树套树 1

线段树/树状数组套名次树

简介

用于实现区间范围的名次树操作，空间复杂度 $O(n\log n)$

模板题

洛谷p3380

维护一种数集结构，支持以下操作：

1. 查询 k 在区间内的排名
2. 查询区间内排名为 k 的值
3. 修改某一位置上的数值
4. 查询 k 在区间内的前驱
5. 查询 k 在区间内的后继

线段树套名次树版本

线段树套名次树 $\text{rank}$$\text{update}$$\text{pre}$$\text{suf}$ 操作和普通线段树类似kth 操作需要二分 $+$ rank 操作。

$\text{rank}$$\text{update}$$\text{pre}$$\text{suf}$ 操作时间复杂度为 $O(\log^2 n)$$\text{kth}$ 操作时间复杂度 $O(log^2 n\log v)$

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
```

```
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=(5e4+5)*4,MAXS=MAXN*20,Inf=0x7fffffff;
template <typename T>
struct Treap{
    int pool[MAXS],top,tot;
    int root[MAXN],ch[MAXS][2],r[MAXS],sz[MAXS],cnt[MAXS];
    T val[MAXS];
    int new_node(T v){
        int id=top?pool[top--]:++tot;
        val[id]=v;r[id]=rand();sz[id]=cnt[id]=1;ch[id][0]=ch[id][1]=0;
        return id;
    }
    void push_up(int id){sz[id]=sz[ch[id][0]]+sz[ch[id][1]]+cnt[id];}
    void Rotate(int &id,int dir){
        int t=ch[id][dir^1];
        ch[id][dir^1]=ch[t][dir];ch[t][dir]=id;id=t;
        push_up(ch[id][dir]);push_up(id);
    }
    void Insert(int &id,T v){
        if(!id)
        return id=new_node(v),void();
        if(v==val[id])
```

```
        cnt[id]++;
        else{
            int dir=v<val[id]?0:1;
            Insert(ch[id][dir],v);
            if(r[id]<r[ch[id][dir]])
            Rotate(id,dir^1);
        }
        push_up(id);
    }
    void Erase(int &id,T v){
        if(!id)
        return;
        if(v==val[id]){
            if(cnt[id]>1)return    cnt[id]--,push_up(id);
            else if(!ch[id][0])pool[++top]=id,id=ch[id][1];
            else if(!ch[id][1])pool[++top]=id,id=ch[id][0];
            else{
                int d=r[ch[id][0]]>r[ch[id][1]]?1:0;
                Rotate(id,d);Erase(ch[id][d],v);push_up(id);
            }
        }
        else{
            if(v<val[id])Erase(ch[id][0],v);
            else Erase(ch[id][1],v);
            push_up(id);
        }
    }
    int Rank(int id,T v){//有多少个数严格小于v
        if(!id)return 0;
        if(v==val[id])return sz[ch[id][0]];
        else if(v<val[id])return Rank(ch[id][0],v);
        else return sz[ch[id][0]]+cnt[id]+Rank(ch[id][1],v);
    }
    T Kth(int id,int rk){
        if(!id) return -1;//第rk小的节点不存在
        if(rk>sz[ch[id][0]]+cnt[id]) return Kth(ch[id][1],rk-sz[ch[id][0]]-
cnt[id]);
        else if(rk>sz[ch[id][0]]) return val[id];
        else return Kth(ch[id][0],rk);
    }
    T Pre(int id,T v){
        int pos=id,ans=-Inf;
        while(pos){
            if(val[pos]<v){
                ans=ans<val[pos]?val[pos]:ans;
                pos=ch[pos][1];
            }
            else pos=ch[pos][0];
        }
        return ans;
    }
```

