Zhejiang Normal University

School of Mathematical Sciences

ACM-ICPC Code Template

Author:
Jerry Black

Date:
2024 年 11 月 10 日

Contest

1 Graph

1.1 Dijkstra

```cpp
struct Edge
{
    int to,dis;
};
vector<Edge>edge[200005];
struct city
{
    int id,dis;
    bool operator<(const city& c)const
    {
        return dis>c.dis;
    }
};
priority_queue<city>q;
int dis[200005];
int vis[200005];
int n,m;
int s,t;
void init()
{
    for(int i=1;i<=n;i++)
    {
        dis[i]=INF;
        vis[i]=0;
    }
    dis[s]=0;
    q.push({s,0});
}
void dijkstra()
{
    while(!q.empty())
    {
        auto [id,d]=q.top();
        q.pop();
        if(vis[id])continue;
        else vis[id]=1;
        for(auto [it,dd]:edge[id])
        {
            if(dis[it]>dis[id]+dd)
            {
                dis[it]=dis[id]+dd;
                q.push({it,dis[it]});
            }
        }
    }
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        for(int i=0;i<n;i++)edge[i].clear();
        while(!q.empty())q.pop();
        for(int i=1;i<=m;i++)
        {
            int from,to,d;
            scanf("%d%d%d",&from,&to,&d);
            edge[from].push_back({to,d});
            edge[to].push_back({from,d});
        }
        scanf("%d%d",&s,&t);
        init();
        dijkstra();
        if(dis[t]==1e9)printf("-1\n");
        else printf("%d\n",dis[t]);
    }
    return 0;
}
```

1.2 匈牙利算法

```cpp
const int MAXN=505;
int k,n,m;
int Map[MAXN][MAXN];
int p[MAXN];
bool vis[MAXN];
bool match(int i)
{
    for (int j=1;j<=m;j++)
    {
        if(Map[i][j]&&!vis[j])
        {
            vis[j]=1;
            if(p[j]==0||match(p[j]))
            {
                p[j]=i;
                return 1;
            }
        }
    }
```

```
    return 0;
}
int main()
{
    while(scanf("%d",&k) != EOF)
    {
        memset(Map,0,sizeof(Map));
        memset(p,0,sizeof(p));
        if(k==0)return 0;
        scanf("%d%d",&n,&m);
        for (int i=1;i<=k;i++)
        {
            int x,y;
            scanf("%d%d",&x,&y);
            Map[x][y]=1;
        }
        int cnt=0;
        for (int i=1;i<=n;i++)
        {
            memset(vis,0,sizeof(vis));
            if(match(i))cnt++;
        }
        printf("%d\n",cnt);
    }
    return 0;
}
```

1.3 Hopcroft_Karp(匈牙利算法优化)

```
struct Hopcroft_Karp
{
    vector<vector<int>>es;
    vector<int>d,match;
    vector<bool>used,used2;
    const int n,m;
    Hopcroft_Karp(int n,int m):es(n),d(n),match(m),used(n),used2(n),n(n
        ),m(m){}
    void add_edge(int u,int v){es[u].push_back(v);}
    void _bfs()
    {
        fill(begin(d),end(d),-1);
        queue<int>que;
        for(int i=0;i<n;i++)
        {
            if (!used[i]){que.push(i);d[i] = 0;}
        }
        while (!que.empty())
        {
            int i=que.front();
            que.pop();
            for(auto &e:es[i])
            {
                int j=match[e];
                if(j!=-1&&d[j]==-1){que.push(j);d[j]=d[i]+1;}
            }
        }
    }
    bool _dfs(int now)
    {
        used2[now]=true;
        for (auto &e:es[now])
        {
            int u=match[e];
            if(u==-1||(!used2[u]&&d[u]==d[now]+1&& _dfs(u)))
            {
                match[e]=now,used[now]=true;
                return true;
            }
        }
        return false;
    }
    int max_matching() // 右边的i与左边的match[i]匹配。
    {
        fill(begin(match),end(match),-1);
        fill(begin(used),end(used),false);
        int ret=0;
        while(1)
        {
            _bfs();
            fill(begin(used2),end(used2),false);
            int flow=0;
            for(int i=0;i<n;i++){if(!used[i]&&_dfs(i))flow++;}
            if(flow==0)break;
            ret+=flow;
        }
        return ret;
    }
};

string s[305];
int x[305][305],y[305][305];

void solve()
```

```
{
    int n,m;
    cin>>n>>m;
    for(int i=0;i<n;i++)cin>>s[i];
    int l=0,r=0;
    memset(x,-1,sizeof(x));
    memset(y,-1,sizeof(y));
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            if(s[i][j]=='#')continue;
            x[i][j]=l;
            while(j+1<m&&s[i][j+1]=='.'){x[i][j+1]=l;j++;}
            l++;
        }
    }
    for(int j=0;j<m;j++)
    {
        for(int i=0;i<n;i++)
        {
            if(s[i][j]=='#')continue;
            y[i][j]=r;
            while(i+1<n&&s[i+1][j]=='.'){y[i+1][j]=r;i++;}
            r++;
        }
    }
    if(l==0&&r==0)
    {
        cout<<0;
        return;
    }
    Hopcroft_Karp BM(l,r);
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            if(x[i][j]!=-1&&y[i][j]!=-1)BM.add_edge(x[i][j],y[i][j]);
        }
    }
    cout<<BM.max_matching();
}
```

1.4 二分图染色 + 拓扑排序

```
#include<bits/stdc++.h>
using namespace std;
struct edge
{
    int type,u,v;
};
vector<int>adj[200005];
int col[200005],topo[200005];
void dfs(int v)
{
    for(int u:adj[v])if(col[u]==-1)
    {
        col[u]=col[v]^1;
        dfs(u);
    }
}
bool BipartiteColoring(int n)
{
    for(int i=1;i<=n;i++)col[i]=-1;
    for(int i=1;i<=n;i++)if(col[i]==-1)
    {
        col[i]=0;
        dfs(i);
    }
    for(int i=1;i<=n;i++)for(int j:adj[i])if(col[i]==col[j])return 0;
    return 1;
}
bool TopologicalSort(int n)
{
    vector<int>in(n+1,0);
    for(int i=1;i<=n;i++)for(int j:adj[i])in[j]++;
    queue<int>q;
    for(int i=1;i<=n;i++)if(in[i]==0)q.push(i);
    int ord=0;
    while(!q.empty())
    {
        int v=q.front();
        q.pop();
        topo[v]=ord++;
        for(int u:adj[v])
        {
            in[u]--;
            if(in[u]==0)q.push(u);
        }
    }
    return ord==n;
}
int main()
```

```
{
    int n,m;
    scanf("%d%d",&n,&m);
    vector<edge>a(m);
    for(int i=0;i<m;i++)
    {
        scanf("%d%d%d",&a[i].type,&a[i].u,&a[i].v);
        adj[a[i].u].push_back(a[i].v);
        adj[a[i].v].push_back(a[i].u);
    }
    if(!BipartiteColoring(n)){printf("NO\n");return 0;}
    // col = 0 -> orient left, col = 1 -> orient right
    for(int i=1;i<=n;i++)adj[i].clear();
    for(edge e:a)
    {
        if(col[e.u]==1)swap(e.u,e.v);
        if(e.type==1)adj[e.u].push_back(e.v);
        else adj[e.v].push_back(e.u);
    }
    if(!TopologicalSort(n)){printf("NO\n");return 0;}
    printf("YES\n");
    for(int i=1;i<=n;i++)
    {
        col[i]==1?printf("R %d\n",topo[i]):printf("L %d\n",topo[i]);
    }
    return 0;
}
```

1.5 网络流

1.5.1 最大流

```
const int MAXN=1000005;
const int inf=1000000000;
int S,T;
struct EDGE
{
    int v,flow,nxt;
}edge[MAXN<<2];
int head[MAXN],cur[MAXN],num=0;
int dep[MAXN],q[MAXN];
void add_edge(int u,int v,int w)
{
    edge[num].v=v;
    edge[num].flow=w;
    edge[num].nxt=head[u];
    head[u]=num++;
}
void Add_edge(int u,int v,int w)
{
    add_edge(u,v,w);
    add_edge(v,u,0);
}
bool bfs()
{
    memset(dep,0,sizeof(dep));
    dep[S]=1;
    int l=0,r=1;
    q[++l]=S;
    while(l<=r)
    {
        int p=q[l++];
        if(p==T)break;
        for(int i=head[p];~i;i=edge[i].nxt)
        {
            if(!dep[edge[i].v]&&edge[i].flow)
            {
                dep[edge[i].v]=dep[p]+1;
                q[++r]=edge[i].v;
                if(edge[i].v==T)return 1;
            }
        }
    }
    return dep[T];
}
int dfs(int now,int nowflow)
{
    if(now==T)return nowflow;
    int totflow=0;
    for(int& i=cur[now];~i;i=edge[i].nxt)
    {
        if(dep[edge[i].v]==dep[now]+1&&edge[i].flow)
        {
            int canflow=dfs(edge[i].v,min(nowflow,edge[i].flow));
            edge[i].flow-=canflow;
            edge[i^1].flow+=canflow;
            totflow+=canflow;
            nowflow-=canflow;
            if(nowflow<=0)break;
        }
    }
    return totflow;
}
```

```
int dinic()
{
   int ans=0;
   while(bfs())
   {
      memcpy(cur,head,sizeof(head));
      ans+=dfs(S,inf);
   }
   return ans;
}
void init()
{
   memset(q,0,sizeof(q));
   memset(cur,0,sizeof(cur));
   memset(dep,0,sizeof(dep));
   memset(head,-1,sizeof(head));
   for(int i=0;i<=num;i++)edge[i].v=edge[i].flow=edge[i].nxt=0;
   num=0;
}
```

1.5.2 KM

```
//KM算法适用于求解完美匹配下的最大权匹配,时间复杂度O(n^3)
const int INF=0x3f3f3f3f;
const int N=305;
int n,m,match[N],pre[N];
bool vis[N];
int favor[N][N];
int val1[N],val2[N],slack[N];
void bfs(int p)
{
   memset(pre,0,sizeof pre);
   memset(slack,INF,sizeof slack);
   match[0]=p;
   int x=0,nex=0;
   do{
      vis[x]=true;
      int y=match[x],d=INF;
      // 对于当前节点y, bfs有连边的下一点
      for(int i=1;i<=m;i++)
      {
         if(!vis[i])
         {
            if(slack[i]>val1[y]+val2[i]-favor[y][i])
            {
               slack[i]=val1[y]+val2[i]-favor[y][i];
               pre[i]=x;
            }
            if(slack[i]<d)
            {
               d=slack[i];
               nex=i;
            }
         }
      }
      for(int i=0;i<=m;i++)
      {
         if(vis[i])
         val1[match[i]]-=d,val2[i]+=d;
         else
         slack[i]-=d;
      }
      x=nex;
   }while(match[x]);

   // pre数组对bfs访问路径进行记录，在最后一并改变match
   while(x)
   {
      match[x]=match[pre[x]];
      x=pre[x];
   }
}
int KM()
{
   memset(match,0,sizeof match);
   memset(val1,0,sizeof val1);
   memset(val2,0,sizeof val2);
   for(int i=1;i<=n;i++)
   {
      memset(vis,false,sizeof vis);
      bfs(i);
   }
   int res=0;
   for(int i=1;i<=m;i++)
   res+=favor[match[i]][i];
   return res;
}
int main()
{
   // Input favor[i][j]

   printf("%d",KM());

}
```

```
    return 0;
}
```

1.5.3 最小费用最大流

```
const int maxn=1050;

struct MCMF {
    struct E {
        int from, to, cap, v;
        E() {}
        E(int f, int t, int cap, int v) : from(f), to(t), cap(cap), v(v)
            {}
    };
    int n, m, s, t;
    vector<E> edges;
    vector<int> G[maxn];
    bool inq[maxn];
    int dis[maxn], pre[maxn], a[maxn];
    void init(int _n, int _s, int _t) {
        n = _n; s = _s; t = _t;
        for (int i = 0; i <= n; i++)
            G[i].clear();
        edges.clear();
        m = 0;
    }
    void add(int from, int to, int cap, int cost) {
        edges.emplace_back(from, to, cap, cost);
        edges.emplace_back(to, from, 0, -cost);
        G[from].push_back(m++);
        G[to].push_back(m++);
    }
    bool spfa() {
        for (int i = 0; i <= n; i++) {
            dis[i] = 1e9;
            pre[i] = -1;
            inq[i] = false;
        }
        dis[s] = 0, a[s] = 1e9, inq[s] = true;
        queue<int> Q; Q.push(s);
        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            inq[u] = false;
            for (int& idx: G[u]) {
                E& e = edges[idx];
                if (e.cap && dis[e.to] > dis[u] + e.v) {
                    dis[e.to] = dis[u] + e.v;
                    pre[e.to] = idx;
                    a[e.to] = min(a[u], e.cap);
                    if (!inq[e.to]) {
                        inq[e.to] = true;
                        Q.push(e.to);
                    }
                }
            }
        }
        return pre[t] != -1;
    }
    int solve() {
        int flow = 0, cost = 0;
        while (spfa()) {
            flow += a[t];
            cost += a[t] * dis[t];
            int u = t;
            while (u != s) {
                edges[pre[u]].cap -= a[t];
                edges[pre[u] ^ 1].cap += a[t];
                u = edges[pre[u]].from;
            }
        }
        return cost;
    }
}f;
```

1.5.4 最大费用可行流

```
struct MCFF{ // Maximum cost feasible flow 最大费用可行流
    int n, s, t, cost, dis[N], st[N], vis[N];
    int head[N], nxt[M], v[M], w[M], flow[M], ecnt;

    inline void ade(int a, int b, int c, int dis){ v[++ecnt]=b; nxt[
        ecnt]=head[a]; w[ecnt]=dis; flow[ecnt]=c; head[a]=ecnt;}

    inline void addedge(int a, int b, int c, int dis){ ade(a,b,c,dis);
        ade(b,a,0,-dis);}

    inline void init(int _n, int _s, int _t){
        n=_n; s=_s; t=_t; ecnt=1; cost=0;
        memset(head, 0, sizeof head);
    }

    inline int spfa(){
```

```
      queue<int> q; q.push(s);
      memset(st,0,sizeof st);
      memset(dis,0x3f,sizeof dis); dis[s]=0;
      while(q.size()){
         int u=q.front(); q.pop(); vis[u]=0;
         for(int i=head[u];i;i=nxt[i]){
            if(flow[i] and dis[v[i]]>dis[u]+w[i]){
               dis[v[i]]=dis[u]+w[i];
               if(!vis[v[i]]) q.push(v[i]), vis[v[i]]=1;
            }
         }
      } return dis[t]<0;
   }

   int dfs(int x, int f){
      if(x==t) return cost+=dis[t]*f,f;
      st[x]=1; int fl=0;
      for(int i=head[x];f and i;i=nxt[i]){
         if(flow[i] and !st[v[i]] and dis[v[i]]==dis[x]+w[i]){
            int mi=dfs(v[i], min(f, flow[i]));
            flow[i]-=mi; flow[i^1]+=mi; fl+=mi; f-=mi;
         }
      } return fl;
   }

   inline int work(){
      while(spfa()) dfs(s, INF);
      return cost;
   }
}f;
```

1.6 连通分量

1.6.1 强连通分量

```
int dfn[N], low[N], dfncnt, s[N], in_stack[N], tp;
int scc[N], sc; // 结点 i 所在 SCC 的编号
int sz[N]; // 强连通 i 的大小

void tarjan(int u) {
   low[u] = dfn[u] = ++dfncnt, s[++tp] = u, in_stack[u] = 1;
   for (int i = h[u]; i; i = e[i].nex) {
      const int &v = e[i].t;
      if (!dfn[v]) {
         tarjan(v);
         low[u] = min(low[u], low[v]);
      } else if (in_stack[v]) {
         low[u] = min(low[u], dfn[v]);
      }
   }
   if (dfn[u] == low[u]) {
      ++sc;
      while (s[tp] != u) {
         scc[s[tp]] = sc;
         sz[sc]++;
         in_stack[s[tp]] = 0;
         --tp;
      }
      scc[s[tp]] = sc;
      sz[sc]++;
      in_stack[s[tp]] = 0;
      --tp;
   }
}
```

1.6.2 边双连通分量

```
#include <algorithm>
#include <cstdio>
#include <stack>
#include <vector>

using namespace std;
const int N = 5e5 + 5, M = 2e6 + 5;
int n, m;

struct edge {
  int to, nt;
} e[M << 2];

int hd[N << 1], tot = 1;

void add(int u, int v) { e[++tot] = (edge){v, hd[u]}, hd[u] = tot; }

void uadd(int u, int v) { add(u, v), add(v, u); }

int bcc_cnt, sum;
int dfn[N], low[N];
bool vis[N];
vector<vector<int>> ans;
stack<int> st;

```

```
void tarjan(int u, int in) {
   low[u] = dfn[u] = ++bcc_cnt;
   st.push(u), vis[u] = 1;
   for (int i = hd[u]; i; i = e[i].nt) {
      int v = e[i].to;
      if (i == (in ^ 1)) continue;
      if (!dfn[v])
      tarjan(v, i), low[u] = min(low[u], low[v]);
      else if (vis[v])
      low[u] = min(low[u], dfn[v]);
   }
   if (dfn[u] == low[u]) {
      vector<int> t;
      t.push_back(u);
      while (st.top() != u) t.push_back(st.top()), vis[st.top()] = 0,
          st.pop();
      st.pop(), ans.push_back(t);
   }
}

int main() {
   scanf("%d%d", &n, &m);
   int u, v;
   for (int i = 1; i <= m; i++) {
      scanf("%d%d", &u, &v);
      if (u != v) uadd(u, v);
   }
   for (int i = 1; i <= n; i++)
      if (!dfn[i]) tarjan(i, 0);
   printf("%llu\n", ans.size());
   for (int i = 0; i < ans.size(); i++) {
      printf("%llu ", ans[i].size());
      for (int j = 0; j < ans[i].size(); j++) printf("%d ", ans[i][j])
          ;
      printf("\n");
   }
   return 0;
}
```

1.6.3 点双连通分量

```
#include <cstdio>
#include <vector>
using namespace std;
const int N = 5e5 + 5, M = 2e6 + 5;
int n, m;

struct edge {
  int to, nt;
} e[M << 1];

int hd[N], tot = 1;

void add(int u, int v) { e[++tot] = (edge){v, hd[u]}, hd[u] = tot; }

void uadd(int u, int v) { add(u, v), add(v, u); }

int ans;
int dfn[N], low[N], bcc_cnt;
int sta[N], top, cnt;
bool cut[N];
vector<int> dcc[N];
int root;

void tarjan(int u) {
   dfn[u] = low[u] = ++bcc_cnt, sta[++top] = u;
   if (u == root && hd[u] == 0) {
      dcc[++cnt].push_back(u);
      return;
   }
   int f = 0;
   for (int i = hd[u]; i; i = e[i].nt) {
      int v = e[i].to;
      if (!dfn[v]) {
         tarjan(v);
         low[u] = min(low[u], low[v]);
         if (low[v] >= dfn[u]) {
            if (++f > 1 || u != root) cut[u] = true;
            cnt++;
            do dcc[cnt].push_back(sta[top--]);
            while (sta[top + 1] != v);
            dcc[cnt].push_back(u);
         }
      } else
         low[u] = min(low[u], dfn[v]);
   }
}

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

```
        if (u != v) uadd(u, v);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i]) root = i, tarjan(i);
    printf("%d\n", cnt);
    for (int i = 1; i <= cnt; i++) {
        printf("%llu ", dcc[i].size());
        for (int j = 0; j < dcc[i].size(); j++) printf("%d ", dcc[i][j])
            ;
        printf("\n");
    }
    return 0;
}
```

1.6.4 割点

```
#include <bits/stdc++.h>
using namespace std;
int n, m; // n: 点数 m: 边数
int dfn[100001], low[100001], inde, res;
// dfn: 记录每个点的时间戳
// low: 能不经过父亲到达最小的编号, inde: 时间戳, res: 答案数量
bool vis[100001], flag[100001]; // flag: 答案 vis: 标记是否重复
vector<int> edge[100001]; // 存图用的

void Tarjan(int u, int father) { // u 当前点的编号, father 自己爸爸的编号
    vis[u] = true; // 标记
    low[u] = dfn[u] = ++inde; // 打上时间戳
    int child = 0; // 每一个点儿子数量
    for (auto v : edge[u]) { // 访问这个点的所有邻居 (C++11)
        if (!vis[v]) {
        child++; // 多了一个儿子
        Tarjan(v, u); // 继续
        low[u] = min(low[u], low[v]); // 更新能到的最小节点编号
        if (father != u && low[v] >= dfn[u] && !flag[u]) { // 主要代码
            // 如果不是自己, 且不通过父亲返回的最小点符合割点的要求, 并且没有被标记
                过
            // 要求即为: 删了父亲连不上去了, 即为最多连到父亲
            flag[u] = true;
            res++; // 记录答案
        }
        } else if (v != father) {
        // 如果这个点不是自己的父亲, 更新能到的最小节点编号
        low[u] = min(low[u], dfn[v]);
        }
    }
    // 主要代码, 自己的话需要 2 个儿子才可以
    if (father == u && child >= 2 && !flag[u]) {
        flag[u] = true;
        res++; // 记录答案
    }
}

int main() {
    cin >> n >> m; // 读入数据
    for (int i = 1; i <= m; i++) { // 注意点是从 1 开始的
        int x, y;
        cin >> x >> y;
        edge[x].push_back(y);
        edge[y].push_back(x);
    } // 使用 vector 存图
    for (int i = 1; i <= n; i++) // 因为 Tarjan 图不一定连通
        if (!vis[i]) {
        inde = 0; // 时间戳初始为 0
        Tarjan(i, i); // 从第 i 个点开始, 父亲为自己
        }
    cout << res << endl;
    for (int i = 1; i <= n; i++)
        if (flag[i]) cout << i << " "; // 输出结果
    return 0;
}
```

1.6.5 割边

```
int low[MAXN], dfn[MAXN], dfs_clock;
bool isbridge[MAXN];
vector<int> G[MAXN];
int cnt_bridge;
int father[MAXN];

void tarjan(int u, int fa) {
    father[u] = fa;
    low[u] = dfn[u] = ++dfs_clock;
    for (int i = 0; i < G[u].size(); i++) {
        int v = G[u][i];
        if (!dfn[v]) {
        tarjan(v, u);
        low[u] = min(low[u], low[v]);
        if (low[v] > dfn[u]) {
            isbridge[v] = true;
            ++cnt_bridge;
        }
```

```
        } else if (dfn[v] < dfn[u] && v != fa) {
            low[u] = min(low[u], dfn[v]);
        }
    }
}
```

1.6.6 圆方树

```
#include <algorithm>
#include <cstdio>
#include <vector>

const int MN = 100005;

int N, M, cnt;
std::vector<int> G[MN], T[MN * 2];

int dfn[MN], low[MN], dfc;
int stk[MN], tp;

void Tarjan(int u) {
    printf(" Enter : #%d\n", u);
    low[u] = dfn[u] = ++dfc; // low 初始化为当前节点 dfn
    stk[++tp] = u; // 加入栈中
    for (int v : G[u]) { // 遍历 u 的相邻节点
        if (!dfn[v]) { // 如果未访问过
            Tarjan(v); // 递归
            low[u] = std::min(low[u], low[v]); // 未访问的和 low 取 min
            if (low[v] == dfn[u]) { // 标志着找到一个以 u 为根的点双连通分量
                ++cnt; // 增加方点个数
                printf(" Found a New BCC #%d.\n", cnt - N);
                // 将点双中除了 u 的点退栈，并在圆方树中连边
                for (int x = 0; x != v; --tp) {
                    x = stk[tp];
                    T[cnt].push_back(x);
                    T[x].push_back(cnt);
                    printf(" BCC #%d has vertex #%d\n", cnt - N, x);
                }
                // 注意 u 自身也要连边（但不退栈）
                T[cnt].push_back(u);
                T[u].push_back(cnt);
                printf(" BCC #%d has vertex #%d\n", cnt - N, u);
            }
        } else
            low[u] = std::min(low[u], dfn[v]); // 已访问的和 dfn 取 min
    }
    printf(" Exit : #%d : low = %d\n", u, low[u]);
    printf(" Stack:\n ");
    for (int i = 1; i <= tp; ++i) printf("%d, ", stk[i]);
    puts("");
}

int main() {
    scanf("%d%d", &N, &M);
    cnt = N; // 点双 / 方点标号从 N 开始
    for (int i = 1; i <= M; ++i) {
        int u, v;
        scanf("%d%d", &u, &v);
        G[u].push_back(v); // 加双向边
        G[v].push_back(u);
    }
    // 处理非连通图
    for (int u = 1; u <= N; ++u)
        if (!dfn[u]) Tarjan(u), --tp;
    // 注意到退出 Tarjan 时栈中还有一个元素即根，将其退栈
    return 0;
}
```

1.7 2-SAT

```
struct Twosat {
    int n;
    vector<int> g[maxn * 2];
    bool mark[maxn * 2];
    int s[maxn * 2], c;

    bool dfs(int x) {
        if (mark[x ^ 1]) return false;
        if (mark[x]) return true;
        mark[x] = true;
        s[c++] = x;
        for (int i = 0; i < (int)g[x].size(); i++)
            if (!dfs(g[x][i])) return false;
        return true;
    }

    void init(int n) {
        this->n = n;
        for (int i = 0; i < n * 2; i++) g[i].clear();
        memset(mark, 0, sizeof(mark));
    }

```

```
    void add_clause(int x, int y) { // 这个函数随题意变化
        g[x].push_back(y ^ 1); // 选了 x 就必须选 y^1
        g[y].push_back(x ^ 1);
    }

    bool solve() {
        for (int i = 0; i < n * 2; i += 2)
        if (!mark[i] && !mark[i + 1]) {
            c = 0;
            if (!dfs(i)) {
                while (c > 0) mark[s[--c]] = false;
                if (!dfs(i + 1)) return false;
            }
        }
        return true;
    }
};
```

1.8 虚树

1.9 树分治

1.9.1 点分治

```
// 给定一棵有 n 个点的带边权树，m 次询问，每次询问给出 k，询问树上距离为 k 的点对是否存在。
// n <= 10000, m <= 100, k <= 10000000

#include <cmath>
#include <queue>
#include <cstdio>
#include <bitset>
#include <cstring>
#include <iostream>
#include <algorithm>

#define endl '\n'

using namespace std;
using i64 = long long;

const int si = 3e5 + 10;
const int inf = 1e9 + 7;

int n, m, q[si];
int tot = 0, head[si];
struct Edge { int ver, Next, w; } e[si << 1];
inline void add(int u, int v, int w) { e[tot] = (Edge){v, head[u], w},
    head[u] = tot++; }

std::queue<int> rec;
bool tf[10000010], can[si], vis[si];
// tf: 当前子树的可行性。

int cnt = 0, sum = 0;
int maxv[si], rt = 0;
int d[si], dis[si], siz[si];
// d: 当前子树的 节点-根 距离。
void calcsiz(int u, int fa) {
    siz[u] = 1, maxv[u] = 0;
    for(int i = head[u]; ~i; i = e[i].Next) {
        int v = e[i].ver;
        if(v == fa || vis[v]) continue;
        calcsiz(v, u);
        maxv[u] = max(maxv[u], siz[v]), siz[u] += siz[v];
    }
    maxv[u] = max(maxv[u], sum - siz[u]); // 注意这里是当前子树的节点个数。
    if(maxv[rt] > maxv[u]) rt = u;
}
void calcdis(int u, int fa) {
    d[++cnt] = dis[u]; // 这里复制是为了枚举的时候不全部枚举，保证复杂度。
    for(int i = head[u]; ~i; i = e[i].Next) {
        int v = e[i].ver, w = e[i].w;
        if(v == fa || vis[v]) continue;
        dis[v] = dis[u] + w, calcdis(v, u);
    }
}
void dfs(int u, int fa) {
    tf[0] = true, rec.push(0), vis[u] = true; // 打 vis 是为了确保在子树中进行操作，不会递归出去。
    // 或者不妨说，我们是利用 vis，将树划分成了一个个联通块来处理，因为它每次都只会标记到重心嘛。
    for(int i = head[u]; ~i; i = e[i].Next) {
        int v = e[i].ver, w = e[i].w;
        if(v == fa || vis[v]) continue;
        dis[v] = w, calcdis(v, u);
        for(int j = 1; j <= cnt; ++j) {
            for(int k = 1; k <= m; ++k) {
                if(q[k] >= d[j]) can[q[k]] |= tf[q[k] - d[j]];
            }
        } // 先判断再添加，不然算的不是除了自己子树的情况，这样会多算。
        for(int j = 1; j <= cnt; ++j) {
            if(d[j] < 10000010) rec.push(d[j]), tf[d[j]] = true;
        }
```

```
        cnt = 0;
    }

    while(!rec.empty()) tf[rec.front()] = false, rec.pop();
    for(int i = head[u]; ~i; i = e[i].Next) {
        int v = e[i].ver;
        if(v == fa || vis[v]) continue;
        rt = 0, maxv[rt] = inf, sum = siz[v];
        calcsiz(v, u), calcsiz(rt, -1), dfs(rt, u); // 先找重心再递归。
    }
}

int main() {

    cin.tie(0) -> sync_with_stdio(false);
    cin.exceptions(cin.failbit | cin.badbit);

    memset(tf, false, sizeof tf);
    memset(head, -1, sizeof head);
    memset(vis, false, sizeof vis);
    memset(can, false, sizeof can);

    cin >> n >> m;
    for(int i = 1; i < n; ++i) {
        int u, v, w;
        cin >> u >> v >> w;
        add(u, v, w), add(v, u, w);
    }
    for(int nw = 1; nw <= m; ++nw) {
        cin >> q[nw];
    }

    rt = 0, maxv[rt] = inf, sum = n;
    calcsiz(1, -1), calcsiz(rt, -1), dfs(rt, -1); // 因为本题需要用到 tf
        (0) 所以 fa 就用 -1 了。

    for(int nw = 1; nw <= m; ++nw) {
        if(can[q[nw]]) cout << "AYE" << endl;
        else cout << "NAY" << endl;
    }

    return 0;
}
```

1.9.2 点分树

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

using namespace std;

typedef vector<int>::iterator IT;

struct Edge {
  int to, nxt, val;

  Edge() {}

  Edge(int to, int nxt, int val) : to(to), nxt(nxt), val(val) {}
} e[300010];

int head[150010], cnt;

void addedge(int u, int v, int val) {
  e[++cnt] = Edge(v, head[u], val);
  head[u] = cnt;
}

int siz[150010], son[150010];
bool vis[150010];

int tot, lasttot;
int maxp, root;

void getG(int now, int fa) {
  siz[now] = 1;
  son[now] = 0;
  for (int i = head[now]; i; i = e[i].nxt) {
    int vs = e[i].to;
    if (vs == fa || vis[vs]) continue;
    getG(vs, now);
    siz[now] += siz[vs];
    son[now] = max(son[now], siz[vs]);
  }
  son[now] = max(son[now], tot - siz[now]);
  if (son[now] < maxp) {
    maxp = son[now];
    root = now;
  }
}

struct Node {
  int fa;
  vector<int> anc;
```

```
  vector<int> child;
} nd[150010];

int build(int now, int ntot) {
  tot = ntot;
  maxp = 0x7f7f7f7f;
  getG(now, 0);
  int g = root;
  vis[g] = 1;
  for (int i = head[g]; i; i = e[i].nxt) {
    int vs = e[i].to;
    if (vis[vs]) continue;
    int tmp = build(vs, ntot - son[vs]);
    nd[tmp].fa = now;
    nd[now].child.push_back(tmp);
  }
  return g;
}

int virtroot;

int main() {
  int n;
  cin >> n;
  for (int i = 1; i < n; i++) {
    int u, v, val;
    cin >> u >> v >> val;
    addedge(u, v, val);
    addedge(v, u, val);
  }
  virtroot = build(1, n);
}
```

```
int k, a[si];
int stk[si], top = 0;
bool cmp(int x, int y) { return dfn[x] < dfn[y]; }
inline void ADD(int u, int v, int w) { E[Tot] = (Edge){v, Head[u], w},
    Head[u] = Tot++; }
inline void Add(int u, int v) { int w = dist(u, v); ADD(u, v, w), ADD(
    v, u, w); }
void build() {
   sort(a + 1, a + 1 + k, cmp);
   stk[top = 1] = 1, Tot = 0, Head[1] = -1; // 这样清空复杂度才是对的。
   for(int i = 1, Lca; i <= k; ++i) {
      if(a[i] == 1) continue;
      Lca = lca(a[i], stk[top]);
      if(Lca != stk[top]) {
         while(dfn[Lca] < dfn[stk[top - 1]])
            Add(stk[top - 1], stk[top]), --top;
         if(dfn[Lca] > dfn[stk[top - 1]])
            Head[Lca] = -1, Add(Lca, stk[top]), stk[top] = Lca;
         else Add(Lca, stk[top--]); // Lca = stk[top - 1].
      }
      Head[a[i]] = -1, stk[++top] = a[i];
   }
   for(int i = 1; i < top; ++i)
      Add(stk[i], stk[i + 1]);
   return;
}
```

