插队问题

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1 | 1
#define root 1,n,1
const int N=2e5+2;
const int inf=0x3f3f3f3f;
int SUM[4*N];

int scan() {
    int res=0;
    bool flag=false;
    int ch;
    if ( ( ch=getchar() ) == '-') flag=true;
    else if (ch>='0' && ch<='9') res=ch-'0';
    while( (ch=getchar()) >='0' && ch<='9' )
        res=res*10+ch-'0';
    return flag ? -res : res;
}

void Out(int x) {
    if (x>9) Out(x/10);
    putchar(x%10 + '0' );
}

void print(int x) {
    if (x<0) {
        putchar('-');
        Out(-x);
    } else
        Out(x);
}
void PushUp(int rt) {
    SUM[rt]=SUM[rt<<1]+SUM[rt<<1 | 1];
}

void build(int l,int r,int rt) {
    if (l==r){
        SUM[rt]=1;
        return;
    }
    int m=(l+r)>>1;
    build(lson);
    build(rson);
    PushUp(rt);
}
int pos[N],val[N],ans[N];
void update(int p,int v,int l,int r,int rt) {
    --SUM[rt];
    if (l==r){
        ans[l]=v;
        return;
```

```
    }
    int m=(l+r)>>1;
    if (SUM[rt<<1]>=p) update(p,v,lson);
    else update(p-SUM[rt<<1],v,rson);
}
void solve(int n) {
    memset(ans,0,sizeof(ans));
    build(root);
    for (int i=1; i<=n; i++){
        pos[i]=scan();
        pos[i]++;
        val[i]=scan();
    }
    for (int i=n; i>=1; i--) update(pos[i],val[i],root);

    for (int i=1; i<=n; i++) {
        print(ans[i]);
        putchar(' ');
    }
    putchar('\n');
}

int main() {
    //    freopen("datain.txt","r",stdin);
    int n;
    while(~scanf("%d",&n)) {
        solve(n);
    }
    return 0;
}
```

出队问题

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <cmath>
#include <vector>
using namespace std;
typedef long long LL;
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1 | 1
#define root 1,n,1
const int N=5e5+2;
const int inf=0x3f3f3f3f;
int SUM[4*N],MAX[N],D[N];
int read_num() {
    int res=0;
    bool flag=false;
    int ch;
    while( (ch=getchar())==' ' || ch=='\n') ;
    if ( ch == '-') flag=true;
    else if (ch>='0' && ch<='9') res=ch-'0';
    while( (ch=getchar()) >='0' && ch<='9' )
        res=res*10+ch-'0';
    return flag ? -res : res;
}

void Out(int x) {
    if (x>9) Out(x/10);
    putchar(x%10 + '0' );
}

void print(int x) {
    if (x<0) {
        putchar('-');
        Out(-x);
    } else
        Out(x);
}

void build(int l,int r,int rt) {
    SUM[rt]=r-l+1;
    if (l==r) return;
    int m=(l+r)>>1;
    build(lson);
    build(rson);
}

char name[N][15];
int val[N],ans[N];
int Delete(int k,int l,int r,int rt) {
    --SUM[rt];
    if (l==r) return l;
    int m=(l+r)>>1;
    if (SUM[rt<<1]>=k) return Delete(k,lson);
    else return Delete(k-SUM[rt<<1],rson);
}
```

```
void init() {
    memset(D,0,sizeof(D));
    memset(MAX,0,sizeof(MAX));

    for (int i=1; i<N; i++)
        for (int j=i; j<N; j+=i)
            D[j]++;

    MAX[1]=1;
    for (int i=2; i<N; i++) {
        if (D[i]>D[MAX[i-1]]) MAX[i]=i;
        else MAX[i]=MAX[i-1];
    }
}

void solve(int n,int k) {
    build(root);
    for (int i=1; i<=n; i++) {
        scanf("%s",name[i]);
        val[i]=read_num();
    }
    int num=MAX[n];
    int p=0;
    val[0]=0;
    for (int i=0; i<num ; i++) {
        int mod=n-i;
        if (val[p]>0) k=((k-2+val[p])%mod+mod)%mod+1;
        else  k=((k-1+val[p])%mod+mod)%mod+1;
        p=Delete(k,root);
    }
    printf("%s",name[p]);
    putchar(' ');
    print(D[num]);
    putchar('\n');
}

int main() {
    //    freopen("datain.txt","r",stdin);
    int n,p;
    init();
    while(~scanf("%d%d",&n,&p)) {
        solve(n,p);
    }
    return 0;
}
```

