PKU-04833060
算法设计与分析『实验班』

第1讲 引论

罗国杰

gluo@pku.edu.cn

2024年春季学期

主要内容

- 课程概述
- 算法设计
  - 基本概念
  - 问题形式化

算法设计的基本概念

- 算法的定义与基本概念
- 算法的分类
- 算法设计技术
- 算法的伪码描述

算法的基本概念：例1 最少总等待时间调度问题

已知：任务集 $S=\{1,2,\ldots,n\}$，

第 j 项任务加工时间：$t_j \in Z^+$，$j=1,2,\ldots,n$

一个可行调度方案：$1,2,\ldots,n$ 的排列 $i_1,i_2,\ldots,i_n$

求：总等待时间最少的调度方案，

即求 S 的排列 $i_1,i_2,\ldots,i_n$ 使得 $\sum_{k=1}^{n}(n-k)t_{i_k}$ 最小

贪心策略：加工时间短的先做。

如何描述这个方法？这个方法是否对所有的实例都能得到最优解？

如何证明？这个方法的效率如何？

算法的基本概念：例2 排序算法的评价

已有的排序算法：考察元素比较次数

插入排序、冒泡排序：最坏和平均状况下都为 $O(n^2)$

快速排序：最坏状况为 $O(n^2)$，平均状况下为 $O(n \log n)$

堆排序、二分归并排序：最坏和平均状况下都为 $O(n \log n)$

……

那个排序算法效率最高?

是否可以找到更好的算法？排序问题的计算难度如何估计?

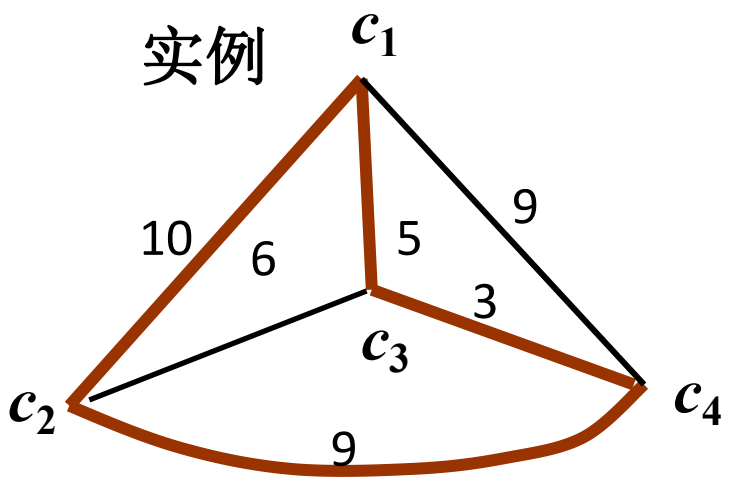
算法的基本概念：例3 货郎问题

货郎问题：

有穷个城市的集合 $C = \{c_1, c_2, \ldots, c_m\}$, 距离

$$d(c_i, c_j) = d(c_j, c_i) \in Z^+, \quad 1 \le i < j \le m$$

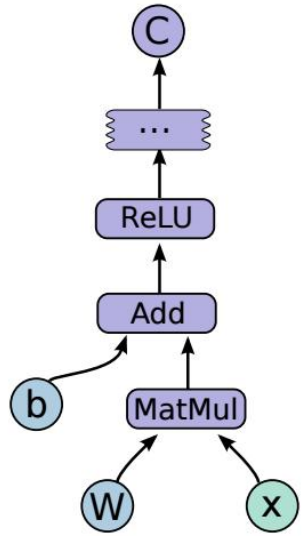
求 $1, 2, \ldots, m$ 的排列 $k_1, k_2, \ldots, k_m$ 使得 $\sum_{i=1}^{m-1} d(c_{k_i}, c_{k_{i+1}}) + d(c_{k_m}, c_{k_1})$ 最小



