# 【HNOI 2004】宠物收养场

# **【题目】**

[传送门](https://www.lydsy.com/JudgeOnline/problem.php?id=1208" \t "https://blog.csdn.net/forever_dreams/article/details/_blank)

****题目描述：****

最近，阿Q开了一间宠物收养所。收养所提供两种服务：收养被主人遗弃的宠物和让新的主人领养这些宠物。

每个领养者都希望领养到自己满意的宠物，阿Q根据领养者的要求通过他自己发明的一个特殊的公式，得出该领养者希望领养的宠物的特点值IMG_256（IMG_257是一个正整数，IMG_258＜IMG_259），而他也给每个处在收养所的宠物一个特点值。这样他就能够很方便的处理整个领养宠物的过程了，宠物收养所总是会有两种情况发生：被遗弃的宠物过多或者是想要收养宠物的人太多，而宠物太少。

1、被遗弃的宠物过多时，假若到来一个领养者，这个领养者希望领养的宠物的特点值为IMG_260，那么它将会领养一只目前未被领养的宠物中特点值最接近IMG_261的一只宠物。（任何两只宠物的特点值都不可能是相同的，任何两个领养者的希望领养宠物的特点值也不可能是一样的）如果有两只满足要求的宠物，即存在两只宠物他们的特点值分别为IMG_262和IMG_263，那么领养者将会领养特点值为IMG_264的那只宠物。

2、收养宠物的人过多，假若到来一只被收养的宠物，那么哪个领养者能够领养它呢？能够领养它的领养者，是那个希望被领养宠物的特点值最接近该宠物特点值的领养者，如果该宠物的特点值为IMG_265，存在两个领养者他们希望领养宠物的特点值分别为IMG_266和IMG_267，那么特点值为IMG_268的那个领养者将成功领养该宠物。

一个领养者领养了一个特点值为IMG_269的宠物，而它本身希望领养的宠物的特点值为IMG_270，那么这个领养者的不满意程度为IMG_271。

你得到了一年当中，领养者和被收养宠物到来收养所的情况，希望你计算所有收养了宠物的领养者的不满意程度的总和。这一年初始时，收养所里面既没有宠物，也没有领养者。

****输入格式：****

第一行为一个正整数IMG_272（IMG_273≤ 80000），表示一年当中来到收养所的宠物和领养者的总数。

接下来的IMG_274行，按到来时间的先后顺序描述了一年当中来到收养所的宠物和领养者的情况。每行有两个正整数IMG_275，IMG_276。其中IMG_277＝0表示宠物，IMG_278＝1表示领养者，IMG_279表示宠物的特点值或是领养者希望领养宠物的特点值。（同一时间呆在收养所中的，要么全是宠物，要么全是领养者，这些宠物和领养者的个数不会超过10000个）

****输出格式：****

输出仅有一个正整数，表示一年当中所有收养了宠物的领养者的不满意程度的总和mod1000000以后的结果。

****样例数据：****

输入

5

0 2

0 4

1 3

1 2

1 5

输出

3

****备注：****

【样例说明】  
IMG_280，最后一个领养者没有宠物可以领养。

# **【分析】**

这道题其实不算很难，但是细节卡了我好久。。。

其实我们可以只用一颗平衡树，若宠物过多则树上就记录的是宠物，否则就是领养者，互换着用就行

还是一样的找前驱和后继，分别与当前数作差比较绝对值，删点，再统计答案即可

**【代码】**

#include<cmath>

#include<ctime>

#include<cstdio>

#include<cstring>

#include<algorithm>

#define N 80005

#define mod 1000000

#define ll long long

#define lc(x) son[x][0]

#define rc(x) son[x][1]

#define inf (1ll<<50ll)

using namespace std;

int tot,val[N],size[N],weight[N],son[N][2];

void pushup(int root){

size[root]=size[lc(root)]+size[rc(root)]+1;

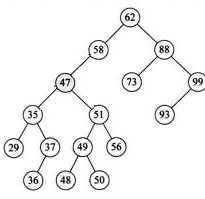
}

void rotate(int &root,int t){

int x=son[root][t];

son[root][t]=son[x][t^1];

son[x][t^1]=root;

 pushup(root),pushup(x);

root=x;

}

void insert(int &root,int x){

if(root){

int t=val[root]<x;