1.10 异或最小生成树

```
/*
给定n个点的无向完全图,
给出每个点的点权ai,
两点之间的边权是两点权异或值,
求最小生成树
ai<2^30
复杂度o(nlogn)
*/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod=998244353;

struct
{
   int nxt[2];
}trie[8000005];
int cnt;

void Insert(int x)
{
   int now=0;
   for(int i=29;i>=0;i--)
   {
      int t=(x>>i)&1;
      if(!trie[now].nxt[t])trie[now].nxt[t]=++cnt;
      now=trie[now].nxt[t];
   }
}
ll Find_MIN(int x)
{
```

```
    ll ans=0;
    int now=0;
    for(int i=29;i>=0;i--)
    {
        int t=(x>>i)&1;
        if(trie[now].nxt[t])
        {
            now=trie[now].nxt[t];
        }
        else
        {
            now=trie[now].nxt[t^1];
            ans|=1<<i;
        }
    }
    return ans;
}

int a[200005];
ll fenzhi(int *a,int l,int r,int id)
{
    if(l>=r||id==-1)return 0;
    int mid=r;
    while(mid>=l&&((a[mid]>>id)&1))mid--;
    ll ans=0;
    ans+=fenzhi(a,l,mid,id-1);
    ans+=fenzhi(a,mid+1,r,id-1);
    if(mid==l-1||mid==r)return ans;
    for(int i=l;i<=mid;i++)Insert(a[i]);
    ll minn=1e18;
    for(int i=mid+1;i<=r;i++)minn=min(minn,Find_MIN(a[i]));
    ans+=minn;
    for(int i=0;i<=cnt;i++)trie[i].nxt[0]=trie[i].nxt[1]=0;
    cnt=0;
    return ans;
}

int main()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);
    sort(a+1,a+n+1);
    ll ans=fenzhi(a,1,n,29);//29是位数
    printf("%lld\n",ans);
    return 0;
}
```

2 Data Structure

2.1 种类并查集

```
#include<bits/stdc++.h>
using namespace std;
int fa[150005];
int findd(int x)
{
    return fa[x]==x?x:fa[x]=findd(fa[x]);
}
int main()
{
    int n,k;
    scanf("%d%d",&n,&k);
    for(int i=1;i<=3*n;i++)fa[i]=i;
    int ans=0;
    for(int i=1;i<=k;i++)
    {
        int d,x,y;
        scanf("%d%d%d",&d,&x,&y);
        if(x>n||y>n){ans++;continue;}
        if(d==1)
        {
            if(findd(x)==findd(y+n)||findd(x)==findd(y+2*n)){ans++;
                continue;}
            fa[findd(y)]=findd(x);
            fa[findd(y+n)]=findd(x+n);
            fa[findd(y+2*n)]=findd(x+2*n);
        }
        else
        {
            if(findd(x)==findd(y+2*n)||findd(x)==findd(y)){ans++;continue
                ;}
            fa[findd(y+n)]=findd(x);
            fa[findd(y+2*n)]=findd(x+n);
            fa[findd(y)]=findd(x+2*n);
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

2.2 二维树状数组

2.2.1 单点修改，区间查询

```
int tree[maxn][maxn];
int n,m;
int lowbit(int x){
   return x&(-x);
}
void add(int x,int y,int w){
   while(x<=n){
      for(int i=y;i<=m;i+=lowbit(i))tree[x][i]+=w;
      x+=lowbit(x);
   }
}
int sum(int x,int y){
   int res=0;
   while(x>0){
      for(int i=y;i>0;i-=lowbit(i)){
         res+=tree[x][i];
      }
      x-=lowbit(x);
   }
   return res;
}
int query(int x1,int y1,int x2,int y2){//左上右下
   return sum(x2,y2)+sum(x1-1,y1-1)-sum(x2,y1-1)-sum(x1-1,y2);
}
```

2.2.2 区间修改，单点查询

```
void update(int x1,int y1,int x2,int y2,int w){//左上右下
   add(x1,y1,w);
   add(x2+1,y1,-w);
   add(x1,y2+1,-w);
   add(x2+1,y2+1,w);
}
int query(int x,int y){
   return sum(x,y);
}
```

2.2.3 区间修改，区间查询

```
int n,m,q;
ll t1[maxn][maxn],t2[maxn][maxn],t3[maxn][maxn],t4[maxn][maxn];
int lowbit(int x){
   return x&(-x);
}
void add(int x,int y,int w){
   for(int i=x;i<=n;i+=lowbit(i)){
      for(int j=y;j<=m;j+=lowbit(j)){
         t1[i][j]+=w;
         t2[i][j]+=x*w;
         t3[i][j]+=w*y;
         t4[i][j]+=w*x*y;
      }
   }
}
void update(int x1,int y1,int x2,int y2,int w){//左上右下
   add(x1,y1,w);
   add(x1,y2+1,-w);
   add(x2+1,y1,-w);
   add(x2+1,y2+1,w);
}
ll ask(int x,int y){
   ll res=0;
   for(int i=x;i;i-=lowbit(i)){
      for(int j=y;j;j-=lowbit(j)){
         res+=(x+1)*(y+1)*t1[i][j]-(y+1)*t2[i][j]
         -(x+1)*t3[i][j]+t4[i][j];
      }
   }
   return res;
}
ll query(int x1,int y1,int x2,int y2){//左上右下
   return ask(x2,y2)-ask(x2,y1-1)-ask(x1-1,y2)+ask(x1-1,y1-1);
}
```

2.3 线段树

2.3.1 维护等差数列

```
#include<bits/stdc++.h>
using namespace std;
namespace Segtree
{
   int n,m;
   double tree[1005<<2],laz1[1005<<2],laz2[1005<<2];
   void pushdown(int p,int l,int r)
   {
      int mid=(l+r)>>1;
      double x=laz1[p],y=laz2[p];
      laz1[p<<1]+=x;
      laz2[p<<1]+=y;
```

```
        laz1[p<<1|1]+=x+y*(mid-l+1);
        laz2[p<<1|1]+=y;
        tree[p<<1]+=(x+x+(mid-l)*y)*(mid-l+1)/2;
        tree[p<<1|1]+=(x+y*(mid-l+1)+x+y*(mid-l+1)+(r-mid-1)*y)*(r-mid)/2;
        laz1[p]=laz2[p]=0;
    }
    void build(int p,int l,int r)
    {
        laz1[p]=laz2[p]=0;
        if(l==r)tree[p]=n-1;
        else
        {
            int mid=(l+r)>>1;
            build(p<<1,l,mid);
            build(p<<1|1,mid+1,r);
            tree[p]=tree[p<<1]+tree[p<<1|1];
        }
    }
    void update(int p,int l,int r,double v,double d,int x,int y)
    {
        if(x<=l&&r<=y)
        {
            double vv=v+(l-x)*d;
            laz1[p]+=vv;
            laz2[p]+=d;
            tree[p]+=(vv+vv+d*(r-l))*(r-l+1)/2;
        }
        else
        {
            int mid=(l+r)>>1;
            pushdown(p,l,r);
            if(x<=mid)update(p<<1,l,mid,v,d,x,y);
            if(mid<y)update(p<<1|1,mid+1,r,v,d,x,y);
            tree[p]=tree[p<<1]+tree[p<<1|1];
        }
    }
    double query(int p,int l,int r,int x,int y)
    {
        if(x<=l&&r<=y)return tree[p];
        else
        {
            pushdown(p,l,r);
            int mid=(l+r)>>1;
            double ans=0;
            if(x<=mid)ans+=query(p<<1,l,mid,x,y);
            if(mid<y)ans+=query(p<<1|1,mid+1,r,x,y);
            tree[p]=tree[p<<1]+tree[p<<1|1];
            return ans;
        }
    }
    int h[100005];
}
using namespace Segtree;
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=0;i<n;i++)scanf("%d",&h[i]);
    build(1,1,1000);
    for(int i=0;i<n-1;i++)
    {
        int minn=min(h[i],h[i+1]);
        int maxx=max(h[i],h[i+1]);
        if(minn>=1)update(1,1,1000,-1,0,1,minn);
        if(maxx>minn)update(1,1,1000,-(1.0-1.0/(maxx-minn)/2),1.0/(maxx-minn),minn+1,maxx);
    }
    while(m--)
    {
        char op[10];
        scanf("%s",op+1);
        if(op[1]=='Q')
        {
            int H;
            scanf("%d",&H);
            if(H==0)printf("0.000\n");
            else printf("%.3f\n",query(1,1,1000,1,H));
        }
        else
        {
            int id,H;
            scanf("%d%d",&id,&H);
            if(id>0)
            {
                if(h[id]>h[id-1])
                {
                    update(1,1,1000,1.0-1.0/(h[id]-h[id-1])/2,-1.0/(h[id]-h[id-1]),h[id-1]+1,h[id]);
                }
                else if(h[id]<h[id-1])
                {
                    update(1,1,1000,-1.0/(h[id-1]-h[id])/2,-1.0/(h[id-1]-h[id]),h[id]+1,h[id-1]);
                }
```

```
            if(H>h[id-1])
            {
                update(1,1,1000,-(1.0-1.0/(H-h[id-1])/2),1.0/(H-h[id
                    -1]),h[id-1]+1,H);
            }
            else if(H<h[id-1])
            {
                update(1,1,1000,1.0/(h[id-1]-H)/2,1.0/(h[id-1]-H),H+1,h
                    [id-1]);
            }
        }
        if(id<n-1)
        {
            if(h[id]>h[id+1])
            {
                update(1,1,1000,1.0-1.0/(h[id]-h[id+1])/2,-1.0/(h[id]-h
                    [id+1]),h[id+1]+1,h[id]);
            }
            else if(h[id]<h[id+1])
            {
                update(1,1,1000,-1.0/(h[id+1]-h[id])/2,-1.0/(h[id+1]-h[
                    id]),h[id]+1,h[id+1]);
            }
            if(H>h[id+1])
            {
                update(1,1,1000,-(1.0-1.0/(H-h[id+1])/2),1.0/(H-h[id
                    +1]),h[id+1]+1,H);
            }
            else if(H<h[id+1])
            {
                update(1,1,1000,1.0/(h[id+1]-H)/2,1.0/(h[id+1]-H),H+1,h
                    [id+1]);
            }
        }
        h[id]=H;
    }
  }
  return 0;
}
```

2.3.2 有加有乘

```
#include<bits/stdc++.h>
using namespace std;
long long mod;
namespace Segtree
{
  int a[100005];
  long long tree[100005<<2],laz1[100005<<2],laz2[100005<<2];
  void build(int p,int l,int r)
  {
    laz1[p]=0,laz2[p]=1;
    if(l==r)tree[p]=a[l]%mod;
    else
    {
      int mid=(l+r)>>1;
      build(p<<1,l,mid);
      build(p<<1|1,mid+1,r);
      tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
    }
  }
  void pushdown(int p,int l,int r)
  {
    int mid=(l+r)>>1;
    long long x=laz1[p],y=laz2[p];
    tree[p<<1]=(tree[p<<1]*y%mod+x*(mid-l+1)%mod)%mod;
    tree[p<<1|1]=(tree[p<<1|1]*y%mod+x*(r-mid)%mod)%mod;
    laz1[p<<1]=(laz1[p<<1]*y%mod+x)%mod;
    laz1[p<<1|1]=(laz1[p<<1|1]*y%mod+x)%mod;
    laz2[p<<1]=laz2[p<<1]*y%mod;
    laz2[p<<1|1]=laz2[p<<1|1]*y%mod;
    laz1[p]=0,laz2[p]=1;
  }
  void update(int p,int l,int r,long long w,int x,int y,int op)
  {
    if(op==2)//加
    {
      if(x<=l&&r<=y)
      {
        tree[p]=(tree[p]+w*(r-l+1)%mod)%mod;
        laz1[p]=(laz1[p]+w)%mod;
      }
      else
      {
        pushdown(p,l,r);
        int mid=(l+r)>>1;
        if(x<=mid)update(p<<1,l,mid,w,x,y,op);
        if(mid<y)update(p<<1|1,mid+1,r,w,x,y,op);
        tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
      }
    }
    else//乘
    {
```

```
        if(x<=l&&r<=y)
        {
            tree[p]=tree[p]*w%mod;
            laz1[p]=laz1[p]*w%mod;
            laz2[p]=laz2[p]*w%mod;
        }
        else
        {
            pushdown(p,l,r);
            int mid=(l+r)>>1;
            if(x<=mid)update(p<<1,l,mid,w,x,y,op);
            if(mid<y)update(p<<1|1,mid+1,r,w,x,y,op);
            tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
        }
    }
}
long long query(int p,int l,int r,int x,int y)
{
    if(x<=l&&r<=y)return tree[p]%mod;
    else
    {
        pushdown(p,l,r);
        long long ans=0;
        int mid=(l+r)>>1;
        if(x<=mid)ans=(ans+query(p<<1,l,mid,x,y))%mod;
        if(mid<y)ans=(ans+query(p<<1|1,mid+1,r,x,y))%mod;
        tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
        return ans;
    }
}
}
using namespace Segtree;
int main()
{
    int n,m;
    scanf("%d%d%lld",&n,&m,&mod);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);
    build(1,1,n);
    while(m--)
    {
        int op;
        scanf("%d",&op);
        if(op==1||op==2)
        {
            int l,r;long long w;
            scanf("%d%d%lld",&l,&r,&w);
            update(1,1,n,w,l,r,op);
        }
        else
        {
            int l,r;
            scanf("%d%d",&l,&r);
            printf("%lld\n",query(1,1,n,l,r));
        }
    }
    return 0;
}
```

2.3.3 最大子段和

```
#include<bits/stdc++.h>
using namespace std;
namespace Segtree
{
    int n,m;
    int a[500005];
    struct node
    {
        int sum,max_l,max_r,maxx;
        node operator+(const node& y)const
        {
            return {sum+y.sum,
                max(max_l,sum+y.max_l),
                max(max_r+y.sum,y.max_r),
                max({maxx,y.maxx,max_r+y.max_l})};
        }
    }tree[500005<<2];
    void build(int p,int l,int r)
    {
        if(l==r)tree[p]={a[l],a[l],a[l],a[l]};
        else
        {
            int mid=(l+r)>>1;
            build(p<<1,l,mid);
            build(p<<1|1,mid+1,r);
            tree[p]=tree[p<<1]+tree[p<<1|1];
        }
    }
    void update(int p,int l,int r,int w,int x)
    {
        if(l==r)tree[p]={w,w,w,w};
        else
        {
```

```
        int mid=(l+r)>>1;
        if(x<=mid)update(p<<1,l,mid,w,x);
        else update(p<<1|1,mid+1,r,w,x);
        tree[p]=tree[p<<1]+tree[p<<1|1];
      }
   }
   node query(int p,int l,int r,int x,int y)
   {
      if(x<=l&&r<=y)return tree[p];
      else
      {
        node ans={0,(int)-1e9,(int)-1e9,(int)-1e9};
        int mid=(l+r)>>1;
        if(x<=mid)ans=ans+query(p<<1,l,mid,x,y);
        if(mid<y)ans=ans+query(p<<1|1,mid+1,r,x,y);
        return ans;
      }
   }
}
using namespace Segtree;
int main()
{
   scanf("%d%d",&n,&m);
   for(int i=1;i<=n;i++)
   scanf("%d",&a[i]);
   build(1,1,n);
   while(m--)
   {
      int op,l,r;
      scanf("%d%d%d",&op,&l,&r);
      if(op==1)
      {
        if(l>r)swap(l,r);
        printf("%d\n",query(1,1,n,l,r).maxx);
      }
      else update(1,1,n,r,l);
   }
   return 0;
}
```

2.4 非递归版本线段树

```
unsigned int bit_ceil(unsigned int n) {
   unsigned int x = 1;
   while (x < (unsigned int)(n)) x *= 2;
   return x;
}

int countr_zero(unsigned int n) {
   return __builtin_ctz(n);
}

template <class S,
        S (*op)(S, S),
        S (*e)(),
        class F,
        S (*mapping)(F, S),
        F (*composition)(F, F),
        F (*id)()>
struct lazy_segtree {
  public:
   lazy_segtree() : lazy_segtree(0) {}
   explicit lazy_segtree(int n) : lazy_segtree(std::vector<S>(n, e()))
        {}
   explicit lazy_segtree(const std::vector<S>& v) : _n(int(v.size()))
       {
      size = (int)bit_ceil((unsigned int)(_n));
      log = countr_zero((unsigned int)size);
      d = std::vector<S>(2 * size, e());
      lz = std::vector<F>(size, id());
      for (int i = 0; i < _n; i++) d[size + i] = v[i];
      for (int i = size - 1; i >= 1; i--) {
         update(i);
      }
   }

   void set(int p, S x) {
      assert(0 <= p && p < _n);
      p += size;
      for (int i = log; i >= 1; i--) push(p >> i);
      d[p] = x;
      for (int i = 1; i <= log; i++) update(p >> i);
   }

   S get(int p) {
      assert(0 <= p && p < _n);
      p += size;
      for (int i = log; i >= 1; i--) push(p >> i);
      return d[p];
   }

   S prod(int l, int r) {
      assert(0 <= l && l <= r && r <= _n);
```

```
    if (l == r) return e();

    l += size;
    r += size;

    for (int i = log; i >= 1; i--) {
      if (((l >> i) << i) != l) push(l >> i);
      if (((r >> i) << i) != r) push((r - 1) >> i);
    }

    S sml = e(), smr = e();
    while (l < r) {
      if (l & 1) sml = op(sml, d[l++]);
      if (r & 1) smr = op(d[--r], smr);
      l >>= 1;
      r >>= 1;
    }

    return op(sml, smr);
  }

  S all_prod() { return d[1]; }

  void apply(int p, F f) {
    assert(0 <= p && p < _n);
    p += size;
    for (int i = log; i >= 1; i--) push(p >> i);
    d[p] = mapping(f, d[p]);
    for (int i = 1; i <= log; i++) update(p >> i);
  }
  void apply(int l, int r, F f) {
    assert(0 <= l && l <= r && r <= _n);
    if (l == r) return;

    l += size;
    r += size;

    for (int i = log; i >= 1; i--) {
      if (((l >> i) << i) != l) push(l >> i);
      if (((r >> i) << i) != r) push((r - 1) >> i);
    }

    {
      int l2 = l, r2 = r;
      while (l < r) {
        if (l & 1) all_apply(l++, f);
        if (r & 1) all_apply(--r, f);
        l >>= 1;
        r >>= 1;
      }
      l = l2;
      r = r2;
    }

    for (int i = 1; i <= log; i++) {
      if (((l >> i) << i) != l) update(l >> i);
      if (((r >> i) << i) != r) update((r - 1) >> i);
    }
  }

  template <bool (*g)(S)> int max_right(int l) {
    return max_right(l, [](S x) { return g(x); });
  }
  template <class G> int max_right(int l, G g) {
    assert(0 <= l && l <= _n);
    assert(g(e()));
    if (l == _n) return _n;
    l += size;
    for (int i = log; i >= 1; i--) push(l >> i);
    S sm = e();
    do {
      while (l % 2 == 0) l >>= 1;
      if (!g(op(sm, d[l]))) {
        while (l < size) {
          push(l);
          l = (2 * l);
          if (g(op(sm, d[l]))) {
            sm = op(sm, d[l]);
            l++;
          }
        }
        return l - size;
      }
      sm = op(sm, d[l]);
      l++;
    } while ((l & -l) != l);
    return _n;
  }

  template <bool (*g)(S)> int min_left(int r) {
    return min_left(r, [](S x) { return g(x); });
  }
  template <class G> int min_left(int r, G g) {
    assert(0 <= r && r <= _n);
```

```
      assert(g(e()));
      if (r == 0) return 0;
      r += size;
      for (int i = log; i >= 1; i--) push((r - 1) >> i);
      S sm = e();
      do {
         r--;
         while (r > 1 && (r % 2)) r >>= 1;
         if (!g(op(d[r], sm))) {
            while (r < size) {
               push(r);
               r = (2 * r + 1);
               if (g(op(d[r], sm))) {
                  sm = op(d[r], sm);
                  r--;
               }
            }
            return r + 1 - size;
         }
         sm = op(d[r], sm);
      } while ((r & -r) != r);
      return 0;
   }

 private:
   int _n, size, log;
   std::vector<S> d;
   std::vector<F> lz;

   void update(int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
   void all_apply(int k, F f) {
      d[k] = mapping(f, d[k]);
      if (k < size) lz[k] = composition(f, lz[k]);
   }
   void push(int k) {
      all_apply(2 * k, lz[k]);
      all_apply(2 * k + 1, lz[k]);
      lz[k] = id();
   }
};
```

2.4.1 用法 1:

```
// operation 1: a[i] -> b*a[i]+c
// operation 2: \sum_{l<=i<r} a[i]

struct S
{
   mint a;
   int siz;
};

struct F
{
   mint a,b;
};

S op(S l,S r){return S{l.a+r.a,l.siz+r.siz};}

S e(){return S{0,0};}

S mapping(F l,S r){return S{r.a*l.a+r.siz*l.b,r.siz};}

F composition(F l,F r){return F{r.a*l.a,r.b*l.a+l.b};}

F id(){return F{1,0};}

void solve()
{
   int n,q;
   cin>>n>>q;
   vector<S>a(n);
   for(int i=0;i<n;i++)
   {
      mint x;
      cin>>x;
      a[i]=S{x,1};
   }

   lazy_segtree<S,op,e,F,mapping,composition,id>seg(a);

   for(int i=0;i<q;i++)
   {
      int t,l,r;
      cin>>t>>l>>r;
      if(t==0)
      {
         mint b,c;
         cin>>b>>c;
         seg.apply(l,r,F{b,c});
      }
      else
      {
```

```
        cout<<seg.prod(l,r).a<<endl;
    }
  }
}
```

2.4.2 用法 2:

```
// a[1],a[2],...,a[n] in [0,1]
// operation 1: forall i in [l,r], 0 -> 1, 1 -> 0
// operation 2: calc the inversion of [l,r]

struct S
{
  ll zero,one,inversion;
};

struct F
{
  bool f;
};

S op(S l,S r)
{
  return S{l.zero+r.zero,l.one+r.one,l.inversion+r.inversion+l.one*r.zero};
}

S e(){return S{0,0,0};}

S mapping(F l,S r)
{
  if(!l.f)return r;
  return S{r.one,r.zero,r.zero*r.one-r.inversion};
}

F composition(F l,F r){return F{(!l.f&&r.f)||(l.f&&!r.f)};}

F id(){return F{false};}

void solve()
{
  int n,q;
  cin>>n>>q;
  vector<S>a(n);
  for(int i=0;i<n;i++)
  {
    int x;
    cin>>x;
    a[i]=S{x==0,x==1,0};
  }

  lazy_segtree<S,op,e,F,mapping,composition,id>seg(a);

  for(int i=0;i<q;i++)
  {
    int t,l,r;
    cin>>t>>l>>r;
    if(t==1)
    {
      seg.apply(l-1,r,F{1});
    }
    else
    {
      cout<<seg.prod(l-1,r).inversion<<endl;
    }
  }
}
```

2.5 动态开点线段树

```
int root=1;
template<typename T,int Sz,int Rng>
struct DynamicSegTree
{
  int cnt=1;
  T val[Sz];
  int ls[Sz];
  int rs[Sz];
  void clear(){cnt=1;}
  int newnode()
  {
    cnt++;
    ls[cnt]=rs[cnt]=0;
    val[cnt]=0;
    return cnt;
  }
  void update(T w,int x,int y,int&p=root,int l=1,int r=Rng)
  {
    if(!p)p=newnode();
    if(x<=l&&r<=y)
    {
      val[p]+=w;
```

```
        }
        else
        {
            int mid=(l+r)>>1;
            if(x<=mid)update(w,x,y,ls[p],l,mid);
            if(mid<y)update(w,x,y,rs[p],mid+1,r);
        }
    }
    T query(int x,int p=root,int l=1,int r=Rng,T ans=0)
    {
        if(!p)return ans;
        ans+=val[p];
        if(l==r)return ans;
        else
        {
            int mid=(l+r)>>1;
            if(x<=mid)return query(x,ls[p],l,mid,ans);
            else return query(x,rs[p],mid+1,r,ans);
        }
    }
};

DynamicSegTree<int,40000005,1000000000>seg;
```

2.6 Segment Tree Beats

2.6.1 区间取 min 、max $O(nlogn)$

```
/*
1s
长度 n,询问 m 都为 1e6
两种操作:
1 x y——将第 x 个数变为 y;
2 y——将所有小于 y 的数修改为 y;
*/
#include<bits/stdc++.h>
using namespace std;

inline int qread(){
    int s=0,w=1;char ch=getchar();
    for(;!isdigit(ch);ch=getchar())if(ch=='-')w=-1;
    for (;ch>='0'&&ch<='9';ch=getchar())s=(s<<1)+(s<<3)+(ch^48);
    return (w==-1?-s:s);}

int n,m;
int a[200050];
struct node{
    int mi,smi,cnt,tag;
}tr[800050];
void pushup(int p){
    if(tr[2*p].mi<tr[2*p+1].mi){
        tr[p].mi=tr[2*p].mi;tr[p].cnt=tr[2*p].cnt;
        tr[p].smi=min(tr[2*p+1].mi,tr[2*p].smi);
    }else if(tr[2*p].mi>tr[2*p+1].mi){
        tr[p].mi=tr[2*p+1].mi;tr[p].cnt=tr[2*p+1].cnt;
        tr[p].smi=min(tr[2*p].mi,tr[2*p+1].smi);
    }else {
        tr[p].mi=tr[2*p].mi;tr[p].cnt=tr[2*p].cnt+tr[2*p+1].cnt;
        tr[p].smi=min(tr[2*p].smi,tr[2*p+1].smi);
    }
}
void build(int p,int l,int r){
    tr[p].tag=-1;
    if(l==r){
        tr[p].mi=a[l];
        tr[p].smi=1e9+7;
        tr[p].cnt=1;
        return;
    }
    int mid=l+r>>1;
    build(2*p,l,mid);
    build(2*p+1,mid+1,r);
    pushup(p);
}
void pushdown(int p){
    if(tr[p].tag==-1)return ;
    if(tr[p].tag>tr[2*p].mi){
        tr[2*p].mi=tr[2*p].tag=tr[p].tag;
    }
    if(tr[p].tag>tr[2*p+1].mi){
        tr[2*p+1].mi=tr[2*p+1].tag=tr[p].tag;
    }
    tr[p].tag=-1;
}
void update(int p,int l,int r,int w){
    if(tr[p].mi>=w)return;
    if(tr[p].smi>w){
        tr[p].mi=tr[p].tag=w;
        return;
    }
    pushdown(p);
    int mid=l+r>>1;
    update(2*p,l,mid,w);update(2*p+1,mid+1,r,w);
```

```
    pushup(p);
}
void update1(int p,int l,int r,int x,int w){
    if(l==r){
        tr[p].mi=w;
        return;
    }
    int mid=l+r>>1;pushdown(p);
    if(x<=mid)update1(2*p,l,mid,x,w);
    else update1(2*p+1,mid+1,r,x,w);
    pushup(p);
}
int query(int p,int l,int r,int x){
    if(l==r)return tr[p].mi;
    int mid=l+r>>1;
    pushdown(p);
    if(x<=mid)return query(2*p,l,mid,x);
    else return query(2*p+1,mid+1,r,x);
}
int main()
{
    n=qread();
    for(int i=1;i<=n;i++){
        a[i]=qread();
    }
    build(1,1,n);
    m=qread();
    for(int i=1;i<=m;i++){
        int op;
        op=qread();
        if(op==1){
            int a,b;
            a=qread();b=qread();
            update1(1,1,n,a,b);
        }else {
            int a;a=qread();
            update(1,1,n,a);
        }
    }
    for(int i=1;i<=n;i++){
        printf("%d ",query(1,1,n,i));
    }
    return 0;
}
```

2.6.2 区间取 min 、max + 区间加减 $O(nlog^2n)$

```
/*
1s
数组长度n，操作数m都为5e5;
五种操作:
1.给一个区间 [L,R] 加上一个数 x
2.把一个区间 [L,R] 里小于 x 的数变成 x
3.把一个区间 [L,R] 里大于 x 的数变成 x
4.求区间 [L,R] 的和
5.求区间 [L,R] 的最大值
6.求区间 [L,R] 的最小值

分别维护最小值、最大值、其他值的加标记;
区间加操作 就是把三个标记都加; 区间取 min 操作就是在最大值的加标记上修改; 取 max同理。

两个注意点:
以最大值上的加减标记为例。下传这个标记时要判断子区间内是否包含最大值，如果不包含则应下传其他值的加减标记;
如果一个区间的值域很小（只有 1 或 2 个数），可能会发生一个值既是最大值又是次小值这种情况，也就是发生了数域的重叠。这种情况要特判，分辨到底该被哪个标记作用。
*/
#include<bits/stdc++.h>
using namespace std;
#define ll long long
inline int qread(){
    int s=0,w=1;char ch=getchar();
    for(;!isdigit(ch);ch=getchar())if(ch=='-')w=-1;
    for (;ch>='0'&&ch<='9';ch=getchar())s=(s<<1)+(s<<3)+(ch^48);
    return (w==-1?-s:s);}

const int INF=2e9;
int n,m;
int s[500050];
struct node{
    //最大值、次大值、最大值个数、最小值、次小值、最小值个数
    int mx,smx,cmx,mi,smi,cmi;
    ll sum;
    //最小值、最大值、其他值的修改标记
    int add1,add2,add3;
}tr[2000050];
#define ls (p<<1)
#define rs (p<<1|1)
void pushup(int p){
    tr[p].sum=tr[ls].sum+tr[rs].sum;
    if(tr[ls].mi==tr[rs].mi){
```

```
        tr[p].mi=tr[ls].mi;
        tr[p].smi=min(tr[ls].smi,tr[rs].smi);
        tr[p].cmi=tr[ls].cmi+tr[rs].cmi;
    }else if(tr[ls].mi<tr[rs].mi){
        tr[p].mi=tr[ls].mi;
        tr[p].smi=min(tr[ls].smi,tr[rs].mi);
        tr[p].cmi=tr[ls].cmi;
    }else {
        tr[p].mi=tr[rs].mi;
        tr[p].smi=min(tr[ls].mi,tr[rs].smi);
        tr[p].cmi=tr[rs].cmi;
    }
    if(tr[ls].mx==tr[rs].mx){
        tr[p].mx=tr[ls].mx;
        tr[p].smx=max(tr[ls].smx,tr[rs].smx);
        tr[p].cmx=tr[ls].cmx+tr[rs].cmx;
    }else if(tr[ls].mx>tr[rs].mx){
        tr[p].mx=tr[ls].mx;
        tr[p].smx=max(tr[ls].smx,tr[rs].mx);
        tr[p].cmx=tr[ls].cmx;
    }else {
        tr[p].mx=tr[rs].mx;
        tr[p].smx=max(tr[ls].mx,tr[rs].smx);
        tr[p].cmx=tr[rs].cmx;
    }
}
//对tr[p]的修改，k1、k2、k3 分别为最小值、最大值、其他值的标记
void update(int p,int l,int r,int k1,int k2,int k3){
    if(tr[p].mi==tr[p].mx){
        //如果区间只有一种值，应该只作用最大值、最小值的加减标记
        if(k1==k3)k1=k2;
        else k2=k1;
        tr[p].sum+=1LL*k1*tr[p].cmi;
    }else{
        tr[p].sum+=1LL*k1*tr[p].cmi+1LL*k2*tr[p].cmx+1LL*k3*(r-l+1-tr[p
            ].cmi-tr[p].cmx);
    }
    //次小值为最大值，应被最大值标记作用
    if(tr[p].smi==tr[p].mx)tr[p].smi+=k2;
    else if(tr[p].smi!=INF)tr[p].smi+=k3;//否则被其他值的标记作用

    if(tr[p].smx==tr[p].mi)tr[p].smx+=k1;
    else if(tr[p].smx!=INF)tr[p].smx+=k3;

    tr[p].mi+=k1;tr[p].mx+=k2;
    tr[p].add1+=k1;tr[p].add2+=k2;tr[p].add3+=k3;
}
```

