分块

```
//区间更新
#include<stdio.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<string.h>
#define M 100005
#define LL long long
using namespace std;
LL sum[M],add[M];
int K,A[M],n,cnt=0;
void down(int k){
	if(add[k]==0)return;
	for(int i=k*K;i<min(n,(k+1)*K);i++)
		A[i]+=add[k];
	add[k]=0;
}
void up(int k){
	sum[k]=0;
	for(int i=k*K;i<min(n,(k+1)*K);i++)
		sum[k]+=A[i];
}
void update(int a,int b,int x){
	int ka=a/K,kb=b/K,i;
	down(ka);
	if(ka==kb){
		for(i=a;i<=b;i++)
			A[i]+=x;
		up(ka);
	}else{
		for(i=a;i<(ka+1)*K;i++)
			A[i]+=x;
		up(ka);
		for(i=(ka+1);i<kb;i++){
			add[i]+=x;
			sum[i]+=K*x;
		}
		down(kb);
		for(i=kb*K;i<=b;i++)
			A[i]+=x;
		up(kb);
	}
}
```

```
LL query(int a,int b){
	int ka=a/K,kb=b/K,i;
	LL res=0;
	down(ka);
	if(ka==kb){
		for(i=a;i<=b;i++)
			res+=A[i];
	}else{
		for(i=a;i<(ka+1)*K;i++)
			res+=A[i];
		for(i=(ka+1);i<kb;i++){
			res+=sum[i];
		}
		down(kb);
		for(i=kb*K;i<=b;i++)
			res+=A[i];
	}
	return res;
}
int main(){
	int m,i,j,a,b,x;
	scanf("%d %d",&n,&m);
	for(i=0;i<n;i++)
		scanf("%d",&A[i]);
	K=int(sqrt(n*1.0));
	for(i=0;i<n;i+=K){
		sum[cnt]=0;
		for(j=i;j<i+K;j++)
			sum[cnt]+=A[j];
		cnt++;
	}
	char str[100];
	while(m--&&scanf("%s %d %d",str,&a,&b)){
		if(str[0]=='Q')cout<<query(a-1,b-1)<<endl;
		else{
			scanf("%d",&x);
			update(a-1,b-1,x);
		}
	}
	return 0;
}
```

主席树

```
int n,q,m,tot;
int a[N],t[N];
int T[M],lson[M],rson[M],c[M];
void Init_hash(){
    for(int i=1;i<=n;i++)
        t[i]=a[i];
    sort(t+1,t+1+n);
    m=unique(t+1,t+1+n)-t-1;
}
int bulid(int l,int r){
    int root=tot++;
    c[root]=0;
    if(l!=r){
        int mid=(l+r)>>1;
        lson[root]=bulid(l,mid);
        rson[root]=bulid(mid+1,r);
    }
    return root;
}
int hash(int x){
    return lower_bound(t+1,t+1+m,x)-t;
}
int update(int root,int pos,int val){
    int newroot=tot++,tmp=newroot;
    c[newroot]=c[root]+val;
    int l=1,r=m;
    while(l<r){
        int mid=(l+r)>>1;
        //cout<<l<<" "<<r<<endl;
        if(pos<=mid){
            lson[newroot]=tot++;rson[newroot]=rson[root];
            newroot=lson[newroot];root=lson[root];
            r=mid;
        }
        else{
            rson[newroot]=tot++;lson[newroot]=lson[root];
            newroot=rson[newroot];root=rson[root];
            l=mid+1;
        }
        c[newroot]=c[root]+val;
    }
    return tmp;
}
```

```
int query(int left_root,int right_root,int k){
	int l=1,r=m;
	while(l<r){
		int mid=(l+r)>>1;
		if(c[lson[left_root]]-c[lson[right_root]]>=k){
			r=mid;
			left_root=lson[left_root];
			right_root=lson[right_root];
		}
		else{
			l=mid+1;
			k-=c[lson[left_root]]-c[lson[right_root]];
			left_root=rson[left_root];
			right_root=rson[right_root];
		}
	}
	return l;
}
int main(){
	while(scanf("%d%d",&n,&q)!=EOF){
		tot=0;
		for(int i=1;i<=n;i++)
			scanf("%d",&a[i]);
		Init_hash();
		T[n+1]=bulid(1,m);
		for(int i=n;i;i--){
			int pos=hash(a[i]);
			T[i]=update(T[i+1],pos,1);
		}
		while(q--){
			int l,r,k;
			scanf("%d%d%d",&l,&r,&k);
			printf("%d\n",t[query(T[l],T[r+1],k)]);
		}
	}
	return 0;
}
```