中国科学技术大学 计算机科学与技术系
University of Science and Technology of China
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

算法基础
Foundation of Algorithms

主讲人 徐 云

Fall 2018, USTC

第14章 数据结构的扩张

14.1 动态顺序统计

14.2 如何扩充一个数据结构

14.3 区间树

14.1 动态顺序统计

- OS树的定义
- 选择问题及算法
- 求秩问题及算法
- OS树的维护：插入
- OS树的维护：删除

OS树的定义（1）

- Def：

  OS(Order-Statistic)树是一棵红黑树在每个节点上扩充一个域size[x]而得到的，它是以x为根的子树中内部节点的总数(包括x)，即子树大小。

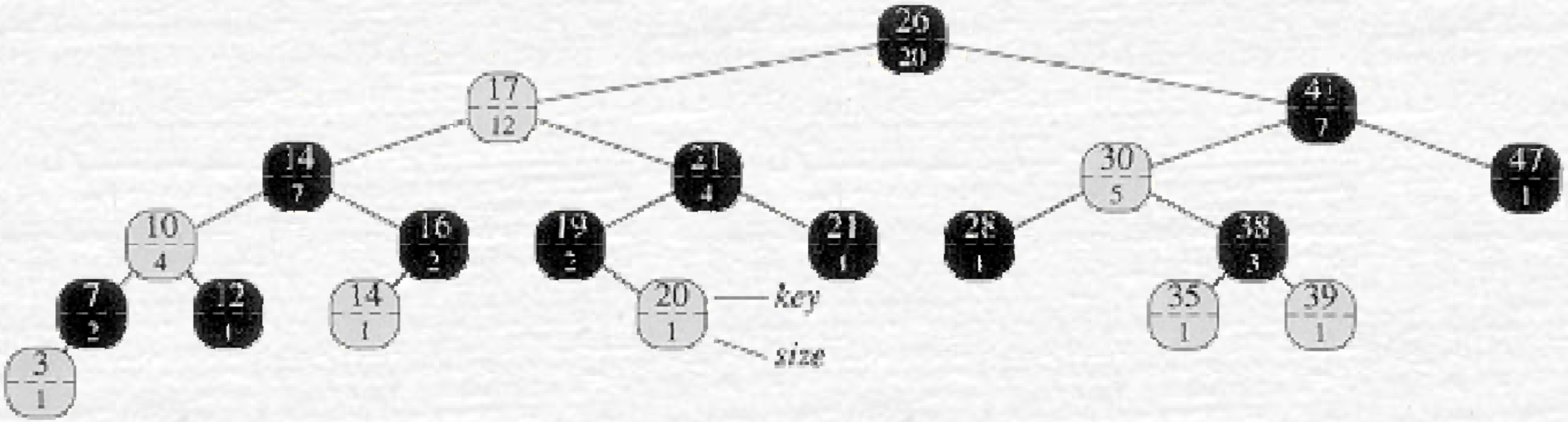
$$size[x]=\begin{cases} 0 & \text{if } x=nil[T] \\ size[left[x]]+size[right[x]]+1 & other \end{cases}$$

- 节点结构：

key
size

OS树的定义（2）

- OS树的一个图例：Fig. 14.1



选择问题及算法

- 选择问题：在以x为根的子树中，查找第i个最小元素。
- 算法：

OSSelect(x, i)

{

(1): r ← size[left[x]]+1;

(2): ①若i=r，则返回x；

②若i < r，则递归地在x的左子树中继续找第i个元素；

③若i > r，则递归地在x的右子树中继续找第i-r个元素；

}

时间：O(logn)

求秩问题及算法

- 求秩问题：在OS树中，给定元素x求其rank。
- 算法：

step 1: 在以x为根的子树中，x的秩：

r ← size[left[x]]+1；

step 2: ①若x是根，则返回r；

②若x是双亲的左子，则x在以p[x]为根的子树中的秩是r；

③若x是双亲的右子，则x在以p[x]为根的子树中的秩是r ← r+size[left[p[x]]]+1；

④x上移至p[x]；

重复②，③，④直至①成立时终止；

时间：O(logn)