```
    T Suf(int id,T v){
        int pos=id,ans=Inf;
        while(pos){
            if(v<val[pos]){
                ans=ans<val[pos]?ans:val[pos];
                pos=ch[pos][0];
            }
            else pos=ch[pos][1];
        }
        return ans;
    }
    void insert(int root_id,T v){Insert(root[root_id],v);}
    void erase(int root_id,T v){Erase(root[root_id],v);}
    int rank(int root_id,T v){return Rank(root[root_id],v);}//如果需要，记得+1
    T kth(int root_id,int rk){return Kth(root[root_id],rk);}
    T pre(int root_id,T v){return Pre(root[root_id],v);}
    T suf(int root_id,T v){return Suf(root[root_id],v);}
};
struct Tree{
    Treap<int> S;
    int a[MAXN],lef[MAXN],rig[MAXN];
    void build(int k,int L,int R){
        lef[k]=L,rig[k]=R;
        _rep(i,L,R)
        S.insert(k,a[i]);
        if(L==R)
        return;
        int M=L+R>>1;
        build(k<<1,L,M);
        build(k<<1|1,M+1,R);
    }
    void build(int n){build(1,1,n);}
    int Rank(int k,int L,int R,int v){
        if(L<=lef[k]&&rig[k]<=R)
        return S.rank(k,v);
        int mid=lef[k]+rig[k]>>1;
        if(mid>=R)
        return Rank(k<<1,L,R,v);
        else if(mid<L)
        return Rank(k<<1|1,L,R,v);
        else
        return Rank(k<<1,L,R,v)+Rank(k<<1|1,L,R,v);
    }
    int rank(int L,int R,int v){return Rank(1,L,R,v)+1;}
    int kth(int L,int R,int rk){
        int lef=0,rig=1e8,mid,trk,ans=0;
        while(lef<=rig){
            mid=lef+rig>>1;
            trk=rank(L,R,mid);
            if(trk<=rk){
```

```
                ans=mid;
                lef=mid+1;
            }
            else if(trk>rk)
            rig=mid-1;
        }
        return ans;
    }
    void update(int k,int pos,int v){
        S.erase(k,a[pos]);
        S.insert(k,v);
        if(lef[k]==rig[k]){
            a[pos]=v;
            return;
        }
        int mid=lef[k]+rig[k]>>1;
        if(pos<=mid)
        update(k<<1,pos,v);
        else
        update(k<<1|1,pos,v);
    }
    void update(int pos,int v){update(1,pos,v);}
    int Pre(int k,int L,int R,int v){
        if(L<=lef[k]&&rig[k]<=R)
        return S.pre(k,v);
        int mid=lef[k]+rig[k]>>1;
        if(mid>=R)
        return Pre(k<<1,L,R,v);
        else if(mid<L)
        return Pre(k<<1|1,L,R,v);
        else
        return max(Pre(k<<1,L,R,v),Pre(k<<1|1,L,R,v));
    }
    int pre(int L,int R,int v){return Pre(1,L,R,v);}
    int Suf(int k,int L,int R,int v){
        if(L<=lef[k]&&rig[k]<=R)
        return S.suf(k,v);
        int mid=lef[k]+rig[k]>>1;
        if(mid>=R)
        return Suf(k<<1,L,R,v);
        else if(mid<L)
        return Suf(k<<1|1,L,R,v);
        else
        return min(Suf(k<<1,L,R,v),Suf(k<<1|1,L,R,v));
    }
    int suf(int L,int R,int v){return Suf(1,L,R,v);}
}tree;
int main()
{
    int n=read_int(),m=read_int(),opt,l,r,pos,k,ans;
    _rep(i,1,n)
```

