优雅的数据结构
---树状数组

2016

SJTU ACM Class

lwher

什么是树状数组？

- 其原理是什么？
- 它能做什么？
- 时空复杂度？
- 好不好写？
- ……

- 太麻烦了！
- 我们先来直接讲个例子，感受一下它的神奇吧！

问题

- 给一个长度为n的数组，每次可以进行两种操作：
- (1)单点增加一个数值
- (2)查询给定区间的和

- n <= 10^5 操作数<=10^5

- 怎么做？

- 每次直接修改，然后查询的时候for一下？
- TLE!

- 利用前缀和每次快速查询，然后修改的时候重构？
- TLE!

- 啥？你说你会线段树？还会分块？
- ...Naive!

- 让我们来看看树状数组的代码是怎么写的吧。

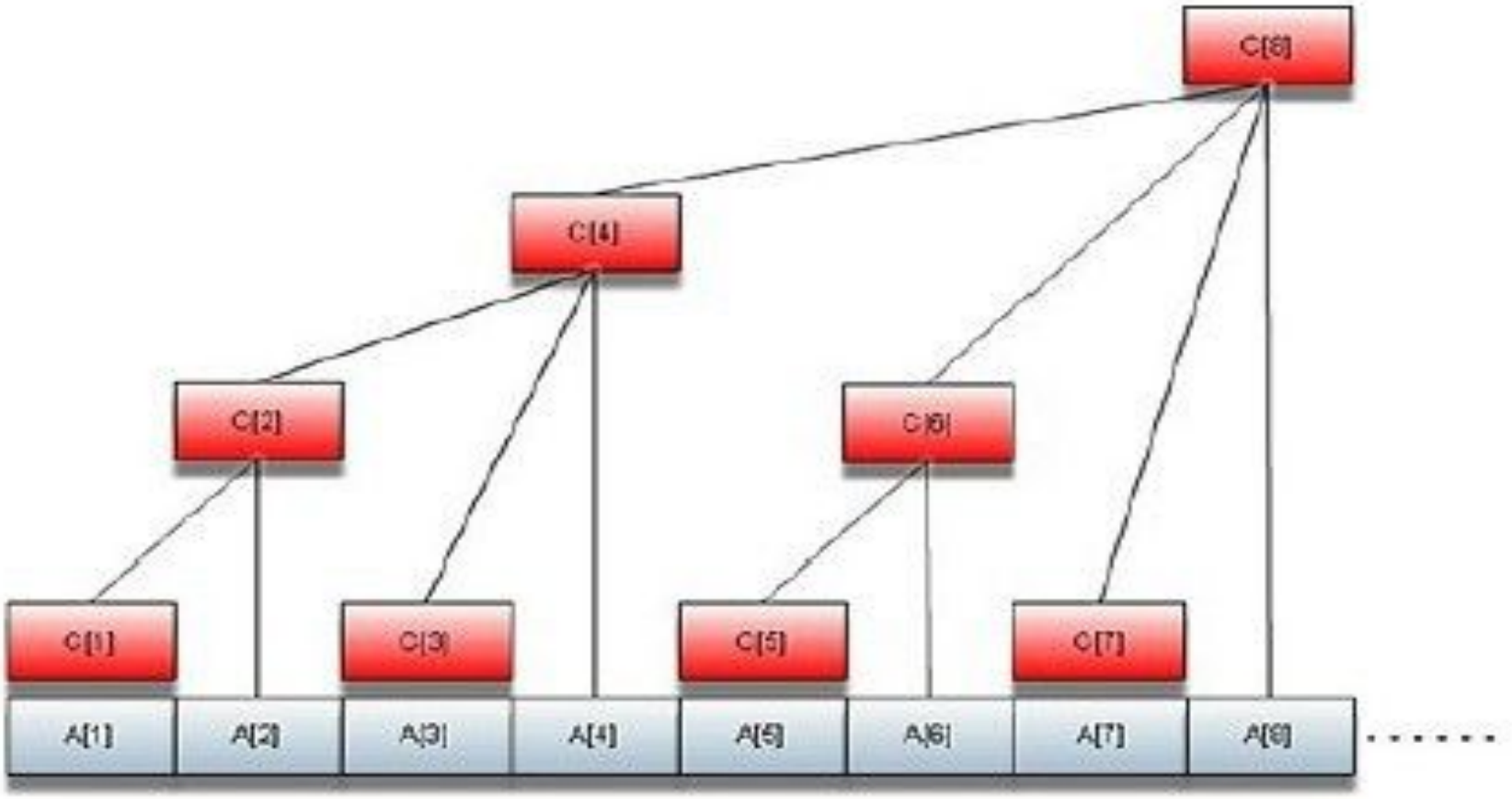
```
void Add(int x, int p){ //在x处增加p的值
    for(int i = x; i <= n; i+= (i & -i)) T[i] += p;
}
int Query(int x){//查询1~x的区间和
    int res = 0;
    for(int i = x; i; i -= (i & -i)) res += T[i];
    return res;
}
```