OS树的维护：插入 (1)

- 插入算法
  - Phase 1：从根向下插入新节点，将搜索路径上所经历的每个节点的size+1，新节点的size置为1；
    - 附加成本：O(logn)
  - Phase 2：采用变色和旋转方法，从叶子向上调整；
    - 变色不改变size；
    - 旋转可能改变size：

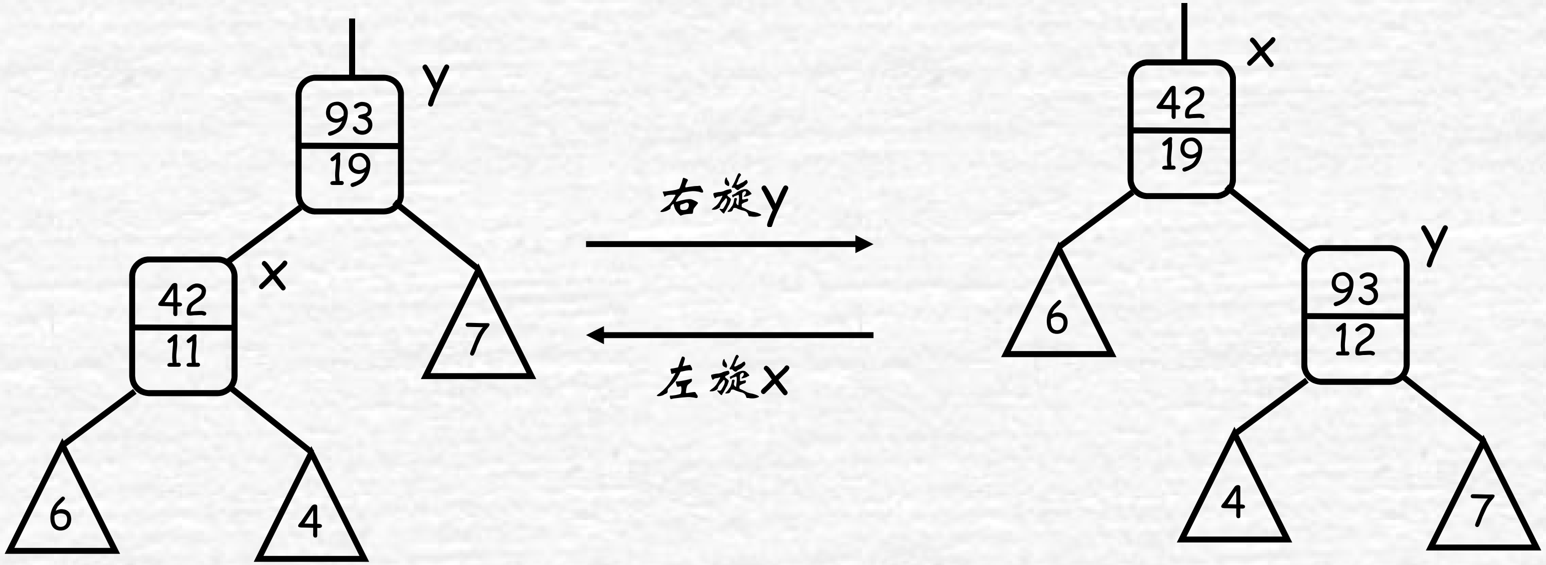
      ∵ 旋转是局部操作，

      又，只有轴上的两个节点的size可能违反定义

      ∴ 只需要在旋转操作后，对违反节点size进行修改
    - 附加成本：旋转为O(1)，总成本为O(logn)，

OS树的维护：插入（2）

- 例： LeftRotate(T, x)



- 左旋x后：

  size[y] ← size[x]

  size[x] ← size[left[x]]+size[right[x]]+1

  ∵插入过程至多有2个旋转

  ∴附加成本为O(1)

