

程序设计实习（实验班-2024春）

课程介绍

授课教师：姜少峰

助教：冯施源 吴天意

Email: shaofeng.jiang@pku.edu.cn

个人履历

现任北京大学计算机学院 前沿计算研究中心 助理教授，博雅青年学者

香港大学博士->以色列魏茨曼科学院博士后->芬兰阿尔托大学助理教授->北大



我的研究

使用数学方法严格证明算法性能，不做实验

理论计算机科学 (Theoretical computer science), 侧重算法研究

- 数据科学基础Foundations of Data Science
- 大数据算法Algorithms for Massive Datasets

我的相关编程经验

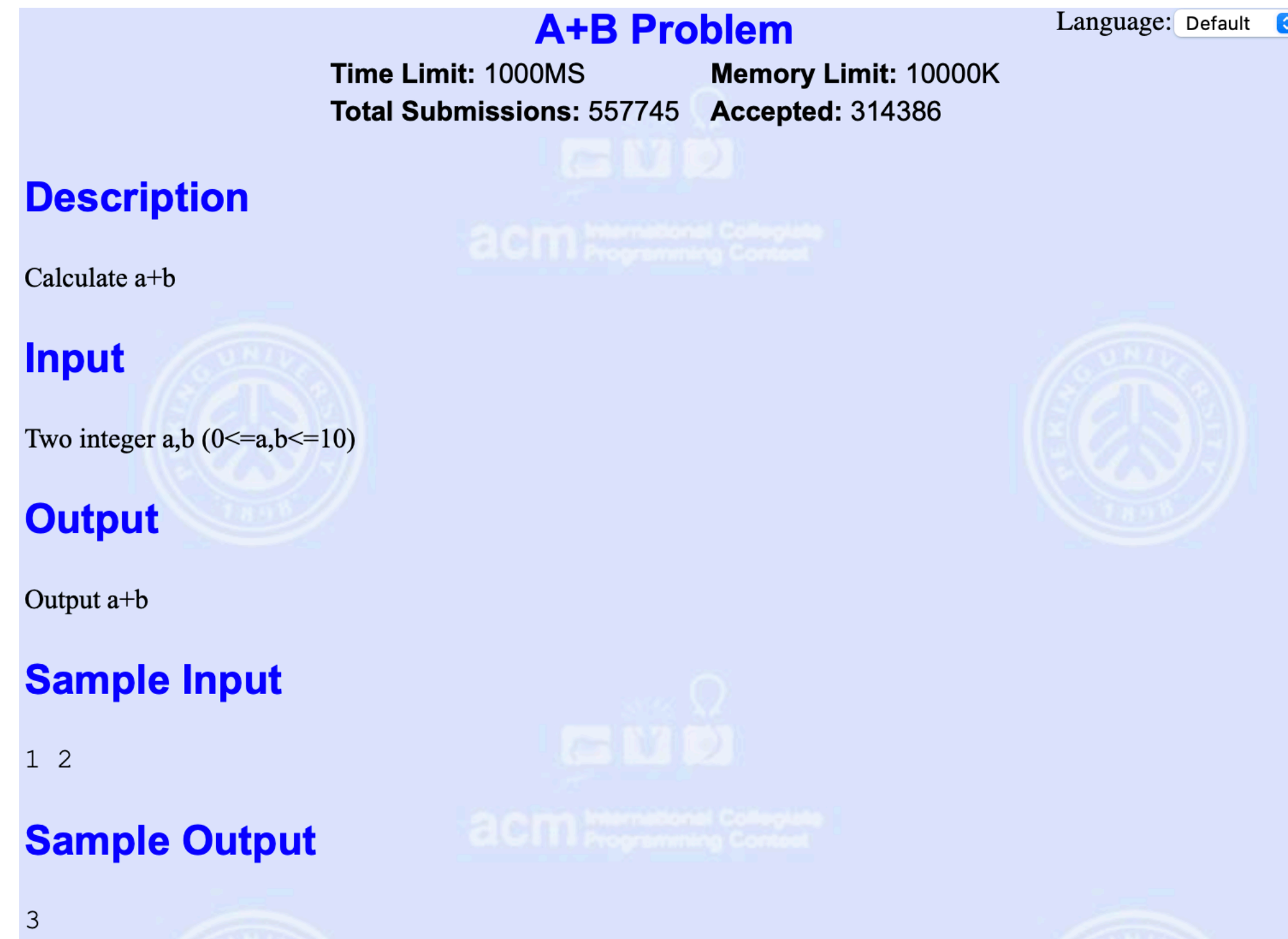
- 本科期间ACM-ICPC regional金奖, 博士期间担任香港大学ACM-ICPC教练
- 我的研究也经常考虑实际, 实验验证提出的大数据算法的有效性

助教介绍

课程内容概述

程序设计

“程序设计”是为了解决“问题”



A+B Problem Language: Default

Time Limit: 1000MS Memory Limit: 10000K
Total Submissions: 557745 Accepted: 314386

Description

Calculate $a+b$

Input

Two integer a, b ($0 \leq a, b \leq 10$)

Output

Output $a+b$

Sample Input

```
1 2
```

Sample Output

```
3
```

第一部分：程序设计的科学角度

问题已经清晰建模为“计算问题”

- 挑战：如何设计高效算法
- 有了算法又如何实现？

算法理论研究的核心

尤其关注实际效果，很多挑战无法被算法理论涵盖

算法：“传统”与“现代”