```
void pushdown(int p,int l,int r){
    int mi=min(tr[ls].mi,tr[rs].mi);
    int mx=max(tr[ls].mx,tr[rs].mx);
    int mid=l+r>>1;
    update(ls,l,mid,tr[ls].mi==mi?tr[p].add1:tr[p].add3,
    tr[ls].mx==mx?tr[p].add2:tr[p].add3,tr[p].add3);
    update(rs,mid+1,r,tr[rs].mi==mi?tr[p].add1:tr[p].add3,
    tr[rs].mx==mx?tr[p].add2:tr[p].add3,tr[p].add3);

    tr[p].add1=tr[p].add2=tr[p].add3=0;
}
void build(int p,int l,int r){
    tr[p].add1=tr[p].add2=tr[p].add3=0;
    if(l==r){
        tr[p].sum=tr[p].mi=tr[p].mx=s[l];
        tr[p].smi=INF;tr[p].smx=-INF;
        tr[p].cmi=tr[p].cmx=1;
        return;
    }
    int mid=l+r>>1;
    build(ls,l,mid);
    build(rs,mid+1,r);
    pushup(p);
}
void update1(int p,int l,int r,int x,int y,int w){
    if(x<=l&&r<=y){
        update(p,l,r,w,w,w);
        return;
    }
    int mid=l+r>>1;
    pushdown(p,l,r);
    if(x<=mid)update1(2*p,l,mid,x,y,w);
    if(mid<y)update1(2*p+1,mid+1,r,x,y,w);
    pushup(p);
}
void update2(int p,int l,int r,int x,int y,int w){
    if(tr[p].mi>=w)return;
    if(x<=l&&r<=y&&tr[p].smi>w){
        update(p,l,r,w-tr[p].mi,0,0);
        return;
    }
    pushdown(p,l,r);
    int mid=l+r>>1;
    if(x<=mid)update2(ls,l,mid,x,y,w);
    if(mid<y)update2(rs,mid+1,r,x,y,w);
    pushup(p);
}
```

```
void update3(int p,int l,int r,int x,int y,int w){
    if(tr[p].mx<=w)return;
    if(x<=l&&r<=y&&tr[p].smx<w){
        update(p,l,r,0,w-tr[p].mx,0);
        return;
    }
    int mid=l+r>>1;
    pushdown(p,l,r);
    if(x<=mid)update3(ls,l,mid,x,y,w);
    if(mid<y)update3(rs,mid+1,r,x,y,w);
    pushup(p);
}
ll query1(int p,int l,int r,int x,int y){
    if(x<=l&&r<=y)return tr[p].sum;
    int mid=l+r>>1;
    pushdown(p,l,r);
    ll ans=0;
    if(x<=mid)ans+=query1(ls,l,mid,x,y);
    if(mid<y)ans+=query1(rs,mid+1,r,x,y);
    return ans;
}
int query2(int p,int l,int r,int x,int y){
    if(x<=l&&r<=y)return tr[p].mx;
    int mid=l+r>>1,ans=-INF;
    pushdown(p,l,r);
    if(x<=mid)ans=max(ans,query2(ls,l,mid,x,y));
    if(mid<y)ans=max(ans,query2(rs,mid+1,r,x,y));
    return ans;
}
int query3(int p,int l,int r,int x,int y){
    if(x<=l&&r<=y)return tr[p].mi;
    int mid=l+r>>1,ans=INF;
    pushdown(p,l,r);
    if(x<=mid)ans=min(ans,query3(ls,l,mid,x,y));
    if(mid<y)ans=min(ans,query3(rs,mid+1,r,x,y));
    return ans;
}
#undef ls
#undef rs
int main(){
    n=qread();
    for(int i=1;i<=n;i++)s[i]=qread();
    build(1,1,n);
    m=qread();
    while(m--){
        int op=qread(),l=qread(),r=qread();
        int x;
        if(op<=3)x=qread();
        if(op==1)update1(1,1,n,l,r,x);
        else if(op==2)update2(1,1,n,l,r,x);
        else if(op==3)update3(1,1,n,l,r,x);
        else if(op==4)printf("%lld\n",query1(1,1,n,l,r));
        else if(op==5)printf("%d\n",query2(1,1,n,l,r));
        else printf("%d\n",query3(1,1,n,l,r));
    }
    return 0;
}
```

2.7 可持久化线段树

2.7.1 静态版本，权值线段树

```
ll cnt=0,n,m,len; //cnt记录当前节点数目，len代表离散化后的区间大小;
ll a[200060],b[200050],root[200050];
struct node{
    ll l,r,num;
}z[maxn];
ll clone(ll x){
    cnt++;
    z[cnt]=z[x];
    return cnt;
}
void update(ll id1,ll &id2,ll l,ll r,ll x){
    id2=clone(id1);z[id2].num++;
    if(l>=r)return;
    ll mid=(l+r)>>1;
    if(x<=mid)update(z[id1].l,z[id2].l,l,mid,x);
    else update(z[id1].r,z[id2].r,mid+1,r,x);
}
ll query(ll id1,ll id2,ll l,ll r,ll k){
    if(l>=r)return l;
    ll mid=(l+r)>>1;
    ll num=z[z[id2].l].num-z[z[id1].l].num;
    if(k<=num)return query(z[id1].l,z[id2].l,l,mid,k);
    //这里的判断条件小心，容易出错!
    //k<=num才往左子树走
    else return query(z[id1].r,z[id2].r,mid+1,r,k-num);
}
int main(){
    n=qread(),m=qread();
    for(int i=1;i<=n;i++){
        a[i]=qread();b[i]=a[i];
    }
```

```
    sort(b+1,b+1+n);
    len=unique(b+1,b+1+n)-b-1;
    for(int i=1;i<=n;i++){
        ll x=lower_bound(b+1,b+1+len,a[i])-b;
        update(root[i-1],root[i],1,len,x);
    }
    for(int i=1;i<=m;i++){
        ll l=qread(),r=qread(),k=qread();
        ll w=query(root[l-1],root[r],1,len,k);
        printf("%lld\n",b[w]);
    }
    return 0;
}
```

2.7.2 动态版本，支持修改

```
ll n,m,cnt=0;
ll a[1000060],root[maxn];
struct node{
    ll l,r,num;
}z[maxn];
ll clone(ll x){
    cnt++;z[cnt]=z[x];
    return cnt;
}
ll build(ll id,ll l,ll r){
    id=++cnt;
    if(l==r){
        z[cnt].num=a[l];return cnt;
    }
    ll mid=(l+r)>>1;
    z[id].l=build(z[id].l,l,mid);
    z[id].r=build(z[id].r,mid+1,r);
    return id;
}
void update(ll id1,ll &id2,ll l,ll r,ll x,ll w){
    id2=clone(id1);
    if(l>=r){
        z[id2].num=w; return;
    }
    ll mid=(l+r)>>1;
    if(x<=mid)update(z[id1].l,z[id2].l,l,mid,x,w);
    else update(z[id1].r,z[id2].r,mid+1,r,x,w);
}
ll query(ll id,ll l,ll r,ll x){
    if(l>=r)return z[id].num; //这里要return z[id].num;
    ll mid=(l+r)>>1;
    if(x<=mid)return query(z[id].l,l,mid,x);
    else return query(z[id].r,mid+1,r,x);
}
int main(){
    n=qread(),m=qread();
    for(int i=1;i<=n;i++){
        a[i]=qread();
    }
    root[0]=build(root[0],1,n);
    for(int i=1;i<=m;i++){
        ll q=qread(),p=qread(),w=qread();
        if(p==1){
            ll k=qread();
            update(root[q],root[i],1,n,w,k);
        }else{
            root[i]=clone(root[q]);
            printf("%lld\n",query(root[q],1,n,w));
        }
    }
    return 0;
}
```

2.8 线段树合并 & 分裂

```
// 给出一个可重集 a（编号为 1），它支持以下操作：
// 0 p x y：将可重集 p 中大于等于 x 且小于等于 y 的值移动到一个新的可重集中（新可重集编号为从 2 开始的正整数，是上一次产生的新可重集的编号+1）。
// 1 p t：将可重集 t 中的数放入可重集 p，且清空可重集 t（数据保证在此后的操作中不会出现可重集 t）。
// 2 p x q：在 p 这个可重集中加入 x 个数字 q。
// 3 p x y：查询可重集 p 中大于等于 x 且小于等于 y 的值的个数。
// 4 p k：查询在 p 这个可重集中第 k 小的数，不存在时输出 -1。

namespace Segtree
{
    int n,m;
    int ls[(n+1)*40],rs[(n+1)*40],root[n+1];
    ll tree[(n+1)*40];
    int cnt=0,top=1;
    void update(int& p,int l,int r,ll w,int x)
    {
        if(!p)p=++cnt;
        if(l==r)tree[p]+=w;
        else
        {
```

```
        int mid=(l+r)>>1;
        if(x<=mid)update(ls[p],l,mid,w,x);
        else update(rs[p],mid+1,r,w,x);
        tree[p]=tree[ls[p]]+tree[rs[p]];
    }
}
int merge(int a,int b,int l,int r)
{
    if(!a||!b)return a+b;
    else if(l==r)
    {
        tree[a]+=tree[b];
        return a;
    }
    else
    {
        int mid=(l+r)>>1;
        ls[a]=merge(ls[a],ls[b],l,mid);
        rs[a]=merge(rs[a],rs[b],mid+1,r);
        tree[a]=tree[ls[a]]+tree[rs[a]];
        return a;
    }
}
void split(int& p,int& q,int l,int r,int x,int y)
{
    if(x<=l&&r<=y)
    {
        p=q;
        q=0;
    }
    else
    {
        if(!p)p=++cnt;
        int mid=(l+r)>>1;
        if(x<=mid)split(ls[p],ls[q],l,mid,x,y);
        if(mid<y)split(rs[p],rs[q],mid+1,r,x,y);
        tree[p]=tree[ls[p]]+tree[rs[p]];
        tree[q]=tree[ls[q]]+tree[rs[q]];
    }
}
ll query1(int p,int l,int r,int x,int y)
{
    if(x<=l&&r<=y)return tree[p];
    else
    {
        int mid=(l+r)>>1;
        ll ans=0;
        if(x<=mid)ans+=query1(ls[p],l,mid,x,y);
        if(mid<y)ans+=query1(rs[p],mid+1,r,x,y);
        return ans;
    }
}
int query2(int p,int l,int r,ll x)
{
    if(l==r)return l;
    int mid=(l+r)>>1;
    if(tree[ls[p]]>=x)return query2(ls[p],l,mid,x);
    else if(tree[rs[p]]>=x-tree[ls[p]])return query2(rs[p],mid+1,r,x
        -tree[ls[p]]);
    else return -1;
}
}
using namespace Segtree;
void solve()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        int x;
        scanf("%d",&x);
        update(root[top],1,n,x,i);
    }
    for(int i=1;i<=m;i++)
    {
        int op;
        scanf("%d",&op);
        if(op==0)
        {
            int p,x,y;
            scanf("%d%d%d",&p,&x,&y);
            split(root[++top],root[p],1,n,x,y);
        }
        else if(op==1)
        {
            int p,t;
            scanf("%d%d",&p,&t);
            root[p]=merge(root[p],root[t],1,n);
            root[t]=0;
        }
        else if(op==2)
        {
            int p,x,q;
            scanf("%d%d%d",&p,&x,&q);
            update(root[p],1,n,x,q);
```

```
        }
        else if(op==3)
        {
            int p,x,y;
            scanf("%d%d%d",&p,&x,&y);
            printf("%lld\n",query1(root[p],1,n,x,y));
        }
        else if(op==4)
        {
            int p,k;
            scanf("%d%d",&p,&k);
            printf("%d\n",query2(root[p],1,n,k));
        }
        //  printf("query%d:\n",i);
        //  for(int j=1;j<=top;j++)
        //  {
            //   printf("%d:",j);
            //   for(int k=1;k<=n;k++)printf("%d%c",query1(root[j],1,n,k,
                k),(k==n)?'\n':' ');
            //  }
    }
}
```

2.9 分块

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1000005;
const int sqN=1005;

int n,sq;
int st[sqN];//第i号块的第一个元素的下标
int ed[sqN];//第i号块最后一个元素的下标
int id[N];//元素i的分块编号

ll arr[N];//零散数组
ll sum[N];//分块和数组
ll laz[N];//分块公共加数组

void init()
{
    sq=(int)sqrt(n);
    for(int i=1;i<=sq;i++)
    {
        st[i]=n/sq*(i-1)+1;
        ed[i]=n/sq*i;
    }
    ed[sq]=n;

    for(int i=1;i<=sq;i++)
    {
        for(int j=st[i];j<=ed[i];j++)
        {
            id[j]=i;
            sum[i]+=arr[j];
        }
    }
}

void update(int l,int r,ll x)
{
    if(id[l]==id[r])
    {
        for(int i=l;i<=r;i++)arr[i]+=x;
        sum[id[l]]+=x*(r-l+1);
    }
    else
    {
        for(int i=l;i<=ed[id[l]];i++)arr[i]+=x;
        sum[id[l]]+=x*(ed[id[l]]-l+1);
        for(int i=st[id[r]];i<=r;i++)arr[i]+=x;
        sum[id[r]]+=x*(r-st[id[r]]+1);
        for(int i=id[l]+1;i<id[r];i++)laz[i]+=x;
    }
}

ll query(int l,int r)
{
    ll ans=0;
    if(id[l]==id[r])
    {
        for(int i=l;i<=r;i++)ans+=arr[i];
        ans+=laz[id[l]]*(r-l+1);
    }
    else
    {
        for(int i=l;i<=ed[id[l]];i++)ans+=arr[i];
        ans+=laz[id[l]]*(ed[id[l]]-l+1);
        for(int i=st[id[r]];i<=r;i++)ans+=arr[i];
        ans+=laz[id[r]]*(r-st[id[r]]+1);
        for(int i=id[l]+1;i<id[r];i++)
        ans+=sum[i]+laz[i]*(ed[i]-st[i]+1);
```

```
    }
    return ans;
}

int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%d",&arr[i]);
    init();
    for(int i=1;i<=n;i++)
    {
        int op;
        scanf("%d",&op);
        if(op==0)
        {
            int l,r,c;
            scanf("%d%d%d",&l,&r,&c);
            update(l,r,c);
        }
        else
        {
            int l,r,c;
            scanf("%d%d%d",&l,&r,&c);
            printf("%lld\n",query(l,r)%(c+1));
        }
    }
    return 0;
}
```

2.10 珂朵莉树

```
/*
例题:
操作1: 区间加x
操作2: 区间赋值x
操作3: 求区间第K大
操作4: 求区间的x幂和 模y;
*/
#include<bits/stdc++.h>
using namespace std;
#define ll long long

struct node{
    int l,r;
    mutable ll val;
    bool operator < (const node &ths)const{
        return l<ths.l;
    }
    node(int L,int R,ll Val):l(L),r(R),val(Val){}
    node(int L):l(L){}
};

set<node>s;
using si=set<node>::iterator;

si split(int pos){
    si it=s.lower_bound(node(pos));
    if(it!=s.end()&&it->l==pos)return it;
    --it;
    int l=it->l,r=it->r;
    ll val=it->val;
    s.erase(it);
    s.insert(node(l,pos-1,val));
    return s.insert(node(pos,r,val)).first;
}
void assign(int l,int r,int val){
    si itr=split(r+1),itl=split(l);
    s.erase(itl,itr);
    s.insert(node(l,r,val));
}
void add(int l,int r,ll val){
    si itr=split(r+1),itl=split(l);
    for(si it=itl;it!=itr;++it)
    it->val+=val;
}
ll kth(int l,int r,int k){
    si itr=split(r+1),itl=split(l);
    vector<pair<ll,int>>v;
    v.clear();
    for(si it=itl;it!=itr;++it)
    v.push_back(pair<ll,int>(it->val,it->r-it->l+1));
    sort(v.begin(),v.end());
    for(int i=0;i<v.size();i++){
        k-=v[i].second;
        if(k<=0)return v[i].first;
    }
    return -1;
}
ll query(int l,int r,int x,int y){
    si itr=split(r+1),itl=split(l);
    ll res=0;
    for(si it=itl;it!=itr;++it)
    res=(res+(it->r-it->l+1)*qp(it->val,x,y)%y)%y;
```

```
    return res;
}
int n,m,vmax;
ll seed;
int rnd(){
    int ret=(int)seed;
    seed=(seed*7+13)%1000000007;
    return ret;
}
int main(){
    scanf("%d%d%lld%d",&n,&m,&seed,&vmax);
    for(int i=1;i<=n;i++){
        int a=rnd()%vmax+1;
        s.insert(node(i,i,a));
    }
    s.insert(node(n+1,n+1,0));
    for(int i=1;i<=m;i++){
        int l,r,x,y;
        int op=rnd()%4+1;
        l=rnd()%n+1,r=rnd()%n+1;
        if(l>r)swap(l,r);
        if(op==3)x=rnd()%(r-l+1)+1;
        else x=rnd()%vmax+1;
        if(op==4)y=rnd()%vmax+1;
        if(op==1)add(l,r,x);
        else if(op==2)assign(l,r,x);
        else if(op==3)printf("%lld\n",kth(l,r,x));
        else printf("%lld\n",query(l,r,x,y));
    }
}
```

2.11 莫队

2.11.1 普通莫队

```
struct Query
{
    int l,r,id,pos;
    bool operator<(const Query &x)const
    {
        return (pos^x.pos)?pos<x.pos:(pos&1)?r<x.r:r>x.r;
    }
}a[100005];
int b[100005],n,m;
ll cnt[100005],Ans[100005];

void solve()
{
    read(n,m);
    int siz=n/sqrt(m);
    if(!siz)siz++;
    for(int i=1;i<=n;i++)read(b[i]);
    for(int i=1;i<=m;i++)
    {
        read(a[i].l,a[i].r);
        a[i].pos=(a[i].l-1)/siz+1;
        a[i].id=i;
    }
    sort(a+1,a+m+1);
    int l=1,r=0;
    int ans=0;
    for(int i=1;i<=m;i++)
    {
        while(l>a[i].l)ans+=(++cnt[b[--l]]==2);
        while(r<a[i].r)ans+=(++cnt[b[++r]]==2);
        while(l<a[i].l)ans-=(--cnt[b[l++]]==1);
        while(r>a[i].r)ans-=(--cnt[b[r--]]==1);
        Ans[a[i].id]=ans;
    }
    for(int i=1;i<=m;i++)
    {
        if(Ans[i])println("No");
        else println("Yes");
    }
}
```

2.11.2 带修莫队

```
struct Query
{
    int l,r,t,id,posl,posr;
    bool operator<(const Query &x)const
    {
        return (posl^x.posl)?posl<x.posl:((posr^x.posr)?posr<x.posr:t<x.
            t);
    }
}a[200005];
struct Replace
{
    int p,col;
}c[200005];
int b[200005],n,m;
```

```
int cnta,cntc;
ll cnt[1000005],Ans[200005];

void solve()
{
   read(n,m);
   int siz=pow(n,2.0/3.0);
   for(int i=1;i<=n;i++)read(b[i]);
   for(int i=1;i<=m;i++)
   {
      string s;int x,y;
      read(s,x,y);
      if(s=="Q")
      {
         a[++cnta].l=x;a[cnta].r=y;
         a[cnta].t=cntc;a[cnta].id=cnta;
         a[cnta].posl=(a[cnta].l-1)/siz+1;
         a[cnta].posr=(a[cnta].r-1)/siz+1;
      }
      else{c[++cntc].p=x;c[cntc].col=y;}
   }
   sort(a+1,a+cnta+1);
   int l=1,r=0,t=0;ll ans=0;
   for(int i=1;i<=m;i++)
   {
      while(l<a[i].l)ans-=!--cnt[b[l++]];
      while(l>a[i].l)ans+=!cnt[b[--l]]++;
      while(r<a[i].r)ans+=!cnt[b[++r]]++;
      while(r>a[i].r)ans-=!--cnt[b[r--]];
      while(t<a[i].t)
      {
         t++;
         if(a[i].l<=c[t].p&&c[t].p<=a[i].r)
         ans-=!--cnt[b[c[t].p]]-!cnt[c[t].col]++;
         swap(b[c[t].p],c[t].col);
      }
      while(t>a[i].t)
      {
         if(a[i].l<=c[t].p&&c[t].p<=a[i].r)
         ans-=!--cnt[b[c[t].p]]-!cnt[c[t].col]++;
         swap(b[c[t].p],c[t].col);
         t--;
      }
      Ans[a[i].id]=ans;
   }
   for(int i=1;i<=cnta;i++)println(Ans[i]);
}
```

2.11.3 树上莫队

```
struct Query
{
   int s,lca,l,r,id,pos;
   bool operator<(const Query &x)const
   {
      return (pos^x.pos)?pos<x.pos:(pos&1)?r<x.r:r>x.r;
   }
}a[100005];
int b[40005],d[40005],cntd,n,m;
int cnt[40005],vis[40005],Ans[100005];
vector<int>edge[40005];
int fa[40005][20],dep[40005];
int c[80005],tot,fir[40005],las[40005];
void dfs(int x,int father)
{
   fa[x][0]=father;dep[x]=dep[father]+1;
   for(int i=1;i<20;i++)fa[x][i]=fa[fa[x][i-1]][i-1];
   c[++tot]=x;fir[x]=tot;
   for(auto it:edge[x])if(it!=father)dfs(it,x);
   c[++tot]=x;las[x]=tot;
}
int cal_lca(int x,int y)
{
   if(dep[x]<dep[y])swap(x,y);
   for(int i=19;i>=0;i--)if(dep[fa[x][i]]>=dep[y])x=fa[x][i];
   if(x==y)return x;
   for(int i=19;i>=0;i--)if(fa[x][i]!=fa[y][i])x=fa[x][i],y=fa[y][i];
   return fa[x][0];
}

void solve()
{
   read(n,m);
   int siz=(int)sqrt(2*n);
   for(int i=1;i<=n;i++)
   {
      read(b[i]);
      d[++cntd]=b[i];
   }
   sort(d+1,d+cntd+1);
   cntd=unique(d+1,d+cntd+1)-d-1;
   for(int i=1;i<=n;i++)
   b[i]=lower_bound(d+1,d+cntd+1,b[i])-d;
   for(int i=1;i<n;i++)
   {
```

```
        int x,y;read(x,y);
        edge[x].push_back(y);
        edge[y].push_back(x);
    }
    dfs(1,1);
    for(int i=1;i<=m;i++)
    {
        int x,y;read(x,y);
        if(fir[x]>fir[y])swap(x,y);a[i].s=x;
        ((a[i].lca=cal_lca(x,y))==x)?a[i].l=fir[x]:a[i].l=las[x];
        a[i].r=fir[y];a[i].pos=(a[i].l-1)/siz+1;a[i].id=i;
    }
    sort(a+1,a+m+1);
    int l=1,r=0,ans=0;
    for(int i=1;i<=m;i++)
    {
        while(l>a[i].l)(vis[c[--l]]?ans-=!--cnt[b[c[l]]]:
        ans+=!cnt[b[c[l]]]++),vis[c[l]]^=1;
        while(l<a[i].l)(vis[c[l]]?ans-=!--cnt[b[c[l]]]:
        ans+=!cnt[b[c[l]]]++),vis[c[l++]]^=1;
        while(r<a[i].r)(vis[c[++r]]?ans-=!--cnt[b[c[r]]]:
        ans+=!cnt[b[c[r]]]++),vis[c[r]]^=1;
        while(r>a[i].r)(vis[c[r]]?ans-=!--cnt[b[c[r]]]:
        ans+=!cnt[b[c[r]]]++),vis[c[r--]]^=1;
        Ans[a[i].id]=((a[i].lca==a[i].s)?ans:ans+
        (vis[a[i].lca]?((cnt[b[a[i].lca]]==1)?-1:0):((cnt[b[a[i].lca
            ]]==0)?1:0)));
    }
    for(int i=1;i<=m;i++)println(Ans[i]);
}
```

2.11.4 回滚莫队

```
struct Query
{
    int l,r,posl,posr,id;
    bool operator<(const Query&x)const
    {
        return (posl^x.posl)?posl<x.posl:r<x.r;
    }
}a[100005];
int b[100005],d[100005],cntd,n,m;
ll cnt[100005],_cnt[100005],Ans[100005];
int L[100005],R[100005];

void solve()
{
    read(n,m);
    int siz=(int)sqrt(n),T=n/siz;
    for(int i=1;i<=n;i++)
    {
        read(b[i]);
        d[++cntd]=b[i];
    }
    sort(d+1,d+cntd+1);
    cntd=unique(d+1,d+cntd+1)-d-1;
    for(int i=1;i<=n;i++)
    b[i]=lower_bound(d+1,d+cntd+1,b[i])-d;
    for(int i=1;i<=T;i++)
    {
        if(i*siz>n)break;
        L[i]=(i-1)*siz+1;
        R[i]=i*siz;
    }
    if(R[T]<n)T++,L[T]=R[T-1]+1,R[T]=n;
    for(int i=1;i<=m;i++)
    {
        read(a[i].l,a[i].r);
        a[i].posl=(a[i].l-1)/siz+1;
        a[i].posr=(a[i].r-1)/siz+1;
        a[i].id=i;
    }
    sort(a+1,a+m+1);
    int l=1,r=0;
    ll ans=0;
    int laspos=0;
    for(int i=1;i<=m;i++)
    {
        if(a[i].posl==a[i].posr)
        {
            for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]++;
            ll tmp=0;
            for(int j=a[i].l;j<=a[i].r;j++)
            tmp=max(tmp,_cnt[b[j]]*d[b[j]]);
            for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]--;
            Ans[a[i].id]=tmp;
        }
        else
        {
            if(laspos^a[i].posl)
            {
                while(l<R[a[i].posl]+1)cnt[b[l++]]--;
                while(l>R[a[i].posl]+1)cnt[b[--l]]++;
```

```
            while(r>R[a[i].posl])cnt[b[r--]]--;
            while(r<R[a[i].posl])
            ans=0,laspos=a[i].posl;
        }
        while(r<a[i].r)ans=max(ans,++cnt[b[++r]]*d[b[r]]);
        ll tmp=ans;
        while(l>a[i].l)tmp=max(tmp,++cnt[b[--l]]*d[b[l]]);
        while(l<R[a[i].posl]+1)cnt[b[l++]]--;
        Ans[a[i].id]=tmp;
    }
  }
  for(int i=1;i<=m;i++)println(Ans[i]);
}
```

```
struct Query
{
  int l,r,posl,posr,id;
  bool operator<(const Query&x)const
  {
    return (posl^x.posl)?posl<x.posl:r>x.r;
  }
}a[200005];
int b[200005],n,m;
int cnt[200005],_cnt[200005],Ans[200005];
int L[200005],R[200005];

void solve()
{
  read(n,m);
  int siz=(int)sqrt(n),T=n/siz;
  for(int i=1;i<=n;i++)read(b[i]);
  for(int i=1;i<=T;i++)
  {
    if(i*siz>n)break;
    L[i]=(i-1)*siz+1;
    R[i]=i*siz;
  }
  if(R[T]<n)T++,L[T]=R[T-1]+1,R[T]=n;
  for(int i=1;i<=m;i++)
  {
    read(a[i].l,a[i].r);
    a[i].posl=(a[i].l-1)/siz+1;
    a[i].posr=(a[i].r-1)/siz+1;
    a[i].id=i;
  }
  sort(a+1,a+m+1);
  int l=1,r=0;
  int ans=0;
  int laspos=0;
  for(int i=1;i<=m;i++)
  {
    if(a[i].posl==a[i].posr)
    {
      for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]++;
      ll tmp=0;
      while(_cnt[tmp])tmp++;
      for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]--;
      Ans[a[i].id]=tmp;
    }
    else
    {
      if(laspos^a[i].posl)
      {
        while(r<n)cnt[b[++r]]++;
        while(l>L[a[i].posl])cnt[b[--l]]++;
        ans=0;
        while(cnt[ans])ans++;
        while(l<L[a[i].posl])!--cnt[b[l]]?ans=min(ans,b[l++]):l++;
        laspos=a[i].posl;
      }
      while(r>a[i].r)!--cnt[b[r]]?ans=min(ans,b[r--]):r--;
      int tmp=ans;
      while(l<a[i].l)!--cnt[b[l]]?tmp=min(tmp,b[l++]):l++;
      while(l>L[a[i].posl])cnt[b[--l]]++;
      Ans[a[i].id]=tmp;
    }
  }
  for(int i=1;i<=m;i++)println(Ans[i]);
}
```

2.12 CDQ 分治

2.12.1 二维

```
//求p[i].x<=p[j].x&&p[i].y<=p[j].y&&i!=j的二元组的个数(此题y已有序)
struct Point
{
  int x,id;
}p[15005],tmp[15005];
int ans[15005],cnt[15005];

void CDQ(int l,int r)
{
```

```
    if(l==r)return;
    int mid=(l+r)>>1;
    CDQ(l,mid);CDQ(mid+1,r);
    int i=l,j=mid+1,pos=l;
    while(i<=mid&&j<=r)
    {
        if(p[i].x<=p[j].x)tmp[pos++]=p[i++];
        else{ans[p[j].id]+=i-l;tmp[pos++]=p[j++];}
    }
    while(i<=mid)tmp[pos++]=p[i++];
    while(j<=r){ans[p[j].id]+=i-l;tmp[pos++]=p[j++];}
    for(int k=l;k<=r;k++)p[k]=tmp[k];
}

void solve()
{
    int n;
    while(read(n))
    {
        for(int i=1;i<=n;i++)
        {
            ans[i]=cnt[i-1]=0;
            int b;read(p[i].x,b);
            p[i].id=i;
        }
        CDQ(1,n);
        for(int i=1;i<=n;i++)cnt[ans[p[i].id]]++;
        for(int i=0;i<n;i++)println(cnt[i]);
    }
}
```

2.12.2 三维

```
//求p[i].x<=p[j].x&&p[i].y<=p[j].y&&p[i].z<=p[j].z&&i!=j的二元组的个数
struct Point
{
    int x,y,z,w,id;
    bool operator<(const Point&p)const
    {
        if(x!=p.x)return x<p.x;
        if(y!=p.y)return y<p.y;
        return z<p.z;
    }
}p[100005],tmp[100005];
int ans[100005],cnt[100005];
int tree[200005];
int lowbit(int x){return x&(-x);}
void update(int x,int w)
{
    while(x<=200000)
    {
        tree[x]+=w;
        x+=lowbit(x);
    }
}
int query(int x)
{
    int sum=0;
    while(x)
    {
        sum+=tree[x];
        x-=lowbit(x);
    }
    return sum;
}

void CDQ(int l,int r)
{
    if(l==r)return;
    int mid=(l+r)>>1;
    CDQ(l,mid);CDQ(mid+1,r);
    int i=l,j=mid+1,pos=l;
    while(i<=mid&&j<=r)
    {
        if(p[i].y<=p[j].y)
        {
            update(p[i].z,p[i].w);
            tmp[pos++]=p[i++];
        }
        else
        {
            ans[p[j].id]+=query(p[j].z);
            tmp[pos++]=p[j++];
        }
    }
    while(i<=mid)
    {
        update(p[i].z,p[i].w);
        tmp[pos++]=p[i++];
    }
    while(j<=r)
    {
        ans[p[j].id]+=query(p[j].z);
```

```
        tmp[pos++]=p[j++];
    }
    for(int k=l;k<=mid;k++)update(p[k].z,-p[k].w);
    for(int k=l;k<=r;k++)p[k]=tmp[k];
}

void solve()
{
    int n,k;
    read(n,k);
    for(int i=1;i<=n;i++)
    {
        read(p[i].x,p[i].y,p[i].z);
        p[i].w=1;p[i].id=i;
    }
    sort(p+1,p+n+1);
    int tot=1;
    for(int i=2;i<=n;i++)
    {
        if(p[tot].x==p[i].x&&p[tot].y==p[i].y&&p[tot].z==p[i].z)p[tot].w
            ++;
        else p[++tot]=p[i];
    }
    CDQ(1,tot);
    for(int i=1;i<=tot;i++)cnt[ans[p[i].id]+p[i].w-1]+=p[i].w;
    for(int i=0;i<n;i++)println(cnt[i]);
}
```

2.13 链式前向星

```
const int maxn=200005;
struct EDGE
{
    int next,to,w;
}edge[maxn<<1];
int head[maxn];
int cnt;
void add(int u,int v,int w)
{
    edge[cnt].w=w;
    edge[cnt].to=v;
    edge[cnt].next=head[u];
    head[u]=cnt++;
}

for(int i=head[x];~i;i=edge[i].next)//遍历
{

}
```