```
    tree.a[i]=read_int();
    tree.build(n);
    while(m--){
        opt=read_int();
        if(opt==3){
            pos=read_int(),k=read_int();
            tree.update(pos,k);
        }
        else{
            l=read_int(),r=read_int(),k=read_int();
            switch(opt){
                case 1:
                    ans=tree.rank(l,r,k);
                    break;
                case 2:
                    ans=tree.kth(l,r,k);
                    break;
                case 4:
                    ans=tree.pre(l,r,k);
                    break;
                case 5:
                    ans=tree.suf(l,r,k);
                    break;
            }
            enter(ans);
        }
    }
    return 0;
}
```

树状数组套名次树版本

树状数组套名次树 rank、update 操作和普通树状数组类似、kth 操作需要二分 $+$ rank 操作、 pre、suf 操作需要 $\text{rank}+\text{kth}$、

rank、update 操作时间复杂度为 $O(\log^2 n)$、pre、suf、kth 操作时间复杂度 $O(log^2 n\log v)$、

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
```

```
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=5e4+5,MAXS=MAXN*20,Inf=0x7fffffff;
template <typename T>
struct Treap{
    int pool[MAXS],top,tot;
    int root[MAXN],ch[MAXS][2],r[MAXS],sz[MAXS],cnt[MAXS];
    T val[MAXS];
    int new_node(T v){
        int id=top?pool[top--]:++tot;
        val[id]=v;r[id]=rand();sz[id]=cnt[id]=1;ch[id][0]=ch[id][1]=0;
        return id;
    }
    void push_up(int id){sz[id]=sz[ch[id][0]]+sz[ch[id][1]]+cnt[id];}
    void Rotate(int &id,int dir){
        int t=ch[id][dir^1];
```

```
        ch[id][dir^1]=ch[t][dir];ch[t][dir]=id;id=t;
        push_up(ch[id][dir]);push_up(id);
    }
    void Insert(int &id,T v){
        if(!id)
        return id=new_node(v),void();
        if(v==val[id])
        cnt[id]++;
        else{
            int dir=v<val[id]?0:1;
            Insert(ch[id][dir],v);
            if(r[id]<r[ch[id][dir]])
            Rotate(id,dir^1);
        }
        push_up(id);
    }
    void Erase(int &id,T v){
        if(!id)
        return;
        if(v==val[id]){
            if(cnt[id]>1)return    cnt[id]--,push_up(id);
            else if(!ch[id][0])pool[++top]=id,id=ch[id][1];
            else if(!ch[id][1])pool[++top]=id,id=ch[id][0];
            else{
                int d=r[ch[id][0]]>r[ch[id][1]]?1:0;
                Rotate(id,d);Erase(ch[id][d],v);push_up(id);
            }
        }
        else{
            if(v<val[id])Erase(ch[id][0],v);
            else Erase(ch[id][1],v);
            push_up(id);
        }
    }
    int Rank(int id,T v){//有多少个数严格小于v
        if(!id)return 0;
        if(v==val[id])return sz[ch[id][0]];
        else if(v<val[id])return Rank(ch[id][0],v);
        else return sz[ch[id][0]]+cnt[id]+Rank(ch[id][1],v);
    }
    T Kth(int id,int rk){
        if(!id) return -1;//第rk小的节点不存在
        if(rk>sz[ch[id][0]]+cnt[id]) return Kth(ch[id][1],rk-sz[ch[id][0]]-
cnt[id]);
        else if(rk>sz[ch[id][0]]) return val[id];
        else return Kth(ch[id][0],rk);
    }
    T Pre(int id,T v){
        int pos=id,ans=-Inf;
        while(pos){
```

```
            if(val[pos]<v){
                ans=ans<val[pos]?val[pos]:ans;
                pos=ch[pos][1];
            }
            else pos=ch[pos][0];
        }
        return ans;
    }
    T Suf(int id,T v){
        int pos=id,ans=Inf;
        while(pos){
            if(v<val[pos]){
                ans=ans<val[pos]?ans:val[pos];
