有序数组还原

薛振梁　18307130172
2019 年 11 月 17 日

概况

基于段切割的排序算法。设序列长度为 n，交换次数为 k，算法查询次数上界为：

$$n + 10k + (4k + 1)\log(2k + 2) = \mathrm{O}(n + k \log k)$$

想法

由于数据只交换了 k 次，所以最多有 $2k$ 个元素是错位的（我们称为 touched，其余的元素称为 untouched）。这 $2k$ 个 touched 元素会将原序列切分为最多 $4k + 1$ 个连续段。连续段（consecution）是一个序列，其在原序列上是**连续排布**的，并且**值域是连续**的，且**内部已经有序**。例如，在下面的序列中：

```
1 2 7 4 5 6 3 8 9
```

其中 “`1 2`”、“`7`”、“`4 5 6`”、“`3`” 以及 “`8 9`” 是连续段。不难发现，如果按照每个连续段的第一个元素排序，就可得到原序列的一个排序。

所以算法的主要目标在于识别出所有的连续段并将它们排序。直接识别连续段是很难的，但是我们可以通过一个 $\mathrm{O}(n)$ 的扫描得到一些类似于连续段的东西：单增段。这里简单点就称为 segment。第一遍扫描时，我们贪心地将原序列划分为一些 segment。例如，上面的序列就会被划分为：

```
1 2 7 | 4 5 6 | 3 8 9
```

不难发现这样的划分使得每个 segment 都是由一个或多个 consecution 组成的。接下来的目标就是通过适当的切割，逐渐还原出所有的 consecution。考虑 segment 在值域序列上的样子：

```
value: 1 2 3 4 5 6 7 8 9
1 2 7: * *         *
4 5 6:       * * *
3 8 9:     *         * *
```

每个 segment 都是由几个值域区间（实际上就是连续段）组成的。如果按照每个 segment 的第一

个元素排序，对于相邻两个 segment，后一个 segment 的开头元素就可能是前一个 segment 的切割点，因此我们可以用二分的方法找出切割位置。被切割的 segment 会变为两个 segment，其中第一个 segment 位置不变，而第二个 segment 需要重新参与排序，这里使用插入排序即可。

如果一个 consecution 内的元素都是 untouched 的，则称其为 untouched consecution。不难发现，每个 segment 内最多包含一个 untouched consecution，否则这个 segment 内部一定不是单调的。这意味着大部分 segment 只有首尾有一些 touched elements，而这些元素的个数最多只有 $2k$ 个，因此实际上完全可以暴力切割。

具体算法

算法主体大致分为 3 步：

1. 通过 $n-1$ 次比较，每次比较原序列中相邻两个数的大小关系，贪心地划分出 segment。
2. 将划分出的 segment 按首元素大小插入排序。
3. 从前往后依次扫描相邻的 segment。如果可以切割，则一定能切出一个最靠前的 consecution，以及另外一个需要重新插入排序的 segment。不断重复这个过程直到所有 segment 均被确认为不可切割的。

对于切割过程，我们暴力地同时从两端扫描，如果某一段扫到了一个切割点则直接切开。如果这个切割点是被右端的扫描扫到的，说明被切下来的左边一定是 untouched consecution，那么可以给切出的 segment 打个标记，表示以后如果继续切割这个 segment，就不需要从左边扫描了。

最后依次输出每个 consecution 的内容即可。

分析

对于第一步，首先需要 $n-1$ 次比较。对于贪心划分的结果，如果有一个 segment 中没有 untouched consecution，则 (1) 这个 segment 中仅有一个 touched element，那么这个元素前后至少有另外一个 touched element 或者是在序列的开头或结尾，这种情况下 untouched consecution 数量上界会减 1；(2) segment 中有至少两个 touched elements，由于它们是相邻的，所以 untouched consecution 的数量上界至少减 1。否则划分出的 segment 中必有一个 untouched consecution。设 $\#\text{untouched}$ 为包含一个 untouched consecution 的 segment 的数量，$\#\text{touched}$ 为不包含 untouched consecution 的 segment 的个数，则根据以上讨论有 $\#\text{untouched} \leqslant 2k+1-\#\text{touched}$，故贪心算法划分出的 segment 的数量至多为 $2k+1$。接下来我们设 $c=\#\text{touched}+\#\text{untouched}$ 表示贪心划分出的段数，用 d 表示总共的 consecution 的数量。由之前的讨论不难得出 $d-c \leqslant \#\text{touched} \leqslant 2k$。

