x * b.x + a.y * b.y;
37 }
38
39 T cross(const Point &a, const Point &b) {
40     return a.x * b.y - a.y * b.x;
41 }

```

## 七、杂项

### 矩阵快速幂

```

1  struct Matrix{
2      int n , m ;
3      vector<vector<ll>> s;
4
5      Matrix(int n , int m):n(n) ,m(m) , s(n , vector<ll>(m ,0)){}
6
7      friend Matrix operator * (Matrix a , Matrix b){
8          assert(a.m == b.n);
9          Matrix res(a.n , b.m);
10         for(int k = 0 ; k < a.m ; k ++ )
11             for(int i = 0 ; i < a.n ; i ++ )
12                 for(int j = 0 ; j < b.m ; j ++ )
13                     res.s[i][j] = (res.s[i][j] + a.s[i][k] * b.s[k][j] %
mod) % mod;
14         return res;
15     }
16
17     Matrix qmi(ll b){
18         assert(n == m);
19         Matrix res(n , n);
20         for(int i = 0 ; i < n ; i ++ )

```

```

21         res.s[i][i] = 1;
22         while(b){
23             if(b & 1)res = ((*this) * res );
24             b >>= 1;
25             *this = (*this) * (*this);
26         }
27         return (*this) = res;
28     };
29
30 };

```

## 组合数

```

1  ll fact[N] = {1}, inv[N] = {1};
2  ll C(ll x, ll y)
3  {
4      return(((fact[x] * inv[y])% MOD * inv[x-y]) % MOD);
5  }
6
7  ll P(ll x, ll y)
8  {
9      return fact[x] * inv[x - y] % MOD;
10 }
11
12 ll ksm(ll x, ll y)
13 {
14     ll ans = 1;
15     x %= MOD;
16     while(y)
17     {
18         if(y&1)
19         {
20             ans = ans * x % MOD;
21         }
22         x = x * x % MOD;
23         y /= 2;
24     }
25     return ans;
26 }
27
28 void build()
29 {
30     for(int i = 1 ; i < N ; i++ )
31     {
32         fact[i] = fact[i-1] * i % MOD;
33     }
34     for(int i = 1 ; i < N ; i++ )
35     {
36         inv[i] = inv[i-1] * ksm(i, MOD-2) % MOD;
37     }
38 }

```