OS树的维护：删除

- 删除算法
  - Phase 1：物理上删除y，在删除y时从y上溯至根，将所经历的节点的size均减1；
    - 附加成本：O(logn)
  - Phase 2：采用变色和旋转方法，从叶子向上调整；
    - 变色不改变size；
    - 旋转可能改变size，至多有3个旋转；
    - 附加成本：O(logn)

- Remark：上面介绍的插入和删除均是有效维护，有效维护保证扩充前后的基本操作的渐近时间不变。

第14章 数据结构的扩张

14.1 动态顺序统计

14.2 如何扩充一个数据结构

14.3 区间树

14.2 如何扩充一个数据结构

- 扩充的目的
- 扩充的步骤
- 扩充红黑树的定理

扩充的目的和步骤

- 扩充的目的
  - 开发新的操作
  - 加速已有的操作
- 扩充的步骤
  ① 选择基本结构；
  ② 确定附加信息：如，数据、指针；
  ③ 维护附加信息和有效性；
  ④ 开发新操作。
- 例：OS树
  ① 选择红黑树作为OS树的基本结构；
  ② 在红黑树上增加size；
  ③ 有效性证明；
  ④ Select，Rank操作；

扩充红黑树的定理

- 定理14.1

假设f是红黑树T的n个节点上扩充的域，

对∀x∈T，假设x的f域的内容能够仅通过节点x、left[x]、right[x]的信息（包括f域）的计算就可以得到，则

扩充树上的插入和删除维护操作（包括对f域的维护）不改变原有的渐近时间O(logn)。

第14章 数据结构的扩张

14.1 动态顺序统计

14.2 如何扩充一个数据结构

14.3 区间树

14.3 区间树

- 基本概念
- 区间重叠的三分律
- 红黑树的扩充：区间树
- 查找算法及正确性

基本概念

- 区间：一个事件占用的时间
- 闭区间：实数的有序对$[t_1,t_2]$，使$t_1 \le t_2$
- 区间的对象表示：$[t_1,t_2]$可以用对象i表示，有两个属性：

