南京市金陵中学 蒋炎岩

线段树

在区间统计方面，有着很大的优势：区间上的最大、最小值以及它们的统计、推广

平衡树

和线段树相同，在统计时有着时间上的优势，而且，与线段树最大的不同在于，平衡树不依赖于处理数据范围的大小。随着 **Treap** 等数据结构的发展，逐步在信息学奥赛中代替了难于实现的 **AVL** 和红黑树，发挥着越来越大的作用

但是

对于大数据量的修改维护、删除操作，这两种数据结构就显得力不从心了在线段树和平衡树中，删除元、插入元素的复杂度均为 **O(logn)** 。在大量的维护面前，它们只能力不从心！

我们看国家集训队队员龙凡演讲中提到的问题

士兵排队

他用转化的思想，巧妙地解决了问题。不过我要提出的，并不是转化问题的模型，而是直接设计一种数据结构进行维护操作，解决问题。

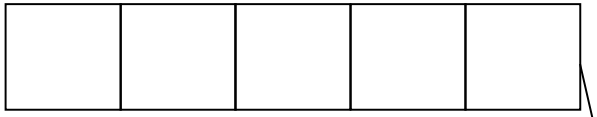
要快速地移动大量连续的元素，最快速的实现方法是链表。

要快速地定位元素，最快速的实现方法是数组。

为了二者效率的协调、统一，必须把它们转化，然

我们可以直接把它们结合起来！

每一个整体元素是一个数组



从理想的角度考虑，
一个表有 **n** 个元素，
那么，理想的复杂度，
无论是插入、删除、移动
或是其他数据操作
我们的时间复杂度

$$\sqrt{n}$$

都是

数组和数组的关系是链接

在块状链表中间插入元素

在每一个块中，我们都维护一个当前表中存储数据的长度。因为数据可能插入在某一个块的中间，所以我们必须对块内的元素进行移动。当然，这可能会造成某一处的元素增加，超过 **Sqrt(n)** 。所以要在表中设置一个缓冲区，当表中元素到达 **2*Sqrt(n)** 的时候，我们把这个表断裂成两个等长的表。

在块状链表中删除元素

与插入算法相同，某一个元素删除以后，表的长度都会减小。当相邻的两个表的长度总和小于 **Sqrt(n)** 时，我们就将这两个表合并。

块状链表的扩张

同样的，我们可以在块状链表中加上某一个块的最大、最小值；块中的元素等，同样的，我们也可以将块状链表与其他数据结构联合。

回到问题上，块状链表正好适应了问题的要求！
我们将带位置的数据插入到块状链表当中，
然后，假如需要移动的化，我们除了需要将某一块链中的 **Sqrt(n)** 个元素移动掉，其他的链表操作，只需要 **sqrt(n)** 的时间． **Sqrt(n)** 完美地将复杂度平均，无论是数组，还是链表，都逃不出这个时间复杂度上限。问题至此被解决。

而且，这个算法的常数非常小，完全没有线段树等复杂的操作和分析，全部都是最简单、最基本的数据操作！

我们看国家集训队队员黄刚演讲中提到的问题

Dynamic Ranking

黄刚同学用数据结构的联合解决了这个问题。不可避免的是，线段树和平衡树的“利益”不很统一最终必须还得借助二分的帮忙。同时维护两个数据结构的代价不小，加上常数，我们不妨换一个角度考虑——

块状链表

同样的，维护一个块状链表，记录所有的元素。当然，仅仅这个还是不够的：将所有的数都导入块状链表，而且，对于链表里的每一个块，都维护一个排序后的表。这个表的 **Replace Element** 仍然可以在 **Sqrt(n)** 时间内解决。然后，如黄刚同学所说，二分那个所求的最大值，然后可以轻松地扫描块，并统计出那个数是否为第 **k** 大，问题解决，时间复杂度为

$$(\mathrm{A}*\sqrt{\mathrm{N}}\mathrm{log}\mathit{MaxLong}\log N+B*\sqrt{\mathrm{N}})+\mathrm{N}\mathrm{log}\mathit{N}$$

$$=\mathrm{N}\sqrt{\mathrm{N}}\mathrm{log}\mathit{MaxLong}\log N$$

看起来，这似乎比黄刚同学的算法要慢一个数量级。可是，维护平衡树和线段树的代价是很大的！

AC就是硬道理

块状链表的出色表现说明，有时候，某个问题，并不是渐进意义复杂度小的办法，就是最好的！

Thank You!