数据结构 & STL

堆

```
#include<cstdio>
#include<queue>
#include<vector>
using namespace std;
int n,m;
priority_queue<int,vector<int>,greater<int> >q;
int main(){
	scanf("%d",&n);
	for(int i=1;i<=n;i++){
		scanf("%d",&m);
		if(m==1){
			scanf("%d",&m);
			q.push(m);
		}
		else if(m==2){
			printf("%d\n",q.top());
		}
		else{
			q.pop();
		}
	}
	return 0;
}

```

map

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<map>
using namespace std;
map<string,int>mp;
char a[200]="aaaaa";
char b[220]="aaaaa";
struct Edge{
	int u,v,w,next;
	bool operator<(const Edge &x)const{return w<x.w;}
};
int main(){
	mp[a]=1;
	printf("%d",mp.size());
	mp.erase("aaaaa");
	//sort,swap,clear,empty,insert
	return 0;
}
```

rand

```
#include<iostream>
#include<cstdio>
#include <stdlib.h>
#include <time.h>
using namespace std;
int a=7,b=20,n,m;
int main(){
        //freopen("flood.in","w",stdout);
        cin>>n>>m;
        cout<<n<<' '<<m<<endl;
    srand((unsigned)time(NULL));
        for(int i=1;i<=n;i++){
                for(int j=1;j<=m;j++){
                        cout<<(rand()%(b-a+1))<<' ';//
                }
                cout<<endl;
        }
    cout<<endl;

    return 0;
}
```

vector,priority_queue

```
//empty,size,swap,clear
#include<iostream>
#include<cstdio>
#include<vector>
#include<queue>
using namespace std;
priority_queue<int,vector<int>,greater<int> >q;

vector<int>a[22];
int main(){
        int b=5;
        a[1].push_back(b);
        cout<<a[1][0];
        a[1].pop_back();
        return 0;
}
```

stl之字典序(number)

```
#include<iostream>
#include<cstdio>
#include<algorithm>//!!
using namespace std;
```

```
int b[102];
int main(){
	int n;
	scanf("%d",&n);
	for(int i=1;i<=n;i++){
		b[i]=i;
	}
	for(int i=1;i<=n;i++){
		printf("]",b[i]);
	}
	cout<<endl;
	while(next_permutation(b+1,b+1+n)==true){
		for(int i=1;i<=n;i++){
			printf("]",b[i]);
		}
		cout<<endl;
	}
	return 0;
}
```

stl之字典序char

```
#include<iostream>
#include<cstdio>
#include<algorithm>//!!
using namespace std;
char b[102];
int main(){
	int n;
	scanf("%d",&n);
	char a;
	scanf("%c",&a);
	for(int i=1;i<=n;i++){
		scanf("%c",&a);
		b[i]=a;
	}
	sort(b+1,b+1+n);
	for(int i=1;i<=n;i++){
		printf("%c",b[i]);
	}
	cout<<endl;
	while(next_permutation(b+1,b+1+n)==true){//prev
		for(int i=1;i<=n;i++){
			printf("%c",b[i]);
		}
		cout<<endl;
	}
	return 0;
}
```

ST表

```cpp
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
//ST表，静态RMQ
int n,m,a[100020],f[100020][33];
void ST(){
	int k=log(n)/log(2);
	for(int i=1;i<=n;i++){
		f[i][0]=a[i];
	}
	for(int j=1;j<=k;j++){
		for(int i=1;i<=n;i++){
			if(i+(1<<j)-1<=n){
				f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
			}
		}
	}
}
int RMQ(int L,int R){
	int k=log(R-L+1)/log(2);
	return max(f[L][k],f[R-(1<<k)+1][k]);
}
int main(){
	scanf("%d%d",&n,&m);
	for(int i=1;i<=n;i++){
		scanf("%d",&a[i]);
	}
	ST();
	int l,r;
	for(int i=1;i<=m;i++){
		scanf("%d%d",&l,&r);
		printf("%d\n",RMQ(l,r));
	}
	return 0;
}
```

树状数组 单点修改 区间查询

```cpp
//树状数组入门
#include<iostream>
#include<cstdio>
using namespace std;
int a[100020],tree[1000020];
int n,m;
int lowbit(int k){
	return k&-k;
}
void add(int k,int num){
	while(k<=n){
		tree[k]+=num;
		k+=lowbit(k);
	}
}
int read(int k){
        int sum=0;
        while(k){
                sum+=tree[k];
                k-=lowbit(k);
        }
        return sum;
}
int main(){
        scanf("%d%d",&n,&m);
        for(int i=1;i<=n;i++){
                scanf("%d",&a[i]);
                add(i,a[i]);
        }
        int x,y;
        for(int i=1;i<=m;i++){
                scanf("%d%d",&x,&y);
                add(x,y-a[x]);
        }
        printf("%d",read(n));
        return 0;
}
```

树状数组 区间修改 单点查询

