**20174179 杨小川 计科卓越班**

**（1）最长非递增子序列问题可以转换为最长公共子序列问题求解，描述算法思路并证明其正确性。**

设原序列为L=<a1,a2,…,an>，则可将其排序求得其非递增序列Lb=<b1,b2,…,bn>。假设L的递增子序列是Lin=<aK1,ak2,…,akm>，其中k1<k2<…<km且aK1<ak2<…<akm，则L的最长非递增子序列MAX(Lin)为L和Lb的最长公共子序列。由此可将最长非递增子序列问题转换为最长公共子序列问题。

**（2）设a[1..n]为整数序列, L为其中最长非递增子序列的长度，数组b[k] (1<=k<=L)为a[1..n]中长度为k的所有非递增子序列的末尾元素的最大值。证明b[1]>=b[2]>=....>=b[L]。**

b[k]为a[1..n]中非递增子序列X[1..k]的末尾元素，则b[k+1]为a[1..n]中非递增子序列X[1..k+1]的末尾元素。由于X[1..k]和X[1..k+1]同为a[1..n]的非递增子序列，且Length(X[1..k]) < Length(X[1..k+1])，所以。由于X[1..k]和X[1..k+1]都是非递增的，所以。递推可得b[1]>=b[2]>=....>=b[L]。

**（3） 16.1-2 假定我们不再一直选择最早结束的活动，而是选择最晚开始的活动，前提仍然是与之前选出的所有活动均兼容。描述如何利用这一方法设计贪心算法，并证明算法会产生最优解。**

令Sk为活动ak开始前结束的任务集合。若ai是Sk中最早结束的任务，则当我们选择ai，那么剩下的Si是唯一需要求解的子问题。因此可以用贪心算法。此时，，，因此ak与ai不相交。

**（4）16.1-5 考虑活动选择问题的一个变形：每个活动ai除了开始和结束的时间外，还有一个值vi。目标不再是求规模最大的兼容活动子集，而是求值之和最大的兼容活动子集。也就是说，选择一个兼容活动子集A，使得最大化。设计一个多项式时间的算法求解此问题。**

设每个活动an有start[n]，end[n]，value[n]。

S[end[i],start[j]]表示在end[i]~start[j]区间内的带权重活动最大值（即活动a[i]结束之后开始，且在a[j]开始之前结束的那些活动的最大值）。

则可得到递推方程：

S[end[i],start[j]] = S[end[i],start[k]] + value[k] + S[end[k],start[j]]。

由此可以动态规划求解。

**（5）16.2-7 给定两个集合A和B，各包含n个正整数。你可以按需要任意重排每个集合，重排后，令ai为集合A的第i个元素，bi为集合B的第i个元素。于是你得到回报。设计算法最大化你的回报。证明你的算法是正确的，并分析运行时间。**

将A和B排序为单调递减顺序。则对于i < j，有ai < aj，

=>

=>

=>

所以只需将A和B分别均单调递减排序或单调递增排序，即可得到最大的回报。