线段树试题分类

（例题来源 1：HDU-杭州电子科技大学在线评测：http://acm.hdu.edu.cn）

（例题来源 2：POJ-北大在线评测：http://poj.org）

先介绍一些约定俗成的名称：

- maxn 是题目给的最大区间，而节点数要开 4 倍，确切的来说节点数要开大于 maxn 的最小 2^x 的两倍。
- lson 和 rson 分别表示结点的左儿子和右儿子，由于每次传参数的时候都固定是这几个变量，所以可以用预定于比较方便的表示。
- 以前的写法是另外开两个数组记录每个结点所表示的区间，其实这个区间不必保存，一边算一边传下去就行，只需要写函数的时候多两个参数，结合 lson 和 rson 的预定义可以很方便。
- PushUP(int rt)是把当前结点的信息更新到父结点。
- PushDown(int rt)是把当前结点的信息更新给儿子结点。
- rt 表示当前子树的根(root)，也就是当前所在的结点。

线段树的题目整体上可以分成以下四个部分：

一、单点更新（P2）

最最基础的线段树，只更新叶子节点，然后把信息用 PushUP(int r)这个函数更新上来

二、成段更新（P7）（通常这对初学者来说是一道坎）：

需要用到延迟标记(或者说懒惰标记)，简单来说就是每次更新的时候不要更新到底，用延迟标记使得更新延迟到下次需要更新或者询问到的时候。

三、区间合并（P16）

这类题目会询问区间中满足条件的连续最长区间，所以 PushUp 的时候需要对左右儿子的区间进行合并。

四、扫描线（P19）

这类题目需要将一些操作排序,然后从左到右用一根扫描线（当然是在我们脑子里）扫过去。最典型的就是矩形面积并、周长并等题。

一、单点更新：

最最基础的线段树，只更新叶子节点，然后把信息用 PushUP(int r)这个函数更新上来。

- hdu1166 敌兵布阵
- 题意:O(-1)
- 思路:O(-1)
- 线段树功能:update:单点增减 query:区间求和

```
#include <cstdio>
 #define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
const int maxn = 55555;
int sum[maxn<<2];
void PushUP(int rt) {
        sum[rt] = sum[rt<<1] + sum[rt<<1|1];
}
void build(int l,int r,int rt) {
        if (l == r) {
                scanf("%d",&sum[rt]);
                return ;
        }
        int m = (l + r) >> 1;
        build(lson);
        build(rson);
        PushUP(rt);
}
void update(int p,int add,int l,int r,int rt) {
        if (l == r) {
                sum[rt] += add;
                return ;
        }
        int m = (l + r) >> 1;
        if (p <= m) update(p , add , lson);
        else update(p , add , rson);
        PushUP(rt);
}
int query(int L,int R,int l,int r,int rt) {
        if (L <= l && r <= R) {
                return sum[rt];
        }
        int m = (l + r) >> 1;
        int ret = 0;
        if (L <= m) ret += query(L , R , lson);
        if (R > m) ret += query(L , R , rson);
        return ret;
}
```

```
int main() {
	int T , n;
	scanf("%d",&T);
	for (int cas = 1 ; cas <= T ; cas ++) {
		printf("Case %d:\n",cas);
		scanf("%d",&n);
		build(1 , n , 1);
		char op[10];
		while (scanf("%s",op)) {
			if (op[0] == 'E') break;
			int a , b;
			scanf("%d%d",&a,&b);
			if (op[0] == 'Q') printf("%d\n",query(a , b , 1 , n , 1));
			else if (op[0] == 'S') update(a , -b , 1 , n , 1);
			else update(a , b , 1 , n , 1);
		}
	}
	return 0;
}