对于第二步，插入排序的比较次数上界为：

$$\begin{aligned}\sum_{i=1}^{c}\lceil\log i\rceil &\leqslant c+\sum_{i=1}^{c}\lfloor\log i\rfloor-\lfloor\log c\rfloor\\ &\leqslant c+\int_1^{c+1}\log x\,\mathrm{d}x-\lfloor\log c\rfloor\\ &\leqslant (c+1)\log(c+1)-\left(\frac{1}{\ln 2}-1\right)c-\lfloor\log c\rfloor\\ &\leqslant c\log(c+1)-\alpha c+\varepsilon_0\end{aligned}$$

其中 $\alpha=1/\ln 2-1\approx 0.4427...$ 以及 $\varepsilon_0=\log(c+1)-\lfloor\log c\rfloor\leqslant 1$（由于 c 是整数，所以这里的

上界不是 2）。

对于第三步，每一次切割都可以确定一个最小的 consecution（因此不用参与后续的排序），以及一个需要重新插入的 segment，故每次插入的比较次数上界均为 $\lceil \log c \rceil$。总共为 $(d-c)\lceil \log c \rceil$。

对于切割，我们考虑其均摊复杂度。我们只用考虑对于 touched consecutions 的代价。由于如果被切下的段是在某个 untouched consecution 前面则其需要两倍于自身长度的代价，因为我们是前后同时扫描的。如果被切下的段是在 untouched consecution 的后面，由于这个 segment 会被标记过，因此只有一倍的代价。但是识别出这个标记需要有一次提前的双向扫描，因此均摊代价是三倍。此外，为了识别切割点，通常会需要一次额外的比较才能确定，即对于长度为 l 的段，需要比较到第 $l+1$ 个元素才能知道是否是切割点。设所有 touched consecutions 的长度依次为 $l_1, l_2, ..., l_m$，则切割的比较次数上界为：

$$\begin{aligned}\sum_{i=1}^{m}(3l_i+1) &= 3\sum_{i=1}^{m} l_i + m \\ &\leqslant 6k + 2k = 8k\end{aligned}$$

把所有的上界加在一起我们得到：

$$\begin{aligned}&n - 1 + c\log(c+1) - \alpha c + \varepsilon_0 + (d-c)\lceil \log c \rceil + 8k \\ =\ & n - 1 + 8k + \varepsilon_0 + \varepsilon_1(d-c) + d\log(c+1) - \alpha c\end{aligned}$$

其中 $\varepsilon_1 = \log(c+1) - \lceil \log c \rceil \leqslant 1$。考虑 $d\log(c+1) - \alpha c$ 关于 c 的导数，易知当 $c \leqslant d \leqslant d/\alpha - 1$ 时单增，故上式：

$$\begin{aligned}&\leqslant n - 1 + 8k + \varepsilon_0 + \varepsilon_1(2k) + (4k+1)\log(2k+2) - \alpha(2k+1) \\ &\leqslant n - \alpha + (8 + 2\varepsilon_1 - 2\alpha)k + (4k+1)\log(2k+2) \\ &\leqslant n + 10k + (4k+1)\log(2k+2) \\ &= \mathrm{O}(n + k\log k)\end{aligned}$$

当 $n = 10^4$ 且 $k = 10$ 时估计上界为 10274 次比较。若 $k = 100$，则估计上界为 13982 次比较。对于随机数据，这个上界非常宽松。

优化

排序

段切割算法主要的 overhead 在排序部分。实际上对于随机数据，贪心划分后的 segment 也并不是完全随机的。实际的实现中，初次排序会先扫描所有待排序的 segment，相邻两个 segment 之间比较一下大小，如果是逆序对则交换一下。在二分插入前，将这次要插入的 segment 与上次插入的 segment 比一下大小，从而缩小一点二分范围。这两个操作在理论上没有任何优化，但实际的随机数据中减少了近 300 次比较。~~我也不知道为什么~~

缓存

由于不要求时间复杂度，所以实现了一个 cache，把所有的询问都缓存下来了。当有新的询问时，用 DFS 检查一下这个询问是否可以由以前的询问推出。缓存实测在随机数据上减少了 100 多次比较。