HW 6

PB17111603 刘高聪

T1.

(a). 证： 命题 1：无论区间是否重叠，最大重叠点一定存在。

证明：这个命题是显而易见的。

命题 2：存在一个区间[a,b]其中 a<=b，使得这个区间内的所有点都是最大重叠点。

证明：由命题 1 可知，对于区间集合 S，必然存在最大重叠点 p。假设与点 p 重叠的区间有 n 个，分别为 Interval1，Interval2,,,,, intervaln。n >=1 并且 n<= S 的大小

由于 p 与这 n 个区间都有重叠，那么 p 必然属于这 n 个区间的公共部分.(这也是一个命题，不过结果显而易见，用反证法即可证明)。那么只需要证明任意两个区间的重叠部分也是一个区间，然后再利用 n 个区间的重合部分，也是由两个区间重合不断复合得到的，并且这个重叠区间的所有的点的重叠区间数量和最大重叠点 p 是一样的。

命题 3：任意两个区间的重叠部分也是一个区间.

证明：考虑区间 A = [s,e],B[s1,e1]。如果 e1<s 或者 s1>e 的话那么重叠部分为空区间。如果不是那重叠区间就是[max(s,s1),min(e,e1)]

所以命题得证。

命题 4：最大重叠点一定是其中一个区间的端点。

证明：由命题 2 和命题 3 可以发现由于区间 H 非空，并且 H 是由多个区间复合重叠而成，再结合命题 3.可以看见任意两个区间重叠如果非空，那么重叠区间是由原区间的端点组成了新的重叠区间端点。非空区间 H 一定包含了某个原区间的端点。

(b).

设有 n 个区间，将所有 2n 个点从小到大排序，对于排序后的第 i 个点，若它是某个区间的左端点，则 p[i]=1，若它是某个区间的右端点，则 p[i]=-1。由第一问可知，所求的点一定是某条线段的端点，所以从端点集合中找出被最多区间覆盖的那个。若一个端点是排序后的第 i 个点，则有个 SUM(s[1],s[i])个区间覆盖这个点。

步骤 1：基础数据结构

红黑树，p[x]=1 表示它是区间的左端点，p[x]=-1 表示它是区间的右端点

步骤 2：附加信息

v[x]：以 x 为根的所有结点的 p 值之和

m[x]：以 x 为根的树中，最大重叠数

o[x]： 以 x 为根的所有结点中的最大覆盖点

步骤 3：对信息的维护

$$v[x] = v[left[x]] + p[x] + v[right[x]] ,$$

$$m[x] = \max \begin{cases} m[left[x]] & (\text{max is in } x\text{'s left subtree}) , \\ v[left[x]] + p[x] & (\text{max is at } x) , \\ v[left[x]] + p[x] + m[right[x]] & (\text{max is in } x\text{'s right subtree}) \end{cases}$$

.o[x] = o[left[x]] (max is in x’ s left subtree)

. p[x] (max is at x)

. o[right[x]] (max is in x’ s right subtree)

. v,m,o 都只依赖于左右子树和本身，故于定理 14.1，插入和删除操作渐进时间依旧为 O(lgn).而 FIND-POM 操作只需 return T.root->o ,时间复杂度为 O(1)。

T2.

(a). 实际上第七行所需要的时间与 x 的孩子数成正比，因为需要将其孩子的父指针指向 NIL.

(b). O(c + x.degree).

T3.

Algorithm 1 MAKE-SET(x):

```
1.  Let o be an object with three fields, next, value, and set
2.  Let L be a linked list object with head = tail = o ,
3.  o.next = NIL
4.  o.set = L
5.  o.value = x
6.  L.length = 1
7.  return L
```

```
1.  Algorithm 2 FIND-SET(x)
2.      return  o.set.head.value
```

```
1.  Algorithm 3 UNION(x,y)
2.      If  x.length  >  y.length
3.          L1= x.set
4.          L2 = y.set
5.      Else
6.          L1=y.set
7.          L2=x.set
8.      L1.tail.next = L2.head
9.      z = L2.head
10.     while z.next != NIL do
```

```
            z.set = L1
        end while
        L1.tail = L2.tail
        L1.length = L1.length + L2.length
        return L1
```