2.14 Kruskal 重构树

```
namespace Kruskal_Rebuild_Tree
{
    struct EDGE
    {
        int a,b,c;
        bool operator<(const EDGE& e)const{return c<e.c;}
    };
    vector<EDGE>e(m+1);
    vector<int>edge[n+m+1];
    int st[n+m+1],fa[n+m+1],dep[n+m+1];
    int f[n+m+1][20];
    int findd(int a)
    {
        return fa[a]==a?a:fa[a]=findd(fa[a]);
    };
    void kruskal()
    {
        sort(e.begin()+1,e.end());
        for(int i=1;i<=m;i++)
        {
            int u=findd(e[i].a),v=findd(e[i].b),w=e[i].c;
            if(u!=v)
            {
                st[++n]=w;
                fa[n]=fa[u]=fa[v]=n;
                edge[n].push_back(u);
                edge[n].push_back(v);
                edge[u].push_back(n);
                edge[v].push_back(n);
            }
        }
    }
    void lca_init(int u,int father)
    {
        f[u][0]=father;dep[u]=dep[father]+1;
        for(int i=1;i<20;i++)f[u][i]=f[f[u][i-1]][i-1];
        for(auto v:edge[u])
        {
            if(v==father)continue;
            lca_init(v,u);
```

```
        }
    }
    int cal_lca(int x,int y)
    {
        if(dep[x]<dep[y])swap(x,y);
        for(int i=19;i>=0;i--)if(dep[f[x][i]]>=dep[y])x=f[x][i];
        if(x==y)return x;
        for(int i=19;i>=0;i--)if(f[x][i]!=f[y][i])x=f[x][i],y=f[y][i];
        return f[x][0];
    }
}
using namespace Kurskal_Rebuild_Tree;
void solve()
{
    int n,m,q;
    read(n,m,q);
    for(int i=1;i<=m;i++)read(e[i].a,e[i].b,e[i].c);
    for(int i=1;i<=n;i++)fa[i]=i;

    kruskal();
    lca_init(n,0);
    for(int i=1;i<=q;i++)
    {
        int u,v,w;
        read(u,v,w);
        int lca=cal_lca(u,v);
        if(st[lca]>w)println("Yes");
        else println("No");
    }
}
```

2.15 树链剖分

```
#include<bits/stdc++.h>
using namespace std;
int n,q;
long long a[100005];
vector<int>edge[100005];
int dep[100005],siz[100005],son[100005],fa[100005];
int id[100005],top[100005],tot=0;
void dfs(int x,int father)
{
    siz[x]=1;
    dep[x]=dep[father]+1;
    int maxsize=0;
    for(auto it:edge[x])if(it!=father)
    {
        fa[it]=x;
        dfs(it,x);
        siz[x]+=siz[it];
        if(siz[it]>maxsize)
        {
            maxsize=siz[it];
            son[x]=it;
        }
    }
}
long long k[100005];
void dfs1(int x,int father)
{
    id[x]=++tot;
    k[id[x]]=a[x];
    if(son[x])
    {
        top[son[x]]=top[x];
        dfs1(son[x],x);
    }
    for(auto it:edge[x])if(it!=father&&it!=son[x])
    {
        top[it]=it;
        dfs1(it,x);
    }
}
long long tree[100005<<2];
void build(int p,int l,int r)
{
    if(l==r)
    {
        tree[p]=k[l];
        return;
    }
    int mid=(l+r)>>1;
    build(p<<1,l,mid);
    build(p<<1|1,mid+1,r);
    tree[p]=tree[p<<1]^tree[p<<1|1];
}
void update(int p,int l,int r,long long w,int x)
{
    if(l==r)
    {
        tree[p]=w;
        return;
    }
```

```
    int mid=(l+r)>>1;
    if(x<=mid)update(p<<1,l,mid,w,x);
    else update(p<<1|1,mid+1,r,w,x);
    tree[p]=tree[p<<1]^tree[p<<1|1];
}
long long query(int p,int l,int r,int x,int y)
{
    if(x<=l&&r<=y)return tree[p];
    long long ans=0;
    int mid=(l+r)>>1;
    if(x<=mid)ans^=query(p<<1,l,mid,x,y);
    if(mid<y)ans^=query(p<<1|1,mid+1,r,x,y);
    return ans;
}
long long Query(int x,int y)
{
    long long ans=0;
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        ans^=query(1,1,n,id[top[x]],id[x]);
        x=fa[top[x]];
    }
    if(dep[x]<dep[y])swap(x,y);
    ans^=query(1,1,n,id[y],id[x]);
    return ans;
}
int main()
{
    scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++)
    scanf("%lld",&a[i]);
    for(int i=1;i<n;i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    dep[1]=1;
    dfs(1,0);
    top[1]=1;
    dfs1(1,0);
    build(1,1,n);
    while(q--)
    {
        long long op,l,r;
        scanf("%lld%lld%lld",&op,&l,&r);
        if(op==1)
        {
            update(1,1,n,r,id[l]);
        }
        else
        {
            printf("%lld\n",Query(l,r));
        }
    }
    return 0;
}
```

2.16 ST 表

```
int LOG[2000005];
void init()
{
    LOG[0]=-1;
    for(int i=1;i<=2000000;i++)
        LOG[i]=LOG[i/2]+1;
}

template<typename T,int N,int K,T(*F)(T&,T&)>
struct Sparse_Table
{
    T st[K][N];
    template<typename Iterator>
    void build(Iterator bg,Iterator ed)
    {
        int now=0;
        for(auto i=bg;i!=ed;i++) // 下标从 1 开始
        {
            st[0][++now]=(*i);
        }
        for(int k=1;k<K;k++)
        {
            for(int i=1;i+(1<<k)-1<=now;i++)
            {
                st[k][i]=F(st[k-1][i],st[k-1][i+(1<<(k-1))]);
            }
        }
    }
    T query(int l,int r)
    {
        int k=LOG[r-l+1];
```

```
        return F(st[k][l],st[k][r-(1<<k)+1]);
    }
    // 记得 init !!!
};

int Max(int &a,int &b)
{
    if(a>=b)return a;
    return b;
}

Sparse_Table<int,2000005,25,Max>st; // [](int &a,int &b){return max(a,b);}

void solve()
{
    init();
    vector<int>a(20);
    for(int i=0;i<20;i++)a[i]=i+1;
    st.build(a.begin(),a.end()); // st.build(a+1,a+n+1);
    cout<<st.query(3,10)<<endl;
}
```

2.17 DSU on Tree

```
#include<bits/stdc++.h>
using namespace std;
int n;
int a[100005];
vector<int>edge[100005];
int siz[100005],son[100005],rev[100005],dfn[100005],num;
void dfs(int u,int fa)
{
    siz[u]=1;rev[dfn[u]=++num]=u;
    for(int v:edge[u])if(v!=fa)
    {
        dfs(v,u);siz[u]+=siz[v];
        if(siz[son[u]]<siz[v])son[u]=v;
    }
}
long long ans[100005],res;
int cnt[100005],maxx;
inline void ins(int x)
{
    cnt[x]++;
    if(maxx<cnt[x])maxx=cnt[x],res=0;
    if(maxx==cnt[x])res+=x;
}
void dsu(int u,int fa,bool kp)
{
    for(int v:edge[u])if(v!=fa&&v!=son[u])dsu(v,u,0);
    if(son[u])dsu(son[u],u,1);
    ins(a[u]);
    for(auto v:edge[u])if(v!=fa&&v!=son[u])
    for(int j=dfn[v];j<=dfn[v]+siz[v]-1;j++)
    ins(a[rev[j]]);
    ans[u]=res;
    if(!kp)
    {
        maxx=res=0;
        for(int i=dfn[u];i<=dfn[u]+siz[u]-1;i++)
        cnt[a[rev[i]]]=0;
    }
}
int main()
{
    read(n);
    for(int i=1;i<=n;i++)read(a[i]);
    for(int i=1;i<n;i++)
    {
        int x,y;
        read(x);read(y);
        edge[x].push_back(y);
        edge[y].push_back(x);
    }
    dfs(1,0);
    dsu(1,0,1);
    for(int i=1;i<=n;i++){print(ans[i]);putchar(' ');}
    return 0;
}
```

2.18 线性基

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

const int N = 65;

int n;
ll b[N];
ll tmp[N];
```

```
bool flag;

void ins(ll x) //插入数字，获取线性基
{
   for(int i = 62; i >= 0; i --) {
      if(x >> i & 1) {
         if(!b[i]) {
            b[i] = x;
            return;
         }
         else x ^= b[i];
      }
   }
   flag = true;
}

ll find_max() //找到能被表示出来的最大值
{
   ll ans = 0;
   for(int i = 62; i >= 0; i --) {
      ans = max(ans, ans ^ b[i]);
   }
   return ans;
}

ll find_min() //找到能被表示出来的最小值
{
   for(int i = 0; i <= 62; i ++) {
      if(b[i]) {
         return b[i];
      }
   }
}

ll get_kth(ll k) //找到第k小的能被表示出来的数
{
   ll res = 0;
   int cnt = 0;
   k -= flag;
   if(!k) return 0;
   for(int i = 0; i <= 62; i ++) {
      for(int j = i - 1; j >= 0; j --) {
         if(b[i] >> j & 1) b[i] ^= b[j];
      }
      if(b[i]) tmp[cnt++] = b[i];
   }
   if(k >= (1ll << cnt)) return -1;
   for(int i = 0; i < cnt; i ++) {
      if(k >> i & 1) {
         res ^= tmp[i];
      }
   }
   return res;
}

bool find_x(ll x) //x是否能被线性基表示出来
{
   for(int i = 62; i >= 0; i --) {
      if(x >> i & 1) {
         if(!b[i]) return false;
         x ^= b[i];
      }
   }
   return true;
}

int main()
{
   scanf("%d", &n);
   for(int i = 1; i <= n; i ++) {
      ll x;
      scanf("%lld", &x);
      ins(x);
   }
   ll ans = find_max();
   printf("%lld\n", ans);
   return 0;
}
```

2.19 平衡树

2.19.1 替罪羊树

```
namespace Scapegoat_Tree
{
   const double alpha=0.75;
   int ls[100005],rs[100005];
   int siz[100005],val[100005];
   int fac[100005]/*实际大小*/,cnt[100005]/*存在标记*/;
   int tot,root;
   void newnode(int&now,int w)
   {
      now=++tot;
```

```
    val[now]=w;
    siz[now]=fac[now]=1;
    cnt[now]=1;
}
int imbalance(int now)//判断是否平衡
{
    if(max(siz[ls[now]],siz[rs[now]])>siz[now]*alpha
    ||siz[now]-fac[now]>siz[now]*0.3)return 1;
    return 0;
}
vector<int>v;
void ldr(int now)
{
    if(!now)return;
    ldr(ls[now]);
    if(cnt[now])v.push_back(now);
    ldr(rs[now]);
}
void lift(int l,int r,int &now)
{
    if(l==r)
    {
        now=v[l];
        ls[now]=rs[now]=0;
        siz[now]=fac[now]=cnt[now];
        return;
    }
    int mid=(l+r)>>1;
    now=v[mid];
    if(l<mid)lift(l,mid-1,ls[now]);
    else ls[now]=0;
    lift(mid+1,r,rs[now]);
    siz[now]=siz[ls[now]]+siz[rs[now]]+cnt[now];
    fac[now]=fac[ls[now]]+fac[rs[now]]+cnt[now];
}
void rebuild(int &now)//中序遍历之后拎起来
{
    v.clear();
    ldr(now);
    if(v.empty())
    {
        now=0;
        return;
    }
    lift(0,v.size()-1,now);
}
void update(int now,int end)
{
    if(!now)return;
    if(val[end]<val[now])update(ls[now],end);
    else update(rs[now],end);
    siz[now]=siz[ls[now]]+siz[rs[now]]+cnt[now];
}
void check(int &now,int end)
{
    if(now==end)return;
    if(imbalance(now))
    {
        rebuild(now);
        update(root,now);
        return;
    }
    if(val[end]<val[now])check(ls[now],end);
    else check(rs[now],end);
}
void insert(int &now,int w)
{
    if(!now)
    {
        newnode(now,w);
        check(root,now);
        return;
    }
    siz[now]++;
    fac[now]++;
    if(w<val[now])insert(ls[now],w);
    else if(w==val[now])cnt[now]++;
    else insert(rs[now],w);
}
void del(int now,int w)
{
    if(cnt[now]&&val[now]==w)
    {
        cnt[now]--;
        fac[now]--;
        check(root,now);
        return;
    }
    fac[now]--;
    if(w<val[now])del(ls[now],w);
    else if(w==val[now])cnt[now]--;
    else del(rs[now],w);
}
int queryrnk(int w)
```

```
    {
        int now=root,rnk=1;
        while(now)
        {
            if(w<val[now])now=ls[now];
            else if(w==val[now])
            {
                rnk+=fac[ls[now]];
                break;
            }
            else
            {
                rnk+=cnt[now]+fac[ls[now]];
                now=rs[now];
            }
        }
        return rnk;
    }
    int querynum(int rnk)
    {
        int now=root;
        while(now)
        {
            if(cnt[now]&&fac[ls[now]]+1<=rnk&&fac[ls[now]]+cnt[now]+1>rnk
                )break;
            else if(fac[ls[now]]>=rnk)now=ls[now];
            else
            {
                rnk-=fac[ls[now]]+cnt[now];
                now=rs[now];
            }
        }
        return val[now];
    }
    int pre(int w)
    {
        return querynum(queryrnk(w)-1);
    }
    int sub(int w)
    {
        return querynum(queryrnk(w+1));
    }
}
using namespace Scapegoat_Tree;

void solve()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        int op,x;
        scanf("%d%d",&op,&x);
        if(op==1)insert(root,x);
        else if(op==2)del(root,x);
        else if(op==3)printf("%d\n",querynum(x));
        else if(op==4)printf("%d\n",querynum(x));
        else if(op==5)printf("%d\n",pre(x));
        else if(op==6)printf("%d\n",sub(x));
    }
}
```

2.19.2 FHQ_Treap

```
//普通版本
namespace FHQ_Treap
{
    int tot,root;
    int ls[100005],rs[100005];
    int siz[100005],val[100005],rnd[100005];
    #include<random>
    mt19937 rand(233);
    inline int newnode(int w)
    {
        val[++tot]=w;
        rnd[tot]=rand();
        siz[tot]=1;
        return tot;
    }
    inline void pushup(int now)
    {
        siz[now]=siz[ls[now]]+siz[rs[now]]+1;
    }
    void split(int now,int w,int &x,int &y)//按值分裂
    {
        if(!now)x=y=0;
        else
        {
            if(val[now]<=w)
            {
                x=now;
                split(rs[now],w,rs[now],y);
            }
```

```
        else
        {
            y=now;
            split(ls[now],w,x,ls[now]);
        }
        pushup(now);
    }
}
int merge(int x,int y)
{
    if(!x||!y)return x+y;
    if(rnd[x]>rnd[y])//>、>=、<、<=都可以
    {
        rs[x]=merge(rs[x],y);
        pushup(x);
        return x;
    }
    else
    {
        ls[y]=merge(x,ls[y]);
        pushup(y);
        return y;
    }
}
int x,y,z;
inline void insert(int w)
{
    split(root,w,x,y);
    root=merge(merge(x,newnode(w)),y);
}
inline void del(int w)
{
    split(root,w,x,z);
    split(x,w-1,x,y);
    y=merge(ls[y],rs[y]);
    root=merge(merge(x,y),z);
}
inline int queryrnk(int w)
{
    split(root,w-1,x,y);
    int ans=siz[x]+1;
    root=merge(x,y);
    return ans;
}
inline int querynum(int rnk)
{
    int now=root;
    while(now)
    {
        if(siz[ls[now]]+1==rnk)break;
        else if(siz[ls[now]]>=rnk)now=ls[now];
        else
        {
            rnk-=siz[ls[now]]+1;
            now=rs[now];
        }
    }
    return val[now];
}
inline int pre(int w)
{
    split(root,w-1,x,y);
    int now=x;
    while(rs[now])now=rs[now];
    root=merge(x,y);
    return val[now];
}
inline int sub(int w)
{
    split(root,w,x,y);
    int now=y;
    while(ls[now])now=ls[now];
    root=merge(x,y);
    return val[now];
}
}
using namespace FHQ_Treap;

void solve()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        int op,w;
        scanf("%d%d",&op,&w);
        if(op==1)insert(w);
        else if(op==2)del(w);
        else if(op==3)printf("%d\n",queryrnk(w));
        else if(op==4)printf("%d\n",querynum(w));
        else if(op==5)printf("%d\n",pre(w));
        else printf("%d\n",sub(w));
    }
}
```

```
//区间翻转
namespace FHQ_Treap
{
    int tot,root;
    int ls[100005],rs[100005],laz[100005];
    int siz[100005],val[100005],rnd[100005];
    #include<random>
    mt19937 rand(233);
    inline int newnode(int w)
    {
        val[++tot]=w;
        rnd[tot]=rand();
        siz[tot]=1;
        return tot;
    }
    inline void pushup(int now)
    {
        siz[now]=siz[ls[now]]+siz[rs[now]]+1;
    }
    inline void pushdown(int now)
    {
        swap(ls[now],rs[now]);
        laz[ls[now]]^=1;
        laz[rs[now]]^=1;
        laz[now]=0;
    }
    void split(int now,int size,int &x,int &y)//按大小分裂
    {
        if(!now)x=y=0;
        else
        {
            if(laz[now])pushdown(now);
            if(siz[ls[now]]<size)
            {
                x=now;
                split(rs[now],size-siz[ls[now]]-1,rs[now],y);
            }
            else
            {
                y=now;
                split(ls[now],size,x,ls[now]);
            }
            pushup(now);
        }
    }
    int merge(int x,int y)
    {
        if(!x||!y)return x+y;
        if(rnd[x]>rnd[y])
        {
            if(laz[x])pushdown(x);
            rs[x]=merge(rs[x],y);
            pushup(x);
            return x;
        }
        else
        {
            if(laz[y])pushdown(y);
            ls[y]=merge(x,ls[y]);
            pushup(y);
            return y;
        }
    }
    void reverse(int l,int r)
    {
        int x,y,z;
        split(root,l-1,x,y);
        split(y,r-l+1,y,z);
        laz[y]^=1;
        merge(merge(x,y),z);
    }
    void ldr(int now)//中序遍历
    {
        if(!now)return;
        if(laz[now])pushdown(now);
        ldr(ls[now]);
        printf("%d ",val[now]);
        ldr(rs[now]);
    }
}
using namespace FHQ_Treap;

void solve()
{
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    root=merge(root,newnode(i));
    for(int i=1;i<=m;i++)
    {
        int l,r;
        scanf("%d%d",&l,&r);
        reverse(l,r);
    }
```

```
    ldr(root);
}
```

2.19.3 Splay

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

#define ll long long
//#define int ll
constexpr int inf=0x3f3f3f3f;
constexpr ll INF=0x3f3f3f3f3f3f3f3f;

const int N = 100005;

struct Splay
{
    int rt,tot,fa[N],ch[N][2],val[N],cnt[N],sz[N];
    void maintain(int x){sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+cnt[x];}
    bool get(int x){return x==ch[fa[x]][1];}
    void clear(int x){ch[x][0]=ch[x][1]=fa[x]=val[x]=sz[x]=cnt[x]=0;}
    void rotate(int x)
    {
        int y=fa[x],z=fa[y],chk=get(x);
        ch[y][chk]=ch[x][chk^1];
        if(ch[x][chk^1])fa[ch[x][chk^1]]=y;
        ch[x][chk^1]=y;
        fa[y]=x;
        fa[x]=z;
        if(z)ch[z][y==ch[z][1]]=x;
        maintain(x);
        maintain(y);
    }
    void splay(int x)
    {
        for(int f=fa[x];f=fa[x],f; rotate(x))
        if(fa[f])rotate(get(x)==get(f)?f:x);
        rt=x;
    }
    void ins(int k)//插入操作
    {
        if(!rt)
        {
            val[++tot]=k;
            cnt[tot]++;
            rt=tot;
            maintain(rt);
            return;
        }
        int cur=rt,f=0;
        while(1)
        {
            if(val[cur]==k)
            {
                cnt[cur]++;
                maintain(cur);
                maintain(f);
                splay(cur);
                break;
            }
            f=cur;
            cur=ch[cur][val[cur]<k];
            if(!cur)
            {
                val[++tot]=k;
                cnt[tot]++;
                fa[tot]=f;
                ch[f][val[f]<k]=tot;
                maintain(tot);
                maintain(f);
                splay(tot);
                break;
            }
        }
    }
    int rk(int k)//查询x的排名
    {
        int res=0,cur=rt;
        while(1)
        {
            if(k<val[cur])
            {
                cur=ch[cur][0];
            }
            else
            {
                res+=sz[ch[cur][0]];
                if(k==val[cur])
                {
                    splay(cur);
                    return res+1;
                }
                res+=cnt[cur];
```

```
            cur=ch[cur][1];
        }
    }
}

int kth(int k)//查询排名x的数
{
    int cur=rt;
    while(1)
    {
        if(ch[cur][0]&&k<=sz[ch[cur][0]])
        {
            cur=ch[cur][0];
        }
        else
        {
            k-=cnt[cur]+sz[ch[cur][0]];
            if(k<=0)
            {
                splay(cur);
                return val[cur];
            }
            cur=ch[cur][1];
        }
    }
}
int pre()//查询前驱
{
    int cur=ch[rt][0];
    if(!cur)return cur;
    while(ch[cur][1])cur=ch[cur][1];
    splay(cur);
    return cur;
}
int nxt()//查询后继
{
    int cur=ch[rt][1];
    if(!cur)return cur;
    while(ch[cur][0])cur=ch[cur][0];
    splay(cur);
    return cur;
}
void del(int k)//删除操作
{
    rk(k);
    if(cnt[rt]>1)
    {
        cnt[rt]--;
        maintain(rt);
        return;
    }
    if(!ch[rt][0]&&!ch[rt][1])
    {
        clear(rt);
        rt=0;
        return;
    }
    if(!ch[rt][0])
    {
        int cur=rt;
        rt=ch[rt][1];
        fa[rt]=0;
        clear(cur);
        return;
    }
    if(!ch[rt][1])
    {
        int cur=rt;
        rt=ch[rt][0];
        fa[rt]=0;
        clear(cur);
        return;
    }
    int cur=rt;
    int x=pre();
    fa[ch[cur][1]]=x;
    ch[x][1]=ch[cur][1];
    clear(cur);
    maintain(rt);
}
}tree;


void solve()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        int op,x;
        scanf("%d%d",&op,&x);
        if(op==1)tree.ins(x);
        else if(op==2)tree.del(x);
        else if(op==3)printf("%d\n",tree.rk(x));
```

```
        else if(op==4)printf("%d\n",tree.kth(x));
        else if(op==5)
        {
            tree.ins(x);
            printf("%d\n",tree.val[tree.pre()]);
            tree.del(x);
        }
        else if(op==6)
        {
            tree.ins(x);
            printf("%d\n",tree.val[tree.nxt()]);
            tree.del(x);
        }
    }
}

signed main()
{
    // ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);



    // int _;scanf("%d",&_);while(_--)
    solve();
    return 0;
}
```

2.20 LCT

```
//0 x y 代表询问从x到y的路径上的点的权值的xor和。保证x到y是联通的。
//1 x y 代表连接x到y，若x到y已经联通则无需连接。
//2 x y 代表删除边(x,y)，不保证边(x,y)存在。
//3 x y 代表将点x上的权值变成y。
#include<bits/stdc++.h>
using namespace std;

#define ll long long
//#define int ll
constexpr int inf=0x3f3f3f3f;
constexpr ll INF=0x3f3f3f3f3f3f3f3f;

constexpr int N=500005;

struct Link_Cut_Tree
{
    #define get(x) (son[1][fa[x]]==x)//获取x是父亲的那个儿子
    int son[2][N],fa[N],siz[N],val[N];
    inline void pushup(int x)
    {
        siz[x]=siz[son[0][x]]^siz[son[1][x]]^val[x];
    }
    bool rev[N];
    inline void pushdown(int x)
    {
        if(!rev[x])return;
        rev[son[0][x]]^=1;rev[son[1][x]]^=1;
        swap(son[0][x],son[1][x]);rev[x]=0;
    }
    inline bool isroot(int x)//判断x是否是所在树的根
    {
        return !(son[0][fa[x]]==x||son[1][fa[x]]==x);
    }
    inline void rotate(int x)//将x向上旋转一层的操作
    {
        int y=fa[x],z=fa[y];
        if(!isroot(y))son[y==son[1][z]][z]=x;
        bool is=(son[1][y]==x);
        son[is][y]=son[!is][x];fa[son[!is][x]]=y;
        son[!is][x]=y;fa[y]=x;fa[x]=z;pushup(y);pushup(x);
    }
    int stk[N],top;
    inline void splay(int x)//通过和rotate操作联动实现把x转移到当前splay的根
    {
        stk[++top]=x;
        for(int i=x;!isroot(i);i=fa[i])stk[++top]=fa[i];
        while(top)pushdown(stk[top--]);
        while(!isroot(x))
        {
            int y=fa[x],z=fa[y];//cout<<x<<" "<<y<<" "<<z<<endl;
            if(!isroot(y))
            {
                if((son[1][y]==x)^(son[1][z]==y))rotate(x);
                else rotate(y);
            }
            rotate(x);
        }
    }
    inline int access(int x)//把从根到x的所有点放在一条实链里,使根到x成为一条
        实路径,并且在同一棵splay里
    {
        int p;
        for(p=0;x;p=x,x=fa[x])
        splay(x),son[1][x]=p,pushup(x);
```

```
        return p;
        //连续两次access操作时,第二次access操作的返回值等于这两个节点的LCA
        //表示x到根的链所在的splay树的根.这个节点一定已经被旋转到了根节点,且父亲一定为空
    }
    void update(int x)//在access操作之后,递归地从上到下pushdown信息
    {
        if(!isroot(x))update(fa[x]);
        pushdown(x);
    }
    inline void makeroot(int x)//使x点成为其所在树的根
    {
        access(x);splay(x);rev[x]^=1;
    }
    inline int find(int x)//找到x所在树的根节点编号
    {
        access(x);splay(x);
        while(son[0][x])x=son[0][x];
        return x;
    }
    inline void link(int x,int y)//把x,y两点间连一条边
    {
        if(find(x)==find(y))return;
        makeroot(x);fa[x]=y;
    }
    inline void split(int x,int y)//提取出x,y间的路径,方便做区间操作,并以y为根
    {
        makeroot(x);access(y);splay(y);
    }
    inline void cut(int x,int y)//把x,y两点间边删掉
    {
        split(x,y);
        if(son[0][y]==x&&son[1][x]==0)son[0][y]=fa[x]=0;
    }
    inline int lca(int x,int y)//求两点LCA
    {
        access(x);splay(x);
        return access(y);
    }
}T;

void solve()
{
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&T.val[i]);
        T.siz[i]=T.val[i];
    }
    for(int i=1;i<=m;i++)
    {
        int op,x,y;
        scanf("%d%d%d",&op,&x,&y);
        if(op==0)T.split(x,y),printf("%d\n",T.siz[y]);
        else if(op==1)T.link(x,y);
        else if(op==2)T.cut(x,y);
        else T.makeroot(x),T.val[x]=y,T.pushup(x);
    }
}

signed main()
{
    // ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);


    // int _;scanf("%d",&_);while(_--)
    solve();
    return 0;
}
```

3 String

3.1 KMP

```cpp
#include<bits/stdc++.h>
using namespace std;
const ll mod = 223092870;
const int MAXN = 1000005;
char s1[MAXN],s2[MAXN];
int nex[MAXN];
int main()
{
    scanf("%s%s",s1+1,s2+1);
    int n=strlen(s1+1),m=strlen(s2+1);
    nex[1]=0;
    for(int i=2,j=0;i<=m;i++)
    {
        while(j>0&&s2[i]!=s2[j+1])j=nex[j];
        if(s2[i]==s2[j+1])j++;
        nex[i]=j;
    }
    for(int i=1,j=0;i<=n;i++)
    {
        while(j>0&&s1[i]!=s2[j+1])j=nex[j];
        if(s1[i]==s2[j+1])j++;
        if(j==m)printf("%d\n",i-m+1),j=nex[j];
    }
    for(int i=1;i<=m;i++)printf("%d ",nex[i]);
    return 0;
}
```

3.2 马拉车

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

const int N=100005;
int len[2*N+5];

void manacher(string& s)
{
    int siz=s.size();
    string str="#";
    for(int i=0;i<siz;i++)
        str=str+s[i]+"#";
    cout<<str<<'\n';
    siz=str.size();
    for(int i=0,l=0,r=-1;i<siz;i++)
    {
        int k=(i>r)?1:min(len[l+r-i],r-i+1);
        while(0<=i-k&&i+k<siz&&str[i-k]==str[i+k])
        {
            k++;
        }
        len[i]=k--;
        if(i+k>r)
        {
            l=i-k;
            r=i+k;
        }
    }
}

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

    int n;
    cin>>n;
    string s;
    cin>>s;
    manacher(s);
    for(int i=0;i<2*n+1;i++)
        cout<<len[i]<<" \n"[i==2*n];

    return 0;
}
```

3.3 哈希

3.3.1 一维

```cpp
#include<bits/stdc++.h>
typedef unsigned long long ull;
using namespace std;
char t[1000005];
char s[1000005];
ull h[1000005],p[1000005];
ull h1[1000005];
ull get(int l,int r)
{
    return h[r]-h[l-1]*p[r-l+1];
```

```
}
ull get1(int l,int r)
{
    return h1[r]-h1[l-1]*p[r-l+1];
}
unordered_map<ull,bool>mp;
int main()
{
    scanf("%s",t+1);
    int len=strlen(t+1);
    p[0]=1;
    h[0]=0;
    for(int i=1;i<=1000000;i++)
    p[i]=p[i-1]*131;
    for(int i=1;i<=len;i++)
    h[i]=h[i-1]*131+t[i]-'a'+1;
    mp[get(1,len)]=1;
    for(int j=1;j<len;j++)
    mp[get(1,j)+get(j+1,len)*p[j]]=1;
    int _;
    scanf("%d",&_);
    while (_--)
    {
        scanf("%s",s+1);
        int l=strlen(s+1);
        h1[0]=0;
        for(int i=1;i<=l;i++)
        h1[i]=h1[i-1]*131+s[i]-'a'+1;
        int num=0;
        for(int i=1;i<=l-len+1;i++)
        if(mp[get1(i,i+len-1)])num++;
        printf("%d\n",num);
    }
    return 0;
}
```

3.3.2 二维

```
#include<bits/stdc++.h>
typedef unsigned int ui;
using namespace std;
ui h[1005][1005];
ui p1[1005],p2[1005];
ui get(int x1,int y1,int x2,int y2)
{
    return h[x2][y2]-h[x2][y1-1]*p1[y2-y1+1]-h[x1-1][y2]*p2[x2-x1+1]+h[
        x1-1][y1-1]*p1[y2-y1+1]*p2[x2-x1+1];
}
unordered_map<ui,bool>mp;
int main()
{
    char x;
    p1[0]=1;p2[0]=1;
    for(int i=1;i<=1000;i++)
    {
        p1[i]=p1[i-1]*131;
        p2[i]=p2[i-1]*233;
    }
    int n,m,a,b;
    scanf("%d%d%d%d",&n,&m,&a,&b);
    getchar();
    for(int j=0;j<=m;j++)h[0][j]=0;
    for(int i=1;i<=n;i++,getchar())
    {
        h[i][0] = 0;
        for(int j=1;j<=m;j++)
        {
            x=getchar();
            h[i][j]=h[i][j-1]*131+x;
        }
    }
    for(int i=1;i<=n;i++)
    for(int j=1;j<=m;j++)
    h[i][j]+=h[i-1][j]*233;
    for(int i=1;i<=n-a+1;i++)
    for(int j=1;j<=m-b+1;j++)
    {
        ui z=get(i,j,i+a-1,j+b-1);
        if(!mp.count(z))mp[z]=1;
    }
    int _;
    scanf("%d",&_);getchar();
    while(_--)
    {
        for(int i=1;i<=a;i++,getchar())
        {
            h[i][0]=0;
            for(int j=1;j<=b;j++)
            {
                x=getchar();
                h[i][j]=h[i][j-1]*131+x;
            }
```

```
        }
        for(int i=1;i<=a;i++)
        h[i][b]+=h[i-1][b]*233;
        if(mp.count(h[a][b]))printf("1\n");
        else printf("0\n");
    }
    return 0;
}
```

3.4 Trie 树