“传统”算法设计：

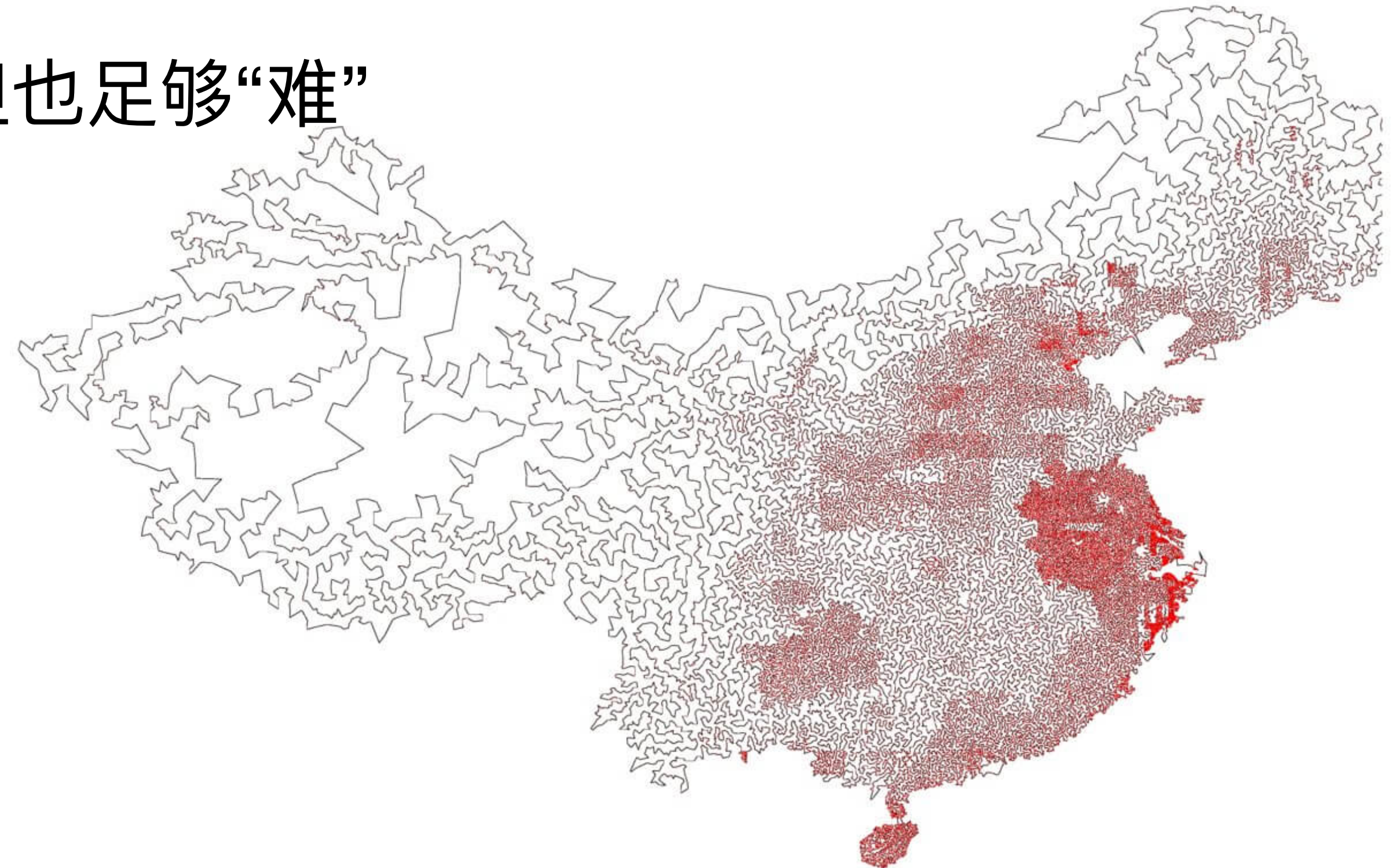
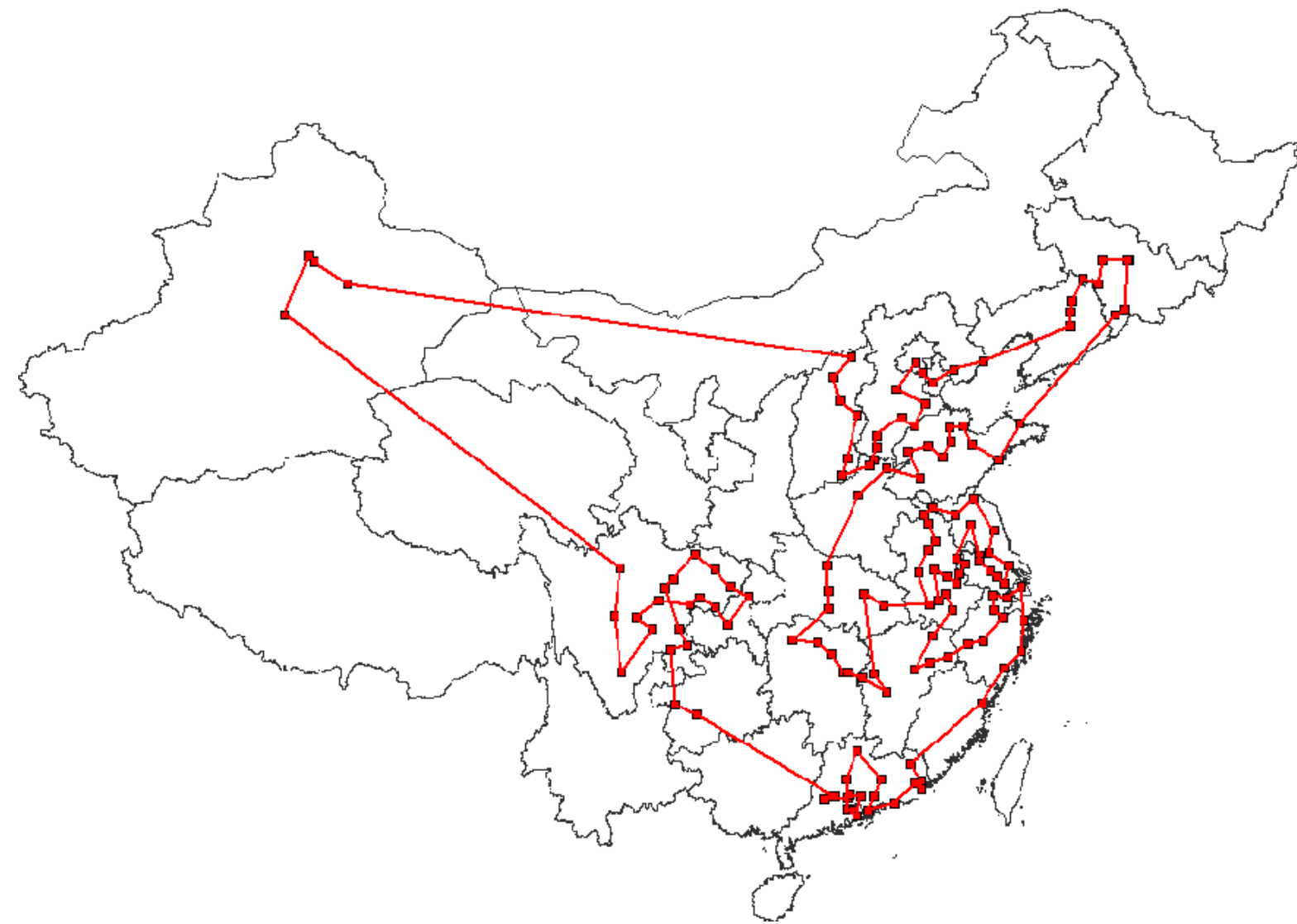
- 有效算法 = 多项式时间
- 追求“精确解”——尤其是“经典”算法/数据结构
- 例子：传统的“算法与数据结构”课程的主要内容，如排序/搜索/字典数据结构；图上的最小生成树/拓扑排序/最短路