                pos=ch[pos][0];
            }
            else pos=ch[pos][1];
        }
        return ans;
    }
    int Count(int id,T v){
        int pos=id;
        while(pos){
            if(v<val[pos])
            pos=ch[pos][0];
            else if(val[pos]<v)
            pos=ch[pos][1];
            else
            return cnt[pos];
        }
        return 0;
    }
    void insert(int root_id,T v){Insert(root[root_id],v);}
    void erase(int root_id,T v){Erase(root[root_id],v);}
    int rank(int root_id,T v){return Rank(root[root_id],v);}//如果需要，记得+1
    T kth(int root_id,int rk){return Kth(root[root_id],rk);}
    T pre(int root_id,T v){return Pre(root[root_id],v);}
    T suf(int root_id,T v){return Suf(root[root_id],v);}
    int count(int root_id,T v){return Count(root[root_id],v);}
};
#define lowbit(x) x&(-x)
struct Tree{
    Treap<int> S;
    int n,a[MAXN];
    void build(int pos,int v){
        while(pos<=n){
            S.insert(pos,v);
            pos+=lowbit(pos);
        }
    }
    void build(int n){
        this->n=n;
```

```
        _rep(i,1,n)
        build(i,a[i]);
    }
    int Rank(int L,int R,int v){
        int ans=0,pos1=L-1,pos2=R;
        while(pos1){
            ans-=S.rank(pos1,v);
            pos1-=lowbit(pos1);
        }
        while(pos2){
            ans+=S.rank(pos2,v);
            pos2-=lowbit(pos2);
        }
        return ans;
    }
    int rank(int L,int R,int v){return Rank(L,R,v)+1;}
    int kth(int L,int R,int rk){
        int lef=0,rig=1e8,mid,trk,ans=0;
        while(lef<=rig){
            mid=lef+rig>>1;
            trk=rank(L,R,mid);
            if(trk<=rk){
                ans=mid;
                lef=mid+1;
            }
            else if(trk>rk)
            rig=mid-1;
        }
        return ans;
    }
    void update(int pos,int v){
        int t=pos;
        while(t<=n){
            S.erase(t,a[pos]);
            S.insert(t,v);
            t+=lowbit(t);
        }
        a[pos]=v;
    }
    int count(int L,int R,int v){
        int ans=0,pos1=L-1,pos2=R;
        while(pos1){
            ans-=S.count(pos1,v);
            pos1-=lowbit(pos1);
        }
        while(pos2){
            ans+=S.count(pos2,v);
            pos2-=lowbit(pos2);
        }
        return ans;
```

```
    }
    int pre(int L,int R,int v){
        int rk=Rank(L,R,v);
        if(rk)
        return kth(L,R,rk);
        else
        return -Inf;
    }
    int suf(int L,int R,int v){
        int rk=rank(L,R,v)+count(L,R,v);
        if(rk<=R-L+1)
        return kth(L,R,rk);
        else
        return Inf;
    }
}tree;
int main()
{
    int n=read_int(),m=read_int(),opt,l,r,pos,k,ans;
    _rep(i,1,n)
    tree.a[i]=read_int();
    tree.build(n);
    while(m--){
        opt=read_int();
        if(opt==3){
            pos=read_int(),k=read_int();
            tree.update(pos,k);
        }
        else{
            l=read_int(),r=read_int(),k=read_int();
            switch(opt){
                case 1:
                    ans=tree.rank(l,r,k);
                    break;
                case 2:
                    ans=tree.kth(l,r,k);
                    break;
                case 4:
                    ans=tree.pre(l,r,k);
                    break;
                case 5:
                    ans=tree.suf(l,r,k);
                    break;
            }
            enter(ans);
        }
    }
    return 0;
}
```