现状：至今没有找到有效的算法，
存在大量问题与它难度等价

问题：是否存在有效算法？
如何处理这类问题？

算法的基本概念



- 如何描述待解问题？ ⇒ 问题形式化
- 如何描述解决方法？⇒ 算法表达（流程图、代码、伪代码）
- 是否对所有输入都能得到合法/最优解？ ⇒ 正确性证明
- 方法的效率如何？如何比较不同方法的效率？ ⇒ 时空复杂度和渐近复杂度分析
- 问题的难度如何？是否存在更高效的方法？ ⇒ 问题复杂度

算法的基本概念：问题

问题：

需要回答的一般性提问，通常含有若干参数

问题描述所包含的内容：

对问题参数的一般性描述

解满足的条件

问题的实例：

对问题的参数的一组赋值

一个问题是由它的全体实例构成的集合

注：第10讲《问题的复杂度分析》，将具体定义“问题”和“算法”

算法的基本概念：算法

非形式定义

有限条指令的序列

确定了解决某个问题的运算或操作

输入个数大于等于0

输出个数大于0

形式定义

对所有的有效输入停机的Turing机、RAM机等等

算法 A 解问题 P

把问题P的任何实例作为算法A的输入，A能够在有限步停机，并输出该实例的正确的解

算法的基本概念

- 算法的时间复杂度
  - 针对问题指定基本运算，计数算法所做的基本运算次数
- 最坏情况下的时间复杂度
  - 算法求解输入规模为n的实例所需要的最长时间$W(n)$
- 平均情况下的时间复杂度
  - 在指定输入的概率分布下，算法求解输入规模为n的实例所需要的平均时间$A(n)$
- 其他时间复杂度:
  - 平摊时间（Amortized）、平滑分析（Smoothed）

检索问题的时间估计

- 检索问题
  - 输入：非降顺序排列的数组 L，元素数为 n； 数 x
  - 输出：j ——若 x 在 L 中， j 是 x 首次出现的序标；否则 $j = 0$
  - 算法：顺序搜索

  - 最坏情况时间：$W(n) = n$
  - 平均情况时间：实例集 S，实例 $I \in S$ 出现的概率是 p_I，算法对 I 的基本运算次数为 t_I，平均情况下的时间复杂度为

$$A(n) = \sum_{I \in S} p_I t_I$$

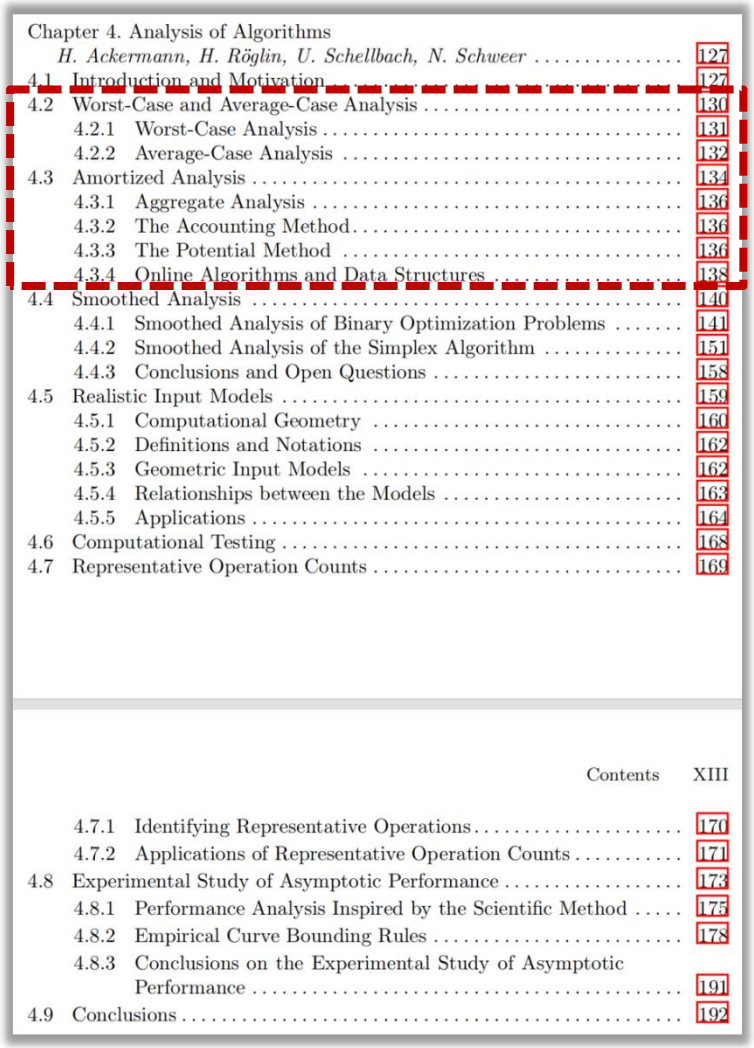
    - 对于顺序搜索算法，假设 $x \in L$ 的概率为 p，x 在 L 不同位置等概分布，则