NP-hardness

暂可以粗略理解为“无多项式时间算法”

实际中的问题通常更为复杂，NP-hard更加常见

- 但是很多问题，即便是“简单”的问题，都可能是NP-hard
- TSP是一个典型：足够“简单”，但也足够“难”



近似算法

- “近似算法”研究领域：探索NP-hard问题在多项式时间内可以近似到多么精确
- 另外，即使是本来存在多项式时间算法的问题，允许近似解可潜在改进复杂度

如何衡量近似算法：近似比

最大化问题需要对应修改

对于优化（最小化）问题：

最小化问题的近似比在 $[1, \infty)$ 范围，

- 一般采用近似比衡量解的质量：最坏情况下的 $\frac{\text{ALG}}{\text{OPT}}$ （这是一种相对误差）

- 形式化：
$$\max_I \frac{\text{ALG}(I)}{\text{OPT}(I)}$$

- 称ALG是 α -近似的，如果对任何input I ，都有 $\text{ALG}(I) \leq \alpha \cdot \text{OPT}(I)$

这个比值通常是通过分析论证得到

一般无法从数据验证，因为 I 可以有无穷种取法

其他误差衡量？

- 绝对误差： $|\text{ALG} - \text{OPT}| \leq \epsilon$
 - 当OPT较小时更弱： 极端情况 $\text{OPT} = 0$
 - 对于具有下界的问题更适用（比如对人类体验来说，精度小于1毫米没意义）
- 两种的折衷： $\text{ALG} \leq \alpha \text{OPT} + \beta$
- 但相对误差是研究的主流

考虑近似解对解决问题是否有意义？

实际应用中，通常“**没有必要**”追求精确解：

- 一个实际问题是需要建模成计算问题的，这一步建模本身就是一种近似
- 建模后，数据的采集也不是精确的过程，常混有一定噪音
- 因此，**算法即使得到精确解也依然是某种近似**

另外，算法设计时考虑的是**最坏情况**的近似比，但在**实际数据**上经常表现更好

举例：TSP问题的发展

- 近似到常数也是NP-hard
- 2-近似：最小生成树（MST）
- 1.5-近似（Christofides 1976）：先MST再在奇数度顶点上做最小权匹配
- $1.5 - 10^{-36}$ -近似 ([Karlin, Klein, Gharan 2021](#)) 91 pages...

举例：线性时间近似点集直径

- 输入：n个d维平面上的点 x_1, \dots, x_n
- 输出：对于直径 $\max_{i,j} \|x_i - x_j\|_2$ 的估计
精确解需要 $O(n^2)$ 时间
- 2-近似算法：选任一个点作为起点，例如 x_1 ，返回 x_1 到距离 x_1 最远点的距离

随机性

类似地，另一种意义上的“近似”是引入“随机性”

- 随机算法：算法自身有随机性，即使对于确定性的输入也得到随机的输出
- 随机算法的性能保证：
 - 蒙特卡洛：有一定概率出错，不出错时有很好的性能保证
 - 拉斯维加斯：一定不出错，但是运行时间等资源消耗可以是随机的