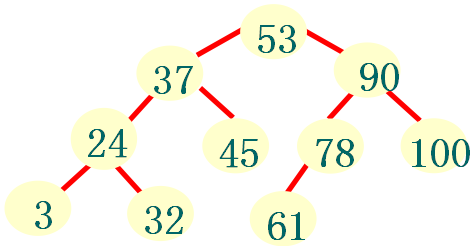
insert(son[root][t],x);

if(weight[son[root][t]]<weight[root])

rotate(root,t);

}

else{

 root=++tot;

val[root]=x;

weight[root]=rand();

}

pushup(root);

}

void remove(int &root,int x){

if(x==inf||x==-inf) return;

if(val[root]==x){

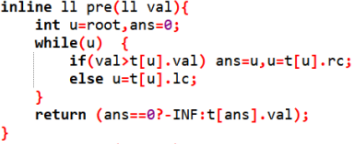
if(!lc(root)&&!rc(root)){

root=0;

return;

}

rotate(root,weight[lc(root)]<weight[rc(root)]);

 remove(root,x);

}

else remove(son[root][val[root]<x],x);

pushup(root);

}

ll findpre(int root,int x){

if(!root) return -inf;

//if(val[root]<=x) return max((ll)val[root],findpre(rc(root),x));

if(val[root]<x) return max((ll)val[root],findpre(rc(root),x));

return findpre(lc(root),x);

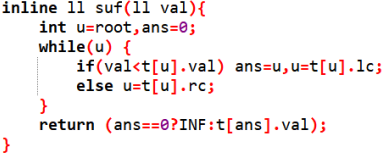
}

ll findsuf(int root,int x){

if(!root) return inf;

//if(val[root]>=x) return min((ll)val[root],findsuf(lc(root),x));

if(val[root]>x) return min((ll)val[root],findsuf(lc(root),x));

 return findsuf(rc(root),x);

}

int main(){

int n,i,s,x;

int ans=0,root=0,pet=0,people=0;

srand(time(0));

scanf("%d",&n);

for(i=1;i<=n;++i) {

scanf("%d%d",&s,&x);

if(pet>0){

if(s==0) pet++,insert(root,x);

else{

ll l=findpre(root,x);

ll r=findsuf(root,x);

ll t=(x-l<=r-x)?l:r;pet--;

remove(root,t),ans=(ans+abs(x-t))%mod;

}

}

else if(people>0){

if(s==1) people++,insert(root,x);

else{

ll l=findpre(root,x);

ll r=findsuf(root,x);

ll t=(x-l<=r-x)?l:r;people--;

remove(root,t),ans=(ans+abs(x-t))%mod;

}

}

else if(pet==0&&people==0){

if(s==0) pet++,insert(root,x);

else people++,insert(root,x);

}

}

printf("%d",ans);

return 0;

}

# NOI2004郁闷的出纳员题解

****描述****  
OIER公司是一家大型专业化软件公司，有着数以万计的员工。作为一名出纳员，我的任务之一便是统计每位员工的工资。这本来是一份不错的工作，但是令人郁闷的是，我们的老板反复无常，经常调整员工的工资。如果他心情好，就可能把每位员工的工资加上一个相同的量。反之，如果心情不好，就可能把他们的工资扣除一个相同的量。我真不知道除了调工资他还做什么其它事情。  
工资的频繁调整很让员工反感，尤其是集体扣除工资的时候，一旦某位员工发现自己的工资已经低于了合同规定的工资下界，他就会立刻气愤地离开公司，并且再也不会回来了。每位员工的工资下界都是统一规定的。每当一个人离开公司，我就要从电脑中把他的工资档案删去，同样，每当公司招聘了一位新员工，我就得为他新建一个工资档案。  
老板经常到我这边来询问工资情况，他并不问具体某位员工的工资情况，而是问现在工资第k多的员工拿多少工资。每当这时，我就不得不对数万个员工进行一次漫长的排序，然后告诉他答案。  
好了，现在你已经对我的工作了解不少了。正如你猜的那样，我想请你编一个工资统计程序。怎么样，不是很困难吧？

