数据结构第五讲

可持久化数据结构和平衡树介绍

丁尧尧

上海交通大学

February 9, 2018

目录

我们所说的可持久数据结构，指的是我们在更新数据结构时，保留其旧的版本，使得我们可以在后面的某个时刻访问前面某个时刻数据结构的版本。因为每次对数据结构修改都修改了整个数据结构的一个小部分，我们可以通过共用其它部分数据的方法来实现同时拥有多个版本。

一般来说，线段树、非均摊复杂度的平衡树是可以可持久化的。

今天我们主要讲线段树的可持久化。

先来看一个问题:

区间第 K 小问题

给出一个序列，要求求出给定区间的第 K 小的数（将区间中的数从小到大排完序后，K 个数字)。

离散化

离散化本质是建立一个双射，将一个集合（可以是浮点数、整数、范围特别大的正整数）中的数映射成 $1, 2, 3, \ldots, n$，其中 n 是集合的大小。这样处理之后我们就缩小了值域范围。比如我们有 10^5 个 $[1, 10^{18}]$ 范围内的数，我们想统计每个数出现的次数，不用数据结构直接处理是有些困难的，但如果我们先离散化，就可以建一个数组 `int cnt[100001];` 来统计了。（当然你也可以直接排序搞搞）

一般来说，如果我们关心的是数的大小关系，而对了具体的值依赖不是特别强时，都可以尝试离散化。

代码很简单

我们先来看看刚才那个计数问题怎么用离散化解决的:

```
int n, a[N];
int disc[N], dtot;
int cnt[N];
int main() {
    for( int i = 1; i <= n; i++ )
        disc[++dtot] = a[i];
    sort( disc+1, disc+1+dtot );
    dtot = unique( disc+1, disc + 1 + dtot )
         - disc - 1;
    for( int i = 1; i <= n; i++ )
        a[i]=lower_bound(disc+1, disc+1+dtot, a[i])-disc;
    for( int i = 1; i <= n; i++ ) cnt[a[i]]++;
    for( int i = 1; i <= dtot; i++ )
        printf( "%d appears %d times\n", disc[i], cnt[i] );
}
```

说明

假如原集合是 A，它被离散化成了 B，如果我们想知道 $a(a \in A)$ 对应的 $b(b \in B)$，我们需要二分找出它在排序去重后的数组中的下标，反过来，只需要访问数组那个位置的值就行了。你可以像上面代码那样，离散化完成后就将原数组整个映射一次，后面如果想要知道原来的数，只需要访问 `disc` 对应的值就行了。

值域线段树

值域线段树不是一种新的数据结构，而是线段树的一种用法：
我们将线段树的每个不同位置当成某个整数变量的取值，这样线段树其实可以充当一个可重集。
如果我们在每个节点处维护一下该区间上数的个数，我们就可以在线段树上通过一种类似二分的方式找到整棵线段树的第 K 大的数。
值域线段树是支持“减法的”。
如果我们将数组中的数从左到右一个一个加到值域线段树中，并且保留下各个时刻的值域线段树版本，那么我们在“做完减法的值域线段树上”二分就可以求出区间第 K 小。

关于线段树可持久化的实现

我们在进行一次修改时，其实最多改变 $O(log(n))$ 个节点，那么此时，如果我们新建 $O(log(n))$ 个节点，原节点保持不变，再将新建的节点弄成改变后的样子，这样就完成了可持久化，并且根节点一定是会被修改的，这样我们就可以得到很多个根，我们通过这些根来访问不同时刻下的线段树。

如果我们是将线段树作为值域线段树，或者线段树一段区间的值不用建子节点就可以直接算出来，那么我们还可以**动态开节点**。

可修改的区间第 K 小

如果我们要修改怎么办？

记得我们讲树状数组时说过，树状树组可以套在其他数据结构外面。

树状数组可以解决单点修改，求前缀和，这里，如果我们把“和”推广到“集合”就可以解决我们的问题了。

我们查询时就不再是两个线段树做减法了，而是 $log(n)$ 个线段树的加减了。

更全面深入的内容可以去看陈立杰的《可持久化数据结构研究》。