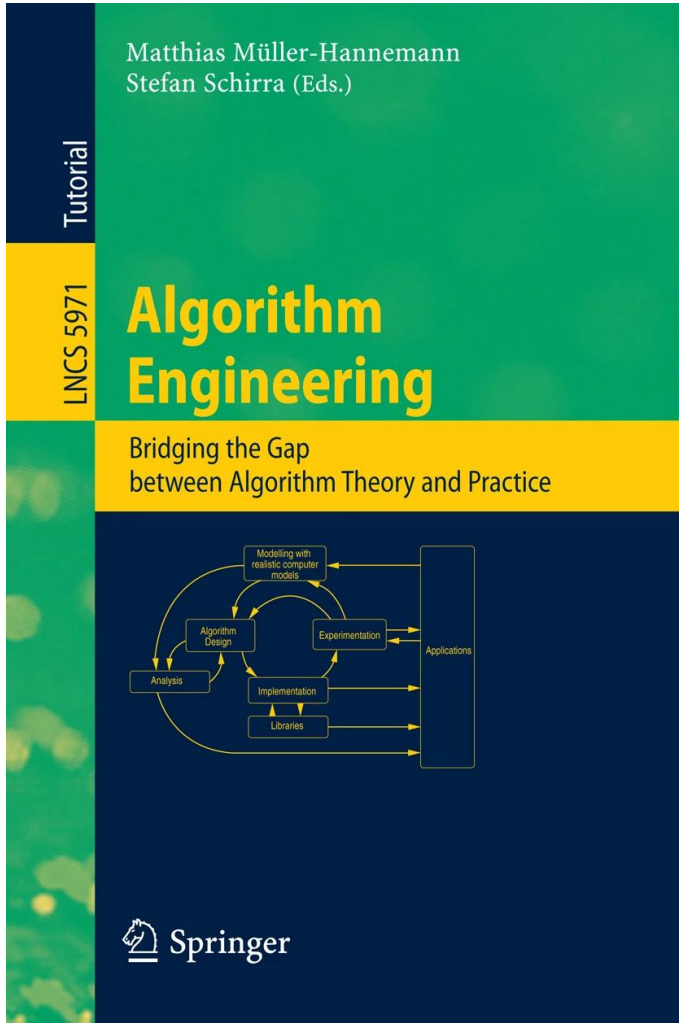
$$A(n) = p \cdot \sum_{i=1}^{n} i \cdot \frac{1}{n} + (1-p) \cdot n = \left(1 - \frac{p}{2}\right) \cdot n + \frac{p}{2}$$