动态开点权值线段树套名次树版本

使用名次树记录每个权值的点的位置，rank 操作查询 $0\sim v-1$ 区间的满足条件的点的个数。

kth 操作类似在线段树上跑类似名次树的操作，update 操作先删除后插入，其他操作基于这两个基础操作即可得到。

空间复杂度 $O\left(n\log v\right)$，所有操作时间复杂度 $O\left(\log n\log v\right)$。

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
```

```
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=5e4+5,MAXS=MAXN*40,Inf=0x7fffffff;
template <typename T>
struct Treap{
    int pool[MAXS],top,tot;
    int root[MAXS],ch[MAXS][2],r[MAXS],sz[MAXS],cnt[MAXS];
    T val[MAXS];
    int new_node(T v){
        int id=top?pool[top--]:++tot;
        val[id]=v;r[id]=rand();sz[id]=cnt[id]=1;ch[id][0]=ch[id][1]=0;
        return id;
    }
    void push_up(int id){sz[id]=sz[ch[id][0]]+sz[ch[id][1]]+cnt[id];}
    void Rotate(int &id,int dir){
        int t=ch[id][dir^1];
        ch[id][dir^1]=ch[t][dir];ch[t][dir]=id;id=t;
        push_up(ch[id][dir]);push_up(id);
    }
    void Insert(int &id,T v){
        if(!id)
        return id=new_node(v),void();
        if(v==val[id])
        cnt[id]++;
        else{
            int dir=v<val[id]?0:1;
            Insert(ch[id][dir],v);
            if(r[id]<r[ch[id][dir]])
            Rotate(id,dir^1);
        }
        push_up(id);
    }
    void Erase(int &id,T v){
        if(!id)
        return;
        if(v==val[id]){
            if(cnt[id]>1)
            return    cnt[id]--,push_up(id);
            else if(!ch[id][0])
            pool[++top]=id,id=ch[id][1];
            else if(!ch[id][1])
            pool[++top]=id,id=ch[id][0];
            else{
                int d=r[ch[id][0]]>r[ch[id][1]]?1:0;
                Rotate(id,d);Erase(ch[id][d],v);push_up(id);
            }
```

```
        }
        else{
            if(v<val[id])
            Erase(ch[id][0],v);
            else
            Erase(ch[id][1],v);
            push_up(id);
        }
    }
    int Rank(int id,T v){//有多少个数严格小于v
        if(!id)
        return 0;
        if(v==val[id])
        return sz[ch[id][0]];
        else if(v<val[id])
        return Rank(ch[id][0],v);
        else
        return sz[ch[id][0]]+cnt[id]+Rank(ch[id][1],v);
    }
    T Kth(int id,int rk){
        if(!id)
        return -1;//第rk小的节点不存在
        if(rk>sz[ch[id][0]]+cnt[id])
        return Kth(ch[id][1],rk-sz[ch[id][0]]-cnt[id]);
        else if(rk>sz[ch[id][0]])
        return val[id];
        else
        return Kth(ch[id][0],rk);
    }
    T Pre(int id,T v){
        int pos=id,ans=-Inf;
        while(pos){
            if(val[pos]<v){
                ans=ans<val[pos]?val[pos]:ans;
                pos=ch[pos][1];
            }
            else
            pos=ch[pos][0];
        }
        return ans;
    }
    T Suf(int id,T v){
        int pos=id,ans=Inf;
        while(pos){
            if(v<val[pos]){
                ans=ans<val[pos]?ans:val[pos];
                pos=ch[pos][0];
            }
            else
            pos=ch[pos][1];
```