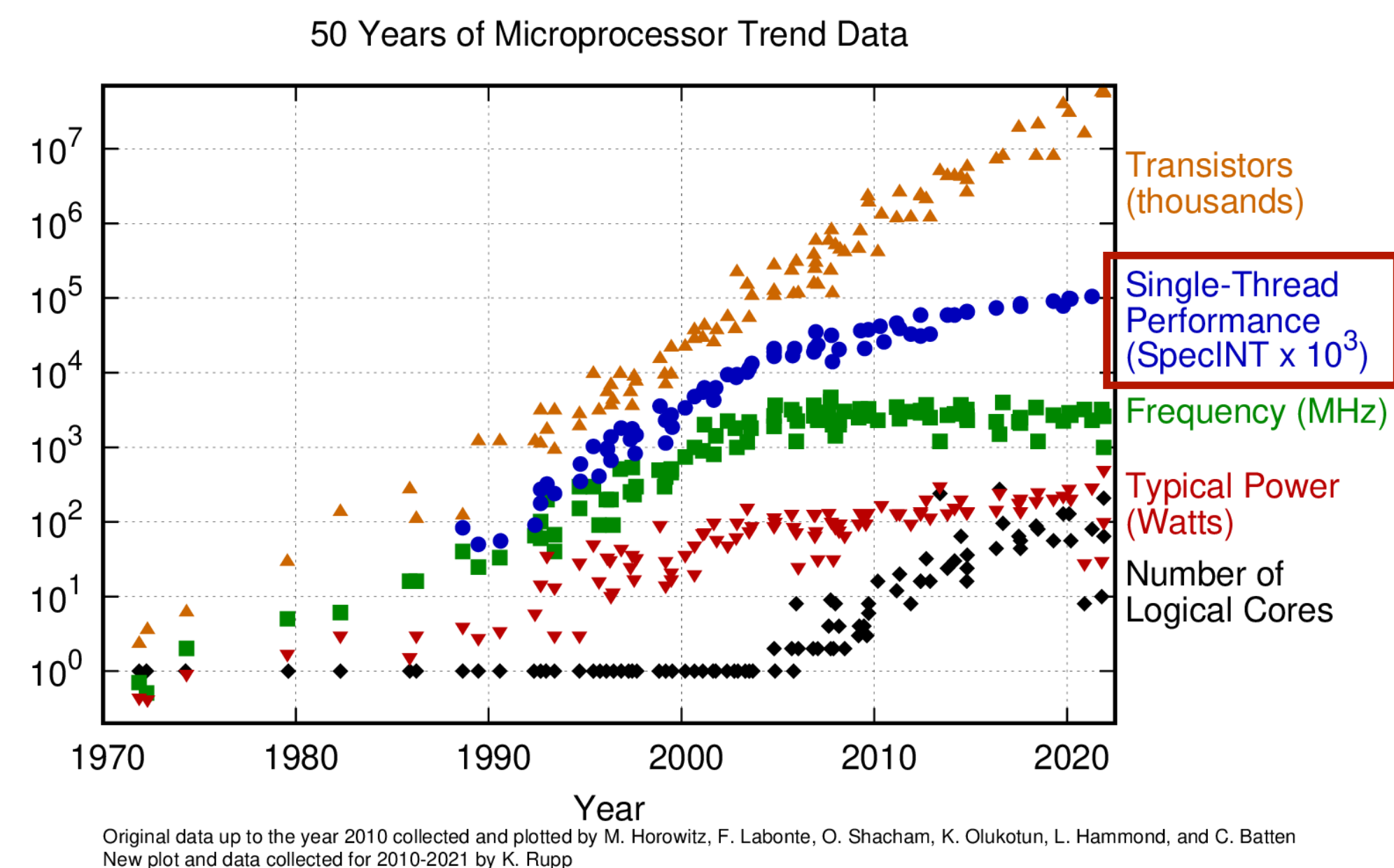
相对确定性算法：由于保证变弱，通常换来大幅度的性能提升

近似 + 随机
成为现代算法设计的主流范式

大数据的时代

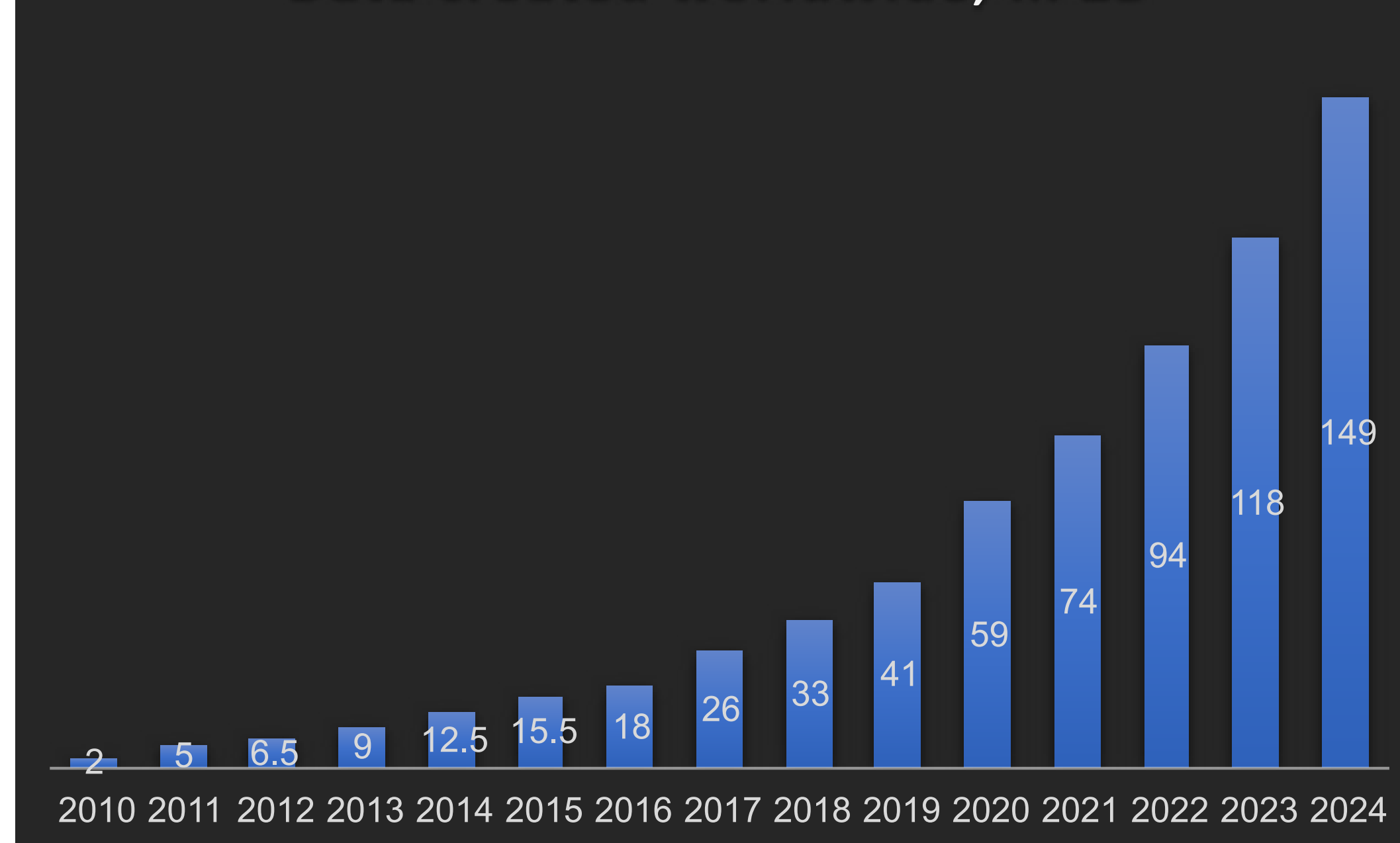
甚至线性时间都不够高效

有效算法 = 多项式时间算法？



单个CPU：每秒 10^{10} 浮点运算；超级计算机：每秒 10^{18} 浮点运算

Data created worldwide, in ZB

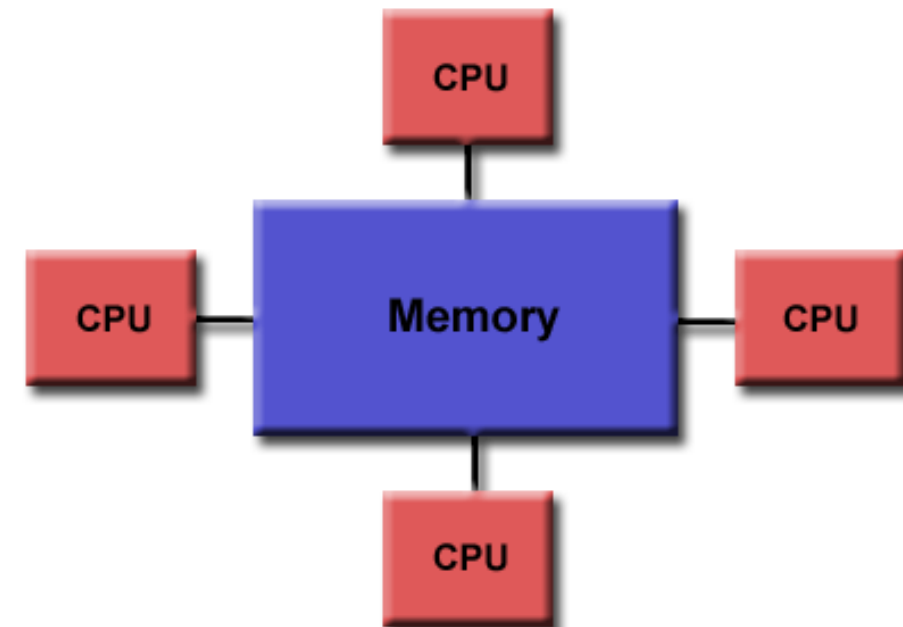


即使仅扫描一遍数据也不切实际；单CPU性能增速跟不上数据

“亚线性”计算模型

- 数据流算法：扫一遍数据流，用较小的空间计算
 - 路由器/物联网设备上直接做数据分析

- 并行算法



- 亚线性时间算法

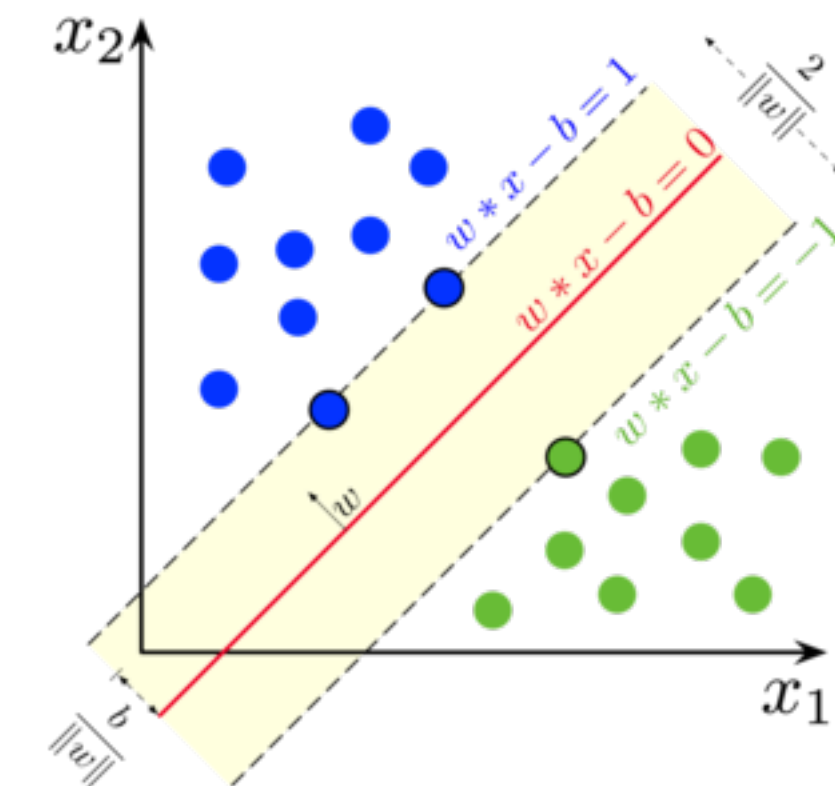
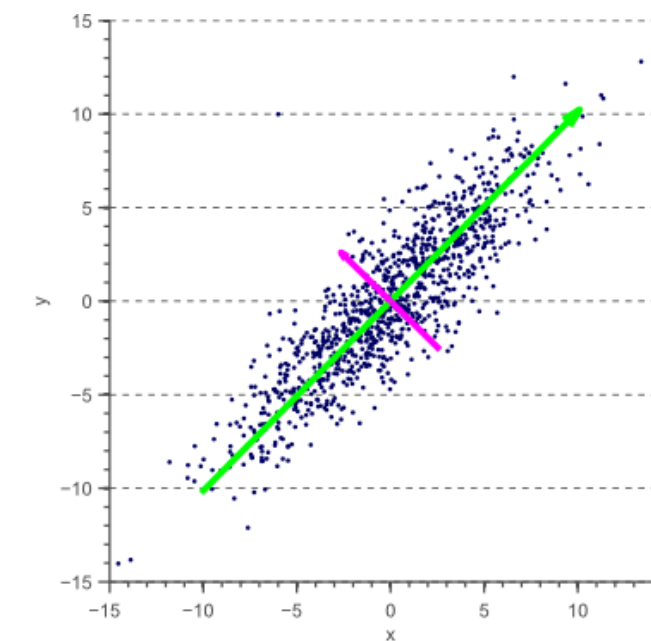
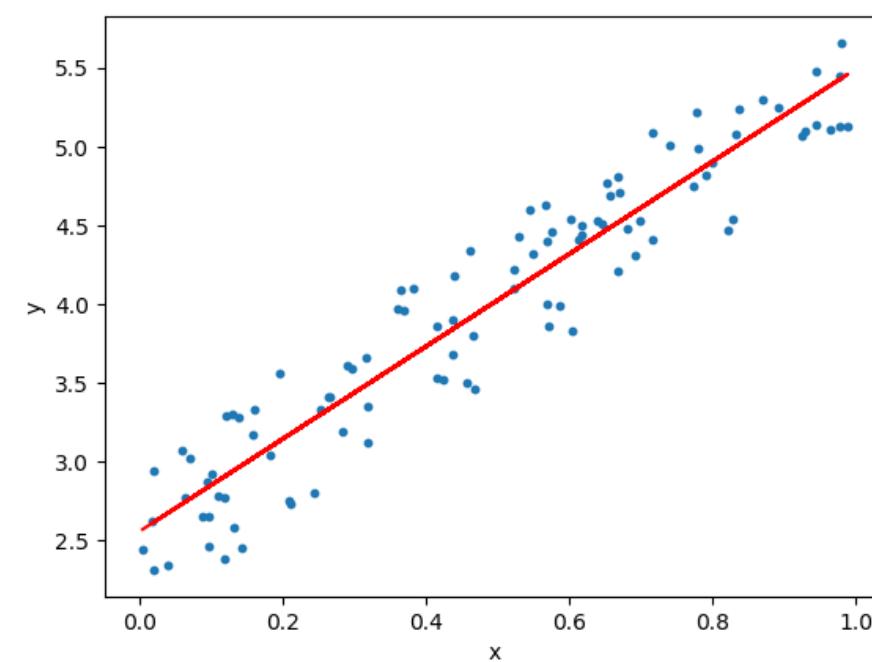
