线段树的合并

——不为人知的实用技巧

杭州二中 黄嘉泰

线段树?

- OI中最常用的数据结构
- 形式化的线段树：

  有一列元素 $a_1, a_2, \ldots, a_n \in S$

  S 上有二元运算+，使得对于任意$a, b \in S$，都有 $a + b \in S$，并且+ 满足结合律。（$\{S, +\}$是半群）

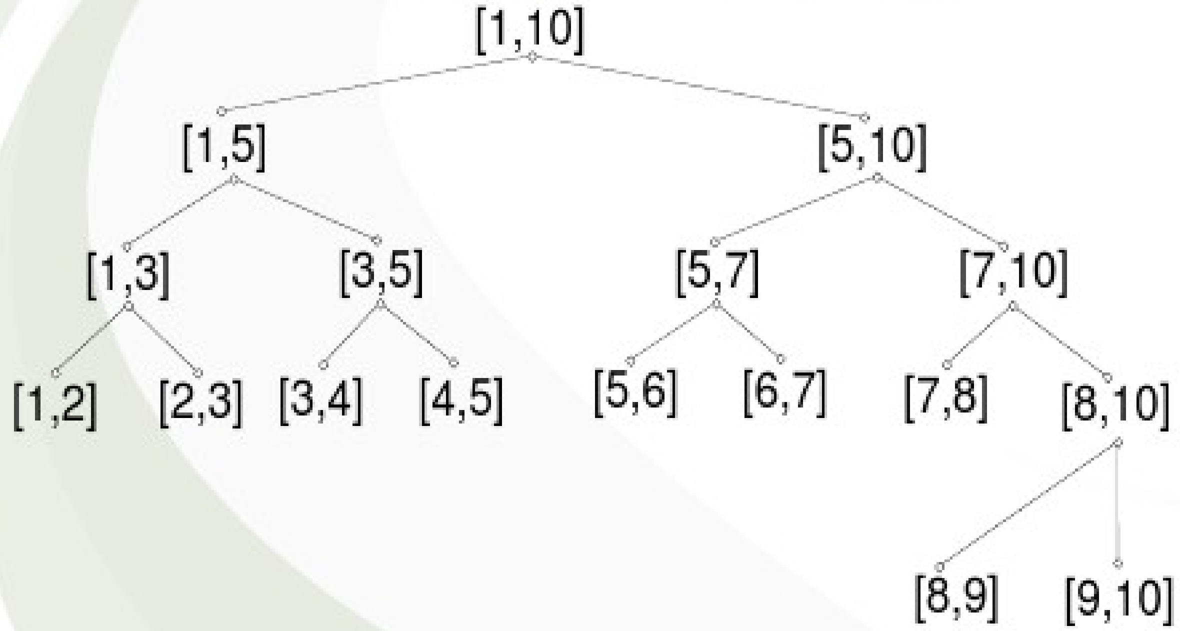
  另外，+运算必须能高效地计算完成，譬如O(1)。
- 要求高效地支持：

  修改某个a_i；

  给定 l, r，回答$a_l + a_{l+1} + \ldots + a_r$。

线段树

- 由于+运算有结合律，设法维护一些连续元素的和，使得每个元素出现在O(logn)个维护的和当中，每个询问能表示成O(logn)个维护的和的和。
- 一个简单有效的办法就是现在常用的“线段树”。从[1,n]开始，把每段连续的元素尽可能均匀地分成两半，直到成为单个元素为止。
- 这个关系构成了一棵近似丰满的二叉树，每个节点代表了一些连续(或单个)元素，我们在上面维护它们的和。



线段树

- 具体实现不再赘述
- 一些常见的问题譬如维护区间和由于其特殊性，可以在支持对连续的一段元素做出某种程度的修改的前提下，仍然高效地维护区间和。一般情况下不一定能这么做。
- 有时也用来实现map，这时key是字长内的整数，同时能维护key连续的一些元素的信息。

