

数据结构 & STL

堆

```
1 #include<cstdio>
2 #include<queue>
3 #include<vector>
4 using namespace std;
5 int n,m;
6 priority_queue<int,vector<int>,greater<int> >q;
7 int main(){
8     scanf("%d",&n);
9     for(int i=1;i<=n;i++){
10         scanf("%d",&m);
11         if(m==1){
12             scanf("%d",&m);
13             q.push(m);
14         }
15         else if(m==2){
16             printf("%d\n",q.top());
17         }
18         else{
19             q.pop();
20         }
21     }
22     return 0;
23 }
24
```

map

```
1 #include<iostream>
2 #include<cstdio>
3 #include<cstring>
4 #include<map>
5 using namespace std;
6 map<string,int>mp;
7 char a[200]="aaaaa";
8 char b[220]="aaaaa";
9 struct Edge{
10     int u,v,w,next;
11     bool operator<(const Edge &x)const{return w<x.w;}
12 };
13 int main(){
14     mp[a]=1;
15     printf("%d",mp.size());
16     mp.erase("aaaaa");
17     //sort,swap,clear,empty,insert
18     return 0;
19 }
```

rand

```
1 #include<iostream>
2 #include<cstdio>
3 #include <stdlib.h>
4 #include <time.h>
5 using namespace std;
6 int a=7,b=20,n,m;
7 int main(){
8     //freopen("flood.in","w",stdout);
9     cin>>n>>m;
10    cout<<n<<' '<<m<<endl;
11    srand((unsigned)time(NULL));
12    for(int i=1;i<=n;i++){
13        for(int j=1;j<=m;j++){
14            cout<<(rand()%(b-a+1))<<' ';//
15        }
16        cout<<endl;
17    }
18    cout<<endl;
19
20    return 0;
21 }
```

vector,priority_queue

```
1 //empty,size,swap,clear
2 #include<iostream>
3 #include<cstdio>
4 #include<vector>
5 #include<queue>
6 using namespace std;
7 priority_queue<int,vector<int>,greater<int> >q;
8
9 vector<int>a[22];
10 int main(){
11     int b=5;
12     a[1].push_back(b);
13     cout<<a[1][0];
14     a[1].pop_back();
15     return 0;
16 }
```

stl之字典序(number)

```
1 #include<iostream>
2 #include<cstdio>
3 #include<algorithm>///!!
4 using namespace std;
```

```

5 int b[102];
6 int main(){
7     int n;
8     scanf("%d",&n);
9     for(int i=1;i<=n;i++){
10         b[i]=i;
11     }
12     for(int i=1;i<=n;i++){
13         printf("]",b[i]);
14     }
15     cout<<endl;
16     while(next_permutation(b+1,b+1+n)==true){
17         for(int i=1;i<=n;i++){
18             printf("]",b[i]);
19         }
20         cout<<endl;
21     }
22     return 0;
23 }

```

stl之字典序char

```

1 #include<iostream>
2 #include<cstdio>
3 #include<algorithm>///!!
4 using namespace std;
5 char b[102];
6 int main(){
7     int n;
8     scanf("%d",&n);
9     char a;
10    scanf("%c",&a);
11    for(int i=1;i<=n;i++){
12        scanf("%c",&a);
13        b[i]=a;
14    }
15    sort(b+1,b+1+n);
16    for(int i=1;i<=n;i++){
17        printf("%c",b[i]);
18    }
19    cout<<endl;
20    while(next_permutation(b+1,b+1+n)==true){//prev
21        for(int i=1;i<=n;i++){
22            printf("%c",b[i]);
23        }
24        cout<<endl;
25    }
26    return 0;
27 }

```

ST表

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cmath>
4  using namespace std;
5  //ST表, 静态RMQ
6  int n,m,a[100020],f[100020][33];
7  void ST(){
8      int k=log(n)/log(2);
9      for(int i=1;i<=n;i++){
10         f[i][0]=a[i];
11     }
12     for(int j=1;j<=k;j++){
13         for(int i=1;i<=n;i++){
14             if(i+(1<<j)-1<=n){
15                 f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
16             }
17         }
18     }
19 }
20 int RMQ(int L,int R){
21     int k=log(R-L+1)/log(2);
22     return max(f[L][k],f[R-(1<<k)+1][k]);
23 }
24 int main(){
25     scanf("%d%d",&n,&m);
26     for(int i=1;i<=n;i++){
27         scanf("%d",&a[i]);
28     }
29     ST();
30     int l,r;
31     for(int i=1;i<=m;i++){
32         scanf("%d%d",&l,&r);
33         printf("%d\n",RMQ(l,r));
34     }
35     return 0;
36 }

```

树状数组 单点修改 区间查询

```

1  //树状数组入门
2  #include<iostream>
3  #include<cstdio>
4  using namespace std;
5  int a[100020],tree[100020];
6  int n,m;
7  int lowbit(int k){
8      return k&-k;
9  }
10 void add(int k,int num){
11     while(k<=n){
12         tree[k]+=num;
13         k+=lowbit(k);
14     }

```