挡板

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
#define ls (rt<<1)
#define rs (rt<<1|1)
#define lson ls,l,m
#define rson rs,m+1,r
#define root 1,0,2*N
const int N=8000+5;
const int inf=0x3f3f3f3f;
int SUM[8*N],color[8*N];
bool mark[N+5][N+5];
void PushDown(int rt,int l,int r) {
    if(color[rt]!=-1) {
        color[ls]=color[rs]=color[rt];
        color[rt]=-1;
    }
}

void PushUp(int rt,int l,int r) {
    if(color[ls]==color[rs]) color[rt]=color[ls];
    else color[rt]=-1;
}
void interval_assign(int L,int R,int val,int rt,int l,int r) {
    if (L<=l && r<=R) {
        color[rt]=val;
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) interval_assign(L,R,val,lson);
    if(R>m) interval_assign(L,R,val,rson);
    PushUp(rt,l,r);
}
void query(int curi,int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R && color[rt]!=-1) {
        mark[curi][color[rt]]=true;
        return;
    }
    if (l==r) return;
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if (L<=m) query(curi,L,R,lson);
    if (R>m)  query(curi,L,R,rson);
    PushUp(rt,l,r);
    return ;
}
struct line {
    int y1,y2,x;
    bool operator < (const line & t) const {
        return x<t.x;
    }
} lines[N];
```

```
void solve() {
    //TODO 1: ≥ı ºª∅
    memset(color,-1,sizeof(color));
    memset(mark,false,sizeof(mark));
    //TODO 2: ∂¡»Î
    int n;
    scanf("%d",&n);
    for (int i=1; i<=n; i++) {
        scanf("%d%d%d",&lines[i].y1,&lines[i].y2,&lines[i].x);
        lines[i].y1<<=1;
        lines[i].y2<<=1;
    }
    //TODO 3:
    sort(lines+1,lines+n+1);
    //TODO 4
    for (int i=1; i<=n; i++) {
        query(i,lines[i].y1,lines[i].y2,root);
        interval_assign(lines[i].y1,lines[i].y2,i,root);
    }
    //TODO 5
    int cnt=0;
    for (int i=1; i<=n; i++) {
        for (int j=1; j<i; j++)
            if (mark[i][j]) {
                for (int k=i+1; k<=n; k++)
                    if (mark[k][i] && mark[k][j]) cnt++;
            }
    }

    printf("%d\n",cnt);
}

int main() {
    //    freopen("datain.txt","r",stdin);
    int T;
    scanf("%d",&T);
    while(T--) {
        solve();
    }
    return 0;
}
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;

/*************************************************************************
                                读入数字
 *************************************************************************/
int read_num() {
    int res=0;
    bool flag=false;
    int ch;
    while( (ch=getchar())==' ' || ch=='\n') ;
    if ( ch == '-') flag=true;
    else if (ch>='0' && ch<='9') res=ch-'0';
    while( (ch=getchar()) >='0' && ch<='9' )
        res=res*10+ch-'0';
    return flag ? -res : res;
}

/*************************************************************************
                                输出数字
 *************************************************************************/
void Out(int x) {
    if (x>9) Out(x/10);
    putchar(x%10 + '0' );
}