```

- hdu1754 I Hate It
- 题意:O(-1)
- 思路:O(-1)
- 线段树功能:update:单点替换 query:区间最值

```
#include <cstdio>
#include <algorithm>
using namespace std;

#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
const int maxn = 222222;
int MAX[maxn<<2];
void PushUP(int rt) {
	MAX[rt] = max(MAX[rt<<1] , MAX[rt<<1|1]);
}
```

```
12 void build(int l,int r,int rt) {
13         if (l == r) {
14             scanf("%d",&MAX[rt]);
15             return ;
16         }
17         int m = (l + r) >> 1;
18         build(lson);
19         build(rson);
20         PushUP(rt);
21 }
22 void update(int p,int sc,int l,int r,int rt) {
23         if (l == r) {
24             MAX[rt] = sc;
25             return ;
26         }
27         int m = (l + r) >> 1;
28         if (p <= m) update(p , sc , lson);
29         else update(p , sc , rson);
30         PushUP(rt);
31 }
32 int query(int L,int R,int l,int r,int rt) {
33         if (L <= l && r <= R) {
34             return MAX[rt];
35         }
36         int m = (l + r) >> 1;
37         int ret = 0;
38         if (L <= m) ret = max(ret , query(L , R , lson));
39         if (R > m) ret = max(ret , query(L , R , rson));
40         return ret;
41 }
42 int main() {
43         int n , m;
44         while (~scanf("%d%d",&n,&m)) {
45             build(1 , n , 1);
46             while (m --) {
47                 char op[2];
48                 int a , b;
49                 scanf("%s%d%d",op,&a,&b);
50                 if (op[0] == 'Q') printf("%d\n",query(a , b , 1 , n , 1));
51                 else update(a , b , 1 , n , 1);
52             }
53         }
54         return 0;
55 }
```

- hdu1394 Minimum Inversion Number
- 题意:求 Inversion 后的最小逆序数
- 思路:用 O(nlogn)复杂度求出最初逆序数后,就可以用 O(1)的复杂度分别递推出其他解
- 线段树功能:update:单点增减 query:区间求和

```
#include <cstdio>
#include <algorithm>
using namespace std;

#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
const int maxn = 5555;
int sum[maxn<<2];
void PushUP(int rt) {
        sum[rt] = sum[rt<<1] + sum[rt<<1|1];
}
void build(int l,int r,int rt) {
        sum[rt] = 0;
        if (l == r) return ;
        int m = (l + r) >> 1;
        build(lson);
        build(rson);
}
void update(int p,int l,int r,int rt) {
        if (l == r) {
                sum[rt] ++;
                return ;
        }
        int m = (l + r) >> 1;
        if (p <= m) update(p , lson);
        else update(p , rson);
        PushUP(rt);
}
int query(int L,int R,int l,int r,int rt) {
        if (L <= l && r <= R) {
                return sum[rt];
        }
        int m = (l + r) >> 1;
        int ret = 0;
        if (L <= m) ret += query(L , R , lson);
        if (R > m) ret += query(L , R , rson);
        return ret;
}
int x[maxn];
```

```
int main() {
	int n;
	while (~scanf("%d",&n)) {
		build(0 , n - 1 , 1);
		int sum = 0;
		for (int i = 0 ; i < n ; i ++) {
			scanf("%d",&x[i]);
			sum += query(x[i] , n - 1 , 0 , n - 1 , 1);
			update(x[i] , 0 , n - 1 , 1);
		}
		int ret = sum;
		for (int i = 0 ; i < n ; i ++) {
			sum += n - x[i] - x[i] - 1;
			ret = min(ret , sum);
		}
		printf("%d\n",ret);
	}
	return 0;
}
```

