

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

# 算法设计与分析

## Design and Analysis of Algorithms

主讲人 徐云  
Fall 2018, USTC

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

- Part 1 Foundation
- Part 2 Sorting and Order Statistics
- Part 3 Data Structure
  - chap 10 Elementary Data Structures
  - chap 11 Hash Tables
  - chap 12 Binary Search Trees
  - chap 13 Red-Black Trees
  - chap 14 **Augmenting Data Structures**
- Part 4 Advanced Design and Analysis Techniques
- Part 5 Advanced Data Structures
- Part 6 Graph Algorithms
- Part 7 Selected Topics
- Part 8 Supplement

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

## 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

2018/10/9 算法设计与分析 4

## 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。

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

- 节点结构:

key
size

2018/10/9 算法设计与分析 5

## OS树的定义 (2)

- OS树的一个图例：Fig. 14.1

2018/10/9 算法设计与分析 6

## 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):  $r \leftarrow size[left[x]] + 1$ ;
  (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:  $O(\log n)$**

2018/10/9

算法设计与分析

7

## 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**:  
 $RankOS(T, x)$ 

```

 $r = size[left[x]] + 1$ 
 $y = x$ 
while  $y \neq root[T]$ :
  if  $y == right[p[y]]$ :
     $r += size[left[y]] + 1$ 
   $y = p[y]$ 
return  $r$ 

```

**step 1**: 在以 $x$ 为根的子树中,  $x$ 的秩:  $r \leftarrow size[left[x]] + 1$ ;  
**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 $r \leftarrow r + size[left[p[x]]] + 1$ ;  
 ④ $x$ 上移至 $p[x]$ ;  
 重复②, ③, ④直至①成立时终止;  
**时间:  $O(\log n)$**

2018/10/9

算法设计与分析

8

## 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:  $O(\log n)$
  - **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 $O(\log n)$ ,

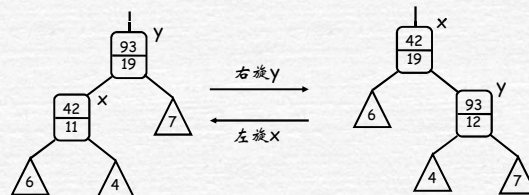
2018/10/9

算法设计与分析

9

## OS树的维护: 插入 (2)

- **例: LeftRotate( $T, x$ )**



➢ 左旋 $x$ 后:

```

 $size[y] \leftarrow size[x]$ 
 $size[x] \leftarrow size[left[x]] + size[right[x]] + 1$ 
∴ 插入过程至多有2个旋转
∴ 附加成本为 $O(1)$ 

```

2018/10/9

算法设计与分析

10

## 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:  $O(\log n)$
  - **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:  $O(\log n)$
- **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。

2018/10/9

算法设计与分析

11

## 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

2018/10/9

算法设计与分析

13

## 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;

2018/10/9

算法设计与分析

14

## 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 $\forall x \in T$ , 假设 $x$ 的 $f$ 域的内容能够仅通过节点 $x$ 、 $\text{left}[x]$ 、 $\text{right}[x]$ 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 $O(\log n)$ 。

2018/10/9

算法设计与分析

15

## 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

2018/10/9

算法设计与分析

17

## 基本概念

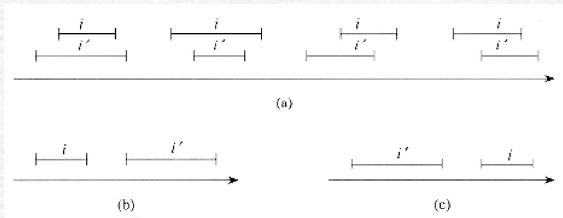
- 区间: 一个事件占用的时间
- 闭区间: 实数的有序对 $[t_1, t_2]$ , 使 $t_1 \leq t_2$
- 区间的对象表示:  $[t_1, t_2]$ 可以用对象 $i$ 表示, 有两个属性:
  - $\text{low}[i] = t_1$  // 起点或低点
  - $\text{high}[i] = t_2$  // 终点或高点
- 区间的重叠:
  - $i \cap i' \neq \emptyset$
  - $\Leftrightarrow (\text{low}[i] \leq \text{high}[i']) \text{ and } (\text{low}[i'] \leq \text{high}[i])$

