数据结构

1.并查集

```
#include<iostream>
using namespace std;
const int N=1e4+1;
int pre[N];
int root(int x){
        return pre[x]==x?x:root(pre[x]);
}
void merge(int x,int y){
        x=root(x),y=root(y);
        if(x==y)return;
        pre[x]=y;
}




int main(){
        int n;
        int m;
        cin>>n>>m;
        for(int i=0;i<N;i++)
        pre[i]=i;//自己是自己的父节点
        while(m--){
                int t,x,y;
                cin>>t>>x>>y;
                if(t==1){
                        merge(x,y);
                }else{
                        cout<<(root(x)==root(y)?'Y':'N')<<'\n';
                }
         }
return 0;
}
```

2.树状数组

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

long long n,m;
const int N=5e5+1;
```

```
int a[N];

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

void add(long long x,long long k){
        while(x! =0&&x< =N){
                a[x]+=k;
                x+=lowbit(x);
        }
}

long long sum(int x){
        long long result=0;
        while(x){
                result+=a[x];
                x=x-lowbit(x);
        }
        return result;
}

int main (){
        long long num;
        cin>>n>>m;
        for(int i=1;i< =n;i++){
                cin>>num;
                add(i,num);
        }
        while(m--){
                int x,y,z;
                cin>>x>>y>>z;
                if(x==1){
                        add(y,z);
                }
                if(x==2){
                        cout<<sum(z)-sum(y-1)<<endl;
                }
         }
        return 0;
 }
```

线段树

线段树1

区间加 区间和

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

using namespace std;

const int maxn=100010;

int a[maxn+2];

struct tree{
    int l,r;
    long long pre,add;
}t[4*maxn+2];

void bulid(int p,int l,int r){
    t[p].l=l;t[p].r=r;
    if(l==r){
        t[p].pre=a[l];
        return;
    }
    int mid=l+r>>1;
    bulid(p*2,l,mid);
    bulid(p*2+1,mid+1,r);
    t[p].pre=t[p*2].pre+t[p*2+1].pre;
}

void spread(int p){
    if(t[p].add){
        t[p*2].pre+=t[p].add*(t[p*2].r-t[p*2].l+1);
        t[p*2+1].pre+=t[p].add*(t[p*2+1].r-t[p*2+1].l+1);
        t[p*2].add+=t[p].add;
        t[p*2+1].add+=t[p].add;
        t[p].add=0;
    }
}

void change(int p,int x,int y,int z){
    if(x≤t[p].l && y≥t[p].r){
        t[p].pre+=(long long)z*(t[p].r-t[p].l+1);
        t[p].add+=z;
        return;
    }
    spread(p);
    int mid=t[p].l+t[p].r>>1;
    if(x≤mid) change(p*2,x,y,z);
    if(y>mid) change(p*2+1,x,y,z);
    t[p].pre=t[p*2].pre+t[p*2+1].pre;
}

long long ask(int p,int x,int y){
```

```
    if(x≤t[p].l && y≥t[p].r) return t[p].pre;
    spread(p);
    int mid=t[p].l+t[p].r>>1;
    long long ans=0;
    if(x≤mid) ans+=ask(p*2,x,y);
    if(y>mid) ans+=ask(p*2+1,x,y);
    return ans;
}

int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=1;i≤n;i++)
    scanf("%d",&a[i]);
    bulid(1,1,n);
    for(int i=1;i≤m;i++)
    {
        int q,x,y,z;
        scanf("%d",&q);
        if(q==1){
            scanf("%d%d%d",&x,&y,&z);
            change(1,x,y,z);
        }
        else {
            scanf("%d%d",&x,&y);
            cout<<ask(1,x,y)<<endl;
        }
    }
    return 0;
}
```