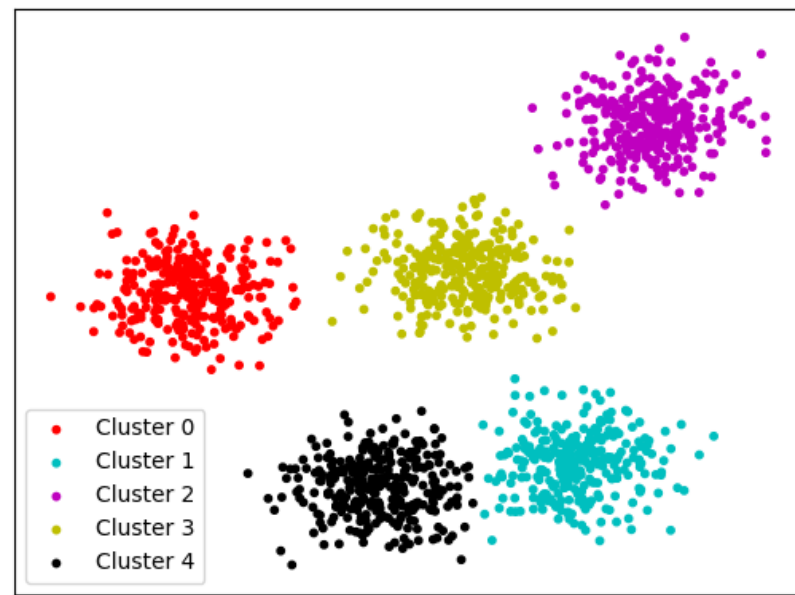
MapReduce框架是分布式计算
最流行的软件实现

- 如何不遍历整个数据库，而是使用较少查询来进行计算？

数据科学

脱胎于统计，使用计算的方法来“理解”大数据的一些总体性质

- 聚类，回归，主成分分析，分类...



这些问题的算法，尤其是大数据下的算法，成为了新的研究焦点

Introduction, *Foundations of Data Science*

Blum, Hopcroft and Kannan

- Computer science ... began in the 1960's.
- In the 1970's, algorithms were added as an important component... The emphasis was on **making computers useful**.
- Today, fundamental change takes place ... the focus is more on **a wealth of applications**.

The enhanced ability to observe, collect, and store data in the **natural sciences**, in **commerce**, and in other fields calls for a change in our understanding of data and **how to handle it in the modern setting**. The emergence of the **web** and **social networks** as central aspects of daily life presents both opportunities and challenges.

- ... this book covers the theory we expect to be **useful in the next 40 years**, just as an understanding of automata theory, algorithms, and related topics gave students an advantage in the last 40 years.

