线段树部分总结

$\mathcal{P}yh$

2017 年 6 月 20 日

目录

1 按位置建线段树的基础应用

有一些典型的数据结构题，即给出几种操作，动态询问区间的某些信息。下面我总结了几类按位置建线段树能够解决的难题。

1.1 一类难点在区间合并与维护信息的线段树问题

有一类问题，初看到问题会无从下手，往往修改操作十分简单，但是询问的内容比较复杂，甚至某些问题完全没有修改操作，只有区间的动态询问。

下面我总结了一些解决这类问题的方法。

1.1.1 从简单情况开始分析

这是一种通用的方法，我们可以首先从简单情况分析起，最后类比推理到复杂的情况。

例题 1. *k-Maximum Subsequence Sum*[1]

给出一个长度为 n 的序列 A, m 次操作, 操作有如下两种:

1. 给出 i,val, 把 A_i 变成 val。

2. 给出 l,r,k, 询问把区间 $[l,r]$ 划分成不超过 k 个不相交的区间，这些区间中数的和的最大值。

数据范围: $n,m \leq 10^5, k \leq 20$

这里介绍一种 $O(k^2(n+m)\log n)$ 的算法（下面将会介绍更优的算法）。

因为是单点修改，我们考虑怎么将两个子区间的信息合并到当前区间。

$k=1$ 的情况就是区间最大子段和，是一个经典问题，不再赘述。

考虑 $k=2$ 的情况。

我们发现选的两个子区间要么都不跨左右子区间的边界，要么一个跨了另一个没跨。

这就相当于在一个子区间中是最大后缀（前缀）和，在另一个子区间中是一端贴着边界，总共包含两个子区间的最大和。

这个相当好维护。那么能不能推广呢？

首先我们在每个子区间维护一个数组 $maxx[1..k]$，$maxx[i]$ 记录该区间划分成 i 个子区间的最大和。

显然首先当前区间划分成的 k 段区间可以完全在左子区间和右子区间，直接统计答案即可。

然后假设选的所有区间不跨两个子区间的边界，我们可以枚举左子区间放了多少个区间，直接统计答案。

跨边界的情况怎么办呢？我们注意到跨边界的那个区间可以拆分成两个部分，一个在左子区间，一个在右子区间。

于是我们对于一个区间，维护一个信息，表示当前区间划分了 k 个子区间，其中最右边的子区间是贴着右边界的最大和。

同理，维护一个最左边的子区间是贴着左边界的最大和（这些都是数组，因为一个子区间可能是放 $1..k$ 中的任意多个区间）。

为了维护上面的这两个“贴着边界最大和”的信息，我们需要再次考虑跨过中点的情况：这次轻车熟路了，我们只需要维护两端都贴着边界的最大和即可。

而维护“两端都贴着边界最大和”的信息，我们可以直接通过左右子区间维护的上述信息维护出来。

[1] Codeforces 280D

因为计算维护的信息数组的每一项需要枚举左右区间各自放了多少个区间，所以区间合并的复杂度是 $O(k^2)$ 的，整体的复杂度为 $O(k^2(n+m)\log n)$。

这是一个很巧妙的算法，但是因为 k^2 比较大而无法通过本题。下文我将会讲述这个题目的正确解法，也是使用了线段树，不过是作为辅助数据结构来使用的。

1.1.2 分析题目性质

解决这类问题的另一种通用方法是分析题目中询问的信息的性质，来达到便捷地维护区间信息的目的。

例题 2. 楼房重建[2]

在二维平面内，给出 m 个操作。

每个操作给出 x,y，把横坐标为 x 处的线段删除，并新建一条从 $(x,0)$ 到 (x,y) 的线段，

每个操作完成后询问在 $(0,0)$ 能够看到多少条线段的顶部（线段可以互相遮挡）。

数据范围：$m \le 10^5$，横坐标 $\le 10^5$

显然问题可以转化成满足下面这个条件的线段的个数：前面的所有线段的 y/x 均严格小于它的 y/x。

所以我们不妨把所有线段的高度变成 y/x。

因为是单点修改，我们同样思考怎样合并两个子区间的信息。

假设当前考虑的区间的前面所有线段的高度最大值为 d，

若左儿子的高度最大值 $< d$，则左儿子的所有线段完全被挡住，左儿子的贡献为 0，但是右儿子的贡献可能不为 0，递归右儿子；

若左儿子的高度最大值 $\ge d$，则左儿子的这个最大的线段完全挡住了左边的区间，在左边区间的修改完全不会影响到右儿子的线段，直接用右儿子在修改前对当前区间的贡献，而左儿子的线段可能会被影响，递归左儿子。

于是，在每次区间合并的过程中，我们要么递归左儿子要么递归右儿子，一次区间合并的复杂度为 $O(\log n)$，总的复杂度为 $O(n\log^2 n)$，可以通过本题。

这个问题的解决我们巧妙地应用了线段遮挡问题的性质，即修改一条线段对后面线段的影响是很少的，所以才能设计出数据结构解决本题。

例题 3. 捉迷藏[3]