- hdu2795 Billboard
- 题意:h*w 的木板,放进一些 1*L 的物品,求每次放**空间能容纳且最上边**的位子
- 思路:每次找到最大值的位子,然后减去 L
- 线段树功能:query:区间求最大值的位子(直接把 update 的操作在 query 里做了)

```
#include <cstdio>
#include <algorithm>
using namespace std;

#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
const int maxn = 222222;
int h , w , n;
int MAX[maxn<<2];
void PushUP(int rt) {
	MAX[rt] = max(MAX[rt<<1] , MAX[rt<<1|1]);
}
void build(int l,int r,int rt) {
	MAX[rt] = w;
	if (l == r) return ;
	int m = (l + r) >> 1;
	build(lson);
	build(rson);
}
```

```
int query(int x,int l,int r,int rt) {
        if (l == r) {
                MAX[rt] -= x;
                return l;
        }
        int m = (l + r) >> 1;
        int ret = (MAX[rt<<1] >= x) ? query(x , lson) : query(x , rson);
        PushUP(rt);
        return ret;
}
int main() {
        while (~scanf("%d%d%d",&h,&w,&n)) {
                if (h > n) h = n;
                build(1 , h , 1);
                while (n --) {
                        int x;
                        scanf("%d",&x);
                        if (MAX[1] < x) puts("-1");
                        else printf("%d\n",query(x , 1 , h , 1));
                }
        }
        return 0;
}
```

- 练习:
  - poj2828 Buy Tickets
  - poj2886 Who Gets the Most Candies?

二、成段更新（通常这对初学者来说是一道坎）：需要用到延迟标记(或者说懒惰标记)，简单来说就是每次更新的时候不要更新到底，用延迟标记使得更新延迟到下次需要更新 or 询问到的时候。

- hdu1698 Just a Hook
- 题意:O(-1)
- 思路:O(-1)
- 线段树功能:update:成段替换 (由于只 query 一次总区间,所以可以直接输出 1 结点的信息)

```cpp
#include <cstdio>
#include <algorithm>
using namespace std;

#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
const int maxn = 111111;
int h , w , n;
int col[maxn<<2];
int sum[maxn<<2];
void PushUp(int rt) {
	sum[rt] = sum[rt<<1] + sum[rt<<1|1];
}
void PushDown(int rt,int m) {
	if (col[rt]) {
		col[rt<<1] = col[rt<<1|1] = col[rt];
		sum[rt<<1] = (m - (m >> 1)) * col[rt];
		sum[rt<<1|1] = (m >> 1) * col[rt];
		col[rt] = 0;
	}
}
void build(int l,int r,int rt) {
	col[rt] = 0;
	sum[rt] = 1;
	if (l == r) return ;
	int m = (l + r) >> 1;
	build(lson);
	build(rson);
	PushUp(rt);
}
void update(int L,int R,int c,int l,int r,int rt) {
	if (L <= l && r <= R) {
		col[rt] = c;
		sum[rt] = c * (r - l + 1);
		return ;
	}
	PushDown(rt , r - l + 1);
	int m = (l + r) >> 1;
	if (L <= m) update(L , R , c , lson);
	if (R > m) update(L , R , c , rson);
	PushUp(rt);
}


```

```
int main() {
        int T , n , m;
        scanf("%d",&T);
        for (int cas = 1 ; cas <= T ; cas ++) {
                scanf("%d%d",&n,&m);
                build(1 , n , 1);
                while (m --) {
                        int a , b , c;
                        scanf("%d%d%d",&a,&b,&c);
                        update(a , b , c , 1 , n , 1);
                }
                printf("Case %d: The total value of the hook is %d.\n",cas , sum[1]);
        }
        return 0;
}
```