规范分析与实证分析

- 规范分析
  - 算法复杂性分析多是理论上的分析；
  - 算法用伪码描述，忽略编程语言、算法实现和运行环境对算法运行时间的影响；
  - 关注于当输入规模 n 趋近于无穷大时算法时间的渐近表示，例如：$O(n)$优于$O(n^2)$
- 实证分析
  - 理论上更优的算法并非在实践上是最实用的算法；
  - 通过基准测试（benchmarks）发现算法优化前后的性能改进；
  - 例：快速排序、线性规划单纯形法、斐波那契堆、SELECT算法

更多的算法分析实践方法





D. Hutchison, J. C. Mitchell, and M. Müller-hannemann, "Algorithm Engineering: Bridging the Gap between Algorithm Theory and Practice," vol. 5971. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

算法的分类

- 递归算法与迭代算法
  - 递归算法：通过反复对自身调用直到某种条件成立为止。例如函数式语言。
  - 迭代算法：通过循环或辅助以栈等数据结构来求解问题。类似处理器运行方式。
- 递归算法和迭代算法是可以相互转换
  - 任何递归算法都可以利用栈转化为迭代算法；
  - 任何迭代算法也可以转化为没有循环的递归算法。

算法的分类

串行算法与并行算法

- 串行算法：设计用于在串行计算机上执行的算法。
  - 算法的正确性可能依赖于算法指令按顺序一个接一个的执行。算法指令执行的乱序可能导致算法输出结果的不正确。
  - 串行算法的时间复杂性等于基本运算的执行次数。
- 并行算法：设计可用于在多个处理器上同时协作执行的算法。很多迭代算法都可以并行化。
  - 并行算法的运行时间不仅与算法的并行加速比有关，还与计算节点间的通信开销相关。
  - Amdahl法则表明，大多数算法的并行加速比与处理器数之间并不能达到线性关系。
  - 多种并行计算模型：PRAM，原子GRAM，格网，超立方体；块GRAM，BSP，CGM，LogP……

$$S = \frac{1}{f + \frac{1-f}{p}}$$

算法的分类

- 精确算法与近似算法
  - 精确算法：多数算法都是精确算法，算法将给出问题的明确的解。
  - 近似算法：对于NPC类问题和NP-hard问题，不太可能找到多项式时间的精确算法，只能在多项式时间内给出问题的近似解。
    - 近似比反映近似解与最优解之间的差距，是描述近似算法质量的重要指标。
    - 近似算法设计中常会采用贪心策略或随机策略。

算法的分类

- 离线算法和在线算法
  - 离线算法：基于在执行算法前输入数据已知的基本假设下，即具有问题完全信息前提下设计出来的算法。
    - 通常的算法都是离线算法：例如选择排序算法
  - 在线算法：通常顺序的一个一个的处理输入，并在得到每个输入时做出算法决策。
    - 在线算法例子：插入排序；页面淘汰。
    - 加拿大旅行者问题：这个问题的目标是在一个有权图中以最小的代价到达一个目标节点，但这个有权图中有些边是不可靠的可能已经被剔除。然而一个旅行者只有到某个边的一个端点时才能确定该边是否已经被移除了。
    - 竞争比：在线算法的解与离线算法最优解之间的费用比。

算法设计技术

- 蛮力（穷举）——Brute-force or exhaustive search
- 分治策略——Divide and conquer
- 动态规划——Dynamic programming
- 贪心法——The greedy method
- 线性规划——Linear programming
- 问题规约——Reduction
- 搜索算法——Search and enumeration
  - 回溯（backtracking）、分支限界（branch and bound）
  - 随机算法（randomized algorithm）、启发式算法（heuristic algorithms）
- ……

算法的伪码描述

- 算法名字
- 算法输入和输出
- 算法描述
  - 赋值语句：←
  - 分支语句：if … then … [else …]
  - 循环语句：while， for，repeat until
  - ~~转向语句：goto~~
  - 输出语句：return
  - 调用：直接写过程的名字
- 算法注释：//…