```

15 }
16 int read(int k){
17     int sum=0;
18     while(k){
19         sum+=tree[k];
20         k-=lowbit(k);
21     }
22     return sum;
23 }
24 int main(){
25     scanf("%d%d",&n,&m);
26     for(int i=1;i<=n;i++){
27         scanf("%d",&a[i]);
28         add(i,a[i]);
29     }
30     int x,y;
31     for(int i=1;i<=m;i++){
32         scanf("%d%d",&x,&y);
33         add(x,y-a[x]);
34     }
35     printf("%d",read(n));
36     return 0;
37 }

```

树状数组 区间修改 单点查询

```

1  #include<iostream>
2  #include<cstdio>
3  using namespace std;
4  int n,m;
5  int a[500020],tree[500020];
6  int lowbit(int k){
7      return k&-k;
8  }
9  void add(int k,int num){
10     while(k<=n){
11         tree[k]+=num;
12         k+=lowbit(k);
13     }
14 }
15 int read(int k){
16     int sum=0;
17     while(k){
18         sum+=tree[k];
19         k-=lowbit(k);
20     }
21     return sum;
22 }
23 int main(){
24     scanf("%d%d",&n,&m);
25     int last=0;
26     for(int i=1;i<=n;i++){
27         scanf("%d",&a[i]);

```

```

28         add(i,a[i]-last);
29         //这里运用了差分思想,假设原本的数据存在a数组中,
30         //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
31         //a[i]=c[i]+c[i-1]+c[i-2]+...+c[2]+c[1].
32         //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。
33         last=a[i];
34     }
35     int command,x,y,l;
36     for(int i=1;i<=m;i++){
37         scanf("%d",&command);
38         if(command==1){
39             scanf("%d%d%d",&x,&y,&l);
40             add(x,l);
41             add(y+1,-l);
42         }
43         else{
44             scanf("%d",&x);
45             printf("%d\n",read(x));
46         }
47     }
48     return 0;
49 }

```

树状数组 区间修改 区间查询

```

1  /*
2  * +-----+
3  * |算法正确性证明|          <=不不不, 这应该叫题解
4  * +-----+
5  *   设原数组第i位的值为a[i],设d[i]=a[i]-a[i-1]
6  *   所以对a的前x位求和,得到式子
7  *       x       x   i
8  *   Σa[i]= Σ  Σd[j]
9  *   i=1       i=1 j=1
10 *   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
11 *       x       x
12 *   Σa[i]=Σ(x-i+1)d[i]
13 *   i=1       i=1
14 *   然后通过进一步推理得到
15 *       x       x       x
16 *   Σa[i]=Σ(x+1)*d[i]-Σi*d[i]
17 *   i=1       i=1       i=1
18 *   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]
19 * */
20 #include<cstdio>
21 #include<iostream>
22 #include<cstring>
23 #include<cctype>
24 using namespace std;
25 const int maxn=200005;
26 typedef long long ull;
27 ull delta[maxn],deltai[maxn];
28 int a[maxn];

```

```

29 int n,q;
30 inline int lowbit(int x){
31     return x&(-x);
32 }
33 inline void add(int x,ull y){
34     for(int i=x;i<=n;i+=lowbit(i)){
35         delta[i]+=y;
36         deltai[i]+=x*y;
37     }
38 }
39 inline ull sum(int x){
40     ull ans=0;
41     for(int i=x;i>0;i-=lowbit(i)){
42         ans+=(x+1)*delta[i]-deltai[i];
43     }
44     return ans;
45 }
46 int main(){
47     scanf("%d",&n);
48     int last=0;
49     for(int i=1;i<=n;i++){
50         int tmp;
51         scanf("%d",&tmp);
52         add(i,tmp-last);
53         last=tmp;
54     }
55     int q;
56     scanf("%d",&q);
57     while(q--){
58         int cho;
59         scanf("%d",&cho);
60         if(cho==1){
61             int t1,t2;
62             long long t3;
63             scanf("%d%d%d",&t1,&t2,&t3);
64             add(t1,t3);
65             add(t2+1,-t3);
66         }
67         else{
68             int t1,t2;
69             scanf("%d%d",&t1,&t2);
70             cout<<sum(t2)-sum(t1-1)<<endl;
71         }
72     }
73     return 0;
74 }

```

线段树求区间最大

