线段树 Interval Tree

J60 唐文斌
failedbamboo@gmail.com

动态统计问题1

- 有一个包含n个元素的整数数组A
  - 每次可以修改一个元素
  - 也可以查询一个区间$[l, r]$内所有元素之和

- 如何设计算法，使得修改和询问操作的时间复杂度尽量低？

动态统计问题1(con't)

- 朴素的数组模拟
  - 修改元素: O(1)
  - 区间查询
    - 依次访问A[$l..r$]的每一个元素并累加
    - $O(n)$
- 如何提高查询的效率?
- 优化思想: 区间加法
  - 例如 A[1..100] = A[1..50] + A[51,100]
  - 若已知A[1..50]和A[51..100]则只需一次运算

Interval Tree

- 线段树(又称区间树)的思想:
  - 通过对区间信息的维护，加速查询操作
  - 使修改和区间求和操作时间均为$O(logN)$

线段树的构造思想

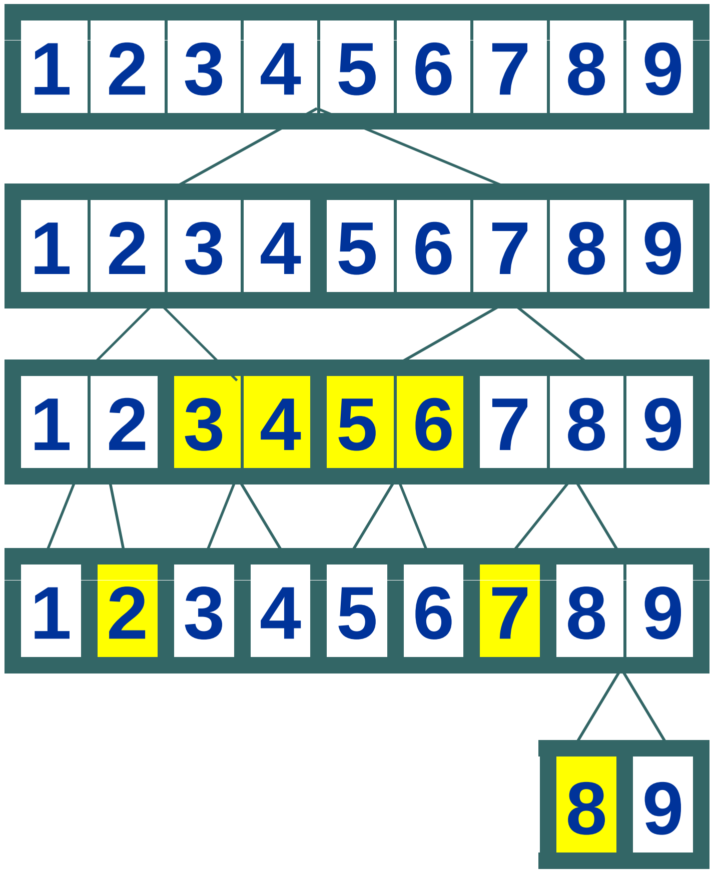
- 线段树是一棵二叉树树中的每一个结点表示了一个区间$[p, r)$ //这里我们使用[)

