ST表

[P3865 【模板】ST 表](https://www.luogu.com.cn/problem/P3865)

[参考1](https://www.luogu.com.cn/blog/_post/31037)

[参考2](https://oi-wiki.org/ds/sparse-table/)

```cpp
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
using namespace std;

const int MAXJ=log2((int)(1e5))+1;//17
const int MAXN=1e6+10;

inline int IntRead(){//快速读入比关闭同步的cin快得多
    char ch=getchar();
    int s=0,w=1;
    while(ch<'0' || ch>'9'){
        if(ch=='-') w=-1;
        ch=getchar();
    }
    while(ch>='0' && ch<='9'){
        s=s*10+ch-'0',
        ch=getchar();
    }
    return s*w;
}

int Max[MAXN][17];

int Query(int l,int r){
    int k=log2(r-l+1);
    return max(Max[l][k],Max[r-(1<<k)+1][k]);//把拆出来的区间分别取最值
}

int main(){
    int n=IntRead(),m=IntRead();
    FOR(i,1,n)
        Max[i][0]=IntRead();
    FOR(j,1,MAXJ)
        for(int i=1;i+(1<<j)-1<=n;i++)//注意这里要控制边界
            Max[i][j]=max(Max[i][j-1],Max[i+(1<<(j-1))][j-1]);
    FOR(i,1,m){
        int l=IntRead(),r=IntRead();
        printf("%d\n",Query(l,r));
    }
    return 0;
}
```

线段树

```cpp
#include<iostream>
using namespace std;
#define maxn 100017  //元素总个数
#define ls l,m,rt<<1
#define rs m+1,r,rt<<1|1
#define LL long long
LL Sum[maxn<<2],Add[maxn<<2];//Sum求和，Add为懒惰标记
LL A[maxn],n;//存原数组数据下标[1,n]
//PushUp函数更新节点信息 ，这里是求和

void PushUp(LL rt){
    Sum[rt]=Sum[rt<<1]+Sum[rt<<1|1];
}

//Build函数建树
void Build(LL l,LL r,LL rt){ //l,r表示当前节点区间，rt表示当前节点编号
    if(l==r) {//若到达叶节点
        Sum[rt]=A[l];//储存数组值
        return;
    }
    LL m=(l+r)>>1;
    //左右递归
    Build(l,m,rt<<1);
    Build(m+1,r,rt<<1|1);
    //更新信息
    PushUp(rt);
}

void UpdateS(LL L,LL C,LL l,LL r,LL rt){//l,r表示当前节点区间，rt表示当前节点编号
    if(l==r){//到叶节点，修改
        Sum[rt]+=C;
        return;
    }
    LL m=(l+r)>>1;
    //根据条件判断往左子树调用还是往右
    if(L <= m) UpdateS(L,C,l,m,rt<<1);
    else       UpdateS(L,C,m+1,r,rt<<1|1);
    PushUp(rt);//子节点更新了，所以本节点也需要更新信息
}

void PushDown(LL rt,LL ln,LL rn){
    //ln,rn为左子树，右子树的数字数量。
    if(Add[rt]){
        //下推标记
        Add[rt<<1]+=Add[rt];
        Add[rt<<1|1]+=Add[rt];
        //修改子节点的Sum使之与对应的Add相对应
        Sum[rt<<1]+=Add[rt]*ln;
        Sum[rt<<1|1]+=Add[rt]*rn;
        //清除本节点标记
        Add[rt]=0;
```

```
    }
}

void Update(LL L,LL R,LL C,LL l,LL r,LL rt){//L,R表示操作区间，l,r表示当前节点区间，rt表示当前节点编号
    if(L <= l && r <= R){//如果本区间完全在操作区间[L,R]以内
        Sum[rt]+=C*(r-l+1);//更新数字和，向上保持正确
        Add[rt]+=C;//增加Add标记，表示本区间的Sum正确，子区间的Sum仍需要根据Add的值来调整
        return ;
    }
    LL m=(l+r)>>1;
    PushDown(rt,m-l+1,r-m);//下推标记
    //这里判断左右子树跟[L,R]有无交集，有交集才递归
    if(L <= m) Update(L,R,C,l,m,rt<<1);
    if(R >  m) Update(L,R,C,m+1,r,rt<<1|1);
    PushUp(rt);//更新本节点信息
}

void UpdateP(LL L,LL R,LL C,LL l,LL r,LL rt){//L,R表示操作区间，l,r表示当前节点区间，rt表示当前节点编号
    if(L <= l && r <= R){//如果本区间完全在操作区间[L,R]以内
        Sum[rt]*=(C*(r-l+1));//更新数字和，向上保持正确
        Add[rt]*=C;//增加Add标记，表示本区间的Sum正确，子区间的Sum仍需要根据Add的值来调整
        return ;
    }
    LL m=(l+r)>>1;
    PushDown(rt,m-l+1,r-m);//下推标记
    //这里判断左右子树跟[L,R]有无交集，有交集才递归
    if(L <= m) UpdateP(L,R,C,l,m,rt<<1);
    if(R >  m) UpdateP(L,R,C,m+1,r,rt<<1|1);
    PushUp(rt);//更新本节点信息
}

LL Query(LL L,LL R,LL l,LL r,LL rt){//L,R表示操作区间，l,r表示当前节点区间，rt表示当前节点编号
    if(L <= l && r <= R){
        //在区间内，直接返回
        return Sum[rt];
    }
    LL m=(l+r)>>1;
    //下推标记，否则Sum可能不正确
    PushDown(rt,m-l+1,r-m);

    //累计答案
    LL ANS=0;
    if(L <= m) ANS+=Query(L,R,l,m,rt<<1);
    if(R >  m) ANS+=Query(L,R,m+1,r,rt<<1|1);
    return ANS;
}
```

```
int main(){
    LL n,m,p;
    cin>>n>>m>>p;
    for(int i=1;i<=n;i++)
        cin>>A[i];
    Build(1,n,1);
    LL o,x,y,k;
    for(int i=1;i<=m;i++){
        cin>>o;
        if(o==1){
            cin>>x>>y>>k;
            for(int j=x;j<=y;j++){
                UpdateS(j,k,1,n,1);
            }
            //UpdateP(x,y,k,1,n,1);
        }
        if(o==2){
            cin>>x>>y>>k;
            Update(x,y,k,1,n,1);
        }
        if(o==3){
            cin>>x>>y;
            cout<<Query(x,y,1,n,1)%p<<endl;
        }
    }
}
```