/*************************************************************************
                                输出
*************************************************************************/
void print(int x) {
    if (x<0) {
        putchar('-');
        Out(-x);
    } else
        Out(x);
}
```

区间异或

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
const int N=65536*2+1;
const int inf=0x3f3f3f3f;
#define ls   (rt<<1)
#define rs    (rt<<1|1)
#define lson ls,l,m
#define rson rs,m+1,r
#define root 1,0,N
int color[N*4+20],XOR[N*4+20],ans[N*4+10];
void SetValue(int val,int rt,int l,int r){
    color[rt]=val;
    XOR[rt]=0;
}

void Reversal(int rt){
    if (color[rt]!=-1) color[rt]^=1;
    else XOR[rt]^=1;
}

void PushDown(int rt,int l,int r) {
    int m=(l+r)>>1;
    if (color[rt]!=-1) {
        SetValue(color[rt],lson);
        SetValue(color[rt],rson);
        color[rt]=-1;
    }
    if (XOR[rt]) {
        Reversal(ls);
        Reversal(rs);
        XOR[rt]=0;
    }
}

void PushUp(int rt,int l,int r){
    return ;
}

void interval_assign(int val,int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R) {
        SetValue(val,rt,l,r);
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) interval_assign(val,L,R,lson);
    if(R>m) interval_assign(val,L,R,rson);
    PushUp(rt,l,r);
}

void FXOR(int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R) {
```

```
        Reversal(rt);
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) FXOR(L,R,lson);
    if(R>m) FXOR(L,R,rson);
    PushUp(rt,l,r);
}
void query(int rt,int l,int r) {
    if (l==r) {
        ans[l]=color[rt];
        return;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    query(lson);
    query(rson);
    return ;
}

void solve() {
    memset(color,0,sizeof(color));
    memset(XOR,0,sizeof(XOR));
    memset(ans,0,sizeof(ans));
    char ch0,ch1,ch2;
    int L,R;
    while(scanf(" %c %c%d,%d%c",&ch0,&ch1,&L,&R,&ch2)!=EOF) {
        L<<=1;
        R<<=1;
        if (ch1=='(') L++;
        if (ch2==')') R--;
        if (L>R) {
            if (ch0=='I' || ch0=='C') {
                color[1]=XOR[1]=0;
            }
            continue;
        }
        switch(ch0) {
            case 'U': {
                interval_assign(1,L,R,root);
                break;
            }
            case 'D': {
                interval_assign(0,L,R,root);
                break;
            }
            case 'I': {
                if(L>0) interval_assign(0,0,L-1,root);
                if (R<N) interval_assign(0,R+1,N,root);
                break;
            }
            case 'S': {
                FXOR(L,R,root);
                break;
            }
            case 'C': {
```

```
                if(L>0) interval_assign(0,0,L-1,root);
                if (R<N) interval_assign(0,R+1,N,root);
                FXOR(L,R,root);
                break;
            }
        }
    }
    query(1,0,N);
    L=0;R=-1;
    for (int i=0; i<=N; i++) {
        if (ans[i]==0 && ans[i+1]==1) L=i+1;
        if (ans[i]==1 && ans[i+1]==0) {
            R=i;
            if (L&1) printf("(%d,",L/2);
            else          printf("[%d,",L/2);
            if (R&1)       printf("%d) ",R/2+1);
            else          printf("%d] ",R/2);
        }
    }
    if (L>R)    printf("empty set\n");
    else printf("\n");
}

int main() {
    //    freopen("datain.txt","r",stdin);
    solve();
    return 0;
}
```

区间最长连续递增子序列

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
#define ls (rt<<1)
#define rs (rt<<1|1)
#define lson ls,l,m
#define rson rs,m+1,r
#define root 1,1,n
const int N=1e5+2;
const int inf=0x3f3f3f3f;
int Rvalue[4*N],Lvalue[4*N],Prefix[4*N],Suffix[4*N],Middle[4*N],MAX[4*N],add[4*N];
void PushUp(int rt,int l,int r) {
    int m=(l+r)>>1;
    Rvalue[rt]=Rvalue[rs];
    Lvalue[rt]=Lvalue[ls];

    if (Rvalue[ls]<Lvalue[ls|1]) {
        Middle[rt]=Suffix[ls]+Prefix[rs];
        if (Prefix[ls]==(m-l+1)) Prefix[rt]=Prefix[ls]+Prefix[rs];
        if (Suffix[rs]==(r-m)) Suffix[rt]=Suffix[ls]+Suffix[rs];
    } else {
        Prefix[rt]=Prefix[ls];
        Suffix[rt]=Suffix[rs];
    }
    MAX[rt]=max( Middle[rt] , max( Prefix[rt] , Suffix[rt] ) );
}

void PushDown(int rt,int l,int r) {
    int m=(l+r)>>1;
    if (add[rt]) {            //«¯º%'^º1
        add[ls]        +=add[rt];
        add[rs]        +=add[rt];
        Rvalue[ls]    +=add[rt];
        Rvalue[rs]    +=add[rt];
        Lvalue[ls]    +=add[rt];
        Lvalue[rs]    +=add[rt];
        add[rt]=0;
    }
}

void build(int rt,int l,int r) {    //
    add[rt]=0;
    if (l==r) {
        cin>>Rvalue[rt];
        Lvalue[rt]=Rvalue[rt];
        MAX[rt]=Prefix[rt]=Suffix[rt]=Middle[rt]=1;
        return ;
    }
    int m=(l+r)>>1;
    build(lson);       //ππΩ®◊Û◊" ˜
    build(rson);       //ππΩ®""◊" ˜
    PushUp(rt,l,r);        //ππΩ®ÕÍ◊Û""¡Ωø≈◊" ˜∫Û£¨∏¸—¬µ±«∞◊" ˜
}
```