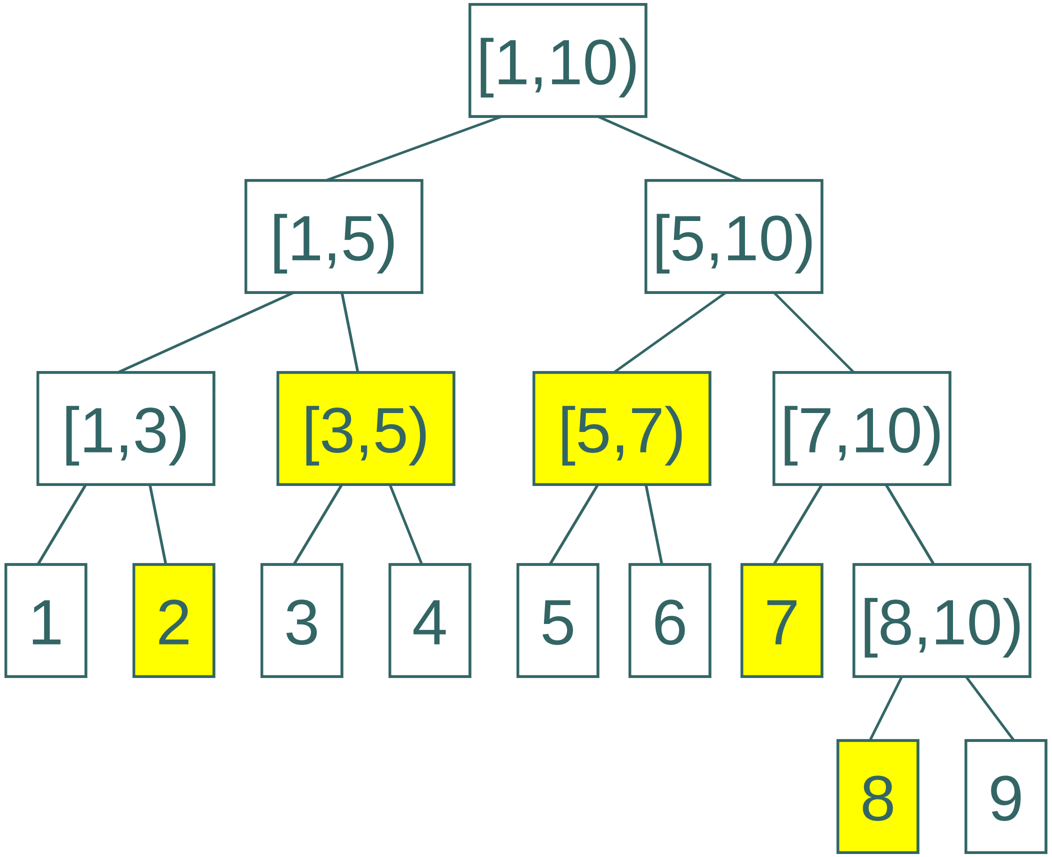
- 每一个叶子节点表示了一个单位区间

- 对于每一个非叶结点所表示的区间$[p, r)$
  - 左儿子表示的区间为$[p, m)$
  - 右儿子表示的区间为$[m, r)$
  - 其中$m = (p + r) / 2$

线段树举例

- 线段[1, 10)的线段树和[2, 9)的分解





线段树的性质(1)