线段树的特点

- 当确定了元素个数n，或者key的范围[1,U]，建出的线段树形态是唯一的。
- 对两棵key的上界相同的线段树进行参数相同的单点更新/区间询问时，所访问到的节点也是一致的。
- 由此我们有了一些喜闻乐见的线段树用法。（详见CLJ去年hw2）
- 今天的内容也依赖于线段树严格的结构

线段树的合并

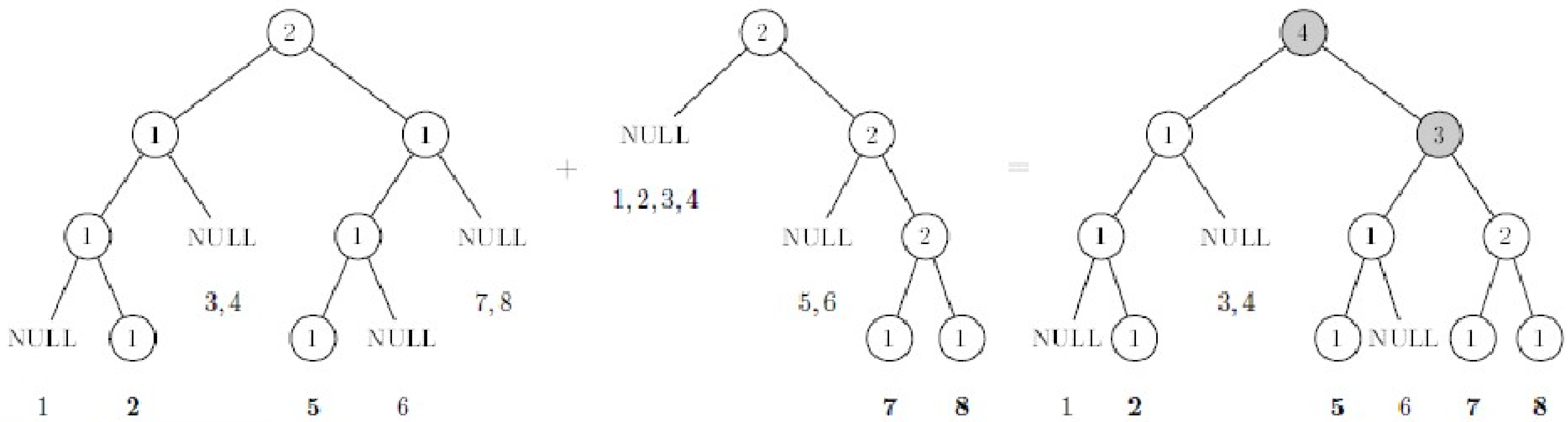
- 由线段树的定义我们不难写出下面的过程，来合并两棵代表范围相同的线段树

```
merge(a,b):
        如果a,b中有一个不含任何元素，就返回另一个
        如果a,b都是叶子，返回merge_leaf(a,b)
        返回merge(a->l,b->l)与merge(a->r,b->r)连接成的树
```

- 由于a,b两棵树结构相同，上面的过程的正确性是显然的。
- a,b中可能存在key相同的元素，我们之前对线段树的定义对这种情况无能为力，所以需要一个merge_leaf过程来给出新树中该位置的元素
- 为了方便确定一棵树是否为空，动态开辟节点。若某棵子树为空，则其父亲的对应指针为空。

例子

- 维护区间内数字个数



复杂度？

- 若merge_leaf过程和+运算的代价都是O(1)
- 容易看出合并的开销正比于两棵树公共的节点数
- 单次merge操作的开销可大可小，但我们可以立即得到一个非常有用的结论：
- 若有n棵含有单个元素的树，经过n-1次merge操作，将他们合并成一棵的代价是O(nlogn)或O(nlogU)
- 理由：这个过程的开销不会比向一棵空树顺序插入n个整数来的大。

