研究生算法课课堂笔记

上课日期：2016-11-24　　第(2)节课

组长学号及姓名：1601214729 刘旭钦
组员学号及姓名：1601214737 谭晓烨
1601214735 尚明月

内容概要：

本节课主要讲授贪心算法中的活动选择问题（interval scheduling），并对上一节课的一个遗留问题进行解答。具体如下：

1. 对于 coin changing 问题，在什么情况下用贪心法能得到最优解？
2. 活动选择问题描述。
3. 活动选择问题贪心算法设计。
4. 活动选择问题贪心算法最优性证明。
5. 活动选择问题与最大独立集的关系。
6. 课后习题解答及其变种问题的探讨。
7. 课室安排问题描述

详细内容：

1. 对于 coin changing 算法，在什么情况下收银员算法可以得到最优解？

当高面值的硬币可以被相邻的低面值硬币拼出时，贪心法得到的解就是最优解。比如：面值为 1, 5, 10, 20, 100 的硬币。但是这只是充分条件，不是必要条件。

2. 活动选择问题（interval scheduling）描述

活动 j 开始于s_j，结束于f_j，如果两个活动的开始时间到结束时间的区间不重合，则这两个活动是兼容的，目标是寻找最大活动子集，使得子集里所有活动相互兼容。图 1 表示了一组活动的分布，其中深色部分为一个最大活动子集解。

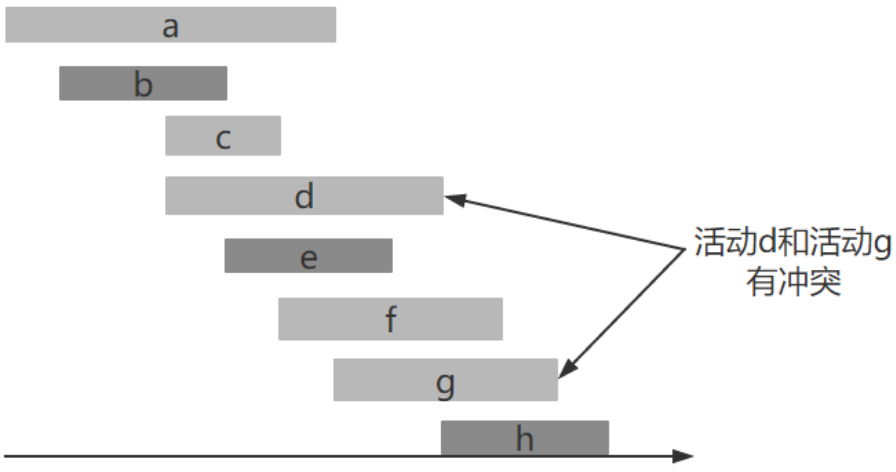

图 1 一组活动分布

Q： 与之前课上讲过的用动态规划选择活动问题的区别？
A： 在贪心法解决活动选择问题时，每个活动没有权重，最终目标只需要选出兼容的活动数最多。

3. 活动选择问题（interval scheduling）算法设计

可选的贪心法思路：

1) 开始时间早的活动优先 [空着也是空着]
2) 结束时间早的活动优先 [结束早给后面的活动更多机会]
3) 活动时长最短的活动优先 [时间短，性价比高]
4) 冲突数最小的活动优先 [例如排课]

贪心法设计 Tip：首先寻找容易想到的自然的思路，然后针对每一种思路寻找是否有反例。如果很难举出反例，则试证明这种思路的正确性。

第一种算法的反例：

有一个活动开始时间最早，但是结束时间最晚。如图 2。

图 2 第一种算法反例

第三种算法的反例：

有一个活动虽然时长最短但是干扰了其他的活动。如图 3。

图 3 第三种算法反例

第四种算法的反例：

有一个活动虽然冲突最少但不是最优解。如图 4。

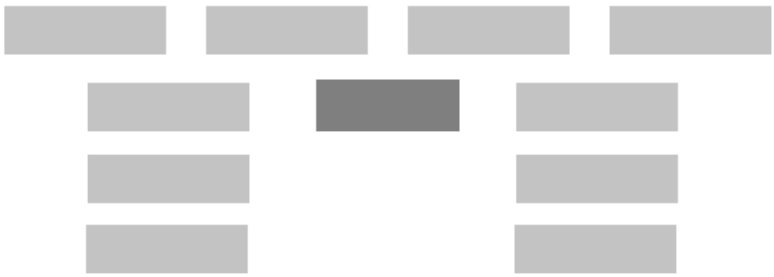

图 4 第四种算法反例

而第二种算法是正确的，算法描述如下。

EARLIEST-FINISH-TIME-FIRST (n, s_1, s_2, … , s_n, f_1, f_2, … , f_n)