- poj3468 A Simple Problem with Integers
- 题意:O(-1)
- 思路:O(-1)
- 线段树功能:update:成段增减 query:区间求和

```
#include <cstdio>
#include <algorithm>
using namespace std;

#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
#define LL long long
const int maxn = 111111;
LL add[maxn<<2];
LL sum[maxn<<2];
void PushUp(int rt) {
        sum[rt] = sum[rt<<1] + sum[rt<<1|1];
}
void PushDown(int rt,int m) {
        if (add[rt]) {
                add[rt<<1] += add[rt];
                add[rt<<1|1] += add[rt];
                sum[rt<<1] += add[rt] * (m - (m >> 1));
                sum[rt<<1|1] += add[rt] * (m >> 1);
                add[rt] = 0;
        }
}
```

```
23 void build(int l,int r,int rt) {
24         add[rt] = 0;
25         if (l == r) {
26                 scanf("%lld",&sum[rt]);
27                 return ;
28         }
29         int m = (l + r) >> 1;
30         build(lson);
31         build(rson);
32         PushUp(rt);
33 }
34 void update(int L,int R,int c,int l,int r,int rt) {
35         if (L <= l && r <= R) {
36                 add[rt] += c;
37                 sum[rt] += (LL)c * (r - l + 1);
38                 return ;
39         }
40         PushDown(rt , r - l + 1);
41         int m = (l + r) >> 1;
42         if (L <= m) update(L , R , c , lson);
43         if (m < R) update(L , R , c , rson);
44         PushUp(rt);
45 }
46 LL query(int L,int R,int l,int r,int rt) {
47         if (L <= l && r <= R) {
48                 return sum[rt];
49         }
50         PushDown(rt , r - l + 1);
51         int m = (l + r) >> 1;
52         LL ret = 0;
53         if (L <= m) ret += query(L , R , lson);
54         if (m < R) ret += query(L , R , rson);
55         return ret;
56 }
57
58
59
60
61
62
63
64
65
66
```

```
int main() {
        int N , Q;
        scanf("%d%d",&N,&Q);
        build(1 , N , 1);
        while (Q --) {
                char op[2];
                int a , b , c;
                scanf("%s",op);
                if (op[0] == 'Q') {
                        scanf("%d%d",&a,&b);
                        printf("%lld\n",query(a , b , 1 , N , 1));
                } else {
                        scanf("%d%d%d",&a,&b,&c);
                        update(a , b , c , 1 , N , 1);
                }
        }
        return 0;
}
```

- poj2528 Mayor’s posters
- 题意:在墙上贴海报,海报可以互相覆盖,问最后可以看见几张海报
- 思路:这题数据范围很大,直接搞超时+超内存,需要离散化:
- 离散化简单的来说就是只取我们**需要的值**来用,比如说区间[1000,2000],[1990,2012] 我们用不到[-∞,999][1001,1989][1991,1999][2001,2011][2013,+∞]这些值,所以我只需要1000,1990,2000,2012 就够了,将其分别映射到 0,1,2,3,在于复杂度就大大的降下来了
- 所以离散化要保存所有需要用到的值,排序后,分别映射到 1~n,这样复杂度就会小很多很多
- 而这题的难点在于每个数字其实表示的是一个单位长度(并且一个点),这样普通的离散化会造成许多错误(包括我以前的代码,poj 这题数据奇弱)
- 给出下面两个简单的例子应该能体现普通离散化的缺陷:
- 1-10 1-4 5-10
- 1-10 1-4 6-10
- 为了解决这种缺陷,我们可以在排序后的数组上加些处理,比如说[1,2,6,10]
- 如果相邻数字间距大于 1 的话,在其中加上任意一个数字,比如加成[1,2,3,6,7,10],然后再做线段树就好了.
- 线段树功能:update:成段替换 query:简单 hash

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1

```

```
const int maxn = 11111;
bool hash[maxn];
int li[maxn] , ri[maxn];
int X[maxn*3];
int col[maxn<<4];
int cnt;

void PushDown(int rt) {
        if (col[rt] != -1) {
                col[rt<<1] = col[rt<<1|1] = col[rt];
                col[rt] = -1;
        }
}
void update(int L,int R,int c,int l,int r,int rt) {
        if (L <= l && r <= R) {
                col[rt] = c;
                return ;
        }
        PushDown(rt);
        int m = (l + r) >> 1;
        if (L <= m) update(L , R , c , lson);
        if (m < R) update(L , R , c , rson);
}
void query(int l,int r,int rt) {
        if (col[rt] != -1) {
                if (!hash[col[rt]]) cnt ++;
                hash[ col[rt] ] = true;
                return ;
        }
        if (l == r) return ;
        int m = (l + r) >> 1;
        query(lson);
        query(rson);
}
int Bin(int key,int n,int X[]) {
        int l = 0 , r = n - 1;
        while (l <= r) {
                int m = (l + r) >> 1;
                if (X[m] == key) return m;
                if (X[m] < key) l = m + 1;
                else r = m - 1;
        }
        return -1;
}
```