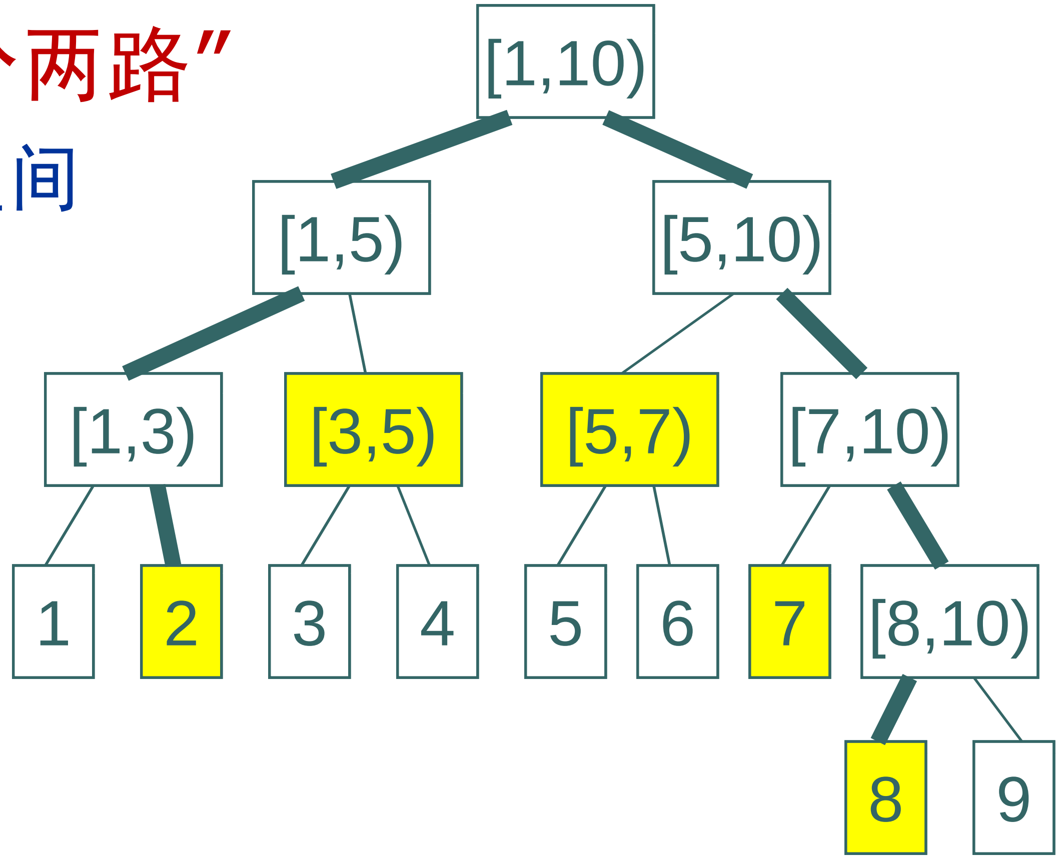
- 若线段树所表示区间为[a, b), 则:
- 每层都是[a, b)的划分. 记$L=b-a$,共$\log_2 L$层
- 任意两个结点的关系
  - 1)包含关系
  - 2)没有公共部分
  - *)不可能部分重叠

线段树的性质(2)

- 对于给定一个叶子p
  - 从根到p路径上所有结点(即p的所有直系祖先)代表的区间都包含点p, 且其他结点代表的区间都不包含点p
- 给定一个区间$[l, r)$, 可以把它分解为不超过$2\log_2 L$条不相交线段的并(why?)

线段树的基本操作

- 找点：根据定义，从根一直走到叶子O(log*L*)
- 区间分解："兵分两路"
  - 每层最多两个区间
  - 总时间$4log_2L$
  - (Flash演示)



如何使用线段树?

- 单纯的线段树
  - 与BST不同，自身没有任何数据
  - 没有序关系
- 为节点(区间)设计附加信息
  - 通过对附加信息的维护与计算
  - 解决各类区间问题

动态统计问题1

- 有一个包含n个元素的整数数组A
  - 每次可以修改一个元素
  - 也可以查询一个区间$[l, r]$内所有元素之和

- 如何设计算法，使得修改和询问操作的时间复杂度尽量低？

动态统计问题1(解答)

- 设计附加信息:
  - Sum(*p*): 结点*p*所表示的区间内所有元素的和
- 操作:
  - 修改元素
    - 修改对应叶子上的Sum为当前键值
    - 直系祖先自底向上维护(只需修改*logn*个sum)
      - Sum(*p*) = Sum(*p*.left) + Sum(*p*.right)
  - 区间求和
    - 将对应区间进行分解,把所有Sum值相加

如何实现线段树？

- C++语言实现演示

线段树 VS. AVL-Tree

内容	Interval Tree	AVL Tree
区间操作	支持	支持
应用范围	只适用区间维护	更广
单次操作时间复杂度	O(log*N*)	O(log*N*)
算法常数(运行速度)	小(快)	大(慢)
编程复杂度	低	极高

动态统计问题2

需要新的"附加信息"