Algorithm 1 An algorithm with caption

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ ▷ This is a comment
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

source: overleaf.com

伪代码例子：求最大公约数

算法 **Euclid(*m*,*n*)**

输入：非负整数 *m*, *n*，其中*m*与*n*不全为0

输出：*m* 与 *n* 的最大公约数

```
1. while m > 0 do
2.       r ← n mod m
3.       n ← m
4.       m ← r
5. return n
```

伪代码例子：顺序检索

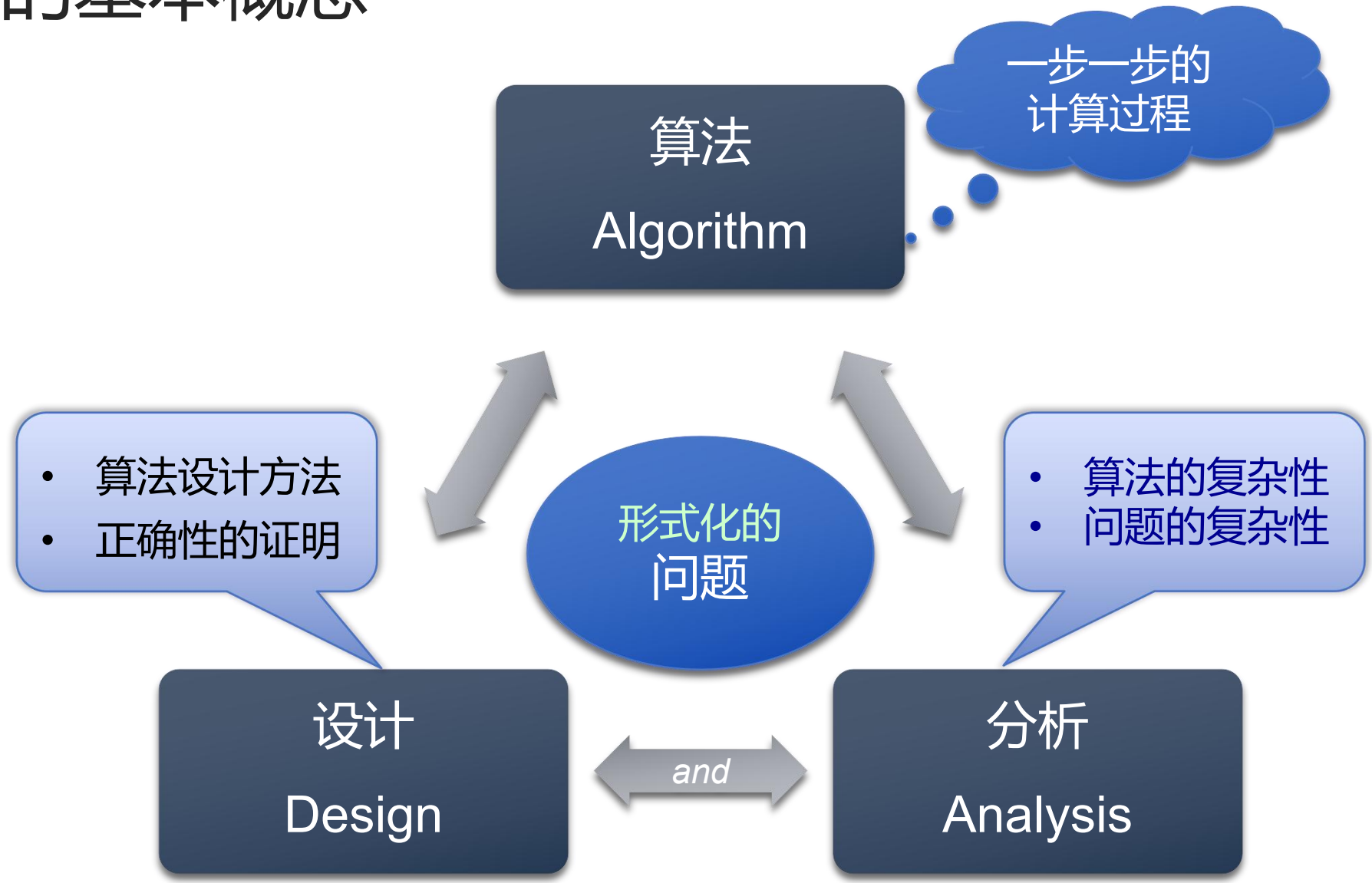
算法 Search(L, x)

输入：数组$L[1..n]$，其元素按照从小到大排列，数 x.