****格式****  
输入格式  
第一行有两个非负整数n和min。n表示下面有多少条命令，min表示工资下界。  
接下来的n行，每行表示一条命令。命令可以是以下四种之一：  
名称 格式 作用  
I命令 I\_k 新建一个工资档案，初始工资为k。如果某员工的初始工资低于工资下界，他将立刻离开公司。  
A命令 A\_k 把每位员工的工资加上k  
S命令 S\_k 把每位员工的工资扣除k  
F命令 F\_k 查询第k多的工资  
\_（下划线）表示一个空格，I命令、A命令、S命令中的k是一个非负整数，F命令中的k是一个正整数。  
在初始时，可以认为公司里一个员工也没有。  
输出格式  
输出文件的行数为F命令的条数加一。  
对于每条F命令，你的程序要输出一行，仅包含一个整数，为当前工资第k多的员工所拿的工资数，如果k大于目前员工的数目，则输出-1。  
输出文件的最后一行包含一个整数，为离开公司的员工的总数。

****样例****  
样例输入  
9 10  
I 60  
I 70  
S 50  
F 2  
I 30  
S 15  
A 5  
F 1  
F 2  
样例输出  
10  
20  
-1  
2

****提示****  
I命令的条数不超过100000  
A命令和S命令的总条数不超过100  
F命令的条数不超过100000  
每次工资调整的调整量不超过1000  
新员工的工资不超过100000

****题解****  
本题涉及单个结点的插入、删除，和所有结点的修改操作，不难想到平衡树，姑且拿这道题练一遍好久不写的splay吧。  
对A与S命令，我们不使用标记，而是直接用一个变量w（可正可负）累积；  
同时，当S命令的效果被累积后，此时工资加上w仍小于min的员工将离开，应找到第一个工资不小于min-w的员工，把他splay到根，那么其左子树即为所有离开的员工，此时维护size域即可维护所有员工的数量。****这里应高度重视边界条件****。  
对I操作，插入时工资变为k-w以维护所有员工工资的相对大小；  
对F操作，用size域直接查询第k大即可；

****Code****

#include <cstdio>

#include <algorithm>

#include <cstring>

#define maxn 100005

#define nil 0

#define oo 1000000000

using namespace std;int n, minp, root, tot, w;int d[maxn], son[maxn][2], fa[maxn], s[maxn];

inline void update(int rot){

s[rot] = s[son[rot][0]] + s[son[rot][1]] + 1;

}

void rotate(int x, int w){

int y = fa[x];

son[y][w ^ 1] = son[x][w];

if(son[x][w] != nil) fa[son[x][w]] = y;

fa[x] = fa[y];

if(fa[y] != nil){

if(y == son[fa[y]][0]) son[fa[y]][0] = x;

else son[fa[y]][1] = x;

}

son[x][w] = y; fa[y] = x;

update(y); update(x);

}

void splay(int x, int poi){

if(x == nil) return;

while(fa[x] != poi){

if(fa[fa[x]] == poi){

if(x == son[fa[x]][0]) rotate(x, 1);

else rotate(x, 0);

}

else{

if(fa[x] == son[fa[fa[x]]][0]){

if(x == son[fa[x]][0])

rotate(fa[x], 1); rotate(x, 1);

else

rotate(x, 0); rotate(x, 1);

}

else{

if(x == son[fa[x]][0])

rotate(x, 1); rotate(x, 0);

else

rotate(fa[x], 0); rotate(x, 0);

}

}

}

if(poi == nil) root = x;

}

void insert(int rot, int x){

++s[rot];

if(d[rot] <= x){

if(son[rot][1] == nil){

d[++tot] = x;

s[tot] = 1;

fa[tot] = rot;

son[rot][1] = tot;

son[tot][0] = son[tot][1] = nil;

**splay(rot, nil);**

}

else{

insert(son[rot][1], x);

}

}

else{

if(son[rot][0] == nil){

d[++tot] = x;

s[tot] = 1;

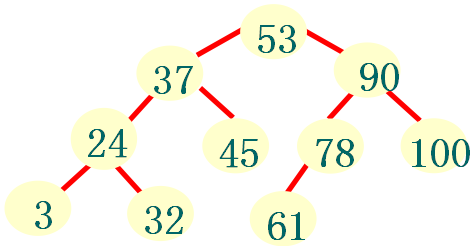
fa[tot] = rot;

son[rot][0] = tot;

son[tot][0] = son[tot][1] = nil;

**splay(rot, nil);**

}

 else{

insert(son[rot][0], x);