- 主函数
- 一开始我们认为数组全是0， 然后如果数组有初始值，就直接看成单点增加就OK了
- 询问？假设问[l, r]和，那么答案就是Q(r) - Q(l - 1)
- 没了.....So 优雅！

- 很神奇吧？
- 那这么优雅的数据结构，它的原理是什么呢？
- 其实这个相对于它的代码来说，还是复杂了一点，我们来慢慢看……

- 首先解决一个问题:
- i & -i 是啥？(后面也写作"lowbit")
- -i 是在 i 的基础上把二进制的符号位变化，并且其他位取反然后+1，&是位运算，a & b 的算法就是把a,b的二进制形式右对齐，然后如果全是1，这位还是1，否则为0
- 举个例子 10100 的 lowbit 就是 100
- i 和 -i 实际上除了i的最后一个1那位，其他位置都不一样，所以我们得到了i的二进制下最右边的那个1

- 接下来，我们来看看树状数组直观上看是长啥样的。



- 二进制十分的美妙！利用lowbit，我们发现从1~n刚好串成了一个树形结构。
- 树状数组是如何实现快速求1~n位的和的呢？
- 其实通俗的来讲，树状数组的每一位C[i]，除了存储num[i]之外，还存了其与一些其他的数字加起来的和，我们可以认为每个C[i]有一个自己负责的"区域"(自己为根的子树)，那么我们在查询的时候，利用lowbit往下走，1~n这个区间又被刚好分成了log个毫无交集的“区域”，并且他们的并集就是1~n，所以相加就好了（每次-=lowbit，那么就是每次消除二进制下最后一个1，最多log个1，所以是log次）

- 修改操作：
- 其实原理上差不多，每次+=lowbit，刚好实现了在树中一层一层往上走的情况。因为每个C[i]统计的是以i为根的子树的和，所以如果修改i的数字，那么i的祖先也都要一一修改。

- (如果觉得上面的解释不好懂，那么我们换一种说法，树状数组可以理解成每个C[i]有一个自己负责的“区域”，然后关系又是一级一级的，每个C[i]最多只会有一个“Boss”，每个C[i]负责自己的下属与下属的下属的和，那么修改的时候就要“层层汇报”，表示一下自己被改了，上级全部修改)

- 从理论上来解释一发
- 存储原理:
- 对于每个C[i]负责的区域，其实是这个i二进制表示下，去掉去lowbit再+1 一直到 i 这个区间。
- 比如： 1101000负责的是1100001 ~ 1101000这个部分
- 所以我们修改的时候，每次加lowbit，相当于把最后的1一直左移，遇到前面也是1的就也消去它再往前走。
- 查询也是一样的，每次查询 [x-lowbit(x)+1,x]然后去掉
- 比如1010 -> 先查了1001 ~ 1010 ，然后x变成1000，
- 接着加上 0001 ~ 1000 这刚好是前缀和。
- 时间复杂度 O(nlogn) 空间复杂度 O(n)

- 树状数组除了刚才的例题还能干嘛？

- 其实树状数组比较容易的用法是单点修改然后区间查询，由于其查询的是前缀区间的和(1~n)，所以要支持区间查询需要满足区间加减性。(比如区间最值，就不满足区间加减性，因此不能用“简单”的树状数组来实现)