```
#include<iostream>
#include<cstdio>
using namespace std;
int n,m;
int a[500020],tree[500020];
int lowbit(int k){
        return k&-k;
}
void add(int k,int num){
        while(k<=n){
                tree[k]+=num;
                k+=lowbit(k);
        }
}
int read(int k){
        int sum=0;
        while(k){
                sum+=tree[k];
                k-=lowbit(k);
        }
        return sum;
}
int main(){
        scanf("%d%d",&n,&m);
        int last=0;
        for(int i=1;i<=n;i++){
                scanf("%d",&a[i]);
                add(i,a[i]-last);
                //这里运用了差分思想，假设原本的数据存在a数组中，
                //那么c数组储存的就是c[i]=a[i]-a[i-1]，如果c[1]=a[1]，那么很明显
                //a[i]=c[i]+c[i-1]+c[i-2]+...+c[2]+c[1].
                //这样我们每次单点查询的时候只要加上c数组的前缀就可以了。
                last=a[i];
        }
        int command,x,y,l;
        for(int i=1;i<=m;i++){
                scanf("%d",&command);
                if(command==1){
                        scanf("%d%d%d",&x,&y,&l);
                        add(x,l);
                        add(y+1,-l);
                }
                else{
                        scanf("%d",&x);
                        printf("%d\n",read(x));
                }
        }
        return 0;
}
```

树状数组 区间修改 区间查询

```
/*
 * +-------------+
 * |算法正确性证明|          <=不不不，这应该叫题解
 * +-------------+
 *   设原数组第i位的值为a[i],设d[i]=a[i]-a[i-1]
 *   所以对a的前x位求和，得到式子
 *     x       x   i
 *     Σa[i]= Σ  Σd[j]
 *    i=1     i=1 j=1
 *   然后通过一些并不显然的变形（可以从求和的意义理解），得到
 *     x      x
 *     Σa[i]=Σ(x-i+1)d[i]
 *    i=1    i=1
 *   然后通过进一步推理得到
 *     x      x            x
 *     Σa[i]=Σ(x+1)*d[i]-Σi*d[i]
 *    i=1    i=1          i=1
 *   所以我们可以维护两个树状数组，一个名叫delta，针对d[i]，一个名叫deltai，针对i*d[i
 * */
#include<cstdio>
#include<iostream>
#include<cstring>
#include<cctype>
using namespace std;
const int maxn=200005;
typedef long long ull;
ull delta[maxn],deltai[maxn];
int a[maxn];
int n,q;
inline int lowbit(int x){
        return x&(-x);
}
inline void add(int x,ull y){
        for(int i=x;i<=n;i+=lowbit(i)){
                delta[i]+=y;
                deltai[i]+=x*y;
        }
}
inline ull sum(int x){
        ull ans=0;
        for(int i=x;i>0;i-=lowbit(i)){
                ans+=(x+1)*delta[i]-deltai[i];
        }
        return ans;
}
int main(){
        scanf("%d",&n);
        int last=0;
        for(int i=1;i<=n;i++){
                int tmp;
                scanf("%d",&tmp);
                add(i,tmp-last);
                last=tmp;
        }
        int q;
        scanf("%d",&q);
        while(q--){
                int cho;
                scanf("%d",&cho);
                if(cho==1){
                        int t1,t2;
                        long long t3;
                        scanf("%d%d%d",&t1,&t2,&t3);
                        add(t1,t3);
                        add(t2+1,-t3);
                }
                else{
                        int t1,t2;
                        scanf("%d%d",&t1,&t2);
                        cout<<sum(t2)-sum(t1-1)<<endl;
                }
        }
        return 0;
}
```

线段树求区间最大

```
#include<iostream>
using namespace std;
const int MAX_N=1<<17;//1e5
const int INT_MIN=-1;
```