线段树2

区间乘 区间加 区间和

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

#define MAXN 100010
#define ll long long

using namespace std;

int n, m, mod;
int a[MAXN];

struct Segment_Tree {
        ll sum, add, mul;
        int l, r;
```

```
}s[MAXN * 4];

void update(int pos) {
        s[pos].sum = (s[pos << 1].sum + s[pos << 1 | 1].sum) % mod;
    return;
}

void pushdown(int pos) { //pushdown的维护
        s[pos << 1].sum = (s[pos << 1].sum * s[pos].mul + s[pos].add *
(s[pos << 1].r - s[pos << 1].l + 1)) % mod;
        s[pos << 1 | 1].sum = (s[pos << 1 | 1].sum * s[pos].mul + s[pos].add
* (s[pos << 1 | 1].r - s[pos << 1 | 1].l + 1)) % mod;

        s[pos << 1].mul = (s[pos << 1].mul * s[pos].mul) % mod;
        s[pos << 1 | 1].mul = (s[pos << 1 | 1].mul * s[pos].mul) % mod;

        s[pos << 1].add = (s[pos << 1].add * s[pos].mul + s[pos].add) % mod;
        s[pos << 1 | 1].add = (s[pos << 1 | 1].add * s[pos].mul +
s[pos].add) % mod;

        s[pos].add = 0;
        s[pos].mul = 1;
        return;
}

void build_tree(int pos, int l, int r) { //建树
        s[pos].l = l;
        s[pos].r = r;
        s[pos].mul = 1;

        if (l == r) {
                s[pos].sum = a[l] % mod;
                return;
        }

        int mid = (l + r) >> 1;
        build_tree(pos << 1, l, mid);
        build_tree(pos << 1 | 1, mid + 1, r);
        update(pos);
        return;
}

void ChangeMul(int pos, int x, int y, int k) { //区间乘法
        if (x ≤ s[pos].l && s[pos].r ≤ y) {
                s[pos].add = (s[pos].add * k) % mod;
                s[pos].mul = (s[pos].mul * k) % mod;
                s[pos].sum = (s[pos].sum * k) % mod;
                return;
        }
```

```
        pushdown(pos);
        int mid = (s[pos].l + s[pos].r) >> 1;
        if (x ≤ mid) ChangeMul(pos << 1, x, y, k);
        if (y > mid) ChangeMul(pos << 1 | 1, x, y, k);
        update(pos);
        return;
}

void ChangeAdd(int pos, int x, int y, int k) { //区间加法
        if (x ≤ s[pos].l && s[pos].r ≤ y) {
                s[pos].add = (s[pos].add + k) % mod;
                s[pos].sum = (s[pos].sum + k * (s[pos].r - s[pos].l + 1)) %
mod;
                return;
        }

        pushdown(pos);
        int mid = (s[pos].l + s[pos].r) >> 1;
        if (x ≤ mid) ChangeAdd(pos << 1, x, y, k);
        if (y > mid) ChangeAdd(pos << 1 | 1, x, y, k);
        update(pos);
        return;
}

ll AskRange(int pos, int x, int y) { //区间询问
        if (x ≤ s[pos].l && s[pos].r ≤ y) {
                return s[pos].sum;
        }

        pushdown(pos);
        ll val = 0;
        int mid = (s[pos].l + s[pos].r) >> 1;
        if (x ≤ mid) val = (val + AskRange(pos << 1, x, y)) % mod;
        if (y > mid) val = (val + AskRange(pos << 1 | 1, x, y)) % mod;
        return val;
}

int main() {
        scanf("%d%d%d", &n, &m, &mod);

        for (int i = 1; i ≤ n; i++) {
                scanf("%d", &a[i]);
        }

        build_tree(1, 1, n);

        for (int i = 1; i ≤ m; i++) {
                int opt, x, y;
                scanf("%d%d%d", &opt, &x, &y);
                if (opt == 1) {
```

```
                        int k;
                        scanf("%d", &k);
                        ChangeMul(1, x, y, k);
                }
                if (opt == 2) {
                        int k;
                        scanf("%d", &k);
                        ChangeAdd(1, x, y, k);
                }
                if (opt == 3) {
                        printf("%lld\n", AskRange(1, x, y));
                }
        }

        return 0;
}
```