```
int main() {
        int T , n;
        scanf("%d",&T);
        while (T --) {
                scanf("%d",&n);
                int nn = 0;
                for (int i = 0 ; i < n ; i ++) {
                        scanf("%d%d",&li[i] , &ri[i]);
                        X[nn++] = li[i];
                        X[nn++] = ri[i];
                }
                sort(X , X + nn);
                int m = 1;
                for (int i = 1 ; i < nn; i ++) {
                        if (X[i] != X[i-1]) X[m ++] = X[i];
                }
                for (int i = m - 1 ; i > 0 ; i --) {
                        if (X[i] != X[i-1] + 1) X[m ++] = X[i-1] + 1;
                }
                sort(X , X + m);
                memset(col , -1 , sizeof(col));
                for (int i = 0 ; i < n ; i ++) {
                        int l = Bin(li[i] , m , X);
                        int r = Bin(ri[i] , m , X);
                        update(l , r , i , 0 , m , 1);
                }
                cnt = 0;
                memset(hash , false , sizeof(hash));
                query(0 , m , 1);
                printf("%d\n",cnt);
        }
        return 0;
}
```

- poj3225 Help with Intervals
- 题意:区间操作,交,并,补等
- 思路:
- 我们一个一个操作来分析:(用 0 和 1 表示是否包含区间,-1 表示该区间内既有包含又有不包含)
- U:把区间[l,r]覆盖成 1
- I:把[-∞,l)(r,∞]覆盖成 0
- D:把区间[l,r]覆盖成 0
- C:把[-∞,l)(r,∞]覆盖成 0，且[l,r]区间 0/1 互换
- S:[l,r]区间 0/1 互换

成段覆盖的操作很简单,比较特殊的就是**区间 0/1 互换**这个操作,我们可以称之为异或操作

很明显我们可以知道这个性质:当一个区间被覆盖后,不管之前有没有异或标记都没有意义了

所以当一个节点得到覆盖标记时把异或标记清空

而当一个节点得到异或标记的时候,先判断覆盖标记,如果是 0 或 1,直接改变一下覆盖标记,不然的话改变异或标记

开区间闭区间只要数字乘以 2 就可以处理(偶数表示端点,奇数表示两端点间的区间)

线段树功能:update:成段替换,区间异或 query:简单 hash

```
#include <cstdio>
#include <cstring>
#include <cctype>
#include <algorithm>
using namespace std;
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1

const int maxn = 131072;
bool hash[maxn];
int cover[maxn<<2];
int XOR[maxn<<2];
void FXOR(int rt) {
	if (cover[rt] != -1) cover[rt] ^= 1;
	else XOR[rt] ^= 1;
}
void PushDown(int rt) {
	if (cover[rt] != -1) {
		cover[rt<<1] = cover[rt<<1|1] = cover[rt];
		XOR[rt<<1] = XOR[rt<<1|1] = 0;
		cover[rt] = -1;
	}
	if (XOR[rt]) {
		FXOR(rt<<1);
		FXOR(rt<<1|1);
		XOR[rt] = 0;
	}
}


```