大数据 + 数据科学
是现代算法设计的新焦点

第一部分大纲：现代算法初探

- 随机算法及其实现
- 哈希方法及其在大数据上的应用
- 常见相似度/距离度量及其有效算法
- 几何近似算法
- 降维
- 压缩感知 & 数学优化

可选覆盖：

- 亚线性算法选讲
- 其他：聚类，社交网络等专题

第二部分： 程序设计的工程角度

实际问题经常是这样的：

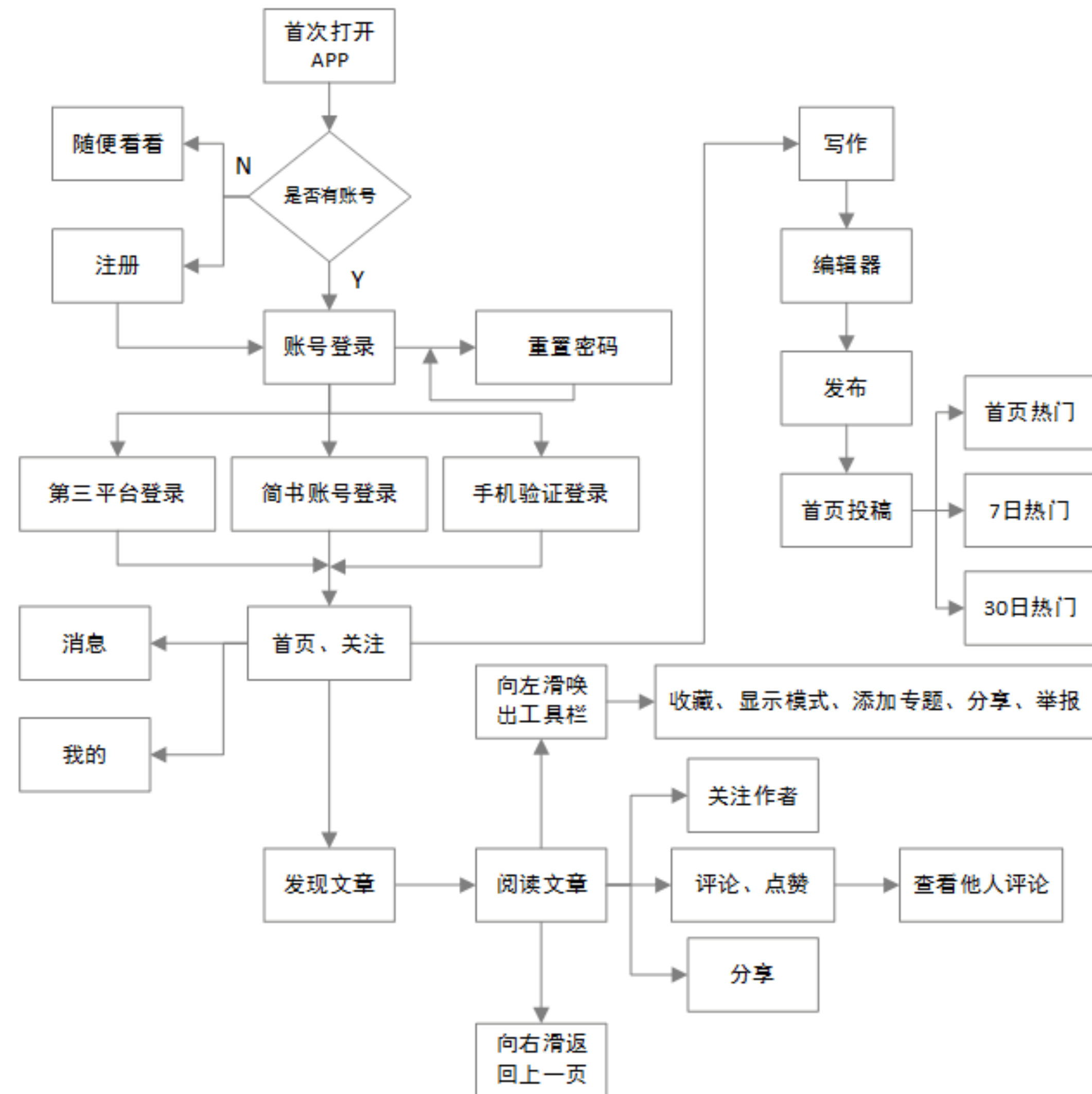
请为P大学开发一套“学生信息管理系统”

请为A公司开发一个网站

有个idea想做个手机App

...

- 现在关键问题还不是计算的复杂性，而是**建模**





若干难点

- 系统复杂且需求经常会改变
 - 如何有效梳理问题逻辑？有改变时如何有效应对？
- 可大规模协作
 - 每个人可以只开发自己的一小块而不必完全掌握大项目的全貌
- 需求/功能经常会重复
 - 如何以较小的代价实现复用？

面向对象

- 面向对象编程就是为上述需求而生的
- 总体上要求：程序层次化、模块化、低耦合度
- 是现代软件开发广为采用的范式

《神奇的数字：7±2：我们信息加工能力的局限》
是美国认知心理学家乔治·A·米勒的一篇重要论文，
1956年发表于《心理学评论》。

他注意到年轻人的记忆广度大约为7个单位（阿拉伯数字、字母、单词或其他单位），称为组块。后来的研究显示广度与组块的类别有关，例如阿拉伯数字为7个，字母为6个，单词为5个，而较长词汇的记忆广度低于较短词汇的记忆广度。

面向对象的原则很好的契合了人类的局限性；
事实上，“面向对象”甚至是数学论文写证明一般采取的方法

程序设计范式之间的关系

- 还有其他范式，比如函数式编程
- 但不同范式从解决问题的“能力”上没有“本质区别”
 - 不论什么编程语言，能力都是图灵机
 - 任何X语言写的程序的功能也可以用Y语言实现
 - 然而实现的难度可以有很大差别：汇编 vs Python脚本
- 区别在于：采用合适的范式可以更少bug，更少努力来处理复杂的需求