给出一棵 n 个点的树，每个点有黑或白一种颜色。

有 m 个操作，每个操作改变一个点的颜色，询问最远的黑点对的距离。

数据范围：$n \le 10^5, m \le 5\times 10^5$

这个题目又是单点修改，我们可以考虑用线段树来解决它。

首先这是一个树上的问题，我们首先用 dfs 序把它转化成区间问题。

感觉还是不好做，因为树上的距离不好直接在线段上表示出来。

这个时候，我们可以考虑树的一个关于距离的性质：在树的括号序列中，一个点到另一个点的距离恰好等于两点之间删去匹配括号之后剩下的括号的数量。

[2] bzoj2957

[3] ZJOI2007

于是对于每个区间，我们维护：

- 删去匹配括号之后的左括号、右括号的数量；
- 所有黑点到左端点、右端点左、右括号的数量的最大值；
- 所有黑点到左端点、右端点左括号数量-右括号数量、右括号数量-左括号数量的最大值。
- 该区间中最远黑远点对的距离。

怎么从子区间合并上述信息并不难，在此不再赘述。
这个算法的时间复杂度为 $O(n\log n)$，可以完美地解决本题。

1.2 一类难点在处理标记的线段树问题

除了上面这种维护的信息比较复杂的问题外，还有一类修改操作比较特别，需要使用懒标记来解决的问题。

1.2.1 从全局出发，一种“操作若干次就不再有效”的问题

这是一类通用的问题，其特点在于操作一旦超过若干次之后作用就可以直接处理。
其一般解决方法就是尝试进行全局操作。

例题 4. *Rikka with Phi*[4]
给出一个长度为 n 的序列 A，有 m 个操作。
1. 给出 l,r，将所有的 $A[i],l\le i\le r$ 全部变为 $\varphi(A[i])$；
2. 给出 l,r,x，将所有的 $A[i],l\le i\le r$ 全部变为 x；
3. 给出 l,r，询问 $\sum_{i=l}^{r}A[i]$
数据范围：$n,m\le 3\times 10^5$

初看觉得十分不可做，但是我们尝试把所有操作看成全局操作：
覆盖操作会导致所有数字变成一样；
取欧拉函数操作最多 $\log(x_i)$ 次就会使所有数变成 1。
于是我们维护当前区间的最大值和最小值，如果最大值等于最小值那么直接打上覆盖标记；否则递归处理下面的区间。
我们来分析复杂度。单看欧拉函数操作，每个数最多 $\log(x_i)$ 次就会变成 1，复杂度显然是 $O(n\log n)$ 的。
如果加上覆盖操作，根据我们的算法，我们可以把相同的一段区间看作一个数字。
那么每一次覆盖操作最多增加一个数字，均摊下来不会降低取欧拉函数操作的复杂度。
所以总的时间复杂度是 $O(n\log n)$。

例题 5. 基础数据结构练习题[5]
给出一个长度为 n 的序列 A，有 m 个操作。
1. 给出 l,r,x，将所有的 $A[i],l\le i\le r$ 变成 $A[i]+x$。
2. 给出 l,r，将所有的 $A[i],l\le i\le r$ 变成 $\lfloor\sqrt{A[i]}\rfloor$
3. 给出 l,r，询问 $\sum_{i=l}^{r}A[i]$

[4] hdu5634
[5] uoj228

我们再次把操作看成全局操作。

我们发现两个数如果开根号的话差会变成 $\sqrt{a}-\sqrt{b}=(a-b)/(\sqrt{a}+\sqrt{b})$。

可以发现最多 $O(\lg\lg n)$ 次就可以变成 0

这样不停地开根号两个数的差就会很快减小到 0。

但是我们发现如果两个数的差为 1，则开根号之后可能差还为 1，所以需要特判差为 1 的情况（发现正好就是区间减法）。

而区间加法只不过是把区间两端的差给变化了。

综上所述，维护区间的最大值和最小值，如果最大值与最小值的差小于等于 1，则直接打上加法标记，否则递归处理子区间。

这种类似的操作还有很多，例如区间取模[6]；区间除法[7]；区间每个数 a_i 都变成 c^{a_i}，c 为常数[8]等等，都可以利用这个思想来解决这些问题，有些还需要特殊的技巧。

1.2.2 标记永久化

标记永久化是一种特殊的思想，其主要原理是把标记打到节点上，然后询问的时候自上而下的经过每个节点的同时更新答案。

例题 6. *Segment*[9]

要求在平面直角坐标系下维护两个操作:

1. 在平面上加入一条线段。记第 i 条被插入的线段的标号为 i。

2. 给定一个数 k, 询问与直线 $x=k$ 相交的线段中, 交点最靠上的线段的编号。

数据范围: 操作数 $\leq 10^5$, 直线的左右端点的横坐标 ≤ 39989, 纵坐标 $\leq 10^9$。

强制在线。

因为是单点查询，所以我们只需要考虑如何插入一条线段并维护信息。

首先找到这个区间所代表的 $O(\log n)$ 个节点。

对于一个节点，如果上面没有线段，直接把线段信息记录在节点上；

如果上面有线段，则比较两根线段。

如果在这个区间中，当前插入的线段完全优于原线段，则覆盖掉节点信息；

如果当前插入的线段完全劣于原线段，则直接退出；

否则比较两根线段的交点，把影响区域大的线段信息记录在节点上，另一根线段的信息递归下放。

这样每经过一个结点需要 $O(\log n)$ 次下放，总的复杂度是 $O(n\log^2 n)$。

每次询问的时候，从上而下进行访问，经过的所有节点的信息就是覆盖在当前节点上的线段的信息。

所以我们可以在 $O(n\log^2 n)$ 的时间内解决这道题。

1.2.3 区间最值操作与历史最值询问

这是一种玄妙的姿势，就是通过记录一些奇怪的信息来支持区间取 min、取 max、询问历史最值的操作。

具体的实现方法在吉如一的 2016 年国家集训队论文里面有详细论述。

[6]某一次省选模拟考试

[7]某一次省选模拟考试

[8]Shoi2017 相逢是问候

[9]Heoi2013 day1

2 权值线段树的基础应用（充当普通平衡树）

权值线段树即按权值建树，适用于一些不要求询问区间的题目（充当普通平衡树的作用），或者是通过树套树或线段树合并乃至可持久化来支持询问一些特殊的东西。

充当普通平衡树，其实就是把元素的权值当作下标插入到线段树中，然后利用线段树来维护这些元素的一些性质。

因为这个很水所以我就不举例子了。

3 可持久化线段树的基础应用

如果两棵线段树之间有许多相同的节点可以共用，那么我们可以考虑用可持久化线段树让它们共用节点。

可持久化线段树最基本的应用就是通过区间的差分来维护给定区间的一些信息，当然也可以用树套树……。

3.1 一类维护序列中某个区间信息的问题

例题 7. 网络管理[10]

给出一棵 n 个点的树，每个点有一个值 w_i。有 m 个操作。

1. 给出 a,b，将 w_a 变成 b。

2. 给出 a,b,k，询问从 a 到 b 的路径中第 k 大的值。

数据范围：$n,m \leq 80000, w_i \leq 10^8$。

这是一个经典问题。

我们对于每个节点开一棵权值线段树，维护从根节点到自己的所有 w。

因为每个节点相对于父亲节点只是插入了一个 w，所以可以用可持久化线段树，每次插入复杂度为 $O(\log n)$。

那么询问的话就是两个端点的权值线段树减去 LCA 的权值线段树（注意细节即可）。

修改一个点的值只会影响它的子树。

于是我们按照 dfs 序建树状数组，树状数组的每个节点维护一棵权值线段树。

修改点值就相当于在它的子树这一段连续的 dfs 序区间中每个点修改值，用树状数组的区间修改、单点查询的方法就能够在 $O(n \log n^2)$ 的时间内解决本题。

~~这就是伟大的树上带修改第 k 大。~~

这类问题满足的性质就是维护的信息满足可加减性，然后就可以用可持久化线段树来动态维护信息了。

例题 8. *k seq*[11]

给出一个长度为 n 的序列 A 和 k。

定义一个子串的 w 为子串中，相同的数字只计算一次，所有的数的和。

求第 k 大的 w。

数据范围：$n,k \leq 10^5$

[10] CTSC2008

[11] hihocoder 1046

因为一个子串总能被表示成一个后缀的前缀，所以我们对于每个位置 i 维护一个普通线段树 $S[i]$，里面下标为 j 的元素是 $w[i..j]$ 的值。同时维护一个最大值。

首先我们先暴力预处理出权值线段树 $S[1]$。

然后往后枚举每一个位置，对于每一个位置 i，记上一个位置的那个元素下一次出现的位置为 pos。

则对于 $j \in [i, pos-1]$，$S[i]$ 中下标为 j 的元素的值就等于 $S[i-1]$ 中下标为 j 的元素的值减去 $val[i-1]$，其他下标的元素的值不变。

使用可持久化线段树，使用懒标记，于是每个位置只需要花费 $O(\log n)$ 的时间就可以构造出线段树。

我们把每个位置的线段树的最大值放到一个堆里面。

每次找出最大的那个位置，然后将最大的那个后缀的最大的那个前缀的位置上的值改成无穷小，求出最大值之后再次插入堆中。

这个操作重复 k 次即找出第 k 大值。

复杂度为 $O((n+k)\log n)$。

3.2 一类关于子序列的问题

4 动态开点线段树的基础应用

5 线段树合并的基础应用

5.1 一类考虑子树对父亲的贡献的问题

5.2 一类有关图的最短路的问题

6 线段树套线段树的基础应用

7 线段树作为辅助数据结构的一些问题

7.1 用线段树优化 dp

7.2 用线段树判断完美匹配

7.3 用线段树模拟费用流

8 线段树问题的一些小技巧

8.1 差分

8.2 下标有关