LIS 问题

李城

2024 年 12 月 17 日

1 复杂度为 O（n^2）算法设计

1.1 想法：

简要想法：运用动态规划的思想，类似数学归纳法，分析某一元素 a 前面已有严格递增序列，给出 a 作为最后一个元素的严格递增序列序列元素个数可能的最大值，最后依照从后往前的顺序，自 LIS 最后一个元素开始给出一个可行的 LIS

1.1.1 描述：

```
关键有一函数 LIS (std::vector<int> A):
        n = A.size()//储存给定序列A的大小
        std::vector<int> max( n, 1 )//储存begin到特定A[i]的最大序列大小
        std::vector<int> prev( n, -1 )//用于跟踪每个A[i]添加后产生更长序列的作用位置
        maxlength = 1//存储整体A中LIS大小
        maxindex = -1//存储整体A中LIS最大元的位置
        for ( i = 0; i < n; ++i)
        {
            for ( j = 0; j < i; ++j)
                if ( A[j] < A[i] && max[i] < max[j] + 1)   //代表找到一个原有的、到A[j]为止的LIS可以添加A[i]元素构成更长子序列
                {
                    max[i] = max[j] + 1
```

```
                    prev[i] = j
                }//结束后prev指向的是最后一个满足的位置j
            if (max[i] > maxlength)//代表整体看来max[i]足够大,
                需要更新begin到A[i]为止的LIS长度和最后元素位置
            {
                maxlength = max[i]
                maxindex = i
            }
        }

        std::vector<int> lis( maxlength, 0 )
        k = 0
        t = maxindex
        while ( t != -1 )//逆序存放最长子序列
        {
            lis[k] = A[t]
            ++k
            t = prev[t]
        }
        std::reverse( lis.begin(), lis.end() )//变为正序

        for( i in lis )//打印
            std::cout << i <<" "
        std::cout << std::endl
```

1.2 例子:

```
    初始状态:
A   :10  9   2   5   3   7   101 18
max :1   1   1   1   1   1   1   1
prev:-1  -1  -1  -1  -1  -1  -1  -1

处理A[1]
A   :10  9   2   5   3   7   101 18
max :1   1   1   1   1   1   1   1
prev:-1  -1  -1  -1  -1  -1  -1  -1

处理A[2]
```

```
A    :10  9   2   5   3   7   101  18
max :1    1   1   1   1   1   1    1
prev:-1   -1  -1  -1  -1  -1  -1   -1

处理A[3]
A    :10  9   2   5   3   7   101  18
max :1    1   1   2   1   1   1    1
prev:-1   -1  -1  2   -1  -1  -1   -1

处理A[4]
A    :10  9   2   5   3   7   101  18
max :1    1   1   2   2   1   1    1
prev:-1   -1  -1  2   2   -1  -1   -1

处理A[5]
A    :10  9   2   5   3   7   101  18
max :1    1   1   2   2   3   1    1
prev:-1   -1  -1  2   2   4   -1   -1

处理A[6]
A    :10  9   2   5   3   7   101  18
max :1    1   1   2   2   3   4    1
prev:-1   -1  -1  2   2   4   5    -1

处理A[7]
A    :10  9   2   5   3   7   101  18
max :1    1   1   2   2   3   4    4
prev:-1   -1  -1  2   2   4   5    5

根据maxlength = 4;maxindex = 6
从101开始作为lis第一个元素
根据prev找到元素 7,3,2
lis是[101,7,3,2]
reverse后变成[2,3,7,101]

输出
```

2 复杂度为 O（$nlogn$）算法设计

2.1 想法：

原有方法中查找较慢，影响效率，只需采用二分查找法即可优化至 O(nlogn) 因为对于已有 A[i], 我们可以知道其性质，因此我们只需要利用 tails[i] 来维护前 i+1 个元素中所有递增子序列的最小尾部元素，这样就是有序的，对 A[i+1]，我们就能利用二分查找快速定位 A[i+1] 的可接位置