- 包含n个元素的数组A
  - Add(i, j, k)：给$A[i]$, $A[i+1]$, … $A[j]$均增加k
  - Query(i)　：求A[i]

- 先看看是否可以沿用刚才的附加信息
  - Query(i)就是读取i对应的结点上的Sum值
  - Add呢？极端情况下，如果是修改整个区间，则所有结点都需要修改！

动态统计问题2（解答）

- 方法一：
  - 设delta(*p*)为对整个*p*所代表区间的Add之和
  - 假设 *p* 所代表区间是 [*i*, *j*+1), delta(p)代表形如Add(*i*, *j*, *k*)的所有*k*之和
    - 如果[*i*, j+1)不是某个p所代表的区间？ --区间分解
    - Add具有叠加性
  - Query(*i*)
    - 把所有*i*结点直系祖先的delta值相加，就是*A*[*i*]的增加量

动态统计问题2（解答）

- 方法二(推荐)

- “Lazy思想”:
  - 如果需要对一个区间中每一个结点进行操作
  - 我们不妨先别忙着操作. 而是区间分解后在所有大区间上做一个”标记”
  - 当操作遇到”标记”时再进行处理(标记传递)

动态统计问题2（解答）

- 对每一个区间p设置一个标记 tot(p)
  - tot(p)定义与delta(p)类似，表示当前需要对整个p所代表区间进行整体增量
- Add(i, j, k)
  - 将[i, j+1)进行区间分解, 修改区间的tot值
- Query(i)
  - 从根结点开始不断往下走
  - 如果走到一个tot(p)不为0的非叶结点p,传递标记
    - tot(p.left)+=tot(p) tot(p.right) += tot(p) tot(p) = 0
  - i所在叶结点的tot值即为该元素的增加量

动态统计问题3

- 包含n个元素的数组A
  - Add(i, j, k) :给$A[i]$, $A[i+1]$, ... $A[j]$均增加k
  - Sum(p, q) :求$A[p]+A[p+1]+\ldots+A[q]$
  - 区间增量 + 区间和查询

- 显然动态统计问题I和II都是它的特殊情况
  - 问题1中, Add操作的$i=j$
  - 问题2中, Sum操作的$p=q$

动态统计问题3（解答）

- 类似《动态统计问题2》中方法一，我们也可以设计类似的静态附加信息，但是较繁琐.
- 故推荐上面提到的基于"Lazy思想"的方法
  - 更易理解
  - 应用范围更大