SORT jobs by finish time so that $f_1 \le f_2 \le \ldots \le f_n$

A ← Ø (set of jobs selected)

FOR j = 1 TO n

  IF job j is compatible with A

    A ← A ∪ { j }

RETURN A

这种算法的时间复杂度为：$n \log n$，如果不考虑排序时间，算法是线性的。因此算法的瓶颈为排序时间。

Q：如果给出的活动是按照开始时间排好顺序的，那么从最晚开始时间向前找，即最晚开始时间优先，得到的结果是最优解吗？

A：这种方法得到的也是最优解。反看时间轴，则相当于按照结束时间来排序。但是与按照结束时间优先的算法得到的结果不一定一样，如图 5 所示。

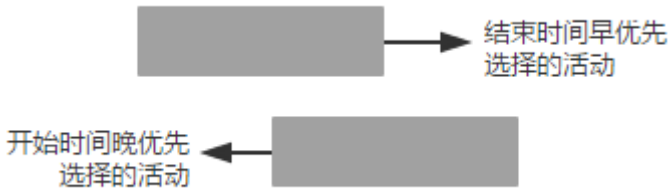

图 5 举例

4. EARLIEST-FINISH-TIME-FIRST 算法的最优性证明：

证明：

假设贪心算法不是最优的。

令$i_1, i_2, \ldots, i_k$表示由贪心算法选择出来的活动序列,

令$j_1, j_2, \ldots, j_m$表示由一个最优的算法选择出来的活动序列,

假设两种算法前 r 个选择出来的活动一样，即$i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$,如图 6 所示，

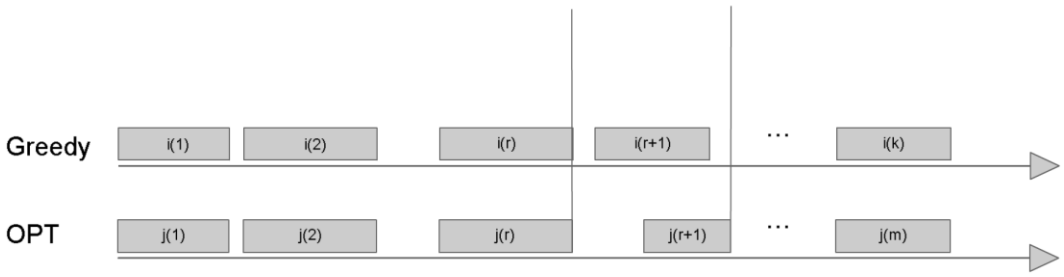

图 6 两种算法活动选择序列

根据贪心法的策略，每次优先选择最早结束的活动，那么对于第$r+1$个活动，贪心法选出来的结束时间一定不会晚于那个最优的算法选择出来的，因此可以将i_{r+1}与j_{r+1}交换，这样不会造成冲突，而且结束时间更早，同理，可以将后续所有活动进行交换，那么最优的算法选择出来的活动序列就变成了由贪心算法选择出来的活动序列，并且结束时间比最优的算法选择出来的活动序列的结束时间更早，则能安排更多的活动，因此贪心算法会变成更优的，与假设矛盾。由此可证得贪心算法 Earliest finish time first 是最优的。

最优解选择的活动数目是唯一的，但是具体选择的活动是不唯一的

5. 活动选择问题与最大独立集的关系。

独立集：对于图$\mathrm{G}=(\mathrm{V},\mathrm{E})$，V是顶点集合，E是边的集合，独立集$V'$满足$V'\subset V$，$\forall u,v\in V'$，$(u,v)\notin E$。

若将每一个活动看作图中的顶点，若两个活动不兼容则表示这两个活动的顶点之间连有一条边，那么可以据此作出一个区间图(interval graph)，求最大活动子集可以转化成求区间图的最大独立集。

Q：最大独立集问题是 NPC 的，而活动选择问题的时间复杂度是O(nlogn)的，是否说明 NPC 的问题能够被nlogn的时间复杂度求解？
A：不能，因为区间图只是一个特例，不是所有的图都能转化成区间图来求解的。例如图 7 是一个反例。