另一种风格的线段树

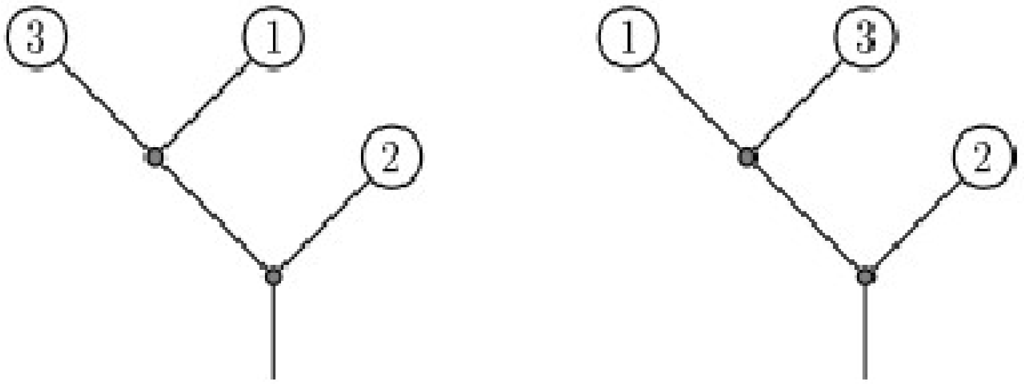
- 以合并操作为核心，我们可以写出另一种风格的线段树
- 关键操作：
  - merge_leaf过程
  - 连接操作
  - merge过程
  - make_leaf过程
- 操作都自顶向下，可以方便地持久化

线段树合并VS启发式合并

- OI中常常遇到一些题目，要将若干物件不断合并，顺便计算一些信息。
- 有些以树为背景的题目也需要完成类似的工作。
- 其中很大一部分需要维护一些元素的有序序列，通常用平衡树的启发式合并解决。复杂度$O(n\log^2 n)$
- 关键字通常是不太大的整数，如果换用这里的线段树合并，就可以降低复杂度的阶，做到$O(n\log n)$或$O(n\log U)$。
- 下面看几个例子

POI18 rot

- 给一棵2n-1个节点的二叉树，每个叶子上有一个1-n的数字，保证每个数字出现且仅出现一次。
- 现在允许任意次交换某两棵兄弟子树
- 对操作完毕的树进行dfs，可以得到一个先序遍历序，它是一个1-n的排列
- 求这个排列最小的逆序对数



POI18 rot

- 若T不是叶子，T的逆序对数=T->l的逆序对数+T->r的逆序对数+(x>y|xϵT->l,y ϵT->r)的对数
- 前两个与是否交换T->l和T->r无关，并且互相独立
- 计算每棵子树经过调整后最小的逆序对数。若子树已经计算完毕，只需知道交换两棵子树与不交换两种情况下新增的逆序对数，选取小的方案。
- 用平衡树维护子树内数字的有序序列，启发式合并时顺便算出需要的信息。
- O(nlog^2n)

POI18 rot

- 合并一些统计区间内数字个数的线段树来解决
- 在执行merge的过程中统计交换与不交换产生的逆序对数
- a是T->l对应线段树的某棵子树,b来自T->r的线段树
- ans0表示不交换时的新增逆序对数,ans1表示交换时的新增逆序对数

```
merge(a,b):
        如果a,b中有一个为空,就返回另一个
        ans0+=cnt(a->r)*cnt(b->l)
        ans1+=cnt(a->l)*cnt(b->r)
        返回merge(a->l,b->l)连接上merge(a->r,b->r)
```

- 时间复杂度O(nlogn)

POI18 rot

- 空间的开销可能是O(nlogn)的，不少这样写的人都MLE了
- 一种办法是借鉴后缀树，线段树可以看成只有两种字符的trie。把无用的边压缩，使得每个节点或者是叶子，或者有两个儿子，空间复杂度就是严格的O(n)。
- 不一定非得这样做。注意到合并完成后线段树所占的空间是O(n)的，MLE的原因是有一些树在递归运算的过程中被晾在一边，白白占用了内存。例子：右偏的一条链
- 每次先递归到较大的子树中去就可以了。

