# 最长递增子序列

对于原问题的最优解，需要去理解算法原型：最长递增子序列（不连续）。最长递增子序列的一般解的时间复杂度为O(n2)，最优解的时间复杂度为O(n\* logn)。

先来看一般解。例子数组r为：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 6 | 4 | 5 | 2 | 7 | 4 |

这个数组的最长递增子序列是1（2）、4、5、7，长度为4。

解法：首先定义一个与该数组等长的一个数组h，i为数组下标值，其中h(i)的含义是：当必须以r(i)结尾的情况下，最长递增子序列的长度。下面是数组h的情况：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2 | ... | .... | ... |  |  |

当i为0，h(0)表示以2为结尾的最长递增子序列，为1。

而当i为2时，r(2)=6，就要去考察前面的值，只有比6小的值才能与6结合成为新的最长递增子序列，所以在不断考察前面的值的过程可以通过对比找到新的最长递增子序列的长度。

再来看最优解。采用相同的例子数组r。

解法：同样地定义一个与该数组等长的一个数组h，i为数组下标值，其中h(i)的含义是：当遍历数组r的某值的时候，在有效区（有值填入的区域）中，长度为i+1的递增子序列的最小末尾（相对于前面是发生变化的）。

流程说明：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 |  |  |  |  |  |  |  |

刚开始时，将r(0)填入，表示长度为1（i是0）的递增子序列的最小末尾是2。

然后继续遍历到r(1)=1，在有效区中寻找第一个比这个值大的位置，将该值填入，如果没找到就扩充有效区。注意，这个有效区是有序的，这里找位置时用的是二分，那就是为什么能做到logn的原因。所以接下来的变化就是：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 6 |  |  |  |  |  |  |

关键点：理解数组h的定义，进而更好地理解流程（如果不是太理解，自己模拟一下流程即可）。