输出：若 x 在L中，输出 x 的位置下标 j；否则输出0.

```
1.   j←1
2.   while j≤n and x>L[j] do
3.       j←j+1
4.   if j>n or x<L[j] then
5.       j←0
6.   return j
```

算法的基本概念


算法
Algorithm
一步一步的
计算过程
• 算法设计方法
• 正确性的证明
形式化的
问题
• 算法的复杂性
• 问题的复杂性
设计
Design
and
分析
Analysis

Polya's Problem-Solving Algorithm

1. Understanding the problem（问题形式化）
2. Devising a plan（设计）
3. Carrying out the plan（算法）
4. Looking back（分析）

Polya, "How to Solve It," 1957.

问题的形式化描述

"Science is the solution of problems whose exact formulations are known only after they are solved". - Yair Censor

问题的形式化描述

- 若干著名问题的形式化描述
  - 稳定匹配问题
  - 区间调度问题

稳定匹配问题的背景

- 美国1940年医院之间对医学院学生需求的竞争
  - 在医学领域，学生通常在后几年学习生涯中需要去医院实习
  - 1940年代，美国医院系统开始大规模发展，但医学院学生的数量很少
  - 医院之间的竞争导致对医学院学生需求的急剧增加
    - 于是很多医院让学生在甚至未选定专业领域时就提前实习
    - 但如果学生拒绝一个医院，往往导致医院再去找第二个学生就太迟了
      - 因为第二个人可能已经被另一个医院抢走了
    - 医院往往会设定最后申请期限，迫使学生在知晓所有机会前就做出选择
  - 市场在这种情况下是极为不稳定的
    - 由于医院未能及时给所有学生提供机会
    - 而学生也未能向所有医院提出及时申请
    - 双方都未能极大化自己的利益

形式化：稳定匹配问题（匹配与完美匹配）

- 学生集合 $S = \{s_1, s_2, \ldots, s_n\}$ 和职位集合 $H = \{h_1, h_2, \ldots, h_n\}$，以及学生-职位的所有有序对 (s, h) 构成的集合 $S \times H$，其中 $s \in S$，$h \in H$。
- 称由 $S \times H$ 中的一组有序对构成的集合 M 为一个匹配，如果 S 和 H 中的每个元素最多只出现在 M 的一个有序对中。
- 如果 S 和 H 中的每个元素恰好都只出现在 M 的某一个有序对中，则称 M 为一个完美匹配。

形式化：稳定匹配问题（偏好与不稳定因素）

- 学生和职位间存在着偏好关系。
  - 每个学生 $s \in S$ 都对所有的职位打分，如果 s 给 h 的打分高于 h'，则称 s 相对于 h' 而言更偏好 h。
  - 称基于学生 s 的打分对职位排序得到的是 s 的偏好列表。
  - 偏好列表中的得分不能相同。
  - 同样的，每个职位也给所有的学生打分。
- 在匹配 M 中，如果存在两个有序对 (s,h) 和 (s',h')，其中 s 相对于 h 更偏好 h'，而 h' 相对于 s' 更偏好 s。则称有序对 (s,h') 为相对于 M 的一个不稳定因素。很显然，(s,h') 不属于 M。

形式化：稳定匹配问题（稳定匹配）

- 问题的目标是找到学生-职位间不存在不稳定因素的录用关系。
- 如果 M 是完美的，并且不存在相对于 M 的不稳定因素，则称匹配 M 是稳定的。
- 两个问题：
  - 对于所有可能的偏好列表，是否都存在稳定匹配？
  - 对于给定的一组偏好列表，如果稳定匹配存在，是否可以把某个稳定匹配高效地构造出来？