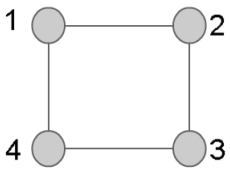

图 7 四边形反例

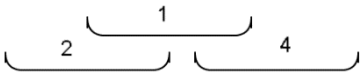

图 8 活动 1,2,4 区间图

若将四个顶点看成四个活动，将活动 1、2、4 绘制成如图 8 所示的区间图，由于活动 3 与活动 2、4 冲突，那么只能画在活动 2 和 4 中间，那么必然与活动 1 冲突，这样就产生了矛盾，说明图 7 不能转化成区间图。

6. 课后习题解答及其变种问题的探讨。

问题：从波士顿到纽约通过卡车运箱子，每个卡车有同样的限重W，每个箱子i有各自的重量w_i，要求箱子必须按照它们到达运输公司的顺序来发送，现在的安排方式是：按照箱子来的顺序一个一个往上装，直到下一个箱子装不上去了，卡车就开走，然后往下一个卡车上装。问：这种方式是否是最省卡车的数目的？

> **Q**：是否可能有若干个箱子在卡车未满时就不放了，改放到下一个卡车上，由此导致的安排顺序的改变会使最后卡车的使用数量减少了
> **A**：不可能，即问题中的贪心算法是最优的

证明问题中的贪心法最优：

假设贪心算法不是最优的。

令$i_1, i_2, \dots, i_k$表示由贪心算法得出来的安放好箱子的卡车序列 Greedy，令$j_1, j_2, \dots, j_m$表示由一个最优的算法得出来的安放好同样箱子的卡车序列 OPT，且$m < k$，拷贝 OPT 序列得到 New 序列。

假设两种算法前 r 个选择出来的箱子安放方式一样，如图 9 所示。

对于第 r+1 辆车，设蓝色块为装在 Greedy 的i_{r+1}中且被装在 OPTj_{r+2}及以后卡车中的箱子，那我们可以将 New 序列的蓝色块调整到j_{r+1}中，这样的序列也是合法的且前$r+1$辆车与 Greedy 相同。类似地，将装在i_{r+2}中且被装在j_{r+3}及以后卡车中的箱子（如果有的话），在 New 序列中调整到j_{r+2}里，也是一个有效的序列，且 New 序列和 Greedy 序列的前$r+2$辆车相同。以此类推到最后，New 变成与 Greedy 相同的序列。但是这样就会得到$m \geq k$（因为可能 New 中最后有的卡车被调整空了），与$m < k$矛盾，所以贪心算法是最优的。

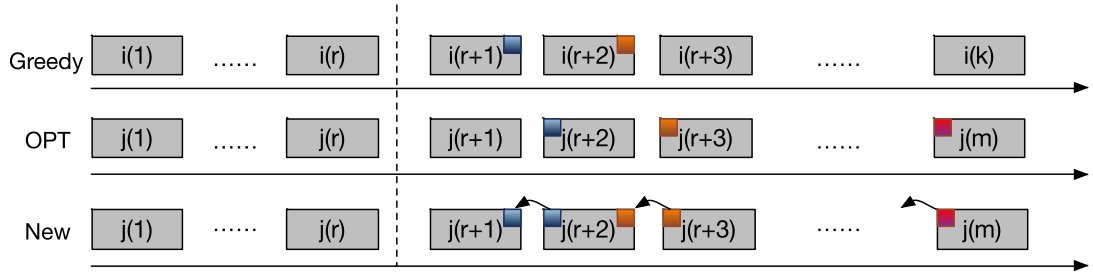

图 9 贪心法、最优算法得出的序列以及一个新序列

这种证明方法叫做”The greedy algorithm always stay ahead”，即贪心法永远是领先的，永远不会吃亏。

Q：如果不要求按照箱子来的顺序装（可以乱序装）呢？
A：实质是背包问题的变种：多个背包，背包限重相同，求需要背包数最少的安排方法

关于背包问题的动态规划解法不是多项式时间算法的解释：利用动态规划算法得出的背包问题的时间复杂度是$O(N*W)$，其中N是物品数目，W是背包的限重。W在计算机中是用$\log W$个 bit 来表示的，而算法复杂度是按照输入规模来看的。记$t=\log W$，则算法复杂度是$O(N*2^t)$，故背包问题的动态规划解法不是多项式时间的算法。

---

7. 课室安排问题描述

课程 j 开始于s_j，结束于f_j，目标是找到最少的课室数量，保证同一时间在同一个课室里不会同时有两个课程。

具体讲解留到下次课。