动态统计问题3（解答）

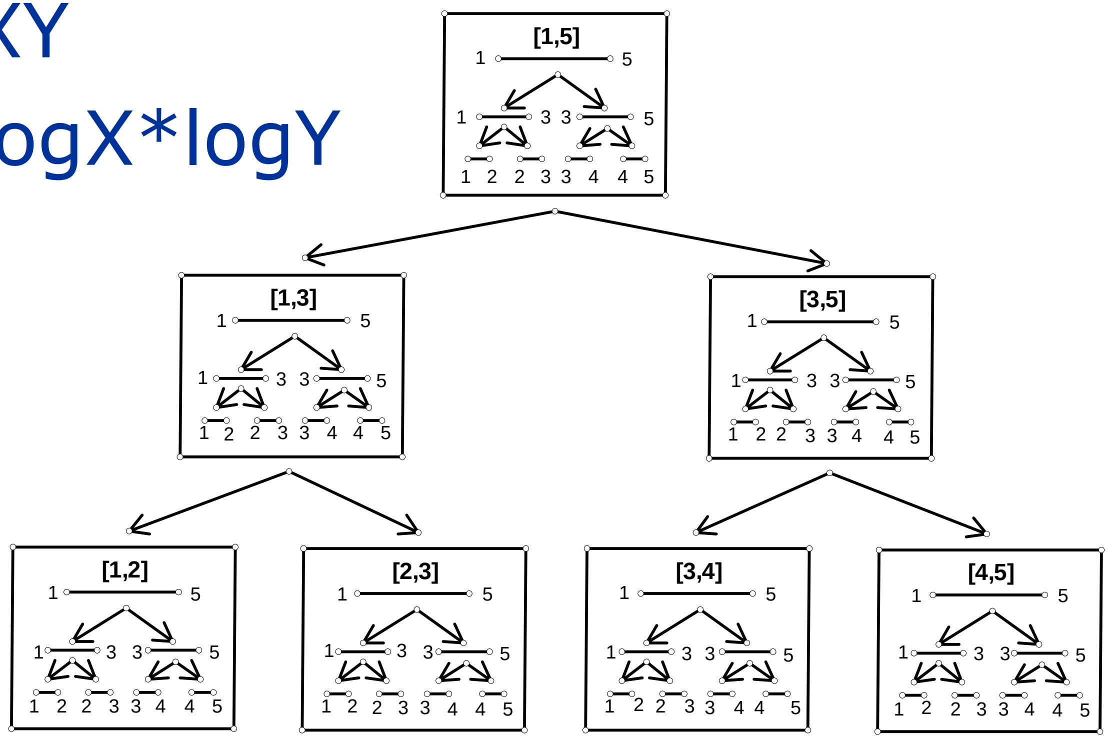
- 附加信息:
  - tot(*p*)：表示当前对整个*p*所代表区间的整体增量
  - sum(*p*): 结点*p*所表示区间内所有元素的和
- Add(*i*, *j*, *k*)
  - 将[*i*, *j*]进行区间分解, 更新区间的tot值(标记)以及其祖先的sum值
- Sum(*p*, *q*)
  - 将[*p*, *q*]进行区间分解,将这些区间sum值累加

动态统计问题3（解答）

- Lazy处理：
  - 在访问到一个tot(*p*)不为0的非叶结点*p*时
  - 进行标记传递操作
- 标记传递：
  - `tot(p.left)+=tot(p);  tot(p.right) += tot(p);`
  - `sum(p.left)+= tot(p) * p.left.length();`
  - `sum(p.right)+= tot(p) * p.right.length();`
  - `tot(p) = 0;`