```
#include<bits/stdc++.h>
using namespace std;
int trie[500005][30];
int color[500005];
int k=1;
void insert(char *w)
{
    int len=strlen(w);
    int p=0;
    for(int i=0;i<len;i++)
    {
        int c=w[i]-'a';
        if(!trie[p][c])
        trie[p][c]=k++;
        p=trie[p][c];
    }
    color[p]=1;
}
int search(char *s)
{
    int len=strlen(s);
    int p=0;
    for(int i=0;i<len;i++)
    {
        int c=s[i]-'a';
        if(!trie[p][c])return 0;
        p=trie[p][c];
    }
    if(color[p]==1){color[p]=2;return 1;}
    else if(color[p]==2)return 2;
    else return 0;
}
char str[55];
int main()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%s",str);
        insert(str);
    }
    int m;
    scanf("%d",&m);
    for(int i=1;i<=m;i++)
    {
        scanf("%s",str);
        int res=search(str);
        if(!res)printf("WRONG\n");
        else if(res==1)printf("OK\n");
        else printf("REPEAT\n");
    }
    return 0;
}
```

3.5 AC 自动机

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN=2000005;
int trie[MAXN][26];
int ed[MAXN];
int fail[MAXN];
int mp[200005];
int tot=0;
int ans[200005];
int vis[200005];
int in[200005];
void insert(char *s,int num)
{
    int len=strlen(s),p=0;
    for(int i=0;i<len;i++)
    {
        int ch=s[i]-'a';
        if(trie[p][ch]==0)trie[p][ch]=++tot;
        p=trie[p][ch];
    }
    if(!ed[p])ed[p]=num;
    mp[num]=ed[p];
}
void build()
```

```
{
    queue<int>q;
    memset(fail,0,sizeof(fail));
    for(int i=0;i<26;i++)if(trie[0][i])q.push(trie[0][i]);
    while(!q.empty())
    {
        int fa=q.front();
        q.pop();
        for(int i=0;i<26;i++)
        {
            if(trie[fa][i])
            {
                fail[trie[fa][i]]=trie[fail[fa]][i];
                in[fail[trie[fa][i]]]++;
                q.push(trie[fa][i]);
            }
            else
            {
                trie[fa][i]=trie[fail[fa]][i];
            }
        }
    }
}
void TopologicalSort()
{
    queue<int>q;
    for(int i=1;i<=tot;i++)if(!in[i])q.push(i);
    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        vis[ed[u]]=ans[u];
        int v=fail[u];
        in[v]--;
        ans[v]+=ans[u];
        if(!in[v])q.push(v);
    }
}
void query(char *s)
{
    int len=strlen(s);
    int p=0;
    for(int i=0;i<len;i++)
        p=trie[p][s[i]-'a'],ans[p]++;
}
char s[200005];
char str[2000005];
int main()
{
    int n;
    scanf("%d",&n);
    tot=0;
    for(int i=1;i<=n;i++)
    {
        scanf("%s",s);
        insert(s,i);
    }
    build();
    scanf("%s",str);
    query(str);
    TopologicalSort();
    for(int i=1;i<=n;i++)printf("%d\n",vis[mp[i]]);
    return 0;
}
```

3.6 回文自动机

```
// BZOJ 3676
// calc max(len(t)*cnt(t)) t为s回文子串，cnt(t)=t出现次数
#include<bits/stdc++.h>
using namespace std;
const int maxn = 3e5+100;
struct Palindromic_AutoMaton{
    //basic
    int s[maxn]{},now{};
    int nxt[maxn][26]{},fail[maxn]{},len[maxn]{},last{},tot{};
    // extension
    int num[maxn]{};/*节点代表的所有回文串出现次数*/
    void clear(){
        //1节点：奇数长度root 0节点：偶数长度root
        s[0]=len[1]=-1;
        fail[0] = tot = now =1;
        last = len[0]=0;
        memset(nxt[0],0,sizeof nxt[0]);
        memset(nxt[1],0,sizeof nxt[1]);
    }
    Palindromic_AutoMaton(){clear();}
    int newnode(int ll){
        tot++;
        memset(nxt[tot],0,sizeof nxt[tot]);
        fail[tot]=num[tot]=0;
        len[tot]=ll;
        return tot;
```

```
    }
    int get_fail(int x){
        while (s[now-len[x]-2]!=s[now-1])x = fail[x];
        return x;
    }
    void add(int ch){
        s[now++] = ch;
        int cur = get_fail(last);
        if(!nxt[cur][ch]){
            int tt = newnode(len[cur]+2);
            fail[tt] = nxt[get_fail(fail[cur])][ch];
            nxt[cur][ch] = tt;
        }
        last = nxt[cur][ch];num[last]++;
    }
    void build(){
        //fail[i]<i, 拓扑更新可以单调扫描。
        for (int i=tot;i>=2;i--){
            num[fail[i]]+=num[i];
        }
        num[0]=num[1]=0;
    }
    void init(char* ss){
        while (*ss){
            add(*ss-'a');ss++;
        }
    }
    void init(const string& str){
        for (char i : str){
            add(i-'a');
        }
    }
    long long query();
}pam;
long long Palindromic_AutoMaton::query(){
    long long ret =1;
    for (int i=2;i<=tot;i++){
        ret = max(ret,1LL*len[i]*num[i]);
    }
    return ret;
}
char s[maxn];
int main(){
    scanf("%s",s);
    pam.init(s);
    pam.build();
    printf("%lld\n",pam.query());
    return 0;
}
```

3.7 广义回文自动机

```
//树上回文自动机，每个点到根的路径代表一个字符串。
//强制在线增加叶子，求最大回文后缀。
#include<bits/stdc++.h>
using namespace std;
const int mod = 998244353;
int lastans = 0;
const int maxn = 5e5+100;
int st[maxn][20];
int pos[maxn];
map<char,int> score;
char a[maxn];
int sum[maxn];
int get_anc(int x,int len){
    while (len){
        int bit = __builtin_ctz(len);
        x = st[x][bit];
        len ^= (1 << bit);
    }
    return x;
}
int Damage[maxn];
struct Palindromic_AutoMaton{
    //basic
    int now;
    int nxt[maxn][26],fail[maxn],len[maxn],tot;
    int damage[maxn];
    int damage_sum[maxn];
    int total[maxn];
    int Link[maxn][26];
    // extension
    int num[maxn];/*节点代表的所有回文串出现次数*/
    void clear(){
        //1节点：奇数长度root 0节点：偶数长度root
        len[1]=-1;
        fail[0] = tot = now =1;
        len[0]=0;
        memset(nxt[0],0,sizeof nxt[0]);
        memset(nxt[1],0,sizeof nxt[1]);
        for (int i = 0;i < 26;i ++)Link[1][i]=1,Link[0][i] = 1;
    }
    Palindromic_AutoMaton(){clear();}
```

```
    int newnode(int ll){
        tot++;
        memset(nxt[tot],0,sizeof nxt[tot]);
        fail[tot]=num[tot]=0;
        len[tot]=ll;
        return tot;
    }
    int add(int i,int F,int last,int ch){
        int cur = Link[last][ch];
        if (a[get_anc(i, len[last] + 1)] - 'A' == ch){
            cur = last;
        }
        if(!nxt[cur][ch]){
            int tt = newnode(len[cur] + 2);
            fail[tt] = nxt[Link[cur][ch]][ch];
            nxt[cur][ch] = tt;
            memcpy(Link[tt],Link[fail[tt]],sizeof Link[tt]);

            int u = get_anc(i, len[fail[tt]]);

            Link[tt][a[u] - 'A'] = fail[tt];

            damage[tt] = sum[i] - sum[get_anc(i, len[tt])];
            damage_sum[tt] = (damage_sum[fail[tt]] + damage[tt])% mod;
        }
        last = nxt[cur][ch];num[last]++;
        return last;
    }
}pam;
char buf[10];
int main(){
    int _;
    scanf("%d",&_);
    score['A'] = _;
    scanf("%d",&_);
    score['C'] = _;
    scanf("%d",&_);
    score['G'] = _;
    scanf("%d",&_);
    score['U'] = _;
    scanf("%s",buf);
    a[1] = buf[0];
    a[0] = 'Z';
    sum[1] = score[buf[0]];
    pos[0] = 1;
    pos[1] = pam.add(1,0,1,buf[0] - 'A');
    lastans = pam.damage[pos[1]];
    printf("%d\n",lastans);
    int total = lastans;
    for (int i = 2;;i ++){
        int F;
        scanf("%d",&F);
        F ^= lastans;
        if (!F)break;
        scanf("%s",buf);
        a[i] = buf[0];
        sum[i] = sum[F] + score[buf[0]];
        st[i][0] = F;
        for (int step = 1;step < 20 and st[i][step-1];step ++){
            st[i][step] = st[st[i][step-1]][step-1];
        }
        pos[i] = pam.add(i,F,pos[F],buf[0] - 'A');
        Damage[i] = pam.damage_sum[pos[i]] % mod;
        (total += Damage[i]) %= mod;
        printf("%d\n",lastans = pam.damage[pos[i]]);
    }
    printf("%d\n",total);
    return 0;
}
```

3.8 后缀数组 (SA)

```
template<int N>
struct suffixarray
{
    char s[N];
    int rk[N],sa[N],ht[N],tmp[N],cnt[N];
    // n:串长 m:字符集大小 s[0..n-1]:字符串 sa[1..n]:字典序第 i 小的是哪个后缀
    // rk[0..n-1]:后缀 i 的排名 ht[i]:lcp(sa[i],sa[i-1])
    void build(int n,int m)
    {
        int i,j,k;n++;
        for(i=0;i<n*2+5;i++)rk[i]=sa[i]=ht[i]=tmp[i]=0;
        for(i=0;i<m;i++)cnt[i]=0;
        for(int i=0;i<n;i++)cnt[rk[i]=s[i]]++;
        for(int i=1;i<m;i++)cnt[i]+=cnt[i-1];
        for(i=0;i<n;i++)sa[--cnt[rk[i]]]=i;
        for(k=1;k<=n;k<<=1)
        {
            for(i=0;i<n;i++)
            {
                j=sa[i]-k;
                if(j<0)j+=n;
```

```
            tmp[cnt[rk[j]]++]=j;
        }
        sa[tmp[cnt[0]=0]]=j=0;
        for(i=1;i<n;i++)
        {
            if(rk[tmp[i]]!=rk[tmp[i-1]]||rk[tmp[i]+k]!=rk[tmp[i-1]+k])
                cnt[++j]=i;
            sa[tmp[i]]=j;
        }
        memcpy(rk,sa,n*sizeof(int));
        memcpy(sa,tmp,n*sizeof(int));
        if(j>=n-1)break;
    }
    for(j=rk[ht[i=k=0]=0];i<n-1;i++,k++)
        while(~k&&s[i]!=s[sa[j-1]+k])ht[j]=k--,j=rk[sa[j]+1];
}
void debug(int n)
{
    cout<<"sa:";for(int i=1;i<=n;i++)cout<<sa[i]<<" \n"[i==n];
    cout<<"rk:";for(int i=0;i<n;i++)cout<<rk[i]<<" \n"[i==n-1];
    cout<<"ht:";for(int i=1;i<=n;i++)cout<<ht[i]<<" \n"[i==n];
}
int mi[__lg(N)+1][N];
void rmqinit(int n)
{
    for(int i=1;i<=n;i++)
    {
        mi[0][i]=ht[i];
    }
    for(int i=1;i<=__lg(n);i++)
    {
        for(int j=1;j+(1<<i)-1<=n;j++)
        {
            mi[i][j]=min(mi[i-1][j],mi[i-1][j+(1<<(i-1))]);
        }
    }
}
int query(int l,int r)
{
    int k=__lg(r-l+1);
    return min(mi[k][l],mi[k][r-(1<<k)+1]);
}
int Query(int L,int R)
{
    int l=min(rk[L],rk[R]);
    int r=max(rk[L],rk[R]);
    return query(l+1,r);
}
}
int cmp(int l1,int r1,int l2,int r2)
{
    int len=Query(l1,l2);
    if(r1-l1+1<=len&&r2-l2+1<=len)return 0;// ==
    else if(r1-l1+1<=len)return -1;// <
    else if(r2-l2+1<=len)return 1;// >
    else
    {
        if(s[l1+len]>s[l2+len])return 1;// >
        else return -1;// <
    }
}
};

suffixarray<200005>sa;
```

3.9 后缀数组 (SA-IS)

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

#define ll long long
#define fi first
#define se second
#define lowbit(x) (x&(-x))
const int mod=998244353;
const double eps=1e-12;
const int inf=0x3f3f3f3f;
const ll INF=0x3f3f3f3f3f3f3f3f;
int dcmp(double x){if(fabs(x)<eps)return 0;return x>0?1:-1;}

// #define int ll

const int N=400050;
int nxt[N];
string s;
const int MAXLEN = 200005;
struct SAIS
{
    // 后缀类型
    #define L_TYPE 0
    #define S_TYPE 1

    int st[MAXLEN];
    int rl[MAXLEN],rk[MAXLEN],lcp[MAXLEN];
```

```
int n;

void init(const string &s)
{
    n = s.size();
    for (int i=1;i<=n;++i) st[i] = s[i-1];
    // st中的字符必须调成正数!!!!!!!!!!!!!!
}

// 判断一个字符是否为LMS字符
inline bool is_lms_char(int *type, int x) {
    return x > 1 && type[x] == S_TYPE && type[x - 1] == L_TYPE;
}

// 判断两个LMS子串是否相同
inline bool equal_substring(int *S, int x, int y, int *type) {
    do {
        if (S[x] != S[y])
        return false;
        x++, y++;
    } while (!is_lms_char(type, x) && !is_lms_char(type, y));

    return S[x] == S[y] && type[x] == type[y];
}

// 诱导排序(从*型诱导到L型、从L型诱导到S型)
// 调用之前应将*型按要求放入SA中
inline void induced_sort(int *S, int *SA, int *type, int *bucket,
    int *lbucket,
int *sbucket, int n, int SIGMA) {
    for (int i = 1; i <= n; i++)
    if (SA[i] > 1 && type[SA[i] - 1] == L_TYPE)
    SA[lbucket[S[SA[i] - 1]]++] = SA[i] - 1;
    for (int i = 0; i <= SIGMA; i++) // Reset S-type bucket
    sbucket[i] = bucket[i];
    for (int i = n; i >= 1; i--)
    if (SA[i] > 1 && type[SA[i] - 1] == S_TYPE)
    SA[sbucket[S[SA[i] - 1]]--] = SA[i] - 1;
}

// SA-IS主体
// S是输入字符串, length是字符串的长度, SIGMA是字符集的大小
int *sais(int *S, int length, int SIGMA) {
    int n = length;
    assert(S[n]==0);
    int *type = new int[n + 5]; // 后缀类型
    int *position = new int[n + 5]; // 记录LMS子串的起始位置
    int *name = new int[n + 5]; // 记录每个LMS子串的新名称
    int *SA = new int[n + 5]; // SA数组
    int *bucket = new int[SIGMA + 5]; // 每个字符的桶
    int *lbucket = new int[SIGMA + 5]; // 每个字符的L型桶的起始位置
    int *sbucket = new int[SIGMA + 5]; // 每个字符的S型桶的起始位置

    // 初始化每个桶
    memset(bucket, 0, sizeof(int) * (SIGMA + 5));
    for (int i = 1; i <= n; i++)
    bucket[S[i]]++;
    for (int i = 0; i <= SIGMA; i++) {
        if (i==0)
        {
            bucket[i] = bucket[i];
            lbucket[i] = 1;
        }else
        {
            bucket[i] += bucket[i - 1];
            lbucket[i] = bucket[i - 1] + 1;
        }
        sbucket[i] = bucket[i];
    }

    // 确定后缀类型(利用引理2.1)
    type[n] = S_TYPE;
    for (int i = n - 1; i >= 1; i--) {
        if (S[i] < S[i + 1])
        type[i] = S_TYPE;
        else if (S[i] > S[i + 1])
        type[i] = L_TYPE;
        else
        type[i] = type[i + 1];
    }
    // 寻找每个LMS子串
    int cnt = 0;
    for (int i = 1; i <= n; i++)
    if (is_lms_char(type,i))
    position[++cnt] = i;
    // 对LMS子串进行排序
    fill(SA, SA + n + 3, -1);
    for (int i = 1; i <= cnt; i++)
    SA[sbucket[S[position[i]]]--] = position[i];
    induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);

    // 为每个LMS子串命名
    fill(name, name + n + 3, -1);
    int lastx = -1, namecnt = 1; // 上一次处理的LMS子串与名称的计数
```

```
        bool flag = false; // 这里顺便记录是否有重复的字符
        for (int i = 2; i <= n; i++) {
            int x = SA[i];

            if (is_lms_char(type, x)) {
                if (lastx >= 0 && !equal_substring(S, x, lastx, type))
                namecnt++;
                // 因为只有相同的LMS子串才会有同样的名称
                if (lastx >= 0 && namecnt == name[lastx])
                flag = true;

                name[x] = namecnt;
                lastx = x;
            }
        } // for
        name[n] = 0;

        // 生成S1
        int *S1 = new int[cnt+5];
        int pos = 0;
        for (int i = 1; i <= n; i++)
        if (name[i] >= 0)
        S1[++pos] = name[i];
        int *SA1;
        if (!flag) {
            // 直接计算SA1
            SA1 = new int[cnt + 5];

            for (int i = 1; i <= cnt; i++)
            SA1[S1[i]+1] = i;
        } else
        SA1 = sais(S1, cnt, namecnt); // 递归计算SA1

        // 从SA1诱导到SA
        for (int i = 0; i <= SIGMA; i++) {
            if (i==0)
            lbucket[i] = 1;
            else
            lbucket[i] = bucket[i - 1] + 1;
            sbucket[i] = bucket[i];
        }
        fill(SA, SA + n + 3, -1);
        for (int i = cnt; i >= 1; i--) // 这里是逆序扫描SA1，因为SA中S型桶是
            倒序的
        SA[sbucket[S[position[SA1[i]]]]--] = position[SA1[i]];
        induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);

        delete[] S1;
        delete[] SA1;
        delete[] bucket;
        delete[] lbucket;
        delete[] sbucket;
        delete[] position;
        delete[] type;
        delete[] name;
        // 后缀数组计算完毕
        return SA;
    }

    void build()
    {
        st[0]=st[n+2]=-1;
        st[n+1]=0;
        int SIGMA = 0;
        for (int i=1;i<=n;++i) SIGMA = max(SIGMA,st[i]);
        int * sa = sais(st,n+1,SIGMA);
        for (int i=2;i<=n+1;++i) rk[sa[i]]=i-1;
        delete[] sa;
        for (int i=1;i<=n;++i) rl[rk[i]]=i;
        for (int i=1,len=0;i<=n;++i)
        {
            if (len) --len;
            while (i+len<=n && rl[rk[i]-1]+len<=n && st[i+len]==st[rl[rk[
                i]-1]+len]) ++len;
            lcp[rk[i]]=len;
        }
    }

    #undef L_TYPE
    #undef R_TYPE
}sa;
int len(int x){int ans=0;while(x)ans++,x>>=1;return ans;}
int mi[25][MAXLEN];
void build(int n)
{
    for(int i=1;i<=n;i++)mi[0][i]=sa.lcp[i];
    for(int j=1;j<25;j++)
    {
        for(int i=1;i<=n-(1<<j)+1;i++)
        {
            mi[j][i]=min(mi[j-1][i],mi[j-1][i+(1<<(j-1))]);
        }
    }
}
```

```
int query(int l,int r)
{
    int k=len(r-l+1)-1;
    return min(mi[k][l],mi[k][r-(1<<k)+1]);
}
int Query(int L,int R)
{
    int l=min(sa.rk[L],sa.rk[R]);
    int r=max(sa.rk[L],sa.rk[R]);
    return query(l+1,r);
}
int cmp(int l1,int r1,int l2,int r2)
{
    int len=Query(l1,l2);
    if(r1-l1+1<=len&&r2-l2+1<=len)return 0;// ==
    else if(r1-l1+1<=len)return -1;// <
    else if(r2-l2+1<=len)return 1;// >
    else
    {
        if(s[l1+len-1]>s[l2+len-1])return 1;// >
        else return -1;// <
    }
}

void solve()
{
    int n;
    cin>>n;
    cin>>s;
    sa.init(s);
    sa.build();
    build(n);
    for(int i=1;i<=n;i++)nxt[i]=1;
    for(int i=n;i>=1;i--)
    {
        int j=i+1;
        while(j<=n)
        {
            if(cmp(i,j-1,j,j+nxt[j]-1)<0)
            {
                j=j+nxt[j];
            }
            else
            {
                break;
            }
        }
        nxt[i]=j-i;
    }
    for(int i=1;i<=n;i++)
    {
        cout<<nxt[i]<<" \n"[i==n];
    }
}

/*

*/

#undef int

int main()
{
    ios::sync_with_stdio(false);cin.tie(nullptr);
    // cout<<fixed<<setprecision(15);

    srand(time(0));

    int _;cin>>_;while(_--)
    {
        solve();
    }
    return 0;
}
```

3.10 后缀自动机

```
//SPOJ substring
// calc ans_i=长度=i的所有子串，出现次数最多的一种出现了多少次。
#define RIGHT
//RIGHT: parent树的dfs序上主席树，求每个点的Right集合
const int maxn = 25e4+100;
#ifdef RIGHT
struct Node{int L,R,val;}Tree[maxn*40];
struct Chairman_Tree{
    int cnt = 0;
    int root[maxn*2];
    void init(){
        memset(root,0,sizeof root);
        cnt =0;
    }
    /* 建T0空树 */
    int buildT0(int l, int r){
```

```
        int k = cnt++;
        Tree[k].val =0;
        if (l==r) return k;
        int mid = (l+r) >>1;
        Tree[k].L = buildT0(l, mid);Tree[k].R = buildT0(mid + 1, r);
        return k;
    }
    /* 上一个版本节点P, [ppos]+=del 返回新版本节点*/
    int update (int p,int l,int r,int ppos,int del){
        assert(cnt < maxn*50);
        int k = cnt++;
        Tree[k].val = Tree[p].val +del;
        if (l==r) return k;
        int mid = (l+r) >>1;
        if (ppos<=mid){
            Tree[k].L = update(Tree[p].L,l,mid,ppos,del);
            Tree[k].R = Tree[p].R;
        }else{
            Tree[k].L = Tree[p].L;
            Tree[k].R = update(Tree[p].R,mid+1,r,ppos,del);
        }
        return k;
    }
    int query(int PL,int PR,int l,int r,int L,int R){
        if (l>R || L>r)return 0;
        if (L <= l && r <= R)return Tree[PR].val - Tree[PL].val;
        int mid = (l + r) >> 1;
        return query(Tree[PL].L,Tree[PR].L,l,mid,L,R) + query(Tree[PL].R
            ,Tree[PR].R,mid+1,r,L,R);
    }
}tree;
#endif
char s[maxn];int n,ans[maxn];
/*注意需要按len将节点基数排序来拓扑更新parent树*/
struct Suffix_Automaton{
    //basic
    int nxt[maxn*2][26],fa[maxn*2],len[maxn*2];
    int last,cnt;
    //nxt[x][c]表示从x点，通过字符c可以到达的点。
    //len[x]表示x点所能表示的最长子串长度。
    //fa[x]表示x在后缀链接上的父亲。
    //extension
    int cntA[maxn*2],A[maxn*2];/*辅助拓扑更新*/
    int num[maxn*2];/*每个节点代表的所有串的出现次数*/
    #ifdef RIGHT
    vector<int> E[maxn*2];
    int dfsl[maxn*2],dfsr[maxn*2],dfn;
    int pos[maxn*2];
    int end_pos[maxn*2];//1基
    #endif
    Suffix_Automaton(){ clear(); }
    void clear(){
        last =cnt=1;
        fa[1]=len[1]=0;
        memset(nxt[1],0,sizeof nxt[1]);
    }
    void init(char *str){
        while (*str){
            add(*str-'a');str++;
        }
    }
    void add(int c){
        int p = last;
        int np = ++cnt;
        memset(nxt[cnt],0,sizeof nxt[cnt]);
        len[np] = len[p]+1;last = np;
        while (p&&!nxt[p][c])nxt[p][c] = np,p = fa[p];
        if (!p)fa[np]=1;
        else{
            int q = nxt[p][c];
            if (len[q]==len[p]+1)fa[np] =q;
            else{
                int nq = ++ cnt;
                len[nq] = len[p]+1;
                memcpy(nxt[nq],nxt[q],sizeof (nxt[q]));
                fa[nq] =fa[q];fa[np] = fa[q] =nq;
                while (nxt[p][c]==q)nxt[p][c] =nq,p = fa[p];
            }
        }
    }
    void build(){
        memset(cntA,0,sizeof cntA);
        memset(num,0,sizeof num);
        for (int i=1;i<=cnt;i++)cntA[len[i]]++;
        for (int i=1;i<=cnt;i++)cntA[i]+=cntA[i-1];
        for (int i=cnt;i>=1;i--)A[cntA[len[i]]--] =i;
        /*更新主串节点*/
        int temp=1;
        for (int i=0;i<n;i++){
            num[temp = nxt[temp][s[i]-'a'] ]=1;
        }
        /*拓扑更新*/
        for (int i=cnt;i>=1;i--){
            //basic
```

```
        int x = A[i];
        num[fa[x]]+=num[x];
        //special
        ans[len[x]] = max(ans[len[x]],num[x]);
    }
    //special
    for (int i=len[last];i>1;i--){
        ans[i-1] = max(ans[i-1],ans[i]);
    }
}

#ifdef RIGHT
int get_right_between(int u,int l,int r){
    return tree.query(tree.root[dfsl[u] - 1],tree.root[dfsr[u]],1,::
        n,l,r);
}
void dfs(int u){
    dfsl[u] = ++ dfn;
    pos[dfn] = u;
    for (int v : E[u]){
        dfs(v);
    }
    dfsr[u] = dfn;
}
void extract_right(){
    int temp = 1;
    for (int i=0;i<n;i++){
        temp = nxt[temp][s[i] - 'a'];
        end_pos[temp] = i+1;
    }
    for (int i=2;i<=cnt;i++){
        E[fa[i]].push_back(i);
    }
    dfn = 0;
    dfs(1);
    tree.root[0] = tree.buildT0(1,n);
    for (int i=1;i<=cnt;i++){
        int u = pos[i];
        if (end_pos[u]){
            int idx = end_pos[u];
            tree.root[i] = tree.update(tree.root[i-1],1,n,idx,1);
        }else{
            tree.root[i] = tree.root[i-1];
        }
    }
}
#endif
```

```
    void debug(){
        for (int i=cnt;i>=1;i--){
            cout<<"num["<<i<<"]="<<num[i]<<" len["<<i<<"]="<<len[i]<<" fa
                ["<<i<<"]="<<fa[i]<<'\n';
        }
    }
}sam;

void solve()
{
    scanf("%s",s);
    /* calc n must before sam.init()*/
    n = strlen(s);
    sam.init(s);
    sam.build();
    for (int i=1;i<=n;i++){
        cout<<ans[i]<<'\n';
    }
}
```

3.11 广义后缀自动机

```
//build sam using trie
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e6+100;
typedef long long ll;
struct Suffix_Automaton{
    int nxt[maxn*2][26],fa[maxn*2],l[maxn*2];
    int last,cnt;
    vector<int> E[maxn*2];
    int Num[maxn*2];
    Suffix_Automaton(){ clear(); }
    void clear(){
        last =cnt=1;
        fa[1]=l[1]=0;
        memset(nxt[1],0,sizeof nxt[1]);
    }
    int add(int pre,int c,int num){
        last = pre;
        int p = last;
        int np = ++cnt;
        Num[np] = num;
        memset(nxt[cnt],0,sizeof nxt[cnt]);
        l[np] = l[p]+1;last = np;
        while (p&&!nxt[p][c])nxt[p][c] = np,p = fa[p];
```

```
        if (!p)fa[np]=1;
        else{
            int q = nxt[p][c];
            if (l[q]==l[p]+1)fa[np] =q;
            else{
                int nq = ++ cnt;
                l[nq] = l[p]+1;
                memcpy(nxt[nq],nxt[q],sizeof (nxt[q]));
                fa[nq] =fa[q];fa[np] = fa[q] =nq;
                while (nxt[p][c]==q)nxt[p][c] =nq,p = fa[p];
            }
        }
        return np;
    }
    int dfsl[maxn*2],dfsr[maxn*2];
    int dfn = 0;
    ll sum[maxn*2];
    void dfs(int u){
        dfsl[u] = ++dfn;
        sum[dfn] = Num[u];
        for (int v : E[u]){
            dfs(v);
        }
        dfsr[u] = dfn;
    }
    void build(){
        for (int i=2;i<=cnt;i++){
            E[fa[i]].push_back(i);
        }
        dfs(1);
        for (int i=1;i<=cnt;i++){
            sum[i] += sum[i-1];
        }
    }
    void query(char * s){
        int temp = 1;
        while (*s){
            int ch = *s - 'A';
            if (!nxt[temp][ch]){
                printf("0\n");
                return;
            }
            temp = nxt[temp][ch];
            s++;
        }
        ll ans = sum[dfsr[temp]] - sum[dfsl[temp] - 1];
        printf("%lld\n",ans);
    }
}sam;
struct Trie{
    int Root = 1;
    int cnt = 2;
    int nxt[maxn][26];
    int num[maxn];
    int sam_pos[maxn];
    int add(int p,int ch){
        if (!nxt[p][ch]){
            nxt[p][ch] = cnt++;
        }
        int now = nxt[p][ch];
        num[now] ++;
        return now;
    }
    void bfs(){
        queue<int> Q;
        Q.push(1);
        sam_pos[1] = 1;
        while (!Q.empty()){
            int head = Q.front();
            Q.pop();
            for (int i=0;i<26;i++){
                if (!nxt[head][i])continue;
                int now = nxt[head][i];
                sam_pos[now] = sam.add(sam_pos[head],i,num[now]);
                Q.push(now);
            }
        }
    }
}trie;
int trie_pos[maxn];
int main(){
    int n,k;
    scanf("%d%d",&n,&k);
    trie_pos[0] = 1;
    for (int i=1;i<=n;i++){
        static char s[5];
        int p;
        scanf("%s%d",s,&p);
        int ch = s[0] - 'A';
        trie_pos[i] = trie.add(trie_pos[p],ch);
    }
    trie.bfs();
    sam.build();
    for (int i=0;i<k;i++){
```

```
        static char t[maxn];
        scanf("%s",t);
        int N = strlen(t);
        reverse(t,t+N);
        sam.query(t);
    }
    return 0;
}
```

4 Mathematics

4.1 快速乘

```
inline ll mull(const ll&a,const ll&b,const ll&mod)
{
    ll tmp=a*b-(ll)((long double)a/mod*b-1e-9)*mod;
    return tmp<0?tmp+mod:tmp;
}
```

4.2 欧几里得算法

```
int gcd(int a,int b)
{
    return b==0?a:gcd(b,a%b);
}
```

4.3 拓展欧几里得算法

```
int exgcd(int a,int b,int &x,int &y)
{
    if(b==0)return x=1,y=0,a;
    int r=exgcd(b,a%b,x,y);
    tie(x,y)=make_tuple(y,x-(a/b)*y);
    return r;
}
```

4.4 BSGS

```
typedef long long ll;
map <ll,ll> f;
ll gcd(ll a,ll b) //最大公约数
{
    if (!b)return a;
    return gcd(b,a%b);
}
void exgcd(ll a,ll b,ll &x,ll &y) //拓欧
{
    if (!b)x=1,y=0;
    else
    {
        exgcd(b,a%b,x,y);
```

```
        ll t=x;
        x=y;
        y=t-a/b*y;
    }
}
ll inv(ll a,ll b/*mod*/) //逆元
{
    ll x,y;
    exgcd(a,b,x,y);
    return (x%b+b)%b;
}
ll poww(ll a,ll n,ll p)
{
    ll ans=1;
    while(n)
    {
        if(n&1)ans=ans*a%p;
        a=a*a%p;
        n>>=1;
    }
    return ans;
}
ll bsgs(ll a,ll b,ll p) //BSGS算法,p为质数 a^x=b(mod p)
{
    f.clear();
    ll m=ceil(sqrt(p));
    b%=p;
    for (ll i=1;i<=m;i++)
    {
        b=b*a%p;
        f[b]=i;
    }
    ll tmp=poww(a,m,p);
    b=1;
    for (ll i=1;i<=m;i++)
    {
        b=b*tmp%p;
        if (f[b])return (i*m-f[b]+p)%p;
    }
    return -1;
}
ll exbsgs(ll a,ll b,ll p)
{
    if (b==1||p==1)return 0; //特殊情况，x=0时最小解
    ll g=gcd(a,p),k=0,na=1;
    while (g>1)
    {
        if (b%g!=0)return -1; //无法整除则无解
        k++;b/=g;p/=g;na=na*(a/g)%p;
        if (na==b)return k; //na=b说明前面的a的次数为0，只需要返回k
        g=gcd(a,p);
    }
    ll ff=bsgs(a,b*inv(na,p)%p,p);
    if (ff==-1)return -1;
    return ff+k;
}
```

4.5 埃氏筛

```
const int maxn=10000000;
bool not_prime[maxn+5];
int prime[maxn+5],tot=0;
for(int i=2;i<=maxn;i++)
{
    if(!not_prime[i])prime[++tot]=i;
    for(int j=2*i;j<=maxn;j+=i)
    {
        not_prime[j]=1;
    }
}
```

4.6 欧拉筛

```
const int maxn=10000000;
bool not_prime[maxn+5];
int prime[maxn+5],tot=0;
for(int i=2;i<=maxn;i++)
{
    if(!not_prime[i])prime[++tot]=i;
    for(int j=1;j<=tot&&i*prime[j]<=maxn;j++)
    {
        not_prime[i*prime[j]]=1;
        if(i%prime[j]==0)break;
    }
}
```

4.7 米勒罗宾素性测试 &Pollard rho 算法

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