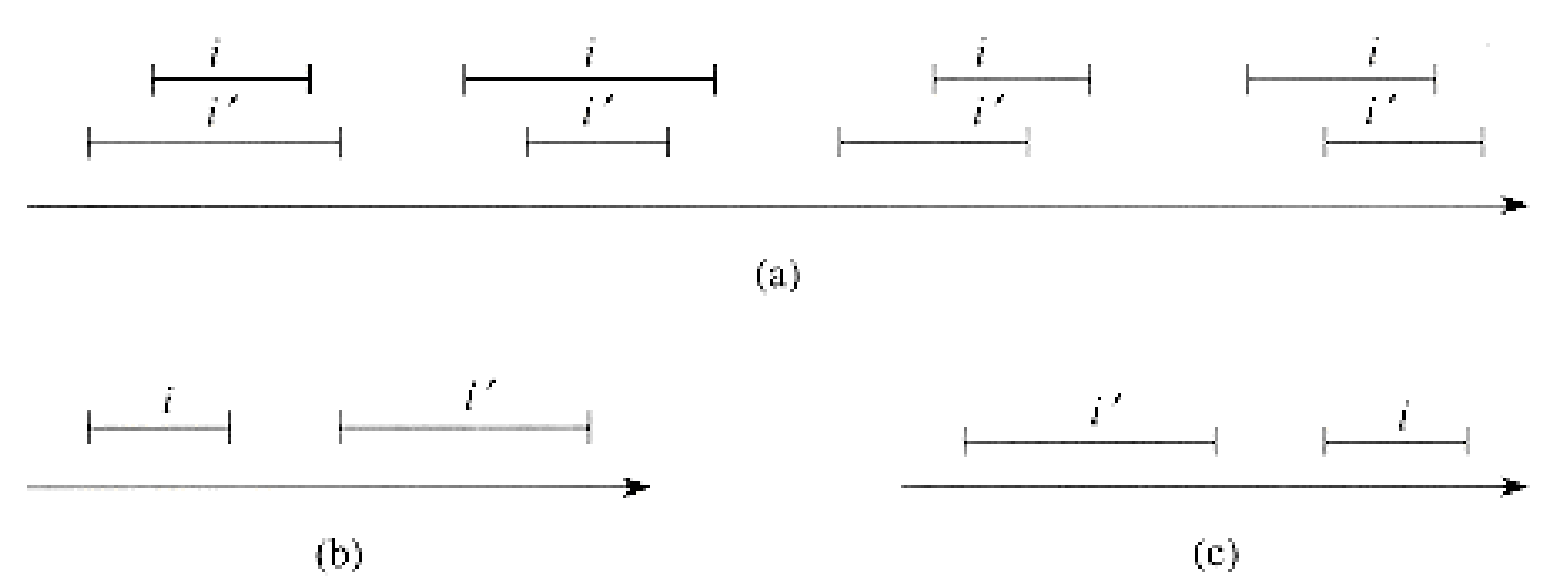
  $low[i]=t_1$　　//起点或低点

  $high[i]=t_2$　　//终点或高点
- 区间的重叠：

  $i \cap i' \neq \varnothing$

  $\Leftrightarrow (low[i] \le high[i'])$ and $(low[i'] \le high[i])$

区间重叠的三分律



(a)i和i'重叠

(b)i在i'前：high[i]<low[i']

(c)i'在i前：high[i']<low[i]

红黑树的扩充：区间树

- Step 1：基本结构

  以红黑树为基础，对∀x∈T，x包含区间int[x]的信息(低点和高点)，key=low[int[x]]。

- Step 2：附加信息

  max[x]=max(high[int[x]], max[left[x]], max[right[x]])

- Step 3：维护附加信息(有效性)

  由定理14.1及max的定义⇒有效

- Step 4：开发新操作

  查找与给定区间重叠的区间

节点x

key
max

查找算法及正确性（1）

- 查找算法IntervalSearch(T, i)

  基本思想：

  step 1：x ← root[T]；//从根开始查找

  step 2：若x≠nil[T]且i与int[x]不重叠

  　　if x的左子树非空且左子树中最大高点≥low[i] then

  　　　x ← left[x]；//到x的左子树中继续查找

  　　else　//左子树必查不到，到右子树查

  　　　x ← right[x]；

  step 3：返回x　//x=nil or i和x重叠

查找算法及正确性（2）

- 算法的正确性
  - 关键说明：如果存在重叠区间，则一定会找到。
  - 定理14.2
    - case 1：若算法从x搜索到左子树，则左子树中包含一个与i重叠的区间或在x的右子树中没有与i重叠的区间；
    - case 2：若从x搜索到右子树时，则在左子树中不会有与i重叠的区间
  - 证明：先证case2，然后再证case1
    - case 2：走x右分支条件是

      left[x]=nil or max[left[x]] < low[i]

      若左子树为空，则左子树不含有与i重叠的区间；

查找算法及正确性（3）

- case 2(cont.)：

  若左子树非空，由于有max[left[x]] < low[i]

  ∵ max[left[x]]是左子树中最大高点

  ∴ 左子树中的区间不可能与i重叠

- case 1：

  若左子树包含与i重叠的区间，则走左分支正确；

  若左子树无区间与i重叠，下证x的右子树中也无区间与i重叠。

  ∵ max[left[x]]≥low[i]

  ∴ 存在区间i′∈{x 左子树的区间}，使得

  high[i′] = max[left[x]]≥low[i]

  ⇒ 只有i′和i重叠或i在i′前

  又左子树无区间与i重叠 ∴只有i在i′前，即 high[i] < low[i′]

  对∀区间i″∈{x右子树的区间}，由BST性质有 low[i′]≤low[i″]

  由此 ⇒ high[i] < low[i″]， ∴ i和i″不重叠

  综上，定理得证。 □

End of Ch14