最长不下降子序列

hyliu

2019 年 8 月 17 日

目录

1 最长不下降子序列

1.1 问题定义

设有一个正整数序列 a[n]: a_1 , a_2 ,..., a_n , 对于下标 $\mathrm{i}_1 < \mathrm{i}_2 < ... < \mathrm{i}_\mathrm{h}$, 若有 $\mathrm{a}_{\mathrm{i}_1}$, $\mathrm{a}_{\mathrm{i}2}$,..., $\mathrm{a}_{\mathrm{i}_\mathrm{h}}$, 则称序列 a[n] 含有一个长度为 h 的不下降子序列。

例如，对于序列 3,7,9,16,38,24,27,38,44,49,21,52,63,15 对于下标 $i_1 = 1$,$i_2 = 4$,$i_3 = 5$,$i_4 = 9$,$i_5 = 13$, 满足 3<16<38<44<63

则存在长度为 5 的不下降子序列。

1.2 状态

dp[i] 表示以 a_i 结尾的最长不下降子序列长度。

1.3 状态转移方程

```
for (int j = 0 ;j< i ;j++){
    if (num[j]<=num[i])
        dp[i] = max(dp[i],dp[j]+1);
}
```

1.4 优化

O(nlogn) 的算法关键是它建立了一个数组 b[]，b[i] 表示长度为 i 的不下降序列中结尾元素的最小值，用 k 表示数组目前的长度，算法完成后 k 的值即为最长不下降子序列的长度。

具体点来讲：

不妨假设，当前已求出的长度为 k，则判断 a[i] 和 b[k]：

如果 b[k] a[i]，即 a[i] 大于长度为 k 的序列中的最后一个元素，这样就可以使序列的长度增加 1，即 k=k+1, 然后更新 b[k]=a[i]；

如果 b[k]>a[i]，那么就在 b[1]⋯b[k] 中找到最大的 j, 使得 b[j] < a[i]，即 a[i] 大于长度为 j 的序列的最后一个元素，显然，b[j+1] a[i]，那么就可以更新长度为 j+1 的序列的最后一个元素，即 b[j+1]=a[i]。

可以注意到：b[i] 单调递增，很容易理解，长度更长了，d[k] 的值是不会减小的，更新公式可以用二分查找，所以算法复杂度为 O(logn)。

1.4.1 具体实现

```
b[0] = 0;
for (int i = 1 ;i<=n ;i++)
    b[i] = INF;
for (int i = 0 ;i< n ;i++){
    int pos = upper_bound(b,b+n+1,num[i]) - b;
    b[pos] = min(b[pos],num[i]);
    dp[i] = pos;
}
```