```
using namespace std;
typedef long long ll;
const int maxn = 105;

ll x[maxn], ans;
queue<ll> aria;

ll multi(ll a, ll b, ll p) //快速乘
{
    ll ans = 0;
    while(b)
    {
        if(b & 1LL)
            ans = (ans+a)%p;
        a = (a+a)%p;
        b >>= 1;
    }
    return ans;
}

ll qpow(ll a, ll b, ll p)
{
    ll ans = 1;
    while(b)
    {
        if(b & 1LL)
            ans = multi(ans, a, p);
        a = multi(a, a, p);
        b >>= 1;
    }
    return ans;
}

bool miller_rabin(ll n)
{
    if(n == 2)
        return true;
    int s = 20, i, t = 0;
    ll u = n-1;
    while(!(u&1))
    {
        t++;
        u >>= 1;
    }
    while(s--)
    {
        ll a = rand()%(n-2)+2;
        x[0] = qpow(a, u, n);
        for(i = 1; i <= t; i++)
        {
            x[i] = multi(x[i-1], x[i-1], n);
            if(x[i] == 1 && x[i-1] != 1 && x[i-1] != n-1)
                return false;
        }
        if(x[t] != 1)
            return false;
    }
    return true;
}

ll gcd(ll a, ll b)
{
    if(b == 0)
        return a;
    else
        return gcd(b, a%b);
}

ll Pollard_Rho(ll n, int c)
{
    ll i = 1, k = 2, x = rand()%(n-1)+1, y = x;
    while(1)
    {
        i++;
        x = (multi(x, x, n)+c)%n;
        ll p = gcd((y-x+n)%n, n);
        if(p != 1 && p != n)
            return p;
        if(y == x)
            return n;
        if(i == k)
        {
            y = x;
            k <<= 1;
        }
    }
}

void calc(ll n, int c)
{
    if(n == 1)
        return;
    if(miller_rabin(n))
    {
```

```
        aria.push(n);
        return;
    }
    ll p = n, k = c;
    while(p >= n)
    {
        p = Pollard_Rho(p, c--);
    }
    calc(p, k);
    calc(n/p, k);
}

int main()
{
    ll n;
    cin>>n;
    calc(n, 107);
    cout << aria.front();
    aria.pop();
    while(!aria.empty())
    {
        cout << "*" << aria.front();
        aria.pop();
    }
    cout << endl;
    return 0;
}
```

4.8 Stein 算法 (大整数的 GCD)

```
//如果a=0或b=0，则(0,b)=b,(a,0)=a
//如果a,b都是偶数，则(a,b)=(a/2,b/2)
//如果a是偶数，b是奇数，则(a,b)=(a/2,b)
//如果a是奇数，b是偶数，则(a,b)=(a,b/2)
//如果a,b都是奇数，不妨设a>b，则(a,b)=(b,a-b)
```

4.9 逆元

```
ll inv(ll a,ll p)
{
    ll x,y;
    exgcd(a,p,x,y);
    return (x%p+p)%p;
}
```

```
const int maxn=10000000;
long long inv[maxn+5];
inv[1]=1;
for(int i=2;i<=maxn;i++)
{
    inv[i]=inv[p%i]*(p-p/i)%p;
}
```

4.10 欧拉函数

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

```
const int maxn=10000000;
bool not_prime[maxn+5];
int prime[maxn+5],tot=0,phi[maxn+5];
for(int i=2;i<=n;i++)
{
    if(!not_prime[i])prime[++tot]=i,phi[i]=i-1;
    for(int j=1;j<=tot&&i*prime[j]<=n;j++)
    {
        not_prime[i*prime[j]]=1;
        if(i%prime[j]==0)
        {
            phi[i*prime[j]]=phi[i]*prime[j];
            break;
        }
        phi[i*prime[j]]=phi[i]*(prime[j]-1);
    }
}
```

4.11 欧拉降幂

$$a^b \mod c = \begin{cases} a^b \mod \varphi(c), & \gcd(a,c)=1 \\ a^b, & \gcd(a,c)\neq 1, b<\varphi(m) \\ a^b \mod \varphi(c)+\varphi(c), & \gcd(a,c)\neq 1, b\geq \varphi(m) \end{cases} \mod c$$

4.12 拓展中国剩余定理

```
ll excrt(ll k, ll *a, ll *r) {
    ll M = r[1], ans = a[1];
    for (int i = 2; i <= k; i++) {
        ll x0, y0;
        ll c = a[i] - ans;
        ll g = exgcd(M, r[i], x0, y0);
        if (c % g != 0) return -1;
        x0 = (__int128)x0 * (c / g) % (r[i] / g);
        ans = x0 * M + ans;
        M = lcm(M, r[i]);
        ans = (ans % M + M) % M;
    }
    return ans;//无解返回-1
}
```

4.13 中国剩余定理

```
ll CRT(int k, ll* a, ll* r) {
    ll n = 1, ans = 0;
    for (int i = 1; i <= k; i++) n = n * r[i];
    for (int i = 1; i <= k; i++) {
        ll m = n / r[i], b, y;
        exgcd(m, r[i], b, y); // b * m mod r[i] = 1
        ans = (ans + a[i] * m * b % n) % n;
    }
    return (ans % n + n) % n;
}
```

4.14 二次剩余

```
struct cipolla
{
    ll n,p,w2,a;//w2=a*a-n mod p

    struct cx{ll x,y;};//Fp2 = {x+yw | x,y belon to Fp}
    cx mul(cx a,cx b){return (cx){(a.x*b.x+a.y*b.y%p*w2)%p,(a.x*b.y+a.y
        *b.x)%p};}
    cx cx_poww(cx a,int n)
    {
        cx ans=(cx){1,0};
        while(n)
        {
            if(n&1)ans=mul(ans,a);
            a=mul(a,a);n>>=1;
        }
        return ans;
    }
    void getsol(ll _n,ll _p,ll&ans1,ll&ans2)
    {
        n=_n,p=_p;
        ans1=ans2=-1;//最多两个解,无解是-1
        if(n==0||p==2){ans1=n;return;}

        if(poww((ll)n,(ll)(p-1)/2,(ll)p)==p-1)return;

        for(a=rand()%p;;a==p-1?(a=0):(++a))
        {
            if(poww((a*a-n+p)%p,(p-1)/2,p)==p-1)
            {
                w2=(a*a-n+p)%p;
                break;
            }
        }
        cx b=(cx){a,1};
        b=cx_poww(b,(p+1)/2);
        ans1=b.x;
        if(ans1!=p-ans1)ans2=p-ans1;
        return;
    }
};
```

4.15 拓展欧拉定理

```
#include<iostream>
#include<algorithm>
#include<math.h>
#include<string.h>
#define ll long long
using namespace std;
ll phi(ll n) {//欧拉函数
    ll ans = n;
```

```
    for (ll i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            ans -= ans / i;//遇到质因数,即X*1/pi
            while (n % i == 0) {
                n /= i;
            }
        }
    }
    if (n > 1)//若n不为1，则还剩下一位质因子
    ans -= ans / n;
    return ans;
}
ll fastpow(ll a, ll b, ll m) {//快速幂
    ll ans = 1;
    while (b > 0) {
        if (b & 1) {
            ans = ans * a % m;
        }
        a = a * a % m;
        b >>= 1;
    }
    return ans;
}
ll eulerDropPow(ll a,char b[],ll c) {//欧拉降幂
    ll eulerNumbers = phi(c);//模数c降幂
    ll descendingPower = 0;
    for (ll i = 0, len = strlen(b); i < len; ++i) {
        descendingPower = (descendingPower * 10 + b[i] - '0') %
            eulerNumbers;
    }
    descendingPower += eulerNumbers-1;
    //descendingPower += eulerNumbers;应该是这样，只是这题要求a^(n-1)
    return fastpow(a, descendingPower, c);
}
long long a=2;
long long c=1000000007;
char b[1000000000];
int main() {
    while(~scanf("%s",b)) {//a是底数，b是幂，c是模
        cout<<eulerDropPow(a,b,c)<<endl;
    }
}
```

4.16 组合数

```
int MAX=1e6+1;
vector<mint>fac,inv,finv;
void binom_init()
{
    fac.resize(MAX);
    finv.resize(MAX);
    inv.resize(MAX);
    fac[0]=fac[1]=1;
    inv[1]=1;
    finv[0]=finv[1]=1;
    for(int i=2;i<MAX;i++)
    {
        fac[i]=fac[i-1]*i;
        inv[i]=-(MOD/i)*inv[MOD%i];
        finv[i]=finv[i-1]*inv[i];
    }
}
mint binom(int n,int r)
{
    if(n<r||n<0||r<0)return 0;
    return fac[n]*finv[r]*finv[n-r];
}
```

```
ll binom(ll n,ll m)
{
    if(n<m||n<0||m<0)return 0;
    if(m>n-m)m=n-m;
    ll ans=1;
    for(int i=0;i<m;i++)
    ans=ans*(n-i)/(i+1); // 取模时记得先单独把 (n-i) 取模
    return ans;
}
```

4.17 卢卡斯定理

```
long long Lucas(long long n, long long m, long long p) {
    if (m == 0) return 1;
    return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
```

4.18 拓展卢卡斯定理

```
LL calc(LL n, LL x, LL P) {
    if (!n) return 1;
    LL s = 1;
```

```
    for (LL i = 1; i <= P; i++)
    if (i % x) s = s * i % P;
    s = Pow(s, n / P, P);
    for (LL i = n / P * P + 1; i <= n; i++)
    if (i % x) s = i % P * s % P;
    return s * calc(n / x, x, P) % P;
}

LL multilucas(LL m, LL n, LL x, LL P) {
    int cnt = 0;
    for (LL i = m; i; i /= x) cnt += i / x;
    for (LL i = n; i; i /= x) cnt -= i / x;
    for (LL i = m - n; i; i /= x) cnt -= i / x;
    return Pow(x, cnt, P) % P * calc(m, x, P) % P * inv(calc(n, x, P),
        P) % P * inv(calc(m - n, x, P), P) % P;
}

LL exlucas(LL m, LL n, LL P) {
    int cnt = 0;
    LL p[20], a[20];
    for (LL i = 2; i * i <= P; i++) {
        if (P % i == 0) {
            p[++cnt] = 1;
            while (P % i == 0) p[cnt] = p[cnt] * i, P /= i;
            a[cnt] = multilucas(m, n, i, p[cnt]);
        }
    }
    if (P > 1) p[++cnt] = P, a[cnt] = multilucas(m, n, P, P);
    return CRT(cnt, a, p);
}
```

4.19 积性函数

```
ll get_f(ll n)
{
    ll ans=1;
    for(int i=2;i<=n/i;i++)
    {
        int cnt=0;
        while(n%i==0)cnt++,n/=i;
        ans*=calc_f(i,cnt);//calc_f(p,k)=f[p^k]
    }
    if(n>1)ans*=calc_f(n,1);
    return ans;
}
```

```
f[1]=1;
for(int i=2;i<=n;i++)
{
    if(!isprime[i])prime[++tot]=i,f[i]=calc_f(i,1);
    for(int j=1;j<=tot&&i*prime[j]<=n;j++)
    {
        if(i%p[j]==0)
        {
            cnt[i*p[j]]=cnt[i]+1;//cnt[n]:n的最小质因子的次数
            f[i*p[j]]=f[i]/calc_f(p[j],cnt[i])*calc_f(p[j],cnt[i]+1);
            break;
        }
        cnt[i*p[j]]=1;
        f[i*p[j]]=f[i]*calc_f(p[j],1);
    }
}
```

4.20 杜教筛

$$sum(n)=\sum_{i=1}^{n}f(i)$$

$$\sum_{i=1}^{n}h(i)=\sum_{i=1}^{n}(f*g)(i)=\sum_{i=1}^{n}\sum_{d|i}g(d)f\left(\frac{i}{d}\right)=\sum_{d=1}^{n}g(d)\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}=\sum_{d=1}^{n}g(d)sum\left(\left\lfloor\frac{n}{d}\right\rfloor\right)$$

$$\Rightarrow g(1)sum(n)=\sum_{i=1}^{n}h(i)-\sum_{i=2}^{n}g(i)sum\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

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

#define ll long long
#define fi first
#define se second
#define lowbit(x) (x&(-x))
const double eps=1e-12;
const int inf=0x3f3f3f3f;
const ll INF=0x3f3f3f3f3f3f3f3f;
int dcmp(double x){if(fabs(x)<eps)return 0;return x>0?1:-1;}

#define int ll

ll sumf1[3000005];
unordered_map<ll,ll>f1;

inline ll F1(ll n)
```

```
{
    if(n<=3e6)return sumf1[n]; // 预处理出 n 较小时的前缀和
    if(f1.count(n))return f1[n]; // 记忆化，如果求过这个值，就不需要再递归一遍了
    ll ans=n*(n+1)/2; // 这是 f * g 的 n 项前缀和
    for(ll l=2,r;l<=n;l=r+1) // 整除分块
        r=n/(n/l),ans-=(r-l+1)*F1(n/l);
        // [l,r] 的 F (n / l) 是一样的，对 g(x) 求个和即可
    return f1[n]=ans; // 别忘了除上 g(1)
}

ll sumf2[3000005];
unordered_map<ll,ll>f2;

inline ll F2(ll n)
{
    if(n<=3e6)return sumf2[n]; // 预处理出 n 较小时的前缀和
    if(f2.count(n))return f2[n]; // 记忆化，如果求过这个值，就不需要再递归一遍了
    ll ans=1; // 这是 f * g 的 n 项前缀和
    for(ll l=2,r;l<=n;l=r+1) // 整除分块
        r=n/(n/l),ans-=(r-l+1)*F2(n/l);
        // [l,r] 的 F (n / l) 是一样的，对 g(x) 求个和即可
    return f2[n]=ans; // 别忘了除上 g(1)
}

int prime[3000005];
int vis[3000005];
int len=0;

void init()
{
    sumf1[1]=sumf2[1]=1;
    for(int i=2;i<=3000000;i++)
    {
        if(!vis[i])
        {
            prime[++len]=i;
            sumf1[i]=i-1;
            sumf2[i]=-1;
        }
        for(int j=1;;j++)
        {
            long long t=1ll*prime[j]*i;
            if(t>3000000)break;
            vis[t]=1;
            if(i%prime[j]==0){sumf1[t]=sumf1[i]*prime[j];sumf2[t]=0;break;}
            else{sumf1[t]=sumf1[i]*(prime[j]-1);sumf2[t]=-sumf2[i];}
        }
    }
    for(int i=2;i<=3000000;i++)
    {
        sumf1[i]+=sumf1[i-1];
        sumf2[i]+=sumf2[i-1];
    }
}

void solve()
{
    ll n;
    cin>>n;
    cout<<F1(n)<<' '<<F2(n)<<'\n';
}

/*

*/

#undef int

int main()
{
    ios::sync_with_stdio(false);cin.tie(nullptr);
    // cout<<fixed<<setprecision(15);

    init();

    int _;cin>>_;while(_--)
    {
        solve();
    }
    return 0;
}
```

4.21 Min_25 筛

To Be Studied...

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int maxn = 2000000+100;

```

```
/*******************************

f()函数中(31-37行) 填函数在质数幂次处的表达式
pow_sum()函数中(38-43行) 填幂和函数（如果需要更高次的话可以在这里添加）
202-205行按要求填写
f_p[][0/1/2/3/ ... ]分别代表质数个数/质数和/质数平方和/质数三次方和/ ... 根据自己需要添加
例：如果该函数在质数处表达式为f(p) = p^2+3*p+1，则表明需要质数个数/质数和/质数平方和，即f_p[][0],f_p[][1],f_p[][2]

*******************************/

ll poww(ll a,ll b){
   ll res = 1;
   ll base = a;
   while(b){
      if(b&1){
         res *= base;
         //res %= mod;
      }
      base *= base;
      //base %= mod;
      b>>=1;
   }
   return res;
}

inline ll f(ll p,int e){
   if(p==1||e==0) return 1;
   ///return f(p^e)
   ll res = poww(p,e);
   return res*res+3*res+1;

}
ll pow_sum(ll n,int k){
   ///return sum(i^k),i from 1 to n.
   if(k==0) return n;
   if(k==1) return n*(n+1)/2;
   if(k==2) return n*(n+1)*(2*n+1)/6;
}
ll f_p[maxn][3];///F_prime(id(n/i))
ll n;
int n_2; ///(int)sqrt(n)
int n_3; ///(int)pow(n,1.0/3.0)
int n_6; ///(int)pow(n,1.0/6.0)
ll val_id[maxn]; ///give the id, return the id-th number like 'n/i' ,(val_id[1] = 1)
int val_id_num; ///how many numbers like 'n/i'
int val_id_num_3; ///how many numbers like 'n/i' below n/n_3;
int p[200000+100];
bool isp[maxn];
int p_sz_2; ///pi(n_2)
int p_sz_3; ///pi(n_3)
int p_sz_6; ///pi(n_6)
void init(){
   n_2 = (int)sqrt(n);
   n_3 = (int)pow(n,1.0/3.0);
   n_6 = (int)pow(n,1.0/6.0);
   val_id_num = 0;
   for(ll i=1;i<=n;){
      val_id[++val_id_num] = i;
      if(i==n) break;
      i = n/(n/(i+1));
   }
   memset(isp,1,sizeof isp);
   isp[1] = 0;
   for(int i=2;i<=n_2;i++){
      if(isp[i]){
         p[++p_sz_2] = i;
         if(i<=n_3) p_sz_3++;
         if(i<=n_6) p_sz_6++;
      }
      for(int j=1;j<=p_sz_2&&p[j]*i<=n_2;j++){
         isp[i*p[j]] = 0;
         if(i%p[j]==0) break;
      }
   }
}
inline int get_id(ll k){ ///give a number like 'n/i', return the id of it
   if(k>n_2) return val_id_num-n/k+1;
   else return k;
}
ll c[maxn];
int lowbit(int n){return n & (-n);}
void add(int x,ll d){
   while(x<maxn){
      c[x]+=d;
      x+=lowbit(x);
   }
}
ll sum(int x){
   ll ans=0;
   while(x){
```

```
        ans+=c[x];
        x-=lowbit(x);
    }
    return ans;
}

struct node{
    int k_max;
    ll val;
    ll f_val;
};
void update_bfs(int k,int type){
    queue<node> q;
    while(!q.empty()) q.pop();
    int e = 1;
    for(ll i=p[k];i<n/n_3;i*=p[k]){
        node st;
        st.k_max = k;
        st.val = i;
        if(type==-1)st.f_val = f(p[k],e);
        else st.f_val = poww(i,type);
        q.push(st);
        e++;
    }
    while(!q.empty()){
        node hd = q.front();
        q.pop();
        if((hd.val!=p[hd.k_max]&&type>=0)||type==-1) {//if(type==-1)cout
            << "****" << hd.val << "****" << hd.f_val << endl;
            ll w = n/hd.val;
            w = n/w;//cout << hd.val << "[" << w<<" , " << val_id[
                val_id_num] << "]" << endl;
            if(type==-1){
                add(get_id(w),hd.f_val);
                add(val_id_num+1,-1ll*hd.f_val);
            }
            else{
                add(get_id(w),-1ll*hd.f_val);
                add(val_id_num+1,hd.f_val);
            }
        }
        for(int i=hd.k_max+1;hd.val*p[i]<n/n_3&&i<=p_sz_2;i++){
            ll res = p[i];
            for(int e=1;;e++){
                if(hd.val*res<n/n_3){
                    node nxt;
                    nxt.k_max = i;
                    nxt.val = hd.val*res;
                    if(type==-1) nxt.f_val = hd.f_val*f(p[i],e);
                    else nxt.f_val = hd.f_val*poww(res,type);
                    q.push(nxt);
                }
                else break;
                res *= p[i];
            }
        }
    }
}
void get_f_p(ll n,int times){
    for(int i=1;i<=val_id_num;i++){
        for(int j=0;j<=times;j++){
            f_p[i][j] = pow_sum(val_id[i],j)-1;
        }
    }
    int now;
    //for(now=1;now<=p_sz_2;now++){
    for(now=1;p[now]<=n_6;now++){
        for(int j=val_id_num;j>=1;j--){
            ll w = val_id[j]/p[now];
            if(w<p[now]) break;
            ll val=1;
            for(int k = 0;k<=times;k++){
                f_p[j][k] = f_p[j][k] - val*(f_p[get_id(w)][k]-f_p[p[now
                    -1]][k]);
                val *= p[now];
            }
        }
    }
    int nnow = now;
    int val = 1;
    for(int tt = 0;tt<=times;tt++){
        now = nnow;
        memset(c,0,sizeof c);
        add(1,f_p[1][tt]);
        for(int i=2;val_id[i]<n/n_3;i++){
            add(i,f_p[i][tt] - f_p[i-1][tt]);
        }
        for(;p[now]<=n_3;now++){
            for(int j=val_id_num;j>=1;j--){
                ll w = val_id[j]/p[now];
                if(val_id[j]<n/n_3) break;
                if(w<p[now]) break;
                if(w<n/n_3) f_p[j][tt] = f_p[j][tt] - (sum(get_id(w)) -
                    sum(p[now-1]))*poww(p[now],tt);
```

```
            else f_p[j][tt] = f_p[j][tt] - (f_p[get_id(w)][tt]-sum(p[
                now-1]))*poww(p[now],tt);
         }
         update_bfs(now,tt);
      }
      for(int i=1;i<=val_id_num&&val_id[i]<n/n_3;i++)
      f_p[i][tt] = sum(i);
      for(;now<=p_sz_2;now++){
         for(int j=val_id_num;j>=1;j--){
            ll w = val_id[j]/p[now];
            if(val_id[j]<n/n_3) break;
            if(w<p[now]) break;
            f_p[j][tt] -= (f_p[get_id(w)][tt]-f_p[p[now-1]][tt])*poww(
                p[now],tt);
         }
      }
   }

   for(int i=1;i<=val_id_num;i++){
      ///if f(p) = p^2+3p+1,then write:f_p[i][0] = f_p[i][2] + 3*f_p[i
          ][1] + f_p[i][0];
      f_p[i][0] = f_p[i][2] + 3*f_p[i][1] + f_p[i][0];
   }

}
ll F[2000000+100];
void get_f_3(ll n){ ///V(F_{pi(n^(1/3))+1},n)
   ll q = p[p_sz_3+1];
   for(int now=1;now<=val_id_num;now++){
      if(val_id[now]<q){
         F[now] = 1;
      }
      else if(val_id[now]<q*q){
         F[now] = 1+(f_p[now][0]-f_p[q-1][0]);
      }
      else{
         F[now] = 1+(f_p[now][0]-f_p[q-1][0]);
         for(int pp=p_sz_3+1;p[pp]<=(int)(sqrt(val_id[now]))&&pp<=
             p_sz_2;pp++){
            F[now] += f(p[pp],2) + (f(p[pp],1))*(f_p[get_id(val_id[now
                ]/p[pp])][0]-f_p[get_id(p[pp])][0]);
         }
      }
   }
}
void get_f_6(ll n){ ///V(F_{pi(n^(1/6))+1},n)
   memset(c,0,sizeof c);
   add(1,F[1]);
   for(int i=2;val_id[i]<n/n_3;i++){
      add(i,F[i] - F[i-1]);
   }
   for(int k=p_sz_3;k>p_sz_6;k--){
      int now = val_id_num;
      for(;val_id[now]>=n/n_3;now--){
         int e = 1;
         ll _p = p[k];
         while(val_id[now]/_p){
            if(val_id[now]/_p>=n/n_3){
               F[now] += F[get_id(val_id[now]/_p)]*f(p[k],e);
            }
            else{
               F[now] += sum(get_id(val_id[now]/_p))*f(p[k],e);
            }
            _p *= p[k];
            e++;
         }
      }
      if(k==1) break;
      //cout << "******" << p[k] << "******" << n/n_3 << endl;
      update_bfs(k,-1);///bfs to update [lpf(i)==P{k-1}]f(i)
   }
   for(int i=1;i<=val_id_num&&val_id[i]<n/n_3;i++)
   F[i] = sum(i);
}
void get_f(ll n){
   for(int k=p_sz_6;k>=1;k--){
      for(int now = val_id_num;now>=1;now--){
         int e = 1;
         ll _p = p[k];
         while(val_id[now]/_p){
            F[now] += F[get_id(val_id[now]/_p)]*f(p[k],e);
            _p *= p[k];
            e++;
         }
      }
   }
}
int main(){//n = 1000000000; //1e10:455052511,0.83s/0.58s 1e12
    :37607912018 9.224s/5.105s
   cin >> n;
   init();
   get_f_p(n,2);
   get_f_3(n);
   get_f_6(n);
```

```
    get_f(n);
    for(int i=1;i<=val_id_num;i++){
        cout << val_id[i] << " : " << F[i] << endl;
    }
}
```

4.22 莫比乌斯函数

```
int prime[10000005];
int vis[10000005];
int mo[10000005];
int len=0;
void init()
{
    mo[1]=1;
    for(int i=2;i<=10000000;i++)
    {
        if(!vis[i])
        {
            prime[++len]=i;
            mo[i]=-1;
        }
        for(int j=1;;j++)
        {
            long long t=1ll*prime[j]*i;
            if(t>10000000)break;
            vis[t]=1;
            if(i%prime[j]==0){mo[t]=0;break;}
            else mo[t]=-mo[i];
        }
    }
}
```

4.23 快速数论变换

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

using u32 = unsigned;
using i64 = long long;
using u64 = unsigned long long;

constexpr int N = 2E5;

std::vector<int> minp, primes;

template<class T>
constexpr T power(T a, i64 b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) {
            res *= a;
        }
    }
    return res;
}

template<int P>
struct MInt {
    int x;
    constexpr MInt() : x{} {}
    constexpr MInt(i64 x) : x{norm(x % getMod())} {}

    static int Mod;
    constexpr static int getMod() {
        if (P > 0) {
            return P;
        } else {
            return Mod;
        }
    }
    constexpr static void setMod(int Mod_) {
        Mod = Mod_;
    }
    constexpr int norm(int x) const {
        if (x < 0) {
            x += getMod();
        }
        if (x >= getMod()) {
            x -= getMod();
        }
        return x;
    }
    constexpr int val() const {
        return x;
    }
    explicit constexpr operator int() const {
        return x;
    }
    constexpr MInt operator-() const {
        MInt res;
        res.x = norm(getMod() - x);
        return res;
```

```
    }
    constexpr MInt inv() const {
        assert(x != 0);
        return power(*this, getMod() - 2);
    }
    constexpr MInt &operator*=(MInt rhs) & {
        x = 1LL * x * rhs.x % getMod();
        return *this;
    }
    constexpr MInt &operator+=(MInt rhs) & {
        x = norm(x + rhs.x);
        return *this;
    }
    constexpr MInt &operator-=(MInt rhs) & {
        x = norm(x - rhs.x);
        return *this;
    }
    constexpr MInt &operator/=(MInt rhs) & {
        return *this *= rhs.inv();
    }
    friend constexpr MInt operator*(MInt lhs, MInt rhs) {
        MInt res = lhs;
        res *= rhs;
        return res;
    }
    friend constexpr MInt operator+(MInt lhs, MInt rhs) {
        MInt res = lhs;
        res += rhs;
        return res;
    }
    friend constexpr MInt operator-(MInt lhs, MInt rhs) {
        MInt res = lhs;
        res -= rhs;
        return res;
    }
    friend constexpr MInt operator/(MInt lhs, MInt rhs) {
        MInt res = lhs;
        res /= rhs;
        return res;
    }
    friend constexpr std::istream &operator>>(std::istream &is, MInt &a
        ) {
        i64 v;
        is >> v;
        a = MInt(v);
        return is;
    }
    friend constexpr std::ostream &operator<<(std::ostream &os, const
         MInt &a) {
        return os << a.val();
    }
    friend constexpr bool operator==(MInt lhs, MInt rhs) {
        return lhs.val() == rhs.val();
    }
    friend constexpr bool operator!=(MInt lhs, MInt rhs) {
        return lhs.val() != rhs.val();
    }
};

template<>
int MInt<0>::Mod = 1;

template<int V, int P>
constexpr MInt<P> CInv = MInt<P>(V).inv();

constexpr int P = 998244353;
using Z = MInt<P>;

std::vector<int> rev;
template<int P>
std::vector<MInt<P>> roots{0, 1};

template<int P>
constexpr MInt<P> findPrimitiveRoot() {
    MInt<P> i = 2;
    int k = __builtin_ctz(P - 1);
    while (true) {
        if (power(i, (P - 1) / 2) != 1) {
            break;
        }
        i += 1;
    }
    return power(i, (P - 1) >> k);
}

template<int P>
constexpr MInt<P> primitiveRoot = findPrimitiveRoot<P>();

template<>
constexpr MInt<998244353> primitiveRoot<998244353> {31};

template<int P>
constexpr void dft(std::vector<MInt<P>> &a) {
    int n = a.size();
```

```

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

    for (int i = 0; i < n; i++) {
        if (rev[i] < i) {
            std::swap(a[i], a[rev[i]]);
        }
    }
    if (roots<P>.size() < n) {
        int k = __builtin_ctz(roots<P>.size());
        roots<P>.resize(n);
        while ((1 << k) < n) {
            auto e = power(primitiveRoot<P>, 1 << (__builtin_ctz(P - 1) -
                 k - 1));
            for (int i = 1 << (k - 1); i < (1 << k); i++) {
                roots<P>[2 * i] = roots<P>[i];
                roots<P>[2 * i + 1] = roots<P>[i] * e;
            }
            k++;
        }
    }
    for (int k = 1; k < n; k *= 2) {
        for (int i = 0; i < n; i += 2 * k) {
            for (int j = 0; j < k; j++) {
                MInt<P> u = a[i + j];
                MInt<P> v = a[i + j + k] * roots<P>[k + j];
                a[i + j] = u + v;
                a[i + j + k] = u - v;
            }
        }
    }
}

template<int P>
constexpr void idft(std::vector<MInt<P>> &a) {
    int n = a.size();
    std::reverse(a.begin() + 1, a.end());
    dft(a);
    MInt<P> inv = (1 - P) / n;
    for (int i = 0; i < n; i++) {
        a[i] *= inv;
    }
}

template<int P = 998244353>
struct Poly : public std::vector<MInt<P>> {
    using Value = MInt<P>;

    Poly() : std::vector<Value>() {}
    explicit constexpr Poly(int n) : std::vector<Value>(n) {}

    explicit constexpr Poly(const std::vector<Value> &a) : std::vector<
        Value>(a) {}
    constexpr Poly(const std::initializer_list<Value> &a) : std::vector
        <Value>(a) {}

    template<class InputIt, class = std::_RequireInputIter<InputIt>>
    explicit constexpr Poly(InputIt first, InputIt last) : std::vector<
        Value>(first, last) {}

    template<class F>
    explicit constexpr Poly(int n, F f) : std::vector<Value>(n) {
        for (int i = 0; i < n; i++) {
            (*this)[i] = f(i);
        }
    }

    constexpr Poly shift(int k) const {
        if (k >= 0) {
            auto b = *this;
            b.insert(b.begin(), k, 0);
            return b;
        } else if (this->size() <= -k) {
            return Poly();
        } else {
            return Poly(this->begin() + (-k), this->end());
        }
    }
    constexpr Poly trunc(int k) const {
        Poly f = *this;
        f.resize(k);
        return f;
    }
    constexpr friend Poly operator+(const Poly &a, const Poly &b) {
        Poly res(std::max(a.size(), b.size()));
        for (int i = 0; i < a.size(); i++) {
            res[i] += a[i];
        }
```

```
    for (int i = 0; i < b.size(); i++) {
      res[i] += b[i];
    }
    return res;
  }
  constexpr friend Poly operator-(const Poly &a, const Poly &b) {
    Poly res(std::max(a.size(), b.size()));
    for (int i = 0; i < a.size(); i++) {
      res[i] += a[i];
    }
    for (int i = 0; i < b.size(); i++) {
      res[i] -= b[i];
    }
    return res;
  }
  constexpr friend Poly operator-(const Poly &a) {
    std::vector<Value> res(a.size());
    for (int i = 0; i < int(res.size()); i++) {
      res[i] = -a[i];
    }
    return Poly(res);
  }
  constexpr friend Poly operator*(Poly a, Poly b) {
    if (a.size() == 0 || b.size() == 0) {
      return Poly();
    }
    if (a.size() < b.size()) {
      std::swap(a, b);
    }
    int n = 1, tot = a.size() + b.size() - 1;
    while (n < tot) {
      n *= 2;
    }
    if (((P - 1) & (n - 1)) != 0 || b.size() < 128) {
      Poly c(a.size() + b.size() - 1);
      for (int i = 0; i < a.size(); i++) {
        for (int j = 0; j < b.size(); j++) {
          c[i + j] += a[i] * b[j];
        }
      }
      return c;
    }
    a.resize(n);
    b.resize(n);
    dft(a);
    dft(b);
    for (int i = 0; i < n; ++i) {
      a[i] *= b[i];
    }
    idft(a);
    a.resize(tot);
    return a;
  }
  constexpr friend Poly operator*(Value a, Poly b) {
    for (int i = 0; i < int(b.size()); i++) {
      b[i] *= a;
    }
    return b;
  }
  constexpr friend Poly operator*(Poly a, Value b) {
    for (int i = 0; i < int(a.size()); i++) {
      a[i] *= b;
    }
    return a;
  }
  constexpr friend Poly operator/(Poly a, Value b) {
    for (int i = 0; i < int(a.size()); i++) {
      a[i] /= b;
    }
    return a;
  }
  constexpr Poly &operator+=(Poly b) {
    return (*this) = (*this) + b;
  }
  constexpr Poly &operator-=(Poly b) {
    return (*this) = (*this) - b;
  }
  constexpr Poly &operator*=(Poly b) {
    return (*this) = (*this) * b;
  }
  constexpr Poly &operator*=(Value b) {
    return (*this) = (*this) * b;
  }
  constexpr Poly &operator/=(Value b) {
    return (*this) = (*this) / b;
  }
  constexpr Poly deriv() const {
    if (this->empty()) {
      return Poly();
    }
    Poly res(this->size() - 1);
    for (int i = 0; i < this->size() - 1; ++i) {
      res[i] = (i + 1) * (*this)[i + 1];
    }
```

```
333         return res;
334     }
335     constexpr Poly integr() const {
336         Poly res(this->size() + 1);
337         for (int i = 0; i < this->size(); ++i) {
338             res[i + 1] = (*this)[i] / (i + 1);
339         }
340         return res;
341     }
342     constexpr Poly inv(int m) const {
343         Poly x{(*this)[0].inv()};
344         int k = 1;
345         while (k < m) {
346             k *= 2;
347             x = (x * (Poly{2} - trunc(k) * x)).trunc(k);
348         }
349         return x.trunc(m);
350     }
351     constexpr Poly log(int m) const {
352         return (deriv() * inv(m)).integr().trunc(m);
353     }
354     constexpr Poly exp(int m) const {
355         Poly x{1};
356         int k = 1;
357         while (k < m) {
358             k *= 2;
359             x = (x * (Poly{1} - x.log(k) + trunc(k))).trunc(k);
360         }
361         return x.trunc(m);
362     }
363     constexpr Poly pow(int k, int m) const {
364         int i = 0;
365         while (i < this->size() && (*this)[i] == 0) {
366             i++;
367         }
368         if (i == this->size() || 1LL * i * k >= m) {
369             return Poly(m);
370         }
371         Value v = (*this)[i];
372         auto f = shift(-i) * v.inv();
373         return (f.log(m - i * k) * k).exp(m - i * k).shift(i * k) *
                power(v, k);
374     }
375     constexpr Poly sqrt(int m) const {
376         Poly x{1};
377         int k = 1;
378         while (k < m) {
379             k *= 2;
380             x = (x + (trunc(k) * x.inv(k)).trunc(k)) * CInv<2, P>;
381         }
382         return x.trunc(m);
383     }
384     constexpr Poly mulT(Poly b) const {
385         if (b.size() == 0) {
386             return Poly();
387         }
388         int n = b.size();
389         std::reverse(b.begin(), b.end());
390         return ((*this) * b).shift(-(n - 1));
391     }
392     constexpr std::vector<Value> eval(std::vector<Value> x) const {
393         if (this->size() == 0) {
394             return std::vector<Value>(x.size(), 0);
395         }
396         const int n = std::max(x.size(), this->size());
397         std::vector<Poly> q(4 * n);
398         std::vector<Value> ans(x.size());
399         x.resize(n);
400         std::function<void(int, int, int)> build = [&](int p, int l, int
                r) {
401             if (r - l == 1) {
402                 q[p] = Poly{1, -x[l]};
403             } else {
404                 int m = (l + r) / 2;
405                 build(2 * p, l, m);
406                 build(2 * p + 1, m, r);
407                 q[p] = q[2 * p] * q[2 * p + 1];
408             }
409         };
410         build(1, 0, n);
411         std::function<void(int, int, int, const Poly &)> work = [&](int
                p, int l, int r, const Poly &num) {
412             if (r - l == 1) {
413                 if (l < int(ans.size())) {
414                     ans[l] = num[0];
415                 }
416             } else {
417                 int m = (l + r) / 2;
418                 work(2 * p, l, m, num.mulT(q[2 * p + 1]).trunc(m - l));
419                 work(2 * p + 1, m, r, num.mulT(q[2 * p]).trunc(r - m));
420             }
421         };
422         work(1, 0, n, mulT(q[1].inv(n)));
423         return ans;
```

```
    }
};

template<int P = 998244353>
Poly<P> berlekampMassey(const Poly<P> &s) {
    Poly<P> c;
    Poly<P> oldC;
    int f = -1;
    for (int i = 0; i < s.size(); i++) {
        auto delta = s[i];
        for (int j = 1; j <= c.size(); j++) {
            delta -= c[j - 1] * s[i - j];
        }
        if (delta == 0) {
            continue;
        }
        if (f == -1) {
            c.resize(i + 1);
            f = i;
        } else {
            auto d = oldC;
            d *= -1;
            d.insert(d.begin(), 1);
            MInt<P> df1 = 0;
            for (int j = 1; j <= d.size(); j++) {
                df1 += d[j - 1] * s[f + 1 - j];
            }
            assert(df1 != 0);
            auto coef = delta / df1;
            d *= coef;
            Poly<P> zeros(i - f - 1);
            zeros.insert(zeros.end(), d.begin(), d.end());
            d = zeros;
            auto temp = c;
            c += d;
            if (i - temp.size() > f - oldC.size()) {
                oldC = temp;
                f = i;
            }
        }
    }
    c *= -1;
    c.insert(c.begin(), 1);
    return c;
}


template<int P = 998244353>
MInt<P> linearRecurrence(Poly<P> p, Poly<P> q, i64 n) {
    int m = q.size() - 1;
    while (n > 0) {
        auto newq = q;
        for (int i = 1; i <= m; i += 2) {
            newq[i] *= -1;
        }
        auto newp = p * newq;
        newq = q * newq;
        for (int i = 0; i < m; i++) {
            p[i] = newp[i * 2 + n % 2];
        }
        for (int i = 0; i <= m; i++) {
            q[i] = newq[i * 2];
        }
        n /= 2;
    }
    return p[0] / q[0];
}

void sieve(int n) {
    minp.assign(n + 1, 0);
    primes.clear();

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

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

int f[N + 1];

void solve() {
    int n;
```

```
    std::cin >> n;

    std::vector<int> a(n);
    for (int i = 0; i < n; i++) {
        std::cin >> a[i];
    }

    bool same = true;
    for (int i = 1; i < n; i++) {
        if (f[a[i]] != f[a[0]]) {
            same = false;
        }
    }
    if (same && (f[a[0]] == 0 || n % 2 == 1)) {
        std::cout << "Bob\n";
    } else {
        std::cout << "Alice\n";
    }
}

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

    sieve(N);

    for (int i = 5; i <= N; i += 2) {
        f[i] = minp[i - 2] == i - 2 ? 1 ^ f[i - 2] : 0;
    }

    Poly<P> p[2];
    p[0].resize(N + 1);
    p[1].resize(N + 1);
    f[4] = 1;
    for (int i = 3; i <= N; i += 2) {
        if (minp[i] == i) {
            p[f[i]][i] = 1;
        }
    }
    auto a = p[0] * p[0];
    auto b = p[1] * p[1];
    for (int i = 6; i <= N; i += 2) {
        if (a[i] != 0 && b[i] != 0) {
            f[i] = 3;
        } else if (a[i] != 0) {
            f[i] = 1;
        }
    }
    }

    int t;
    std::cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}
```