- 形式化过程的假设
  - 学生不能兼职；职位不能增加；存在全序的偏好关系

稳定匹配问题

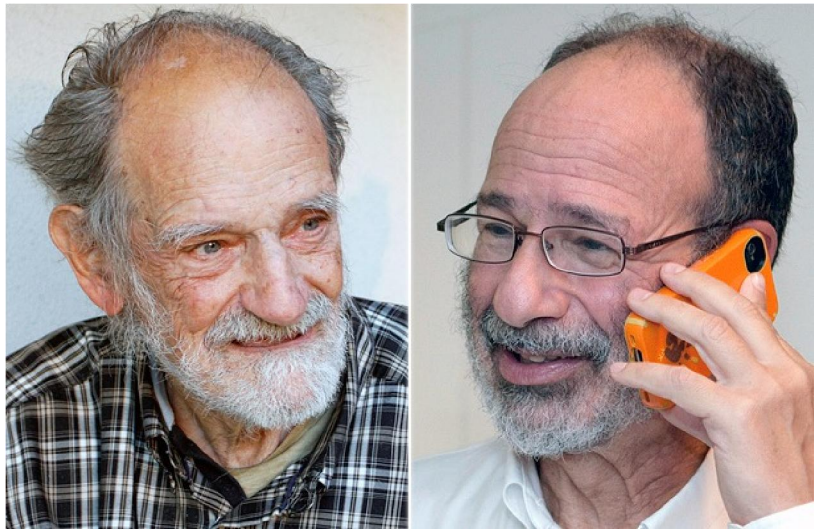
- 2012年的诺贝尔经济学奖：夏普利和罗斯，表彰他们在稳定配对和市场设计方面的理论和实践并重的贡献。

COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only q. Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the q best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept.

-- The American Mathematical Monthly, Vol. 69, No. 1 (Jan., 1962), pp. 9-15



- Lloyd Shapley：稳定匹配理论与Gale-Shapley算法
- Alvin Roth：将Gale-Shapley算法应用于医院-学生匹配、学校-学生匹配、以及器官捐赠者-接受者匹配问题

区间调度问题

- 存在某种资源，例如球场、3D打印机、电子显微镜等等。许多人都希望某段时间来使用这个资源，但资源只能在每段时间内分配给唯一的使用者。当很多人都来申请时，如何能够满足最多的人利用这个资源?

区间调度问题的形式化：例1

- 有 n 个请求，分别编号为 $1, 2, \ldots, n$ 。每个请求 i 都有一个开始时间 s_i 和一个结束时间 f_i 。并且对于所有的 i ，自然的要求 $s_i < f_i$ 成立。如果两个请求 i 和 j 的时间段不重叠，则称 i 与 j 是相容的，即，请求 i 的时间段或者早于请求 j （ $f_i \leq s_j$ ），或者晚于请求 j （ $f_j \leq s_i$ ）。
- 对于请求的子集 A ，如果任意的 $i, j \in A$ ， $i \neq j$ 都是相容的，则称 A 是相容请求子集。
- 目标：求最大相容请求子集。

区间调度问题的形式化：例2（带权重）

- 把区间调度问题更一般化
- 为每个请求区间 i 关联一个数值（或权重） $v_i > 0$，可以表示为：从满足并安排请求 i 中获得的收益。
- 目标：求总收益最大的相容请求子集。

本章小结

- 算法的定义与基本概念
  - 算法的分类
  - 算法设计技术
  - 算法的伪码描述
- 形式化问题描述
  - 求解问题的第一步
  - 两种形式化问题方式
    - 直接形式化：严格定义问题各个方面
    - 问题抽象：将问题转化为一个通用问题
  - 将问题转化为标准的常见问题
  - 通过严格的形式化描述分析问题