- 另一种姿势：权值维护
- 啥意思呢？我们考虑上面那个例题，我们维护的是一个序列，num[i]存着第i个数
- 我们还有另外的姿势，那就是num[i]不存第i个数的位置，而是存权值为i的数的个数，也就是如果i多了一个，那么num[i]++
- 我们用树状数组(Bit)来维护这个序列，那么可以做到每次删减一个数字，询问某个区间里面的数字有多少个。

- 干嘛用的呢？
- 有个问题叫做求逆序对个数(n <= 10^5)
- 也就是对于数列A，求Ai>Aj,i<j 这种数对的对数。
- 我们假设数值范围是1~n
- 我们倒过来扫，然后维护一个权值树状数组，每次对于A[i]，我们先查询一下后面有多少个权值比A[i]小的加进答案，然后把A[i]这个数再扔进去，问题就圆满解决了。O(nlogn)

- 上次机考的第二题：k元逆序对个数
- 由于k很小，不难想到DP
- f[i][j]表示以i结尾的长度为j的逆序对个数是多少
- 然后f[i][j] = sigma(f[k][j-1]) k<i,A[k]>A[i]

- 然后状态数是10^5 * 10，可以接受，但是简单地转移是要O(n)的
- 这时候我们也是利用树状数组，num[i]存A[k] = i 的f[k][j-1]的总和，树状数组维护，然后每次转移的时候去查询一下A[i]-1的前缀和，然后把f[i][j-1]扔到num[A[i]]去就可以了。

- 姿势还是有挺多的。
- 再介绍最后一个，区间修改(听不懂无所谓)
- 首先，我们还是原本的例题，只不过变成可以在区间[l,r]上加上一个数，然后区间查询
- 一个一个改？
- 那不是比暴力还慢？

- 我们引入一个概念：差分
- 什么意思呢？就是根据A数组，我们搞出一个D数组，其中A[i] = sigma(D[1~i])
- 也就是D[1] = A[1] D[2] = A[2] - A[1] D[3] = A[3] - A[2]...以此类推
- 那么我们如果对区间[l,r]进行加值，就可以变成在在D[l] + x然后在D[r+1] - x，这样子的话，就完成了修改。

- 区间查询呢？

- 我们考虑[1, I]区间和的表示形式：
- 其中 A[I] = D[1] + D[2] + ... + D[I]
- A[I-1] = D[1] + D[2] +...+ D[I-1]
- .....
- A[1] = D[1]

- 我们惊(jian)奇(dan)地发现，如果求1~I的和，那么D[1]被算了I次，D[2]是I-1次，D[I]是1次

- 那么，要求[1, l]的和

- 我们只要维护两棵BIT即可

- 第一科维护D[i]
- 第二棵维护D[i] * i

- 查询的时候只要用
- (l+1)* sigma(D[1~l]) - sigma(D[k]*k)(k:1~l)就OK了，为啥是这样大家可以想象一个(l+1)*l的网格图然后自行脑补。

- 这样问题就圆满解决了！
- 其实BIT还有一些姿势，不过由于用(wo)不(tai)多(lan)了所以就不拿出来具体讲了，大家有兴趣可以上网找找看。
- 比如:二维BIT，BIT里面再放一个其他的树(主席树就是一个例子)，上面那种差分的方法还可以推广到二维的情况，我记得要维护4棵貌似，其实BIT可以做区间最值，只不过复杂度会退化成log^2。
- 以上如有不对，还请大家指正。

- 大概就介绍这么多，讲起来容易做起来也许就不是那么简单了，希望以前没玩过的同学可以找几个题目写一写，锻炼一下(虽然BIT插入加查询就那么5行甚至不到的代码.....)
- BIT的题目超多！
- 如果找不到，不妨搜一下什么POJ 树状数组，BZOJ 树状数组
- 似乎有个叫CodeVS的OJ上面有很多裸题？反正我没用过.....大家可以自己去看看。

谢谢大家！

欢迎提问！