SPOJ COT3

- 给定一棵n个点的有根树，每个点是黑的或者白的。
- 两个人在树上博弈，轮流进行以下操作：
- 选择一个当前为白色的点u，把u到根路径上的所有点涂黑
- 不能操作者输
- 判断两人都用最优策略进行游戏时的胜负情况，并输出第一个人第一步所有可行的决策。

SPOJ COT3

- 出处：2010集训队 陈高远
- 数据加强
- 由公平组合游戏的性质不难想到以下的dp：
- dp[u]表示只考虑子树u的SG值
- g[u][v]表示只考虑子树u，v是u的某个白色后代(可能为u)，第一步选择了v，将u到v全部涂黑后的局面的SG值
- dp[u]=mex(g[u])，关键是求g[u]
- 当u是白色时，g[u][u]=sigma{dp[v]|v是u的儿子}
- g[u][w]=g[v][w]+sigma{dp[v]|v是u的儿子且v!=branch[w]}
- g[u][w]=g[v][w]+g[u][u]+dp[branch[w]]

SPOJ COT3

- O(n^2)
- 我们可以做的更好
- 注意到转移过程中，来自同一子树的g值都被xor上了同一个数，最后所有g值被放在一起进行mex
- 如果选用某种数据结构，能够快速地完成整体xor，再合并的操作，并且高效地支持mex运算，就可以改进复杂度。
- 二进制Trie
- 启发式合并，对最大子树的Trie打标记，其余dp值暴力插入
- mex(T)=size(T->l)==cnt(T->l)?size(T->l)+mex(T->r):mex(T->l)
- 复杂度O(nlog^2n)

SPOJ COT3

- 瓶颈是合并操作
- 二进制Trie和线段树非常类似，使用之前的过程高效合并
- 传递标记/询问mex/合并树 都是自顶向下的，不矛盾
- O(nlogn)

ONTAK2010 aut

- 给一棵n个点的树以及m条额外的双向边
- q次询问，统计满足以下条件的u到v的路径：
  - 恰经过一条额外的边
  - 不经过树上u到v的路径上的边

ONTAK2010 aut

- 等价于给定A,B两类树上的路径，问A中的每条路径被B中的多少条所覆盖
- A类路径的形状：^ / \
- 只考虑^形，剩下两种类似
- 对A类路径(u,v)，统计B中两个端点分别在子树u和子树v内的路径
- 求一个dfs序
- 询问B中两端点的dfs序号分别都落在指定区间内的路径数

ONTAK2010 aut

- 解法1
  - 将一个端点在dfs序中的编号<=i的B类路径的另一个端点的dfs序号都插入一棵线段树T[i]
  - 用可持久化线段树可以将T[1..n]都建出来
  - 对^形的询问，若两区间是[a,b]和[c,d]，答案就是在树T[d]中[a,b]内数字的个数减去T[c-1]中[a,b]内数字的个数

  - 在线算法
  - 时间复杂度O(mlogn)-O(logn)，空间复杂度O(mlogn)
- 解法2
  - 离线计算树T[i]中[l,r]内数字个数

ONTAK2010 aut

- 解法3
  - 离线回答^形询问
  - 同样把一个端点在子树u内的B类路径的另一个端点的dfs序号插入线段树
  - 回答(u,?)所需的线段树是u的所有儿子线段树的并，再插入一些序号。
  - 按照dfs序从后往前处理
  - 时间复杂度O((m+q)logn)，空间复杂度与读入同阶。

总结

- 某种程度上替代平衡树启发式合并，改进复杂度
- 一些常用容器的实现
  - Haskell的IntMap
- 在树上做预处理
  - 可持久化
- 其他用途?
  - 整点凸壳(COT6)
  - 二维或者更高维线段树合并(COT7 等)

谢谢大家！

- 感谢王子昱同学和罗雨屏同学的帮助
- ~~欢迎出题~~