

# 论对题目中算法的选择

## 【关键词】

非最优算法，编程复杂度

**【摘要】** 本文介绍了对于一些信息学题目的次好或者差的算法，通过各种举例来说明做题目时并不要一味追求一个完全理想的算法，相对应比较差的算法在实际编程上对于那些好的算法有很大的优势。最后总结出在平常的练习之中，我们需要对一道题目进行多方面的思考，不能抱有知道算法就完事的一种心态，对一道题目，要多考虑新的算法，这样面对形式未见过的题型的时候，就会有更多的思路。

## 【正文】

### 一、引言

计算机竞赛是一项对选手计算机知识、编程能力的综合测试，在平时的训练之中，我们一般都会精益求精，设法想出最好的算法，但是在竞赛的时候，时间和心理状态都是不一样的，如何在竞赛中编出能得分尽量多的程序，是很重要的，所以在平常的练习之中，要有一题多解的概念，把各种算法尝试一下，就会在看似简单的做法中推想出更好的算法。在学习汇编语言的时候，老师这样说，汇编语言编译出的程序是运行速度最快的，为什么，对于一类题目，或者要实现一个目标，人编程用的算法所需要的语句得越多，则机器运行程序速度也就越快，反之人编程用的精力越少，机器运行速度也会越慢。若能找到一个编程难度较低的算法，使程序仍然能在时间限制内运行完成，这样即减轻了人的劳动，又能编写出正确的程序，而且编写的时候错误率也会降低，所以在竞赛中，次好或者差的算法也是可以考虑的。

### 二、算法的复杂度

我在这里所说的复杂度，包括编程复杂度，时间复杂度，空间复杂度，因为在竞赛几个小时的时间里，编程复杂度就不容忽视，知道算法但是程序没完成，这是最不理想的情况，对源程序的最基本的质量要求是正确性和可靠性，只要保证在这两点的情况下，各种算法都是好算法。但编程复杂度取决于程序的易理解性，易测试性和易修改性，程序不可能都是第一遍就能达到理想的状态，人总有失误的地方，如果有错，不易测试以至难以发现错误，或者知道错误而难以修改，就会浪费不必要的时间。或许，自己对知道的算法还不太熟悉，因为我们不可能对每种算法了解都很透彻，在这种情况下编程序是比较吃亏的，因要花许多时间去了解这种方法。所以在这三个复杂度都考虑的情况下，找到最适宜的算法是必要的。

### 三、对于一些题目的思路

在我们做题的时候，我们先想出的算法不一定是最好的算法，但不是说明它是不好的算法，所以，在我们想出一种算法的时候，要考虑一下它的可行性，如果可行的话，不如自己先把该算法编一编，可以锻炼自己编那些非标准算法的能力，这样对自己面对新颖题型的时候是会有好处的。

例1.

[问题描述]

输入:

一个正整数  $n$ , 以及整数数列  $A_1, A_2, A_3, \dots, A_n$ 。

一个正整数  $m$ , 以及整数数列  $B_1, B_2, B_3, \dots, B_m$ 。

其中 ( $1 \leq n \leq 10^6, 1 \leq A_i \leq 10^6, 1 \leq m \leq 1000, 1 \leq B_i \leq 10^6$ )

输出

一共  $m$  行, 每行一个整数, 第  $i$  行所输出的数表示数列  $\{A\}$  中小于等于  $B_i$  的数的数目。

[算法分析]

该题是 IOI 中移动电话的退化。标准算法是利用线段树来解决, 线段树的基本知识可以参阅相关书籍。

但我要说的做法是这样的, 首先, 我们便注意到了  $n$  与  $m$  的范围的差距。于是, 我们采用这样的做法。

其中的记录数组如下。

$L[1..1000000]$  记录直至当前, 其中  $A_i$  中取值为  $k$  的数有  $L[k]$  个

$V[1..1000]$  记录直至当前, 其中  $A_i$  中取值在  $(k-1)*1000+1$  到  $k*1000$  的数总共有  $V[k]$  个。

程序流程如下:

对  $L, V$  数组进行清零。

(1) 读入  $n$ , 依次读入  $A_i$ 。对于每个  $A_i$ , 设  $(k-1)*1000+1 \leq A_i \leq k*1000$  ( $k$  为整数)

$L[A_i]++, V[k]++$

(2) 读入  $m$ , 依次读入  $B_i$ 。对于每个  $B_i$ , 设  $(k-1)*1000+1 \leq B_i \leq k*1000$  ( $k$  为整数)

$$\text{设 } \{A\} \text{ 中小于 } B_i \text{ 的数有 } S \text{ 个, 易知 } S = \sum_{i=1}^{k-1} V[i] + \sum_{i=k}^{B[i]} L[i]$$

输出  $S$

[复杂度分析]

易知计算每个  $S$  的值最多需要运算操作  $1000+1000=2000$  次。

每次将一个  $A_i$  记录的操作有 2 次

利用数据分布的特点, 实际上的运算次数小于  $2000*m$ 。

这是分治法, 是一个过渡阶段的算法, 是一个不完全的算法, 本办法是分成  $1000*1000$ , 很容易联想到分成  $100*100*100$ , 既而逐渐推想到线段树的做法。

让我们看一下两种算法的复杂度比较

| 线段树              | 本算法          | (最坏情况)         |
|------------------|--------------|----------------|
| $(n+m)*\log_2 G$ | $n*2+m*2000$ | (其中 $G=10^6$ ) |