```
void interval_add(int L,int R,int addv,int rt,int l,int r) {
    if (L<=l && r<=R) {
        add[rt]        +=addv;
        Rvalue[rt]    +=addv;
        Lvalue[rt]    +=addv;
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) interval_add(L,R,addv,lson);
    if(R>m)  interval_add(L,R,addv,rson);
    PushUp(rt,l,r);
}
int query_MAXLEN(int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R) {
        return MAX[rt];
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    int res=0,t1=0,t2=0;
    if (L<=m) {
        res= max( res , query_MAXLEN(L,R,lson) );
        t1=min(m-L+1 , Suffix[ls] );
    }

    if (R>m) {
        res= max( res , query_MAXLEN(L,R,rson) );;
        t2=min(R-m   , Prefix[rs] );
    }
    if (Rvalue[ls]<Lvalue[rs]) res=max(res,t1+t2);
    return res;
}

void solve() {
    int n,Q;
    cin>>n>>Q;
    build(root);
    while(Q--) {
        char order;
        cin>>order;
        if (order=='q') {
            int a,b;
            cin>>a>>b;
            cout<<query_MAXLEN(a,b,root)<<endl;
        } else {
            int a,b,c;
            cin>>a>>b>>c;
            interval_add(a,b,c,root);
        }
    }
}

int main() {
    //    freopen("datain.txt","r",stdin);
    ios::sync_with_stdio(false);
    cin.tie(0);
    int T;
```

```
    cin>>T;
    for (int i=1; i<=T; i++) {
        cout<<"Case #"<<i<<":"<<'\n';
        solve();
    }
    return 0;
}
```

贴海报 离散化

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
#include <set>
#include <map>
using namespace std;
typedef long long LL;
#define ls (rt<<1)
#define rs (rt<<1|1)
#define lson ls,l,m
#define rson rs,m+1,r
#define root 1,1,n
const int N=1e5+2;
const int inf=0x3f3f3f3f;
int color[4*N];
bool num[N];

void SetValue(int val,int rt,int l,int r) {
    color[rt]=val;
}
void PushDown(int rt,int l,int r) {
    int m=(l+r)>>1;
    if(color[rt]) {
        SetValue(color[rt],lson);
        SetValue(color[rt],rson);
        color[rt]=0;
    }
}
void interval_assign(int L,int R,int val,int rt,int l,int r) {
    if (L<=l && r<=R) {
        SetValue(val,rt,l,r);
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) interval_assign(L,R,val,lson);
    if(R>m) interval_assign(L,R,val,rson);
}
void query(int l,int r,int rt) {
    if (l==r) {
        num[color[rt]]=true;
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    query(lson);
    query(rson);
    return;
}


struct node{
    int l,r;
}a[N];
set <int> vis;
```

```cpp
vector <int> rnk;
map <int,int> binarysearch;
void solve() {
    memset(color,0,sizeof(color));
    memset(num,false,sizeof(num));
    vis.clear();
    rnk.clear();
    int n;
    cin>>n;
    for (int i=0;i<n;i++) {
        cin>>a[i].l>>a[i].r;
        vis.insert(a[i].l);
        vis.insert(a[i].r);
    }
    rnk.push_back( *vis.begin() );
    for (set<int>::iterator it=++vis.begin();it!=vis.end();it++){
        int t=*it;
        int pre=rnk[rnk.size()-1];
        if (t>pre+1) {
            rnk.push_back(t-1);
            rnk.push_back(t);
        } else{
            rnk.push_back(t);
        }
    }
    int siz=rnk.size();
    for (int i=0;i<siz;i++){
        binarysearch[rnk[i]]=i+1;
    }
    for (int i=0;i<n;i++){
        int L=binarysearch[a[i].l];
        int R=binarysearch[a[i].r];
        interval_assign(L,R,i+1,1,siz,1);
    }
    query(1,siz,1);
    int ans=0;
    for (int i=1;i<=n;i++) if (num[i]) ans++;
    cout<<ans<<'\n';
}

int main() {
    //    freopen("datain.txt","r",stdin);
    ios::sync_with_stdio(false);
    cin.tie(0);
    int T;
    cin>>T;
    while(T--) {
        solve();
    }
    return 0;
}
```