}

}

}

int succ(int rot, int k){

if(rot == nil) return nil;

if(d[rot] >= k){

int t = succ(son[rot][0], k);

if(t == nil) return rot;

else return t;

}

else{

return succ(son[rot][1], k);

}

}

int select(int rot, int k){

if(rot == nil || k < 0) return -1;

if(s[son[rot][1]] + 1 == k) return d[rot] + w;

if(s[son[rot][1]] >= k) return select(son[rot][1], k);

return select(son[rot][0], k - s[son[rot][1]] - 1);

}

int main(){

char opt;

int k, sum = 0;

scanf("%d%d", &n, &minp);

d[root = ++tot] = (oo << 1);

s[tot] = 1;

fa[tot] = son[tot][0] = son[tot][1] = nil;

while(n--){

scanf("\n%c %d", &opt, &k);

switch(opt){

case 'I': if(k >= minp){

insert(root, k - w);

++sum;

}

break;

case 'A': w += k;

break;

case 'S': w -= k;

**splay(succ(root, minp - w), nil);**

s[root] -= s[son[root][0]];

son[root][0] = nil;

break;

case 'F': printf("%d\n", select(root, k + 1));

break;

default : break;

}

}

printf("%d\n", sum - s[root] + 1);

return 0;

}

# **bzoj** **1251: 序列终结者**

Time Limit:20 SecMemory Limit:162 MB  
Submit:2971Solved:1188  
[[Submit](http://www.lydsy.com/JudgeOnline/submitpage.php?id=1251" \t "https://blog.csdn.net/aarongzk/article/details/_blank)][[Status](http://www.lydsy.com/JudgeOnline/problemstatus.php?id=1251" \t "https://blog.csdn.net/aarongzk/article/details/_blank)][[Discuss](http://www.lydsy.com/JudgeOnline/bbs.php?id=1251" \t "https://blog.csdn.net/aarongzk/article/details/_blank)]

## **Description**

网上有许多题，就是给定一个序列，要你支持几种操作：A、B、C、D。一看另一道题，又是一个序列 要支持几种操作：D、C、B、A。尤其是我们这里的某人，出模拟试题，居然还出了一道这样的，真是没技术含量……这样 我也出一道题，我出这一道的目的是为了让大家以后做这种题目有一个“库”可以依靠，没有什么其他的意思。这道题目 就叫序列终结者吧。

【问题描述】

给定一个长度为N的序列，每个序列的元素是一个整数（废话）。要支持以下三种操作：

1. 将[L,R]这个区间内的所有数加上V。

2. 将[L,R]这个区间翻转，比如1 2 3 4变成4 3 2 1。

3. 求[L,R]这个区间中的最大值。 最开始所有元素都是0。

## **Input**

第一行两个整数N，M。M为操作个数。 以下M行，每行最多四个整数，依次为K，L，R，V。K表示是第几种操作，如果不是第1种操作则K后面只有两个数。

## **Output**

对于每个第3种操作，给出正确的回答。

## **Sample Input**

4 4  
1 1 3 2  
1 2 4 -1  
2 1 3  
3 2 4

## **Sample Output**

2  
【数据范围】  
N<=50000，M<=100000。

## **HINT**

## **Source**

[Splay](http://www.lydsy.com/JudgeOnline/problemset.php?search=Splay" \t "https://blog.csdn.net/aarongzk/article/details/_blank)

每次进行序列操作时，把l-1旋转到根，把r+1旋转到根的右儿子，r+1的左子树就是整个区间[l,r]。

我们可以用Splay的每个节点记录该节点对应子树的信息，那么每次询问只要输出r+1的左子树中的最大值，即代码中的mx[t[y][0]]。

为了避免Splay中有节点0，我们将所有节点的编号加1。又因为要旋转l-1和r+1，所以在Splay插入节点为1到n+2。(原因显然…大家自己脑补)

这道题用Splay的提根操作达到了区间操作的目的，方法很巧妙。

另外我觉得这道题有几点需要注意：

①要理解Splay中节点的含义以及节点所记录的信息。

②区间的翻转操作很巧妙，只需要将标记下传并且交换左右子树，并不需要修改节点的max和size。

③每次find操作都要pushdown，这样就可以保证节点x到根的路径上所有点都被更新，便于之后的旋转操作。

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<cstdlib>

#include<cmath>

#define F(i,j,n) for(int i=j;i<=n;i++)