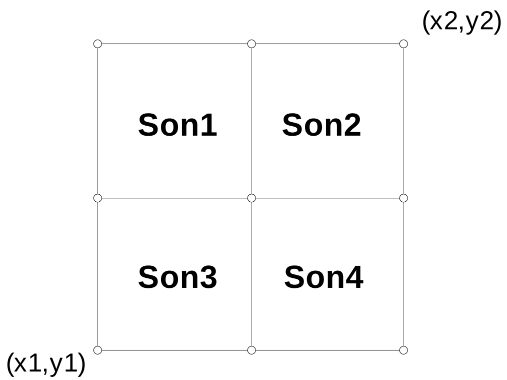
线段树的扩展－二维线段树

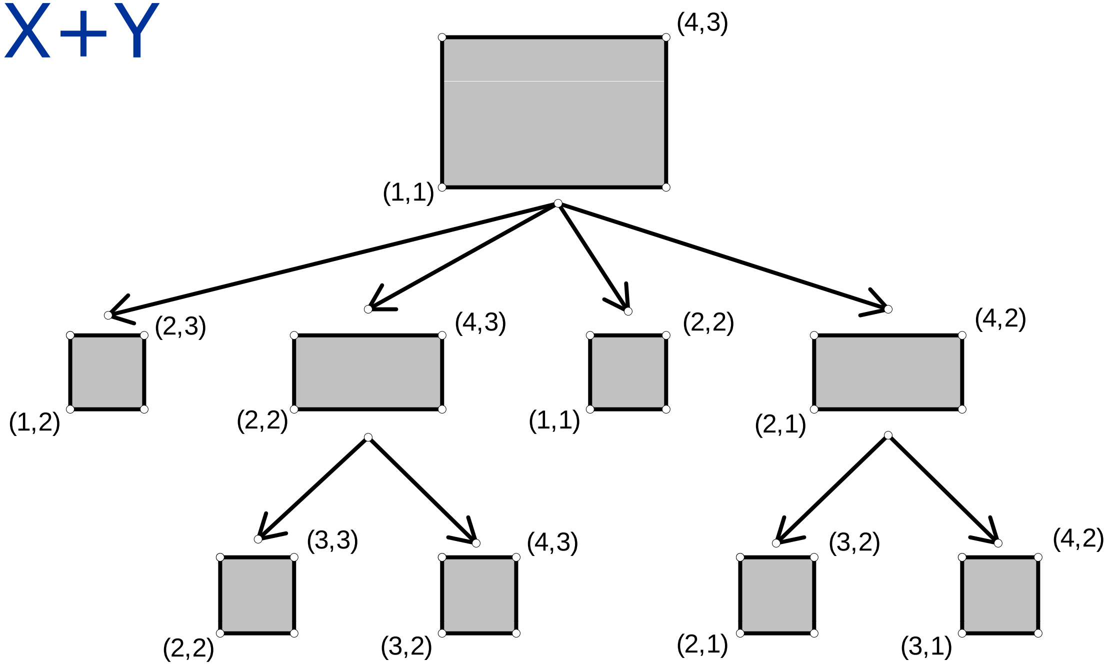
- 给原来线段树中的每个结点都加多一棵线段树(平衡树)，即“树中有树”
  - 空间复杂度：XY
  - 时间复杂度：logX*logY



线段树的扩展 – 矩形树

- 每个矩形分成四份
  - 空间复杂度：XY
  - 时间复杂度：X+Y



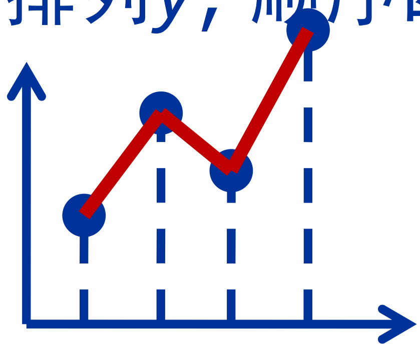


思考题(编程练习题)
区间最小值查询问题

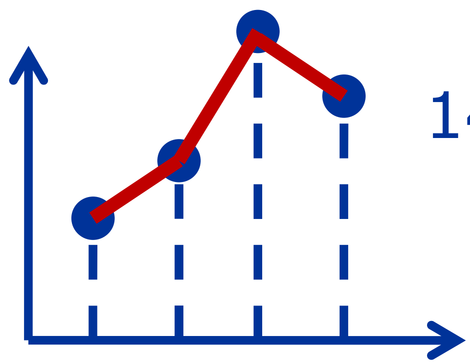
- 包含n个元素的数组A
  - Modify(i, j):设$A[i] = j$
  - Min(p, q)　:求$\mathbf{min}\{A[p],A[p+1],\ldots,A[q]\}$

思考题(Hard!!)
Totem(中国国家队组队赛)

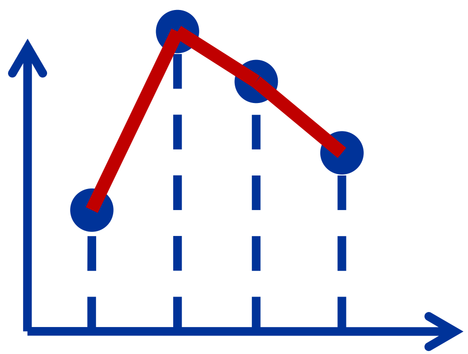
- 给定1~N的一个排列y, 顺序四元组y_a, y_b, y_c, y_d
- 考察相对大小:
  - 闪电：1324



  - 山峰：1243



1432



- 求闪电图腾与山峰图腾的数量差
- $N\leq 200{,}000$

Thank you!