```

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

```

```

5  int n,dat[2*MAX_N-1],x,y;
6  void update(int k,int a){
7      k+=n-1;
8      dat[k]=a;
9      while(k>0){
10         k=(k-1)/2;
11         dat[k]=max(dat[k*2+1],dat[k*2+2]);
12     }
13 }
14 void init(int n_){
15     n=1;
16     while(n<n_){
17         n*=2;
18     }
19     for(int i=0;i<2*n-1;i++){
20         dat[i]=INT_MIN;
21     }
22     for(int i=0;i<n;i++){
23         scanf("%d",&x);
24         update(i,x);
25     }
26 }
27
28 int query(int a,int b,int k,int l,int r){
29     if(r<=a||b<=1){
30         return INT_MIN;
31     }
32     if(a<=l&&r<=b){
33         return dat[k];
34     }
35     else{
36         int vl=query(a,b,k*2+1,l,(l+r)/2);
37         int vr=query(a,b,k*2+2,(l+r)/2,r);
38         return max(vl,vr);
39     }
40 }
41 int main(){
42     cin>>n;
43     init(n);
44     int q;
45     cin>>q;
46     for(int i=1;i<=q;i++){
47         scanf("%d%d",&x,&y);
48         printf("%d\n",query(x,y+1,0,0,n));
49     }
50
51     return 0;
52 }
53

```

线段树区间修改查询（带lazy）

```
1  #include<iostream>
```



```

2  #include<cstring>
3  using namespace std;
4  struct tree
5  {
6      int start,end;
7      long long value,lazy;
8  };
9  int n,q,x,ss,ee,l,r,v;
10 tree a[800010];
11 void make_tree(int start,int end,int now)
12 {
13     int mid;
14     a[now].start=start;
15     a[now].end=end;
16     a[now].value=0;
17     a[now].lazy=0;
18     mid=(start+end)/2;
19     if (start!=end)
20     {
21         make_tree(start,mid,now*2);
22         make_tree(mid+1,end,now*2+1);
23     }
24 }
25 void add(int s,int e,int num,int now)
26 {
27     int mid;
28     mid=(a[now].start+a[now].end)/2;
29     if ((s==a[now].start)&&(e==a[now].end))
30     {
31         a[now].lazy+=num;
32         return;
33     }
34     if ((s<=mid)&&(e<=mid))
35     {
36         a[now].value+=(e-s+1)*num;
37         if (a[now].lazy==0)
38             add(s,e,num,now*2);
39         else
40         {
41             a[now].value+=(a[now].end-a[now].start+1)*a[now].lazy;
42             a[now*2+1].lazy+=a[now].lazy;
43             a[now*2].lazy+=a[now].lazy;
44             a[now].lazy=0;
45             add(s,e,num,now*2);
46         }
47     }
48     if ((s>mid)&&(e>mid))
49     {
50         a[now].value+=(e-s+1)*num;
51         if (a[now].lazy==0)
52             add(s,e,num,now*2+1);
53         else
54         {
55             a[now].value+=(a[now].end-a[now].start+1)*a[now].lazy;
56             a[now*2].lazy+=a[now].lazy;

```

```

57         a[now*2+1].lazy+=a[now].lazy;
58         a[now].lazy=0;
59         add(s,e,num,now*2+1);
60     }
61 }
62 if ((s<=mid)&&(e>mid))
63 {
64     a[now].value+=(e-s+1)*num;
65     if (a[now].lazy==0)
66     {
67         add(s,mid,num,now*2);
68         add(mid+1,e,num,now*2+1);
69     }
70     else
71     {
72         a[now].value+=(a[now].end-a[now].start+1)*a[now].lazy;
73         a[now*2].lazy+=a[now].lazy;
74         a[now*2+1].lazy+=a[now].lazy;
75         a[now].lazy=0;
76         add(s,mid,num,now*2);
77         add(mid+1,e,num,now*2+1);
78     }
79 }
80 }
81 long long find(int s,int e,int now)
82 {
83     int mid;
84     mid=(a[now].start+a[now].end)/2;
85     if ((s==a[now].start)&&(e==a[now].end))
86         if (a[now].lazy==0)
87             return (a[now].value);
88         else
89             return (a[now].value+a[now].lazy*(e-s+1));
90     if ((s<=mid)&&(e<=mid))
91         if (a[now].lazy==0)
92             return (find(s,e,now*2));
93         else
94             return (find(s,e,now*2)+(e-s+1)*a[now].lazy);
95     if ((s>mid)&&(e>mid))
96         if (a[now].lazy==0)
97             return (find(s,e,now*2+1));
98         else
99             return (find(s,e,now*2+1)+(e-s+1)*a[now].lazy);
100     if ((s<=mid)&&(e>mid))
101         if (a[now].lazy==0)
102             return (find(s,mid,now*2)+find(mid+1,e,now*2+1));
103         else
104             return (find(s,mid,now*2)+find(mid+1,e,now*2+1)+(e-s+1)*a[
105 ]
106 int main()
107 {
108     cin>>n;
109     make_tree(1,n,1);
110     for (int i=1;i<=n;i++)
111     {

```

```
112         cin>>x;
113         add(i,i,x,1);
114     }
115     cin>>q;
116     for (int i=1;i<=q;i++)
117     {
118         cin>>x;
119         if (x==1)
120         {
121             cin>>l>>r>>v;
122             add(l,r,v,1);
123         }
124         if (x==2)
125         {
126             cin>>l>>r;
127             cout<<find(l,r,1)<<endl;
128         }
129     }
130     return 0;
131 }
```