#define D(i,j,n) for(int i=j;i>=n;i--)

#define LL long long

#define pa pair<int,int>

#define MAXN 50005

#define INF 1000000000

using namespace std;

int n,m,rt=0,tot=0;

int a[MAXN],fa[MAXN],t[MAXN][2],mx[MAXN],tag[MAXN],size[MAXN];

bool rev[MAXN];

inline int read(){

int ret=0,flag=1;char ch=getchar();

while (ch<'0'||ch>'9'){if (ch=='-') flag=-1;ch=getchar();}

while (ch>='0'&&ch<='9'){ret=ret\*10+ch-'0';ch=getchar();}

return ret\*flag;

}

inline void pushup(int k){

int l=t[k][0],r=t[k][1];

mx[k]=max(max(mx[l],mx[r]),a[k]);

size[k]=size[l]+size[r]+1;

}

inline void pushdown(int k){

int l=t[k][0],r=t[k][1],tg=tag[k];

if (tg){

tag[k]=0;

if (l){tag[l]+=tg;mx[l]+=tg;a[l]+=tg;}

if (r){tag[r]+=tg;mx[r]+=tg;a[r]+=tg;}

}

if (rev[k]){

rev[k]=0;

rev[l]^=1;rev[r]^=1;

swap(t[k][0],t[k][1]);

}

}

inline void rotate(int &k,int x){

int y=fa[x],z=fa[y],l,r;

if (t[y][0]==x) l=0;else l=1;r=l^1;

if (k==y) k=x;

else{if (t[z][0]==y) t[z][0]=x;else t[z][1]=x;}

fa[x]=z;

fa[y]=x;

fa[t[x][r]]=y;

t[y][l]=t[x][r];

t[x][r]=y;

pushup(y);pushup(x);

}

inline void splay(int &k,int x){

while (x!=k){

int y=fa[x],z=fa[y];

if (y!=k){

if ((t[y][0]==x)^(t[z][0]==y)) rotate(k,x);

else rotate(k,y);

}

rotate(k,x);

}

}

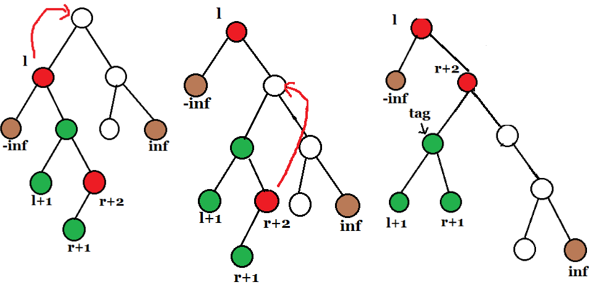
inline int find(int k,int rank){

if (tag[k]||rev[k]) pushdown(k);

int l=t[k][0],r=t[k][1];

if (size[l]+1==rank) return k;

else if (size[l]>=rank) return find(l,rank);

 else return find(r,rank-size[l]-1);

}

inline void add(int l,int r,int val){

int x=find(rt,l),y=find(rt,r+2);

**splay(rt,x);**

**splay(t[x][1],y);**

int z=t[y][0];

tag[z]+=val;mx[z]+=val;a[z]+=val;

}

inline void reverse(int l,int r){

int x=find(rt,l),y=find(rt,r+2);

**splay(rt,x);splay(t[x][1],y);**

rev[t[y][0]]^=1;

}

inline void query(int l,int r){

int x=find(rt,l),y=find(rt,r+2);

**splay(rt,x);splay(t[x][1],y);**

printf("%d\n",mx[t[y][0]]);

}

inline void build(int l,int r,int last){

if (l>r) return;

if (l==r){

fa[l]=last;size[l]=1;

if (l<last) t[last][0]=l;

else t[last][1]=l;

return;

}

int mid=(l+r)>>1;

build(l,mid-1,mid);build(mid+1,r,mid);

fa[mid]=last;pushup(mid);

if (mid<last) t[last][0]=mid;

else t[last][1]=mid;

}

int main(){

mx[0]=-INF;

n=read();m=read();

build(1,n+2,0);

rt=(n+3)>>1;

F(i,1,m){

int flag=read(),l=read(),r=read(),val;

if (flag==1){val=read();add(l,r,val);}

else if (flag==2) reverse(l,r);

else query(l,r);

}

return 0;

}