```
35 void update(char op,int L,int R,int l,int r,int rt) {
36         if (L <= l && r <= R) {
37             if (op == 'U') {
38                 cover[rt] = 1;
39                 XOR[rt] = 0;
40             } else if (op == 'D') {
41                 cover[rt] = 0;
42                 XOR[rt] = 0;
43             } else if (op == 'C' || op == 'S') {
44                 FXOR(rt);
45             }
46             return ;
47         }
48         PushDown(rt);
49         int m = (l + r) >> 1;
50         if (L <= m) update(op , L , R , lson);
51         else if (op == 'I' || op == 'C') {
52             XOR[rt<<1] = cover[rt<<1] = 0;
53         }
54         if (m < R) update(op , L , R , rson);
55         else if (op == 'I' || op == 'C') {
56             XOR[rt<<1|1] = cover[rt<<1|1] = 0;
57         }
58 }
59 void query(int l,int r,int rt) {
60         if (cover[rt] == 1) {
61             for (int it = l ; it <= r ; it ++) {
62                 hash[it] = true;
63             }
64             return ;
65         } else if (cover[rt] == 0) return ;
66         if (l == r) return ;
67         PushDown(rt);
68         int m = (l + r) >> 1;
69         query(lson);
70         query(rson);
71 }
72
73
74
75
76
77
78
```

```
int main() {
        cover[1] = XOR[1] = 0;
        char op , l , r;
        int a , b;
        while ( ~scanf("%c %c%d,%d%c\n",&op , &l , &a , &b , &r) ) {
                a <<= 1 , b <<= 1;
                if (l == '(') a ++;
                if (r == ')') b --;
                if (a > b) {
                        if (op == 'C' || op == 'I') {
                                cover[1] = XOR[1] = 0;
                        }
                } else update(op , a , b , 0 , maxn , 1);
        }
        query(0 , maxn , 1);
        bool flag = false;
        int s = -1 , e;
        for (int i = 0 ; i <= maxn ; i ++) {
                if (hash[i]) {
                        if (s == -1) s = i;
                        e = i;
                } else {
                        if (s != -1) {
                                if (flag) printf(" ");
                                flag = true;
                                printf("%c%d,%d%c",s&1?'(':'[' , s>>1 , (e+1)>>1 , e&1?')':']');
                                s = -1;
                        }
                }
        }
        if (!flag) printf("empty set");
        puts("");
        return 0;
}
```

- 练习:

  - poj1436 Horizontally Visible Segments
  - poj2991 Crane
  - Another LCIS
  - Bracket Sequence

三、区间合并：

这类题目会询问区间中满足条件的连续最长区间，所以 PushUp 的时候需要对左右儿子的区间进行合并。

- poj3667 Hotel
- 题意:1 a:询问是不是有连续长度为 a 的空房间,有的话住进最左边
- 2 a b:将[a,a+b-1]的房间清空
- 思路:记录区间中最长的空房间
- 线段树操作:update:区间替换 query:询问满足条件的最左断点

```
#include <cstdio>
#include <cstring>
#include <cctype>
#include <algorithm>
using namespace std;
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1

const int maxn = 55555;
int lsum[maxn<<2] , rsum[maxn<<2] , msum[maxn<<2];
int cover[maxn<<2];

void PushDown(int rt,int m) {
        if (cover[rt] != -1) {
                cover[rt<<1] = cover[rt<<1|1] = cover[rt];
                msum[rt<<1] = lsum[rt<<1] = rsum[rt<<1] = cover[rt] ? 0 : m - (m >> 1);
                msum[rt<<1|1] = lsum[rt<<1|1] = rsum[rt<<1|1] = cover[rt] ? 0 : (m >> 1);
                cover[rt] = -1;
        }
}
void PushUp(int rt,int m) {
        lsum[rt] = lsum[rt<<1];
        rsum[rt] = rsum[rt<<1|1];
        if (lsum[rt] == m - (m >> 1)) lsum[rt] += lsum[rt<<1|1];
        if (rsum[rt] == (m >> 1)) rsum[rt] += rsum[rt<<1];
        msum[rt] = max(lsum[rt<<1|1] + rsum[rt<<1] , max(msum[rt<<1] , msum[rt<<1|1]));
}
void build(int l,int r,int rt) {
        msum[rt] = lsum[rt] = rsum[rt] = r - l + 1;
        cover[rt] = -1;
        if (l == r) return ;
        int m = (l + r) >> 1;
        build(lson);
        build(rson);
}
```