```
int n,dat[2*MAX_N-1],x,y;
void update(int k,int a){
        k+=n-1;
        dat[k]=a;
        while(k>0){
                k=(k-1)/2;
                dat[k]=max(dat[k*2+1],dat[k*2+2]);
        }
}
void init(int n_){
        n=1;
        while(n<n_){
                n*=2;
        }
        for(int i=0;i<2*n-1;i++){
                dat[i]=INT_MIN;
        }
        for(int i=0;i<n_;i++){
                scanf("%d",&x);
                update(i,x);
        }
}

int query(int a,int b,int k,int l,int r){
        if(r<=a||b<=l){
                return INT_MIN;
        }
        if(a<=l&&r<=b){
                return dat[k];
        }
        else{
                int vl=query(a,b,k*2+1,l,(l+r)/2);
                int vr=query(a,b,k*2+2,(l+r)/2,r);
                return max(vl,vr);
        }
}
int main(){
        cin>>n;
        init(n);
        int q;
        cin>>q;
        for(int i=1;i<=q;i++){
                scanf("%d%d",&x,&y);
                printf("%d\n",query(x,y+1,0,0,n));
        }

        return 0;
}

```

线段树区间修改查询（带lazy）

```
#include<iostream>
```

```cpp
#include<cstring>
using namespace std;
struct tree
{
        int start,end;
        long long value,lazy;
};
int n,q,x,ss,ee,l,r,v;
tree a[800010];
void make_tree(int start,int end,int now)
{
        int mid;
        a[now].start=start;
        a[now].end=end;
        a[now].value=0;
        a[now].lazy=0;
        mid=(start+end)/2;
        if (start!=end)
        {
                make_tree(start,mid,now*2);
                make_tree(mid+1,end,now*2+1);
        }
}
void add(int s,int e,int num,int now)
{
        int mid;
        mid=(a[now].start+a[now].end)/2;
        if ((s==a[now].start)&&(e==a[now].end))
        {
                a[now].lazy+=num;
                return;
        }
        if ((s<=mid)&&(e<=mid))
        {
                a[now].value+=(e-s+1)*num;
                if (a[now].lazy==0)
                        add(s,e,num,now*2);
                else
                {
                        a[now].value+=(a[now].end-a[now].start+1)*a[now].lazy;
                        a[now*2+1].lazy+=a[now].lazy;
                        a[now*2].lazy+=a[now].lazy;
                        a[now].lazy=0;
                        add(s,e,num,now*2);
                }
        }
        if ((s>mid)&&(e>mid))
        {
                a[now].value+=(e-s+1)*num;
                if (a[now].lazy==0)
                        add(s,e,num,now*2+1);
                else
                {
                        a[now].value+=(a[now].end-a[now].start+1)*a[now].lazy;
                        a[now*2].lazy+=a[now].lazy;
                        a[now*2+1].lazy+=a[now].lazy;
                        a[now].lazy=0;
                        add(s,e,num,now*2+1);
                }
        }
        if ((s<=mid)&&(e>mid))
        {
                a[now].value+=(e-s+1)*num;
                if (a[now].lazy==0)
                {
                        add(s,mid,num,now*2);
                        add(mid+1,e,num,now*2+1);
                }
                else
                {
                        a[now].value+=(a[now].end-a[now].start+1)*a[now].lazy;
                        a[now*2].lazy+=a[now].lazy;
                        a[now*2+1].lazy+=a[now].lazy;
                        a[now].lazy=0;
                        add(s,mid,num,now*2);
                        add(mid+1,e,num,now*2+1);
                }
        }
}
long long find(int s,int e,int now)
{
        int mid;
        mid=(a[now].start+a[now].end)/2;
        if ((s==a[now].start)&&(e==a[now].end))
                if (a[now].lazy==0)
                        return (a[now].value);
                else
                        return (a[now].value+a[now].lazy*(e-s+1));
        if ((s<=mid)&&(e<=mid))
                if (a[now].lazy==0)
                        return (find(s,e,now*2));
                else
                        return (find(s,e,now*2)+(e-s+1)*a[now].lazy);
        if ((s>mid)&&(e>mid))
                if (a[now].lazy==0)
                        return (find(s,e,now*2+1));
                else
                        return (find(s,e,now*2+1)+(e-s+1)*a[now].lazy);
        if ((s<=mid)&&(e>mid))
                if (a[now].lazy==0)
                        return (find(s,mid,now*2)+find(mid+1,e,now*2+1));
                else
                        return (find(s,mid,now*2)+find(mid+1,e,now*2+1)+(e-s+1)*a[now].lazy);
}
int main()
{
        cin>>n;
        make_tree(1,n,1);
        for (int i=1;i<=n;i++)
        {
```

```
			cin>>x;
			add(i,i,x,1);
		}
		cin>>q;
		for (int i=1;i<=q;i++)
		{
			cin>>x;
			if (x==1)
			{
				cin>>l>>r>>v;
				add(l,r,v,1);
			}
			if (x==2)
			{
				cin>>l>>r;
				cout<<find(l,r,1)<<endl;
			}
		}
		return 0;
}
```