4.24 FWT

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

//is=+1 : fwt
//is=-1 : ifwt

//basically, we need +,-,* for class T
// for xor_fwt and xor_convolution, we need /2 (or change it to
    *2^(-1) manually)

template <class T>
void and_fwt(T f[],int ldn,int is=1)
{
    int n=(1<<ldn);
    for(int ldm=1;ldm<=ldn;ldm++)
    {
        int m=(1<<ldm),mh=m>>1;
        for(int r=0;r<n;r+=m)
        {
            int t1=r,t2=r+mh;
            for(int j=0;j<mh;j++,t1++,t2++)
            {
                T u=f[t1],v=f[t2];
                if(is>0)f[t1]=u+v,f[t2]=v;
                else f[t1]=u-v,f[t2]=v;
            }
        }
    }
}

template <class T>
void or_fwt(T f[],int ldn,int is=1)
```

```
{
    int n=(1<<ldn);
    for(int ldm=1;ldm<=ldn;ldm++)
    {
        int m=(1<<ldm),mh=m>>1;
        for(int r=0;r<n;r+=m)
        {
            int t1=r,t2=r+mh;
            for(int j=0;j<mh;j++,t1++,t2++)
            {
                T u=f[t1],v=f[t2];
                if(is>0)f[t1]=u,f[t2]=u+v;
                else f[t1]=u,f[t2]=v-u;
            }
        }
    }
}

template <class T>
void xor_fwt(T f[],int ldn,int is=1)
{
    int n=(1<<ldn);
    for(int ldm=1;ldm<=ldn;ldm++)
    {
        int m=(1<<ldm),mh=m>>1;
        for(int r=0;r<n;r+=m)
        {
            int t1=r,t2=r+mh;
            for(int j=0;j<mh;j++,t1++,t2++)
            {
                T u=f[t1],v=f[t2];
                if(is>0)f[t1]=u+v,f[t2]=u-v;
                else f[t1]=(u+v)/2,f[t2]=(u-v)/2;
            }
        }
    }
}

// will **change** f[] and g[], result is stored in f[]
template <class T>
void and_convolution(T f[],T g[],int n)
{
    int ldn=__lg(n);assert((1<<ldn)==n);
    and_fwt(f,ldn);and_fwt(g,ldn);
    for(int i=0;i<n;i++)f[i]=f[i]*g[i];
    and_fwt(f,ldn,-1);
}

template <class T>
void or_convolution(T f[],T g[],int n)
{
    int ldn=__lg(n);assert((1<<ldn)==n);
    or_fwt(f,ldn);or_fwt(g,ldn);
    for(int i=0;i<n;i++)f[i]=f[i]*g[i];
    or_fwt(f,ldn,-1);
}

template <class T>
void xor_convolution(T f[],T g[],int n)
{
    int ldn=__lg(n);assert((1<<ldn)==n);
    xor_fwt(f,ldn);xor_fwt(g,ldn);
    for(int i=0;i<n;i++)f[i]=f[i]*g[i];
    xor_fwt(f,ldn,-1);
}
```

4.25 BM

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(
    b&1)res=res*a%mod;a=a*a%mod;}return res;}
// head

ll n;
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
```

```
    rep(i,0,k+k) _c[i]=0;
    rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
    for (int i=k+k-1;i>=k;i--) if (_c[i])
    rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%
        mod;
    rep(i,0,k) a[i]=_c[i];
  }
  int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+ ...
    ll ans=0,pnt=0;
    int k=SZ(a);
    assert(SZ(a)==SZ(b));
    rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
    Md.clear();
    rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
    rep(i,0,k) res[i]=base[i]=0;
    res[0]=1;
    while ((1ll<<pnt)<=n) pnt++;
    for (int p=pnt;p>=0;p--) {
      mul(res,res,k);
      if ((n>>p)&1) {
        for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
        rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%
            mod;
      }
    }
    rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
    if (ans<0) ans+=mod;
    return ans;
  }
  VI BM(VI s) {
    VI C(1,1),B(1,1);
    int L=0,m=1,b=1;
    rep(n,0,SZ(s)) {
      ll d=0;
      rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
      if (d==0) ++m;
      else if (2*L<=n) {
        VI T=C;
        ll c=mod-d*powmod(b,mod-2)%mod;
        while (SZ(C)<SZ(B)+m) C.pb(0);
        rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
        L=n+1-L; B=T; b=d; m=1;
      } else {
        ll c=mod-d*powmod(b,mod-2)%mod;
        while (SZ(C)<SZ(B)+m) C.pb(0);
        rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
        ++m;
      }
    }
    return C;
  }
  int gao(VI a,ll n) {
    VI c=BM(a);
    c.erase(c.begin());
    rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
    return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
  }
};

int main() {
  /*push_back 进去前 8~10 项左右、最后调用 gao 得第 n 项*/
  vector<int>v;
  v.push_back(3);
  v.push_back(9);
  v.push_back(20);
  v.push_back(46);
  v.push_back(106);
  v.push_back(244);
  v.push_back(560);
  v.push_back(1286);
  v.push_back(2956);
  v.push_back(6794);
  int nCase;
  scanf("%d", &nCase);
  while(nCase--){
    scanf("%lld", &n);
    printf("%lld\n",1LL * linear_seq::gao(v,n-1) % mod); ///求第n项
  }
}
```

4.26 矩阵求逆

```
#include<bits/stdc++.h>
#define int64 long long
using namespace std;
const int64 mod=1e9+7;
int64 a[410][410];
int n,is[410],js[410];
void exgcd(int a,int b,int &x,int &y){
  if(!b)return x=1,y=0,void();
  exgcd(b,a%b,y,x);y-=x*(a/b);
}
int inv(int p){
```

```
    int x,y;exgcd(p,mod,x,y);
    return (x+mod)%mod;
}
void inv(){
    for(int k=1;k<=n;k++){
        for(int i=k;i<=n;i++) // 1
        for(int j=k;j<=n;j++)if(a[i][j]){
            is[k]=i,js[k]=j;break;
        }
        for(int i=1;i<=n;i++) // 2
        swap(a[k][i],a[is[k]][i]);
        for(int i=1;i<=n;i++)
        swap(a[i][k],a[i][js[k]]);
        if(!a[k][k]){
            puts("No Solution");
            exit(0);
        }
        a[k][k]=inv(a[k][k]); // 3
        for(int j=1;j<=n;j++)if(j!=k) // 4
        (a[k][j]*=a[k][k])%=mod;
        for(int i=1;i<=n;i++)if(i!=k) // 5
        for(int j=1;j<=n;j++)if(j!=k)
        (a[i][j]+=mod-a[i][k]*a[k][j]%mod)%=mod;
        for(int i=1;i<=n;i++)if(i!=k) // 就是这里不同
        a[i][k]=(mod-a[i][k]*a[k][k]%mod)%mod;
    }
    for(int k=n;k;k--){ // 6
        for(int i=1;i<=n;i++)
        swap(a[js[k]][i],a[k][i]);
        for(int i=1;i<=n;i++)
        swap(a[i][is[k]],a[i][k]);
    }
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
    scanf("%lld",a[i]+j);
    inv();
    for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
    printf("%lld%c",a[i][j],j==n?'\n':' ');
    return 0;
}
```

4.27 矩阵乘法 & 快速幂

```
template<typename T>
struct matrix
{
    int siz;
    vector<vector<T>>a;
    matrix(int _siz)
    {
        siz=_siz;
        a=vector(siz,vector<T>(siz));
    }
    void resize(int _siz)
    {
        siz=_siz;
        a=vector(siz,vector<T>(siz));
    }
    vector<T>& operator[](int x)
    {
        assert(x>=0&&x<siz);
        return a[x];
    }
    matrix<T> operator*(matrix<T>&b)
    {
        assert(siz==b.siz);
        matrix<T> res(siz);
        for(int i=0;i<siz;i++)
        {
            for(int j=0;j<siz;j++)
            {
                for(int k=0;k<siz;k++)
                {
                    res[i][j]+=a[i][k]*b[k][j];
                }
            }
        }
        return res;
    }
    friend matrix<T> operator^(matrix<T> a,int n)
    {
        matrix<T> res(a.siz);
        for(int i=0;i<a.siz;i++)res[i][i]=1;
        while(n)
        {
            if(n&1)res=res*a;
            a=a*a;n>>=1;
        }
```

```
        return res;
    }
};
```

4.28 整除分块

```
//向上取整
for(int l=1;l<=n;l++)
{
    int k=(n+l-1)/l;
    int r=(n-1)/(k-1);
    solve();
    l=r;
}
```

4.29 自适应辛普森积分

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{6}\left[f(a)+4\times f\left(\frac{a+b}{2}\right)+f(b)\right]$$

```
long double simpson(long double a, long double b) {
    return (f(a) + 4 * f((a + b) / 2) + f(b)) * (b - a) / 6;
}

long double integral(long double L, long double R, long double eps) {
    long double mid = (L + R) / 2;
    long double S_total = simpson(L, R);
    long double S_left = simpson(L, mid);
    long double S_right = simpson(mid, R);
    long double delta = S_left + S_right - S_total;
    if (fabs(delta) < 15 * eps)
    return S_left + S_right + delta / 15;
    return integral(L, mid, eps / 2) + integral(mid, R, eps / 2);
}
```

4.30 最小二乘法

给定 n 个点, 求一条直线 $y=kx+b$ 最匹配 (写出方差式子求偏导)

$$\begin{cases}\Sigma xk+nb-\Sigma y=0\\ \Sigma x^2k+\Sigma xb-\Sigma xy=0\end{cases}\rightarrow\begin{cases}k=\dfrac{-\Sigma x\Sigma y+n\Sigma xy}{\Sigma x\Sigma x-\Sigma x^2n}\\ b=\dfrac{-\Sigma x^2\Sigma y+\Sigma x\Sigma xy}{\Sigma x^2n-\Sigma x\Sigma x}\end{cases}$$

```
void ols(double n, double*x, double*y,double&k,double&b) {//ordinary
    least squares
    double sumx = 0, sumy = 0, sumxx = 0, sumxy = 0;
    for(int i=1;i<=n;++i){
        sumx += x[i];
        sumy += y[i];
        sumxx += x[i] * x[i];
        sumxy += x[i] * y[i];
    }
    k = (sumx * sumy - n * sumxy) / (sumx * sumx - sumxx * n);
    b = -(sumxx * sumy - sumx * sumxy) / (sumx * sumx - sumxx * n);
}
double variance(double* x, double* y, double& k, double& b) {
    double res = 0;
    for (int i = 1; i <= n; ++i) {
        res += (k * x[i] + b - y[i]) * (k * x[i] + b - y[i]);
    }
    return res;
}
```

4.31 高精度

```
struct Int {
    int sign;
    std::vector<int> v;

    Int() : sign(1) {}
    Int(const std::string &s) { *this = s; }
    Int(int v) {
        char buf[21];
        sprintf(buf, "%d", v);
        *this = buf;
    }
    void zip(int unzip) {
        if (unzip == 0) {
            for (int i = 0; i < (int)v.size(); i++)
            v[i] = get_pos(i * 4) + get_pos(i * 4 + 1) * 10 + get_pos(i *
                4 + 2) * 100 + get_pos(i * 4 + 3) * 1000;
        } else
        for (int i = (v.resize(v.size() * 4), (int)v.size() - 1), a; i
            >= 0; i--)
        a = (i % 4 >= 2) ? v[i / 4] / 100 : v[i / 4] % 100, v[i] = (i &
            1) ? a / 10 : a % 10;
        setsign(1, 1);
    }
```

```
21    int get_pos(unsigned pos) const { return pos >= v.size() ? 0 : v[
          pos]; }
22    Int &setsign(int newsign, int rev) {
23       for (int i = (int)v.size() - 1; i > 0 && v[i] == 0; i--)
24       v.erase(v.begin() + i);
25       sign = (v.size() == 0 || (v.size() == 1 && v[0] == 0)) ? 1 : (
             rev ? newsign * sign : newsign);
26       return *this;
27    }
28    std::string to_str() const {
29       Int b = *this;
30       std::string s;
31       for (int i = (b.zip(1), 0); i < (int)b.v.size(); ++i)
32       s += char(*(b.v.rbegin() + i) + '0');
33       return (sign < 0 ? "-" : "") + (s.empty() ? std::string("0") : s
             );
34    }
35    bool absless(const Int &b) const {
36       if (v.size() != b.v.size()) return v.size() < b.v.size();
37       for (int i = (int)v.size() - 1; i >= 0; i--)
38       if (v[i] != b.v[i]) return v[i] < b.v[i];
39       return false;
40    }
41    Int operator-() const {
42       Int c = *this;
43       c.sign = (v.size() > 1 || v[0]) ? -c.sign : 1;
44       return c;
45    }
46    Int &operator=(const std::string &s) {
47       if (s[0] == '-')
48       *this = s.substr(1);
49       else {
50          for (int i = (v.clear(), 0); i < (int)s.size(); ++i)
51          v.push_back(*(s.rbegin() + i) - '0');
52          zip(0);
53       }
54       return setsign(s[0] == '-' ? -1 : 1, sign = 1);
55    }
56    bool operator<(const Int &b) const {
57       return sign != b.sign ? sign < b.sign : (sign == 1 ? absless(b)
             : b.absless(*this));
58    }
59    bool operator==(const Int &b) const { return v == b.v && sign == b.
          sign; }
60    Int &operator+=(const Int &b) {
61       if (sign != b.sign) return *this = (*this) - -b;
62       v.resize(std::max(v.size(), b.v.size()) + 1);
63       for (int i = 0, carry = 0; i < (int)b.v.size() || carry; i++) {
64          carry += v[i] + b.get_pos(i);
65          v[i] = carry % 10000, carry /= 10000;
66       }
67       return setsign(sign, 0);
68    }
69    Int operator+(const Int &b) const {
70       Int c = *this;
71       return c += b;
72    }
73    void add_mul(const Int &b, int mul) {
74       v.resize(std::max(v.size(), b.v.size()) + 2);
75       for (int i = 0, carry = 0; i < (int)b.v.size() || carry; i++) {
76          carry += v[i] + b.get_pos(i) * mul;
77          v[i] = carry % 10000, carry /= 10000;
78       }
79    }
80    Int operator-(const Int &b) const {
81       if (b.v.empty() || b.v.size() == 1 && b.v[0] == 0) return *this;
82       if (sign != b.sign) return (*this) + -b;
83       if (absless(b)) return -(b - *this);
84       Int c;
85       for (int i = 0, borrow = 0; i < (int)v.size(); i++) {
86          borrow += v[i] - b.get_pos(i);
87          c.v.push_back(borrow);
88          c.v.back() -= 10000 * (borrow >>= 31);
89       }
90       return c.setsign(sign, 0);
91    }
92    Int operator*(const Int &b) const {
93       if (b < *this) return b * *this;
94       Int c, d = b;
95       for (int i = 0; i < (int)v.size(); i++, d.v.insert(d.v.begin(),
             0))
96       c.add_mul(d, v[i]);
97       return c.setsign(sign * b.sign, 0);
98    }
99    Int operator/(const Int &b) const {
100      Int c, d;
101      d.v.resize(v.size());
102      double db = 1.0 / (b.v.back() + (b.get_pos((unsigned)b.v.size()
             - 2) / 1e4) +
103      (b.get_pos((unsigned)b.v.size() - 3) + 1) / 1e8);
104      for (int i = (int)v.size() - 1; i >= 0; i--) {
105         c.v.insert(c.v.begin(), v[i]);
106         int m = (int)((c.get_pos((int)b.v.size()) * 10000 + c.get_pos
                ((int)b.v.size() - 1)) * db);
```

```
        c = c - b * m, c.setsign(c.sign, 0), d.v[i] += m;
        while (!(c < b))
        c = c - b, d.v[i] += 1;
      }
      return d.setsign(sign * b.sign, 0);
    }
    Int operator%(const Int &b) const { return *this - *this / b * b; }
    bool operator>(const Int &b) const { return b < *this; }
    bool operator<=(const Int &b) const { return !(b < *this); }
    bool operator>=(const Int &b) const { return !(*this < b); }
    bool operator!=(const Int &b) const { return !(*this == b); }
};
```

5 Dynamic Programming

5.1 最长公共子序列

```
if(a[i]==b[j])dp[i][j]=dp[i-1][j-1]+1;
else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
```

5.2 背包

5.2.1 01 背包

```
//一个旅行者有一个最多能装M公斤的背包，现在有n件物品，它们的重量分别是W1，W2
    ......Wn；它们的价格为C1，C2......Cn，求旅行者能获得的最大总价值.

#include<bits/stdc++.h>
using namespace std;
int w[1003],c[1003],f[1003];
int n,m;
int main()
{
   scanf("%d%d",&n,&m);
   for(int i = 1;i <= n;i++)
   {
      scanf("%d%d",&w[i],&c[i]);
      for(int j = m;j >= w[i];j--)
      {
         f[j] = max(f[j],f[j - w[i]] + c[i]);
      }
   }
   printf("%d",f[m]);
   return 0;
}
```

5.2.2 完全背包

```
//有N种物品和一个容量为V的背包，每种物品都有无限件可以用。第i种物品的的费用是w[i]
    ，价值是c[i]。求解将哪些物品装入背包的费用总和不超过背包容量，且价值总和最大。

#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
int m,n;
int w[1003],c[1003],f[1003];
```

```
int main()
{
    scanf("%d%d",&n,&m);
    for(int i = 1;i <= n;i++)
    {
        scanf("%d%d",&w[i],&c[i]);
        for(int j = w[i];j <= m;j++)//01背包是倒序，完全背包是顺序
        {
            f[j] = max(f[j],f[j - w[i]] + c[i]);
        }
    }
    printf("%d",f[m]);
    return 0;
}
```

5.2.3 多重背包

```
//有N种物品和一个容量为V的背包。第i种物品最多有n[i]，每件费用是w[i]，价值是c[i]。求解将哪些物品放入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
int m,n;
int w[1003],c[1003],s[1003],f[1003];
int main()
{
    scanf("%d%d",&n,&m);
    for(int i = 1;i <= n;i++)
    {
        scanf("%d%d%d",&w[i],&c[i],&s[i]);
    }
    for(int i = 1;i <= n;i++)
    {
        for(int j = m;j >= 0;j--)
        {
            for(int k = 0;k <= s[i];k++)
            {
                if(j - k * w[i] < 0)break;
                f[j] = max(f[j],f[j - k * w[i]] + k * c[i]);
            }
        }
    }
    printf("%d",f[m]);
    return 0;
}
```

5.2.4 多种背包混合问题

```
//一个旅行者有一个最多能装V公斤的背包，现在有n件物品，已知它们的价值和所占体积。有的物品可以一次取一件，有的物品可以取无限次，有的物品可以取得个数有上限。求最大价值。

//无法理解出题人为什么非要把这三个背包放在一起，有毒吧。。。

#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
int m,n;
int w[1003],c[1003],s[1003],f[1003];
int main()
{
    scanf("%d%d",&n,&m);
    for(int i = 1;i <= n;i++)
    {
        scanf("%d%d%d",&w[i],&c[i],&s[i]);
    }
    for(int i = 1;i <= n;i++)
    {
        if(s[i] == 0)
        {
            for(int j = w[i];j <= m;j++)
            {
                f[j] = max(f[j],f[j - w[i]] + c[i]);
            }
        }
        else{
            for(int j = 1;j <= s[i];j++)
            {
                for(int k = m;k >= w[i];k--)
                f[k] = max(f[k],f[k - w[i]] + c[i]);
            }
        }
    }
    printf("%d",f[m]);
    return 0;
}
```

5.2.5 二维费用的背包问题

```cpp
//小橙书上关于这个讲了一大堆，大意就是限制条件有之前的一个变成了两个，就是由之前得背包重量，变成了现在的氢气和氧气的含量。

//例题：潜水员

//潜水员有一个带2种气体的气缸：一个为氧气，一个为氮气。他有一定数量的气缸。每个气缸都有重量和气体容量。潜水员为了完成他的工作需要特定数量的氧和氮。他完成工作所需气缸的总重的最低限度是多少

#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
//啊啊啊de了一万年bug终于过了
int u,v,k;
int f[101][101];
int a[1001],b[1001],c[1001];
int main()
{
    memset(f,10001,sizeof(f));//要把数组初始化为一个很大的数
    f[0][0] = 0;
    scanf("%d%d%d",&v,&u,&k);
    for(int i = 1;i <= k;i++)
    {
        scanf("%d%d%d",&a[i],&b[i],&c[i]);
    }
    for(int i = 1;i <= k;i++)
    {
        for(int j = v;j >= 0;j--)
        {
            for(int x = u;x >= 0;x--)
            {
                int t1 = j + a[i],t2 = x + b[i];//设一个变量方便计算
                if(t1 > v)t1 = v;//如果超出需求量就用需求量代替
                if(t2 > u)t2 = u;//这样就可以用f[v][u]来表示最优解了
                f[t1][t2] = min(f[t1][t2],f[j][x] + c[i]);
            }
        }
    }
    printf("%d",f[v][u]);
    return 0;
}
/*
5 60
5
3 36 120
10 25 129
5 50 250
1 45 130
4 20 119
*/
//正解是最后输出249
```

5.2.6 分组的背包问题

```cpp
//有完没完，难道我真的要把自己的大好人生都用来学习背包吗！！！

//问题：

//有N件物品和一个容量为V的背包。第i件物品的费用是w[ i ],价值是c[ i ]。这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。求解将哪些物品放入背包中可使这鞋物品的费用总和不超过背包容量，且价值总和最大

//对每一组物品，可以选择其中的一件或者一件都不选

#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
//这段代码简直有毒，我照着小橙书一个字一个字地敲得，结果竟然过不了样例
int v,n,t;
int w[33],c[33];
int a[13][33],dp[203];
int main()
{
    scanf("%d%d%d",&v,&n,&t);
    for(int i = 1;i <= n;i++)
    {
        int p;
        scanf("%d%d%d",&w[i],&c[i],&p);
        a[p][++a[p][0]] = i;//记录下来它属于哪个组
    }
    for(int x = 1;x <= t;x++)
    {
        for(int j = v;j >= 0;j--)
        {
            for(int i = 1;i <= a[x][0];i++)
            {
                if(j >= w[a[x][i]])
```

```
            {
                int q = a[x][i];
                dp[j] = max(dp[j - w[q]] + c[q],dp[j]);
            }
        }
    }
}
printf("%d",dp[v]);
return 0;
}
/*
10 6 3
2 1 1
3 3 1
4 8 2
6 9 2
2 8 3
3 9 3
*/

//输出20
```

5.2.7 有依赖的背包问题

```
/*
这种背包问题的物品间存在某种依赖关系。若i依附于j，表示若选物品i，则必须选物品j。对每个主见i的附件集合先进行一次01背包，将主件转化为一个物品组，然后就可以按照分组背包来进行计算了。
详情见金明的预算方案https://blog.csdn.net/qq_42914224/article/details/82594657
题目描述
金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过N元钱就行”。今天一早，金明就开始做预算了，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：
主件 附件
电脑 打印机，扫描仪
书柜 图书
书桌 台灯，文具
工作椅 无
如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有0个、1个或2个附件。附件不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一个重要度，分为5等：用整数1～5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是10元的整数倍）。他希望在不超过N元（可以等于N元）的前提下，使每件物品的价格与重要度的乘积的总和最大。
设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为j1,j2,…,jk，则所求的总和为：
v[j1]×w[j1]+v[j2]×w[j2]+…+v[jk]×w[jk]。
请你帮助金明设计一个满足要求的购物单。
输入输出格式
输入格式：
第1行，为两个正整数，用一个空格隔开：
Nm
（其中N(<32000)表示总钱数，
m(<60)为希望购买物品的个数。） 从第2行到第
m+1行，第j行给出了编号为j－1的物品的基本数据，每行有3个非负整数vpq（其中v表示该物品的价格（v<10000），p表示该物品的重要度（1～5），q表示该物品是主件还是附件。如果q=0，表示该物品为主件，如果q>0，表示该物品为附件，q是所属主件的编号）

输出格式：

一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（<200000）。

输入输出样例
输入样例#1：
1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0
输出样例#1：
2200

对于每个主件和两个附件
可能存在五种情况
只选主件，主件和一个附件，主件和另一个附件，主件和两个附件，全都不选。
所以就枚举每个主件，然后对于这个主件比较五种情况的大小
*/

#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
int n,m;
int c[63][63];
//昨晚没过就是因为数组的第二维开小了
int v[63],p[63],dp[32003];
int main()
{
    scanf("%d%d",&n,&m);
```

```
    n = n / 10;
    for(int i = 1;i <= m;i++)
    {
        int q;
        scanf("%d%d%d",&v[i],&p[i],&q);
        v[i] = v[i] / 10;
        p[i] *= v[i];
        c[q][++c[q][0]] = i;
        //主件为q的附件的个数（q = 0时，还能智能地记录主件个数）
    }
    for(int i = 1;i <= c[0][0];i++)//枚举主件
    {
        for(int j = n;j >= v[i];j--)
        {
            int v1 = (j - v[c[0][i]]) >= 0?dp[j - v[c[0][i]]] + p[c[0][i
                ]]:0;
            int v2 = (j - v[c[0][i]] - v[c[c[0][i]][1]]) >= 0?dp[j - v[c
                [0][i]] - v[c[c[0][i]][1]]] + p[c[0][i]] + p[c[c[0][i
                ]][1]]:0;
            int v3 = (j - v[c[0][i]] - v[c[c[0][i]][2]]) >= 0?dp[j - v[c
                [0][i]] - v[c[c[0][i]][2]]] + p[c[0][i]] + p[c[c[0][i
                ]][2]]:0;
            int v4 = (j - v[c[0][i]] - v[c[c[0][i]][1]] - v[c[c[0][i
                ]][2]]) >= 0?dp[j - v[c[0][i]] - v[c[c[0][i]][1]] - v[c[c
                [0][i]][2]]] + p[c[0][i]] + p[c[c[0][i]][1]] + p[c[c[0][i
                ]][2]]:0;
            dp[j] = max(dp[j],max(v1,max(v2,max(v3,v4))));
        }
    }
    printf("%d",dp[n] * 10);
    return 0;
}
```