```
void update(int L,int R,int c,int l,int r,int rt) {
        if (L <= l && r <= R) {
                msum[rt] = lsum[rt] = rsum[rt] = c ? 0 : r - l + 1;
                cover[rt] = c;
                return ;
        }
        PushDown(rt , r - l + 1);
        int m = (l + r) >> 1;
        if (L <= m) update(L , R , c , lson);
        if (m < R) update(L , R , c , rson);
        PushUp(rt , r - l + 1);
}
int query(int w,int l,int r,int rt) {
        if (l == r) return l;
        PushDown(rt , r - l + 1);
        int m = (l + r) >> 1;
        if (msum[rt<<1] >= w) return query(w , lson);
        else if (rsum[rt<<1] + lsum[rt<<1|1] >= w) return m - rsum[rt<<1] + 1;
        return query(w , rson);
}
int main() {
        int n , m;
        scanf("%d%d",&n,&m);
        build(1 , n , 1);
        while (m --) {
                int op , a , b;
                scanf("%d",&op);
                if (op == 1) {
                        scanf("%d",&a);
                        if (msum[1] < a) puts("0");
                        else {
                                int p = query(a , 1 , n , 1);
                                printf("%d\n",p);
                                update(p , p + a - 1 , 1 , 1 , n , 1);
                        }
                } else {
                        scanf("%d%d",&a,&b);
                        update(a , a + b - 1 , 0 , 1 , n , 1);
                }
        }
        return 0;
}
```

- 练习:

- hdu3308 LCIS
- hdu3397 Sequence operation
- hdu2871 Memory Control
- hdu1540 Tunnel Warfare
- CF46-D Parking Lot

四、扫描线

这类题目需要将一些操作排序,然后从左到右用一根扫描线(当然是在我们脑子里)扫过去。最典型的就是矩形面积并,周长并等题

- hdu1542 Atlantis
- 题意:矩形面积并
- 思路:浮点数先要离散化;然后把矩形分成两条边,上边和下边,对横轴建树,然后从下到上扫描上去,用 cnt 表示该区间下边比上边多几个
- 线段树操作:update:区间增减 query:直接取根节点的值

```
#include <cstdio>
#include <cstring>
#include <cctype>
#include <algorithm>
using namespace std;
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1

const int maxn = 2222;
int cnt[maxn << 2];
double sum[maxn << 2];
double X[maxn];
struct Seg {
        double h , l , r;
        int s;
        Seg(){}
        Seg(double a,double b,double c,int d) : l(a) , r(b) , h(c) , s(d) {}
        bool operator < (const Seg &cmp) const {
                return h < cmp.h;
        }
}ss[maxn];
```

```
void PushUp(int rt,int l,int r) {
        if (cnt[rt]) sum[rt] = X[r+1] - X[l];
        else if (l == r) sum[rt] = 0;
        else sum[rt] = sum[rt<<1] + sum[rt<<1|1];
}
void update(int L,int R,int c,int l,int r,int rt) {
        if (L <= l && r <= R) {
                cnt[rt] += c;
                PushUp(rt , l , r);
                return ;
        }
        int m = (l + r) >> 1;
        if (L <= m) update(L , R , c , lson);
        if (m < R) update(L , R , c , rson);
        PushUp(rt , l , r);
}
int Bin(double key,int n,double X[]) {
        int l = 0 , r = n - 1;
        while (l <= r) {
                int m = (l + r) >> 1;
                if (X[m] == key) return m;
                if (X[m] < key) l = m + 1;
                else r = m - 1;
        }
        return -1;
}
```