所以, 光凭借直觉, 我们不能判断两算法时间效率的优越。虽然在大多数情况下, 我们并不确定到  $n$  与  $m$  的大小, 但是如果确定两种算法都可行的情况下, 用本算法就降低了编程复杂度, 毕竟程序是人编的, 以后要修改, 理解起来也方便一点。

例2. 营业额统计(湖南选拔赛)

[问题描述]

公司的账本上记录了公司成立以来每天的营业额。分析营业情况是一项相当复杂的工作, 营业额会出现一定的波动, 当然一定的波动是能够接受的, 但是在某些时候营业额突变得很高或是很低, 这就证明公司此时的经营状况出现了问题。经济管理学上定义了一种**最小波动值**来衡量这种情况:

该天的最小波动值  $= \min\{ | \text{该天以前某一天的营业额} - \text{该天营业额} | \}$

当最小波动值越大时，就说明营业情况越不稳定。

而分析整个公司的从成立到现在营业情况是否稳定，只需要把每一天的最小波动值加起来就可以了。你的任务就是编写一个程序来计算这一个值。第一天的最小波动值为第一天的营业额。

输入文件第一行为正整数  $n (n \leq 32767)$ ，表示该公司从成立一直到现在的天数，接下来的  $n$  行每行有一个正整数  $a_i (1 \leq a_i \leq 1000000000)$ ，表示第  $i$  天公司的营业额。

输出文件仅有一个正整数，即  $\sum$  每一天的最小波动值。结果小于  $2^{31}$ 。

#### 输入输出样例

|                                 |    |
|---------------------------------|----|
| 6<br>5<br>1<br>2<br>5<br>4<br>6 | 12 |
|---------------------------------|----|

#### [算法分析]

本题题意明了，关键是读入一个数，找到前面已经输入的与此数相差最小的数。

题目的标准算法是通过平衡树来解决的，平衡树，就是一种能自己进行适应，使树的深度达到尽量小的一种检索树，详细的情况可以参考相关书籍。

##### (1) 预处理：

① 把全部数据读入，将  $\{a\}$  从小到大排序，这样得到一个序列  $B_1, B_2, \dots, B_n$ 。

② 累加器  $S$  赋值为 0

##### (2) 主过程：读入将要处理的数据，对每天的营业额进行处理，求出该天的最小波动值。

(这里的  $V, L$  与例 1 中的定义一样。

但这次定义  $L[1..32767]$ ， $V[1..327]$  也就是范围改了一改，以 100 为长度单位分隔区间)

- 读入当天的营业额  $P$
- 用二分查找法在数列  $\{B\}$  中找到  $B_q = P$ 
  - 设  $(k-1)*100+1 \leq q \leq k*100$ ，若  $V[k] > 0$  即该区间内已经记录过数据，在该区间内找到大于等于  $q$  最小的数  $ga$ ，且  $L[ga] > 0$ ，及小于等于  $q$  最大的数  $gb$ ，且  $L[gb] > 0$
  - 若找不到  $ga$  或  $gb$ ，现在说一下找不到  $gb$  的情况 ( $ga$  的情况相同)
  - 依次查询  $V[k-1], V[k-2]$  直至  $V[1]$  找到有  $V[t] > 0$
  - 若找到这样的  $t$ ，则在区间  $((t-1)*100+1, t*100)$  内找到最大的数  $h$ ，使  $L[h] > 0$ ，即  $gb = h$
- 计算当天的最小波动值
  - 若找到这样的  $ga$  或  $gb$ ，则比较  $P$  与  $B_{ga}$  的差或  $P$  与  $B_{gb}$  的差，取较小的值，即为当天的最小波动值。
  - 若同时找不到  $ga$  与  $gb$ ，证明这是第一天，最小波动值即为  $P$ 。
- 将当天的最小波动值加入累加器  $S$
- 将元素  $q$  加入数组，即  $L[q]++$ ， $V[k]++$

##### (3) 最后收尾：输出 $S$

(这里的  $V, L$  与例 1 中的定义一样。

但这次定义  $L[1..32767]$ ， $V[1..327]$  也就是范围改了一改，以 100 为长度单位分隔区间) 详细程序参见 myturn.pas (稍有一些改进)

虽然，程序的长度大致相同，但是，本算法的测试与修改却比平衡树容易的多，因为它是分两步进行，可以分步进行测试而且每步的要求技术都不高。

每个算法都是有它自己的优势，例如，我们对于排序，对于小数据我们就用冒泡，选择等时间复杂度高，但编程复杂度低的算法，反之，我们则会使用堆排序，快速排序，归并排序等，每种算法的效果在不同的场合是不一样的，在编程的过程中，在平常的练习之中，我们就需要对一道题目进行多方面的思考，不能抱有知道算法就完事的一种心态，对一到题目，要多考虑新的算法，这样便能发现每种算法使用的场合，这样面对形式未见过的题型的时候，就会有更多的思路。

|                 |         |     |    |
|-----------------|---------|-----|----|
| 1. 算法与数据结构      | 傅清祥     | 王晓东 | 编著 |
|                 | 电子工业出版社 |     |    |
| 2. 实用算法的分析与程序设计 | 吴文虎     | 王建德 | 编著 |
|                 | 电子工业出版社 |     |    |