2018/10/9

算法设计与分析

18

## 区间重叠的三分律



(a)  $i$  和  $i'$  重叠

(b)  $i$  在  $i'$  前:  $\text{high}[i] < \text{low}[i']$

(c)  $i'$  在  $i$  前:  $\text{high}[i'] < \text{low}[i]$

2018/10/9

算法设计与分析

19

## 红黑树的扩充: 区间树

### Step 1: 基本结构

以红黑树为基础, 对  $\forall x \in T$ ,  $x$  包含区间  $\text{int}[x]$  的信息 (低点和高点),  $\text{key} = \text{low}[\text{int}[x]]$ 。

### Step 2: 附加信息 左子树高点、右子树高点、本结点高点三者最大值

$\text{max}[x] = \max(\text{high}[\text{int}[x]], \text{max}[\text{left}[x]], \text{max}[\text{right}[x]])$

### Step 3: 维护附加信息 (有效性)

由定理14.1及  $\text{max}$  的定义  $\Rightarrow$  有效

节点  $x$

### Step 4: 开发新操作

查找与给定区间重叠的区间



2018/10/9

算法设计与分析

20

## 查找算法及正确性 (1)

查找与给定区间重叠的区间

### 查找算法 $\text{IntervalSearch}(T, i)$

基本思想:

step 1:  $x \leftarrow \text{root}[T]$ ; // 从根开始查找

step 2: 若  $x = \text{nil}[T]$  且  $i$  与  $\text{int}[x]$  不重叠

if  $x$  的左子树非空且左子树中最大高点  $\geq \text{low}[i]$  then

$x \leftarrow \text{left}[x]$ ; // 到  $x$  的左子树中继续查找

else // 左子树必查不到, 到右子树查

$x \leftarrow \text{right}[x]$ ;

step 3: 返回  $x$  //  $x = \text{nil}$  or  $i$  和  $x$  重叠

2018/10/9

算法设计与分析

21

## 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 case 2, 然后再证 case 1

- case 2: 走  $x$  右分支条件是

$\text{left}[x] = \text{nil}$  or  $\text{max}[\text{left}[x]] < \text{low}[i]$

若左子树为空, 则左子树不含有与  $i$  重叠的区间;

2018/10/9

算法设计与分析

22

## 查找算法及正确性 (3)

严格数学证明:

- case 2(cont.):  $\text{high}[i'] < \text{max}[\text{left}[x]] < \text{low}[i]$ , 即  $\text{high}[i'] < \text{low}[i]$  所以  $i'$  在  $i$  之前

若左子树非空, 由于有  $\text{max}[\text{left}[x]] < \text{low}[i]$

$\therefore \text{max}[\text{left}[x]]$  是左子树中最大高点 左子树所有区间都在  $\text{low}[i]$  之前

$\therefore$  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。

$\therefore \text{max}[\text{left}[x]] \geq \text{low}[i]$

$\therefore$  存在区间  $i' \in \{x \text{ 左子树的区间}\}$ , 使得

$\text{high}[i'] = \text{max}[\text{left}[x]] \geq \text{low}[i]$

$\Rightarrow$  只有  $i'$  和  $i$  重叠或  $i$  在  $i'$  前

又左子树无区间与  $i$  重叠  $\therefore$  只有  $i$  在  $i'$  前, 即  $\text{high}[i] < \text{low}[i']$

对  $\forall$  区间  $i'' \in \{x \text{ 右子树的区间}\}$ , 由 BST 性质有  $\text{low}[i'] \leq \text{low}[i'']$

由此  $\Rightarrow \text{high}[i] < \text{low}[i'']$ ,  $\therefore i$  和  $i''$  不重叠  $i$  在  $i''$  之前

综上, 定理得证。  $\square$

2018/10/9

算法设计与分析

23

End of Chap14