5.3 状压 dp

```
#include<bits/stdc++.h>
using namespace std;
const int INF=1e9;
int mp[50][50];
struct node
{
    int now,set;
};
long long dp[21][1<<21];
bool vis[21][1<<21];
///设dp状态i,j表示当前在i城市，已经访问的城市集合为j
queue<node>q;
int main()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)mp[i][j]=INF;

    for(int i=1;i<=n;i++)
    for(int j=0;j<=1<<n;j++)dp[i][j]=INF;

    int m;
    scanf("%d",&m);
    for(int i=1;i<=m;i++)
    {
        int ta,tb,tc;
        scanf("%d%d%d",&ta,&tb,&tc);
        mp[ta][tb]=min(mp[ta][tb],tc);
        mp[tb][ta]=min(mp[tb][ta],tc);
    }
    dp[1][0]=0;
    vis[1][0]=1;
    q.push({1,0});
    while(!q.empty())
    {
        node p=q.front();
        q.pop();
        int now=p.now,set=p.set;
        vis[now][set]=0;
        for(int i=1;i<=n;i++)
        {
            if((set&(1<<(i-1)))||i==now);
            else
            {
                if(dp[i][set|(1<<(i-1))]>dp[now][set]+mp[now][i])
                {
                    dp[i][set|(1<<(i-1))]=dp[now][set]+mp[now][i];

                    if (vis[i][set|(1<<(i-1))]==0)
                    {
                        q.push({i,set|(1<<(i-1))});
                        vis[i][set|(1<<(i-1))]=1;
                    }
                }
            }
        }
    }
```

```
    if (n==1)printf("0");
    else printf("%lld\n",dp[1][(1<<n)-1]);
    ///dis[i]表示城市1到城市i的最短路距离
    ///dis[t]现在还不是最短路，那么肯定存在j,dis[j]+w(j,t)<dis[t]
    ///queue<int>q;q.push(i),q.front();
    return 0;
}
```

5.4 数位 dp

```
#include <bits/stdc++.h>
using namespace std;
int cnt=0;
int a[30];
long long dp[70][2];
long long dfs(int pos,int limit,int lead,int is6){
    if(pos==0) return 1;
    if(!limit&&dp[pos][is6]!=-1) return dp[pos][is6];
    int up=(limit?a[pos]:9);
    long long ans=0;
    for(int i=0;i<=up;i++){
        if(i==4) continue;
        if(i==2&&is6) continue;
        ans+=dfs(pos-1,limit&&(i==up),lead&&i==0,i==6);
    }
    if(!limit) dp[pos][is6]=ans;
    return ans;
}

long long solve(long long x){
    cnt=0;
    while(x){
        a[++cnt]=x%10;
        x/=10;
    }
    return dfs(cnt,1,0,0);
}

int main(){
    long long n,m;
    memset(dp,-1,sizeof(dp));
    while(scanf("%lld %lld",&n,&m)!=EOF){
        if(n==0&&m==0) break;
        printf("%lld\n",solve(m)-solve(n-1));
    }
}
```

5.5 带障碍的路径计数

```
//复杂度为O(m^2),m为障碍的个数,此代码包含m个障碍和n个终点,每个终点互不影响
void solve()
{
    binom_init();
    int n,m;
    cin>>n>>m;
    vector<int>x(m+n+1),y(m+n+1),vis(m+n+1);
    vector<Z>dp(m+n+1);
    for(int i=1;i<=m;i++)cin>>x[i]>>y[i];
    for(int i=1;i<=n;i++)x[m+i]=i,y[m+i]=n-i+1;
    function<Z(int)>dfs=[&](int i)->Z
    {
        if(vis[i])return dp[i];
        vis[i]=1;
        Z res=0;
        for(int j=1;j<=m;j++)
        {
            if(j!=i&&x[j]<=x[i]&&y[j]<=y[i])
            {
                res+=dfs(j)*binom(x[i]-x[j]+y[i]-y[j],x[i]-x[j]);
            }
        }
        return dp[i]=binom(x[i]-1+y[i]-1,x[i]-1)-res;
    };
    Z ans=0;
    for(int i=m+1;i<=n+m;i++)
    {
        ans+=dfs(i);
    }
    cout<<ans;
}
```

6 Geometry

6.1 二维

```
namespace Geometry{
   const double eps=1e-9;
   const double PI=acos(-1.0);
   inline int dcmp(double x){
      if(fabs(x)<eps)return 0;
      return x<0?-1:1;
   }
   inline double Hypot(double a,double b){
      return sqrt(a*a+b*b);
   }
   struct Point{
      double x,y;
      Point(){}
      Point(double x,double y):x(x),y(y){}
   };
   typedef Point Vector;
   Vector operator + (const Vector &a,const Vector &b){
      return Vector(a.x+b.x,a.y+b.y);
   }
   Vector operator - (const Vector &a,const Vector &b){
      return Vector(a.x-b.x,a.y-b.y);
   }
   Vector operator * (const Vector &a,const double k){
      return Vector(a.x*k,a.y*k);
   }
   Vector operator / (const Vector &a,const double k){
      return Vector(a.x/k,a.y/k);
   }
   bool operator == (const Vector &a,const Vector &b){
      return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
   }
   inline double Dot(Vector a,Vector b){
      return a.x*b.x+a.y*b.y;
   }
   inline double Cross(Vector a,Vector b){
      return a.x*b.y-a.y*b.x;
   }
   inline double Length(Vector a){
      return sqrt(Dot(a,a));
   }
   inline double Angle(Vector a,Vector b){
      return acos(Dot(a,b)/(Length(a)*Length(b)));
   }
   inline double Distance(Point a,Point b){
      return Hypot(a.x-b.x,a.y-b.y);
   }
   inline double TriangleArea(Point &a,Point &b,Point &c){
      return fabs(Cross(b-a,c-a)/2);
   }
   inline Vector Rotate(Vector a,double ang){
      return Vector(a.x*cos(ang)-a.y*sin(ang),a.x*sin(ang)+a.y*cos(ang
          ));
   }
   struct StdLine{
      double a,b,c,angle;
      Point s,e;
      StdLine(){}
      StdLine(Point _s,Point _e){
         s=_s;
         e=_e;
         a=s.y-e.y;
         b=e.x-s.x;
         c=s.x*e.y-e.x*s.y;
         angle=atan2(e.y-s.y,e.x-s.x);
      }
   };
   struct Line{
      Point s,e;
      Line(){}
      Line(Point _s,Point _e){
         s=_s;
         e=_e;
      }
      bool isPointOnSegment(Point &p){
         return dcmp(Cross(p-s,e-s))==0&&dcmp(Dot(p-s,p-e))<=0;
      }
      bool isPointOnLine(Point &p){
         return dcmp(Cross(p-s,e-s))==0;
      }
      bool isParallel(Line &l){
         return dcmp(Cross(e-s,l.e-l.s))==0;
      }
      double getDistance(Point &p){
         return fabs(Cross(p-s,p-e)/Distance(s,e));
      }
      double getDistanceToSegment(Point &p){
         if(Dot(e-s,p-s)<0)return Distance(s,p);
         if(Dot(s-e,p-e)<0)return Distance(e,p);
         return getDistance(p);
      }
```

```
90      Point intersectPoint(Line &l){
91          return s+(e-s)*(Cross(l.e-l.s,l.s-s)/Cross(l.e-l.s,e-s));
92      }
93      Point symmetryPoint(Point &p){
94          double a=e.x-s.x,b=e.y-s.y;
95          double t=((p.x-s.x)*a+(p.y-s.y)*b)/(a*a+b*b);
96          return Point(2*s.x+2*a*t-p.x,2*s.y+2*b*t-p.y);
97      }
98      Line getPlumbLine(double inc=20.0){
99          Point p=Point((s.x+e.x)/2.0,(s.y+e.y)/2.0);
100         if(dcmp(s.x-e.x)==0)return Line(p,Point(p.x+inc,p.y));
101         if(dcmp(s.y-e.y)==0)return Line(p,Point(p.x,p.y+inc));
102         return Line(p,Point(p.x+inc,p.y-(inc/((s.y-e.y)/(s.x-e.x)))))
                ;
103     }
104 };
105 typedef Point Segment;
106 inline int getIntersectSegment(Segment a,Segment b,Segment &res){
107     if(a.y<b.x||a.x>b.y){
108         res=Segment(0,0);
109         return 0;
110     }
111     if(a.x<=b.x&&a.y>=b.y)res=b;
112     else if(b.x<=a.x&&b.y>=a.y)res=a;
113     else res=Segment(max(a.x,b.x),min(a.y,b.y));
114     return 1;
115 }
116 struct Rectangle{
117     Point a,b;
118     Rectangle(){}
119     Rectangle(Point a,Point b):a(a),b(b){}
120     double area(){
121         return fabs((a.x-b.x)*(a.y-b.y));
122     }
123     Rectangle getIntersect(Rectangle &r){
124         Segment x,y;
125         getIntersectSegment(Segment(a.x,b.x),Segment(r.a.x,r.b.x),x);
126         getIntersectSegment(Segment(a.y,b.y),Segment(r.a.y,r.b.y),y);
127         return Rectangle(Point(x.x,y.x),Point(x.y,y.y));
128     }
129 };
130 struct Polygon{
131     int n; vector<Point> p;
132     Polygon(){n=0;}
133     void init(){
134         p.clear();
135         n=0;
136     }
137     void push(Point &a){
138         p.push_back(a);
139         n++;
140     }
141     void pop(){
142         p.pop_back();
143         n--;
144     }
145     Point back(){
146         return p[n-1];
147     }
148     bool insidePolygon(Point &a){
149         double ang=0;
150         for(int i=0;i<n;i++)
151         ang+=Angle(p[i]-a,p[(i+1)%n]-a);
152         return dcmp(fabs(ang)-PI*2)==0;
153     }
154     double area(Point &o){
155         double sum=0;
156         for(int i=0;i<n;i++)
157         sum+=Cross(p[i]-o,p[(i+1)%n]-o);
158         return fabs(sum/2);
159     }
160     Polygon graham(){//凸包
161         for(int i=1;i<n;i++)
162         if(p[i].y<p[0].y||dcmp(p[i].y-p[0].y)==0&&p[i].x<p[0].x)
163         swap(p[i],p[0]);
164         sort(p.begin()+1,p.end(),[&](Point a,Point b){
165             double r=Cross(a-p[0],b-p[0]);
166             if(dcmp(r)==0)return dcmp(Distance(p[0],a)-Distance(p[0],b
                    ))<0;
167             return r>0;
168         });
169         Polygon poly;
170         poly.push(p[0]);
171         poly.push(p[1]);
172         for(int i=2;i<n;i++){
173             while(poly.n>=2&&Cross(poly.p[poly.n-1]-poly.p[poly.n-2],p
                    [i]-poly.p[poly.n-2])<0)
174             poly.pop();
175             poly.push(p[i]);
176         }
177         return poly;
178     }
179 };
180 struct Circle{
```

```
    Point o; double r;
    Circle(){}
    Circle(Point o,double r):o(o),r(r){}
    double getArcLength(double ang){
        return ang*r;
    }
    double getSectorArea(double ang){
        return getArcLength(ang)*r/2;
    }
    Point getPoint(double ang){
        return Point(o.x+r*cos(ang),o.y+r*sin(ang));
    }
    int getTangentPoints(Point &p,vector<Point> &vec){
        double d=Distance(p,o),b=atan2(p.y-o.y,p.x-o.x);
        double delta=acos(r/d);
        vec.push_back(getPoint(b+delta));
        if(dcmp(delta)){
            vec.push_back(getPoint(b-delta));
            return 2;
        }
        return 1;
    }
    int getIntersectPoints(Line &l,vector<Point> &vec){
        if(dcmp(l.getDistance(o)-r)>0)return 0;
        Vector v=l.e-l.s;
        Point p=l.s+v*(Dot(o-l.s,v))/Dot(v,v);
        double b=sqrt(r*r-Dot(p-o,p-o));
        if(dcmp(b)==0){
            vec.push_back(p);
            return 1;
        }
        v=v/Length(v)*b;
        vec.push_back(p+v);
        vec.push_back(p-v);
        return 2;
    }
    int getIntersectPoints(Circle &c,vector<Point> &vec){
        double d=Distance(o,c.o);
        if(dcmp(d)==0){
            if(dcmp(r-c.r)==0)return -1;
            return 0;
        }
        if(dcmp(r+c.r-d)<0)return 0;
        if(dcmp(fabs(r-c.r)-d)>0)return 0;
        double ang=Angle(c.o-o,Vector(1,0));
        double delta=acos((r*r+d*d-c.r*c.r)/(2*r*d));
        vec.push_back(getPoint(ang+delta));
        if(dcmp(delta)){
            vec.push_back(getPoint(ang-delta));
            return 2;
        }
        return 1;
    }
};
}
using namespace Geometry;
```

6.2 三维

```
namespace Geometry3 {
    const double eps=1e-9;
    inline int dcmp(double x){
        if(fabs(x)<eps)return 0;
        return x<0?-1:1;
    }
    inline double Hypot(double a,double b,double c){
        return sqrt(a*a+b*b+c*c);
    }
    struct Point3{
        double x,y,z;
        Point3(){}
        Point3(double x,double y,double z):x(x),y(y),z(z){}
    };
    typedef Point3 Vector3;
    Vector3 operator + (const Vector3 &a,const Vector3 &b){
        return Vector3(a.x+b.x,a.y+b.y,a.z+b.z);
    }
    Vector3 operator - (const Vector3 &a,const Vector3 &b){
        return Vector3(a.x-b.x,a.y-b.y,a.z-b.z);
    }
    Vector3 operator * (const Vector3 &a,const double k){
        return Vector3(a.x*k,a.y*k,a.z*k);
    }
    Vector3 operator / (const Vector3 &a,const double k){
        return Vector3(a.x/k,a.y/k,a.z/k);
    }
    bool operator == (const Vector3 &a,const Vector3 &b){
        return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0&&dcmp(a.z-b.z)==0;
    }
    inline double Dot(Vector3 a,Vector3 b){
        return a.x*b.x+a.y*b.y+a.z*b.z;
    }
    inline double Length(Vector3 a){
```

```
35      return sqrt(Dot(a,a));
36   }
37   inline Vector3 CrossVector(Vector3 v1,Vector3 v2){
38      return Vector3(v1.y*v2.z-v1.z*v2.y,v1.z*v2.x-v1.x*v2.z,v1.x*v2.y
            -v1.y*v2.x);
39   }
40   inline double Cross(Vector3 a,Vector3 b){
41      return Length(CrossVector(a,b));
42   }
43   inline double Distance(Point3 a,Point3 b){
44      return Hypot(a.x-b.x,a.y-b.y,a.z-b.z);
45   }
46   struct Line3{
47      Point3 s,e;
48      Line3(){}
49      Line3(Point3 _s,Point3 _e){
50         s=_s;
51         e=_e;
52      }
53      bool isPointOnSegment(Point3 &p){
54         return dcmp(Cross(p-s,e-s))==0&&dcmp(Dot(p-s,p-e))<=0;
55      }
56      bool isPointOnLine(Point3 &p){
57         return dcmp(Cross(p-s,e-s))==0;
58      }
59      bool isParallel(Line3 &l){
60         return dcmp(Cross(e-s,l.e-l.s))==0;
61      }
62      double getDistance(Point3 &p){
63         return fabs(Cross(p-s,p-e)/Distance(s,e));
64      }
65      double getDistanceToSegment(Point3 &p){
66         if(Dot(e-s,p-s)<0)return Distance(s,p);
67         if(Dot(s-e,p-e)<0)return Distance(e,p);
68         return getDistance(p);
69      }
70      double getDistanceToLine(Line3 &l){
71         Vector3 x=CrossVector(s-e,l.s-l.e);
72         return fabs(Dot(x,s-l.s))/Length(x);
73      }
74      double getAngleCosineToLine(Line3 &l){
75         return Dot(s-e,l.s-l.e)/Length(s-e)/Length(l.s-l.e);
76      }
77   };
78   struct Plane{
79      Point3 a,b,c,v;
80      Plane(){}
81      Plane(Point3 _a,Point3 _b,Point3 _c){
82         a=_a;
83         b=_b;
84         c=_c;
85         v=CrossVector(a-b,a-c);
86      }
87      Plane(Point3 _a,Vector3 _v){
88         a=_a;
89         v=_v;
90      }
91      bool isPointOnTriangle(Point3 &p){
92         return dcmp(Cross(a-b,a-c)-Cross(p-a,p-b)-Cross(p-b,p-c)-
               Cross(p-c,p-a))==0;
93      }
94      bool isParallel(Plane &p){
95         return dcmp(Cross(v,p.v))==0;
96      }
97      int getIntersectLine(Plane &p,Line3 &l){
98         Vector3 r=CrossVector(v,p.v),v=CrossVector(v,r);
99         double d=Dot(p.v,v);
100        if(dcmp(d)==0) return 0;
101        l.s=a+v*(Dot(p.v,p.a-a)/d);
102        l.e=l.s+r;
103        return 1;
104     }
105     double getDistance(Point3 &p){
106        return fabs(Dot(v,p-a))/Length(v);
107     }
108     double getAngleCosineToPlane(Plane &p){
109        return Dot(v,p.v)/Length(v)/Length(p.v);
110     }
111     double getAngleSineToLine(Line3 &l){
112        return Dot(l.s-l.e,v)/Length(l.s-l.e)/Length(v);
113     }
114  };
115  struct Circle3{
116     Point3 o; double r;
117     Circle3(){}
118     Circle3(Point3 o,double r):o(o),r(r){}
119     int getIntersectPoints(Line3 &l,vector<Point3> &vec){
120        if(dcmp(l.getDistance(o)-r)>0)return 0;
121        Vector3 v=l.e-l.s;
122        Point3 p=l.s+v*(Dot(o-l.s,v))/Dot(v,v);
123        double b=sqrt(r*r-Dot(p-o,p-o));
124        if(dcmp(b)==0){
125           vec.push_back(p);
126           return 1;
```

```
        }
        v=v/Length(v)*b;
        vec.push_back(p+v);
        vec.push_back(p-v);
        return 2;
    }
  };
}
using namespace Geometry3;
```

7 Others

7.1 头文件

```
#include<algorithm>
#include<cctype>
#include<chrono>//poj不能用
#include<cmath>
#include<cstdio>
#include<cstring>
#include<functional>
#include<iomanip>
#include<iostream>
#include<map>
#include<queue>
#include<random>//poj不能用
#include<set>
#include<sstream>
#include<stack>
#include<string>
#include<utility>
#include<vector>
```

7.2 吸氧大法

```
#pragma GCC optimize(2)
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-funroll-loops")
```

```
#pragma GCC optimize("-fwhole-program")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-fstrict-overflow")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-skip-blocks")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("-funsafe-loop-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
//简易
#pragma GCC optimize(2)
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast,unroll-loops")
```

7.3 迭代器

```
vector<int>v(10,1);
for(vector<int>::iterator it=v.begin();it!=v.end();it++)
{
    // ...
}
```

7.4 lambda 函数

```
function<void(int,int)>dfs=[&](int now,int flag)
{
    // ...
};
function<int(int,int)>func=[&](int x,int y)->int
{
    // ...
};
```

7.5 结构体

```
struct node
{
    string name;
    int cnt;
    node(){}
    node(string s,int c){name=s,cnt=c;}
    bool operator<(const node& a)const // 结构体排序
    {
        if(cnt!=a.cnt)
        return cnt>a.cnt;// 先按数量从大到小
        if(name.size()!=a.name.size())
        return name.size()>a.name.size(); // 再按标签长度从小到大
        return name<a.name; // 最后按字典序，string可以直接进行对比
    }
};
```

7.6 快读快写

```
namespace fastIO
{
    #define gc h==d&&(d=(h=buf)+fread(buf,1,100000,stdin),h==d)?EOF:*h++ //不能用fread则换成getchar
    static char buf[100000],*h=buf,*d=buf;//缓存开大可减少读入时间，看题目给的空间
    template<typename T>
    inline void read(T&x)
    {
        int f=1;x=0;char c(gc);
        while(c>'9'||c<'0'){if(c=='-') f=-1;c=gc;}
        while(c<='9'&&c>='0')x=(x<<1)+(x<<3)+(c^48),c=gc;
        x*=f;
    }
    template<typename T>
    void print(T x)
    {
        if(x<0){putchar('-');x=~(x-1);}
```

```
        static int s[20],top=0;
        while(x){s[++top]=x%10;x/=10;}
        if(!top)s[++top]=0;
        while(top)putchar(s[top--]+'0');
    }
}
using namespace fastIO;
```

```
namespace fastIO
{
    #ifdef LOCAL
    #define DEBUG
    #endif
    #define BUF_SIZE 100000
    #define DECIMAL 6
    #define LL long long
    static bool IOerror=0;
    #ifdef DEBUG
    inline char nc(){char ch=getchar();if(ch==-1)IOerror=1;return ch;}
    #else
    inline char nc(){static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=
        buf+BUF_SIZE;if(p1==pend){p1=buf;pend=buf+fread(buf,1,BUF_SIZE,
        stdin);if(pend==p1){IOerror=1;return -1;}}return *p1++;}
    #endif
    inline bool blank(char ch){return ch==' '||ch=='\n'||ch=='\r'||ch==
        '\t';}
    template<class T> inline bool read(T&x){bool sign=0;char ch=nc();x
        =0;for(;blank(ch);ch=nc());if(IOerror)return false;if(ch=='-')
        sign=1,ch=nc();for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';if(
        sign)x=-x;return true;}
    inline bool read(unsigned long long&x){char ch=nc();x=0;for(;blank(
        ch);ch=nc());if(IOerror)return false;for(;ch>='0'&&ch<='9';ch=
        nc())x=x*10+ch-'0';return true;}
    inline bool read(unsigned int&x){char ch=nc();x=0;for(;blank(ch);ch
        =nc());if(IOerror)return false;for(;ch>='0'&&ch<='9';ch=nc())x=
        x*10+ch-'0';return true;}
    inline bool read(double&x){bool sign=0;char ch=nc();x=0;for(;blank(
        ch);ch=nc());if(IOerror)return false;if(ch=='-')sign=1,ch=nc();
        for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';if(ch=='.'){double
        tmp=1;ch=nc();for(;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(
        ch-'0');}if(sign)x=-x;return true;}
    inline bool read(char*s){char ch=nc();for(;blank(ch);ch=nc());if(
        IOerror)return false;for(;!blank(ch)&&!IOerror;ch=nc())*s++=ch
        ;*s=0;return true;}
    inline bool read(char&c){for(c=nc();blank(c);c=nc());if(IOerror){c
        =-1;return false;}return true;}
    inline bool read(std::string&s){s.clear();char ch=nc();for(;blank(
        ch);ch=nc());if(IOerror)return false;for(;!blank(ch)&&!IOerror;
        ch=nc())s+=ch;return true;}
    template<class T,class ... U> bool read(T&h,U& ... t){return read(h)
        &&read(t ... );}
    struct Ostream_fwrite
    {
        #ifdef DEBUG
        inline void out(char ch) { putchar(ch); }
        #else
        char*buf,*p1,*pend;
        Ostream_fwrite(){buf=new char[BUF_SIZE];p1=buf;pend=buf+BUF_SIZE
            ;}
        inline void out(char ch){if(p1==pend){fwrite(buf,1,BUF_SIZE,
            stdout);p1=buf;}*p1++=ch;}
        #endif
        template<class T> inline void print(T x){static char s[41],*s1;
            s1=s;if(!x)*s1++='0';if(x<0)out('-'),x=-x;while(x)*s1++=x
            %10+'0',x/=10;while(s1--!=s)out(*s1);}
        inline void print(char ch){out(ch);}
        inline void print(unsigned long long x){static char s[41],*s1;s1
            =s;if(!x)*s1++='0';while(x)*s1++=x%10+'0',x/=10;while(s1--!=
            s)out(*s1);}
        inline void print(unsigned int x){static char s[41],*s1;s1=s;if
            (!x)*s1++='0';while(x)*s1++=x%10+'0',x/=10;while(s1--!=s)out
            (*s1);}
        inline void print(double x,int y=DECIMAL){static LL mul
            []={1,10,100,1000,10000,100000,
              1000000,10000000,100000000,1000000000,
              10000000000LL,100000000000LL,
              1000000000000LL,10000000000000LL,
              100000000000000LL,1000000000000000LL,
              10000000000000000LL,100000000000000000LL};
            if(x<-1e-12)out('-'),x=-x;x*=mul[y];LL x1=(LL)floor(x);if(x-
                floor(x)>=0.5)++x1;LL x2=x1/mul[y],x3=x1-x2*mul[y];print(
                x2);if(y>0){out('.');for(size_t i=1;i<y&&x3*mul[i]<mul[y
                ];out('0'),++i);print(x3);}}
        inline void print(char*s){while(*s)out(*s++);}
        inline void print(const char*s){while(*s)out(*s++);}
        inline void print(std::string s){print(s.c_str());}
        #ifndef DEBUG
        inline void flush(){if(p1!=buf){fwrite(buf,1,p1-buf,stdout);p1=
            buf;}}
        ~Ostream_fwrite(){flush();}
        #endif // !DEBUG
    }Ostream;
    template<class T> inline void print(T x){Ostream.print(x);}
    template<class T> inline void println(T x){Ostream.print(x);Ostream
```

```
        .out('\n');}
    inline void print(double x,int y=DECIMAL){Ostream.print(x,y);}
    inline void println(double x,int y=DECIMAL){Ostream.print(x,y);
        Ostream.out('\n');}
    template<class T,class... U> void print(const T&h,const U&... t){
        print(h);Ostream.out(' ');print(t...);}
    template<class T,class... U> void println(const T&h,const U&... t){
        print(h);Ostream.out(' ');println(t...);}
    #ifndef DEBUG
    inline void flush(){Ostream.flush();}
    #endif
    #undef LL
    #undef DECIMAL
    #undef BUF_SIZE
    //本地 getchar/putchar , 提交 fread/fwrite
    //本地支持终端交互输入输出
}
using namespace fastIO;
```

```
void read(int& x)
{
    char c = getchar(); x = 0;
    while (c < '0' || c > '9') c = getchar();
    while (c <= '9' && c >= '0') x = x * 10 + c - 48, c = getchar();
}
inline int read()
{
    int s=0,w=1;char ch=getchar();
    for(;!isdigit(ch);ch=getchar())
    if(ch=='-')w=-1;
    for(;ch>='0'&&ch<='9';ch=getchar())
    s=(s<<1)+(s<<3)+(ch^48);
    return (w==-1?-s:s);
}
inline void print(int x) {
    if (x == 0){printf("0\n");return;}
    int num = 0; char c[15];
    while (x) c[++num] = (x % 10) + 48, x /= 10;
    while (num) putchar(c[num--]);
    putchar('\n');
}
```

7.7 取模

```
template<typename T>
int normalize(T value, int mod) {
    if (value < -mod || value >= 2 * mod) value %= mod;
    if (value < 0) value += mod;
    if (value >= mod) value -= mod;
    return value;
}

template<int mod>
struct static_modular_int {
    using mint = static_modular_int<mod>;

    int value;

    static_modular_int() : value(0) {}
    static_modular_int(const mint &x) : value(x.value) {}

    template<typename T, typename U = std::enable_if_t<std::is_integral
        <T>::value>>
    static_modular_int(T value) : value(normalize(value, mod)) {}

    template<typename T>
    mint power(T degree) const {
        degree = normalize(degree, mod - 1);
        mint prod = 1, a = *this;
        for (; degree > 0; degree >>= 1, a *= a)
            if (degree & 1)
                prod *= a;

        return prod;
    }

    mint inv() const {
        return power(-1);
    }

    mint& operator=(const mint &x) {
        value = x.value;
        return *this;
    }

    mint& operator+=(const mint &x) {
        value += x.value;
        if (value >= mod) value -= mod;
        return *this;
    }

    mint& operator-=(const mint &x) {
        value -= x.value;
```

```
    if (value < 0) value += mod;
    return *this;
  }

  mint& operator*=(const mint &x) {
    value = int64_t(value) * x.value % mod;
    return *this;
  }

  mint& operator/=(const mint &x) {
    return *this *= x.inv();
  }

  friend mint operator+(const mint &x, const mint &y) {
    return mint(x) += y;
  }

  friend mint operator-(const mint &x, const mint &y) {
    return mint(x) -= y;
  }

  friend mint operator*(const mint &x, const mint &y) {
    return mint(x) *= y;
  }

  friend mint operator/(const mint &x, const mint &y) {
    return mint(x) /= y;
  }

  mint& operator++() {
    ++value;
    if (value == mod) value = 0;
    return *this;
  }

  mint& operator--() {
    --value;
    if (value == -1) value = mod - 1;
    return *this;
  }

  mint operator++(int) {
    mint prev = *this;
    value++;
    if (value == mod) value = 0;
    return prev;
  }

  mint operator--(int) {
    mint prev = *this;
    value--;
    if (value == -1) value = mod - 1;
    return prev;
  }

  mint operator-() const {
    return mint(0) - *this;
  }

  bool operator==(const mint &x) const {
    return value == x.value;
  }

  bool operator!=(const mint &x) const {
    return value != x.value;
  }

  bool operator<(const mint &x) const {
    return value < x.value;
  }

  template<typename T>
  explicit operator T() {
    return value;
  }

  friend std::istream& operator>>(std::istream &in, mint &x) {
    std::string s;
    in >> s;
    x = 0;
    for (const auto c : s)
      x = x * 10 + (c - '0');

    return in;
  }

  friend std::ostream& operator<<(std::ostream &out, const mint &x) {
    return out << x.value;
  }

  static int primitive_root() {
    if constexpr (mod == 1000000007) return 5;
    if constexpr (mod == 998244353) return 3;
    if constexpr (mod == 786433) return 10;
```

```
    static int root = -1;
    if (root != -1)
        return root;

    std::vector<int> primes;
    int value = mod - 1;
    for (int i = 2; i * i <= value; i++)
        if (value % i == 0) {
            primes.push_back(i);
            while (value % i == 0)
                value /= i;
        }

    if (value != 1) primes.push_back(value);
    for (int r = 2;; r++) {
        bool ok = true;
        for (auto p : primes) {
            if ((mint(r).power((mod - 1) / p)).value == 1) {
                ok = false;
                break;
            }
        }
        if (ok) return root = r;
    }
  }
};

// constexpr int MOD = 1000000007;
constexpr int MOD = 998244353;
using mint = static_modular_int<MOD>;
```

7.8 Int128

```
std::ostream &operator<<(std::ostream &os, i128 n) {
  std::string s;
  while (n) {
    s += '0' + n % 10;
    n /= 10;
  }
  std::reverse(s.begin(), s.end());
  return os << s;
}

i128 sqrt(i128 x)//开方
{
  i128 y=(i128)ceil(sqrt((long double)x));
  for(;y*y<=x;++y);
  for(--y;y*y>x;--y);
  return y;
}
```

7.9 gp_hash_table

```
#include<bits/extc++.h>
using namespace __gnu_pbds;

struct custom_hash {
  static uint64_t splitmix64(uint64_t x) {
    // http://xorshift.di.unimi.it/splitmix64.c
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }

  size_t operator()(uint64_t x) const {
    static const uint64_t FIXED_RANDOM = chrono::steady_clock::now()
        .time_since_epoch().count();
    return splitmix64(x + FIXED_RANDOM);
  }
};

gp_hash_table<long long, int, custom_hash> safe_hash_table;
```

7.10 关同步流

```
ios::sync_with_stdio(false);cin.tie(0);
```

7.11 计时器

```
clock_t start,finish;
double totaltime;
start=clock();
// ...
finish=clock();
totaltime=(double)(finish-start)/CLOCKS_PER_SEC;
cout<<"\n此程序的运行时间为"<<totaltime<<"秒！"<<endl;
```