Lab6

22302010022 曾华正

Algorithm

动态规划

model.py schedule_model_Dynamic
先删除无效job，然后其他根据**DDL**排序，之后采用类似0-1背包问题的动态规划算法来进行计算,通过遍历m*n的辅助表来找出最优解，m为单次job最长的DDL，n为处理过后的数据量，所以该操作的复杂度为O(m * n),

贪心算法（不适用）

model.py schedule_model_Greedy
以下为基于原本样例的分析

initial

先对jobs数据进行排序，根据其三个特性**DDL**，**Profit**，**Profit/time**，进行排序，三个特征值优先级依次下降,而后进行贪心操作，选取局部最优解，最后形成答案;发现错误，进行优化得到第二种贪心算法的答案

Improve

改变排序,排序优先级改为**Profit**，**DDL**，**Profit/time**，发现第十个样例的结果从160提高到190，其余不变，解决**Shortcoming**中提到的样例；但仍然未显著提高算法鲁棒性。

新样例结果

动态规划算法

160
180
100
160

220
180
140
180
100
190

Greedy

160
100
100
160
220
180
140
180
100
190

由此可见 Greedy算法在此问题上不适用

Efficiency Analysis

动态规划

删除与排序工作时间复杂度为O(nlog(n))
后续动态规划的复杂度为O(m * n),m为单次job最长的DDL，n为处理过后的数据量
所以该操作的复杂度为O(n * m)
综上，以上操作的时间复杂度为O(n * m+n log n)

贪心算法

贪心算法遍历所有的的jobs，时间复杂度为O(N)
前期排序使用sorted函数，其内置使用了merge sort 和 insert sort，时间复杂度为O(Nlog(N))
综上，上述算法的时间复杂度为O(Nlog(N))

Shortcoming

贪心算法对jobs进行排序的量化方式会导致不同的结果，例如：当样例出现 **[(4,60,4),(4,70,7)]** 时，使用initial方法，排序后未改变其顺序，最后的结果仍然为60，理论上最优解为70，这也是贪心算法的一个缺点；如果我们加入更多的特征数值来进行排序，会使得算法鲁棒性更优；鲁棒性问题说明贪心算法不适合于该问题

result

综上，贪心算法并不适用于该问题，在出现特殊情况时候无法解决；动态规划更适用于该问题