```
int main() {
        int n , cas = 1;
        while (~scanf("%d",&n) && n) {
                int m = 0;
                while (n --) {
                        double a , b , c , d;
                        scanf("%lf%lf%lf%lf",&a,&b,&c,&d);
                        X[m] = a;
                        ss[m++] = Seg(a , c , b , 1);
                        X[m] = c;
                        ss[m++] = Seg(a , c , d , -1);
                }
                sort(X , X + m);
                sort(ss , ss + m);
                int k = 1;
                for (int i = 1 ; i < m ; i ++) {
                        if (X[i] != X[i-1]) X[k++] = X[i];
                }
                memset(cnt , 0 , sizeof(cnt));
                memset(sum , 0 , sizeof(sum));
                double ret = 0;
                for (int i = 0 ; i < m - 1 ; i ++) {
                        int l = Bin(ss[i].l , k , X);
                        int r = Bin(ss[i].r , k , X) - 1;
                        if (l <= r) update(l , r , ss[i].s , 0 , k - 1, 1);
                        ret += sum[1] * (ss[i+1].h - ss[i].h);
                }
                printf("Test case #%d\nTotal explored area: %.2lf\n\n",cas++ , ret);
        }
        return 0;
}
```

- hdu1828 Picture
- 题意:矩形周长并
- 思路:与面积不同的地方是还要记录竖的边有几个(numseg 记录),并且当边界重合的时候需要合并(用 lbd 和 rbd 表示边界来辅助)
- 线段树操作:update:区间增减 query:直接取根节点的值

```
#include <cstdio>
#include <cstring>
#include <cctype>
#include <algorithm>
using namespace std;
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1

const int maxn = 22222;
struct Seg{
        int l , r , h , s;
        Seg() {}
        Seg(int a,int b,int c,int d):l(a) , r(b) , h(c) , s(d) {}
        bool operator < (const Seg &cmp) const {
                if (h == cmp.h) return s > cmp.s;
                return h < cmp.h;
        }
}ss[maxn];
bool lbd[maxn<<2] , rbd[maxn<<2];
int numseg[maxn<<2];
int cnt[maxn<<2];
int len[maxn<<2];
void PushUP(int rt,int l,int r) {
        if (cnt[rt]) {
                lbd[rt] = rbd[rt] = 1;
                len[rt] = r - l + 1;
                numseg[rt] = 2;
        } else if (l == r) {
                len[rt] = numseg[rt] = lbd[rt] = rbd[rt] = 0;
        } else {
                lbd[rt] = lbd[rt<<1];
                rbd[rt] = rbd[rt<<1|1];
                len[rt] = len[rt<<1] + len[rt<<1|1];
                numseg[rt] = numseg[rt<<1] + numseg[rt<<1|1];
                if (lbd[rt<<1|1] && rbd[rt<<1]) numseg[rt] -= 2;//两条线重合
        }
}

```

```
void update(int L,int R,int c,int l,int r,int rt) {
        if (L <= l && r <= R) {
                cnt[rt] += c;
                PushUP(rt , l , r);
                return ;
        }
        int m = (l + r) >> 1;
        if (L <= m) update(L , R , c , lson);
        if (m < R) update(L , R , c , rson);
        PushUP(rt , l , r);
}
int main() {
        int n;
        while (~scanf("%d",&n)) {
                int m = 0;
                int lbd = 10000, rbd = -10000;
                for (int i = 0 ; i < n ; i ++) {
                        int a , b , c , d;
                        scanf("%d%d%d%d",&a,&b,&c,&d);
                        lbd = min(lbd , a);
                        rbd = max(rbd , c);
                        ss[m++] = Seg(a , c , b , 1);
                        ss[m++] = Seg(a , c , d , -1);
                }
                sort(ss , ss + m);
                int ret = 0 , last = 0;
                for (int i = 0 ; i < m ; i ++) {
                        if (ss[i].l < ss[i].r) update(ss[i].l , ss[i].r - 1 , ss[i].s , lbd , rbd - 1 , 1);
                        ret += numseg[1] * (ss[i+1].h - ss[i].h);
                        ret += abs(len[1] - last);
                        last = len[1];
                }
                printf("%d\n",ret);
        }
        return 0;
}
```

练习

- hdu3265 Posters
- hdu3642 Get The Treasury
- poj2482 Stars in Your Window
- poj2464 Brownie Points II
- hdu3255 Farming
- ural1707 Hypnotoad's Secret
- uva11983 Weird Advertisement