```
        }
        return ans;
    }
    void insert(int root_id,T v){Insert(root[root_id],v);}
    void erase(int root_id,T v){Erase(root[root_id],v);}
    int rank(int root_id,T v){return Rank(root[root_id],v);}
    T kth(int root_id,int rk){return Kth(root[root_id],rk);}
    T pre(int root_id,T v){return Pre(root[root_id],v);}
    T suf(int root_id,T v){return Suf(root[root_id],v);}
    bool empty(int root_id){return sz[root_id]==0;}
};
const int MAXV=1e8;
struct Tree{
    Treap<int> S;
    int root,a[MAXN],pool[MAXS],top,tot,lson[MAXS],rson[MAXS];
    int New(){
        int k=top?pool[top--]:++tot;
        lson[k]=rson[k]=0;
        return k;
    }
    void Del(int &k){
        pool[++top]=k;
        k=0;
    }
    void Insert(int &k,int lef,int rig,int v,int pos){
        if(!k)k=New();
        S.insert(k,pos);
        if(lef==rig)
        return;
        int mid=lef+rig>>1;
        if(v<=mid)
        Insert(lson[k],lef,mid,v,pos);
        else
        Insert(rson[k],mid+1,rig,v,pos);
    }
    void insert(int pos,int v){Insert(root,0,MAXV,v,pos);}
    void build(int n){
        _rep(i,1,n)
        insert(i,a[i]);
    }
    void Erase(int &k,int lef,int rig,int v,int pos){
        S.erase(k,pos);
        if(lef==rig){
            if(S.empty(k))
            Del(k);
            return;
        }
        int mid=lef+rig>>1;
        if(v<=mid)
        Erase(lson[k],lef,mid,v,pos);
        else
```

```
        Erase(rson[k],mid+1,rig,v,pos);
        if(S.empty(k))
        Del(k);
    }
    void erase(int pos,int v){Erase(root,0,MAXV,v,pos);}
    int Rank(int k,int lef,int rig,int v,int pos1,int pos2){
        if(!k)
        return 0;
        if(rig<=v)
        return S.rank(k,pos2)-S.rank(k,pos1);
        int mid=lef+rig>>1;
        if(mid>=v)
        return Rank(lson[k],lef,mid,v,pos1,pos2);
        else
        return
Rank(lson[k],lef,mid,v,pos1,pos2)+Rank(rson[k],mid+1,rig,v,pos1,pos2);
    }
    int rank(int L,int R,int v){return Rank(root,0,MAXV,v-1,L,R+1)+1;}
    int kth(int L,int R,int rk){
        if(rk==0)
        return -Inf;
        else if(rk>R-L+1)
        return Inf;
        int k=root,lef=0,rig=MAXV,mid,trk;
        R++;rk--;
        while(lef<rig){
            trk=S.rank(lson[k],R)-S.rank(lson[k],L);
            mid=lef+rig>>1;
            if(rk>=trk)
            rk-=trk,k=rson[k],lef=mid+1;
            else
            k=lson[k],rig=mid;
        }
        return lef;
    }
    void update(int pos,int v){
        erase(pos,a[pos]);
        insert(pos,v);
        a[pos]=v;
    }
    int count(int L,int R,int v){
        int k=root,lef=0,rig=MAXV,mid;
        while(lef!=rig){
            if(!k)
            return 0;
            mid=lef+rig>>1;
            if(v<=mid){
                k=lson[k];
                rig=mid;
            }
```

```
            else{
                k=rson[k];
                lef=mid+1;
            }
        }
        return S.rank(k,R+1)-S.rank(k,L);
    }
    int pre(int L,int R,int v){return kth(L,R,rank(L,R,v)-1);}
    int suf(int L,int R,int v){return kth(L,R,rank(L,R,v)+count(L,R,v));}
}tree;
int main()
{
    int n=read_int(),m=read_int(),opt,l,r,pos,k,ans;
    _rep(i,1,n)
    tree.a[i]=read_int();
    tree.build(n);
    while(m--){
        opt=read_int();
        if(opt==3){
            pos=read_int(),k=read_int();
            tree.update(pos,k);
        }
        else{
            l=read_int(),r=read_int(),k=read_int();
            switch(opt){
                case 1:
                    ans=tree.rank(l,r,k);
                    break;
                case 2:
                    ans=tree.kth(l,r,k);
                    break;
                case 4:
                    ans=tree.pre(l,r,k);
                    break;
                case 5:
                    ans=tree.suf(l,r,k);
                    break;
            }
            enter(ans);
        }
    }
    return 0;
}
```