第二部分大纲：面向对象编程（C++）

- C++的面向对象语言特性
 - 类/对象
 - 继承/多态
 - 模板类/模板方法，STL等
- 设计模式选讲
- 案例选讲

课程安排

选课指导

目的是让这部分有基础的同学依然可以学到新东西，
而又不与后面的课程有过大重合

- 主要面向竞赛生/有良好编程基础的同学
- 这门课的风格：非传统，前沿内容未必能组织成教科书一样的知识体系
 - “展示”：选择重要idea展示出来，通过编程、解题来深化理解
 - 开阔眼界，对未来更深入的学习有启发作用
- 内容上C++部分与普通班有一些重叠，“现代算法”部分则为全新
- 没有期中考试，对于C++面向对象也不在考试中考察（只有大作业）

课业负担

预计~20个题目

- 每1 - 2次课后布置编程小作业/实验报告，每个作业1 - 3分不等（平均约2分）
- 面向对象会有一个大作业（与普通班类似），约占10分
- 没有期中考试
- 期末上机考试：不专门考察面向对象，主要考察算法（经典和现代）
- 分数：平时作业50%，期末考试50%
- 每个作业题目给至少3周时间（大作业更长）

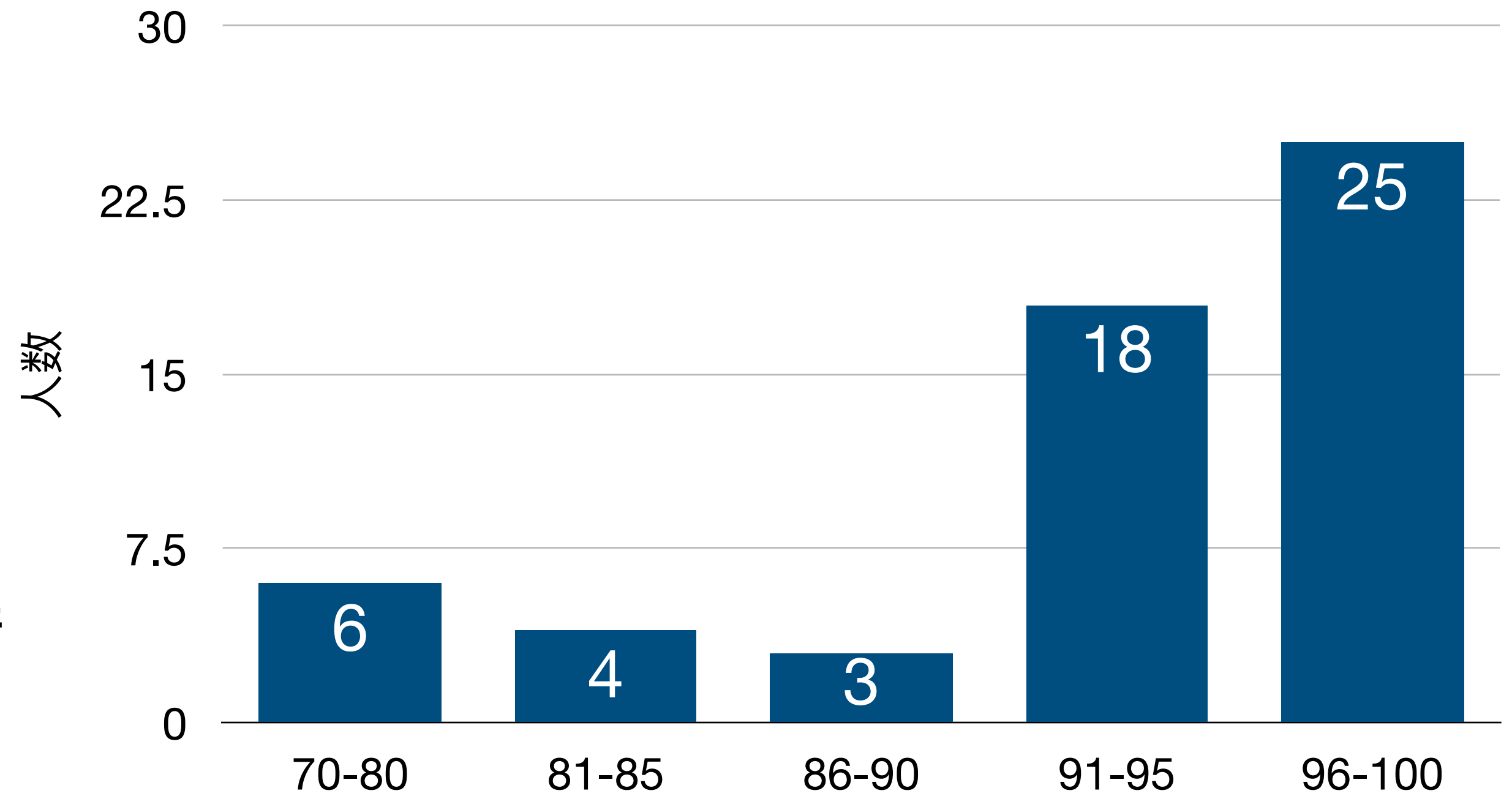
经典为主（内容上可与普通班类比）
但想得90+需要能解出一些现代算法的题目

授课计划

- 每周 周三，单周 周五上课
 - 计划算法与面向对象穿插进行
- 现代算法：介绍性为主，主要理解算法过程、会实现，以及直觉上的设计原理
 - 但也会有一些“活学活用”的作业题目

去年的一些情况

- 期末成绩：优秀率80%+
- 去年期末考试题
- 今年（拟）新加
 - 丰富作业题目
 - 面向对象部分加入更多案例讲解
 - 期末考试过多错误提交扣分



课程资料

- 课程讲义和其他阅读材料

- 教学网

- 课程网页： <https://shaofengjiang.cn/programming-course/>

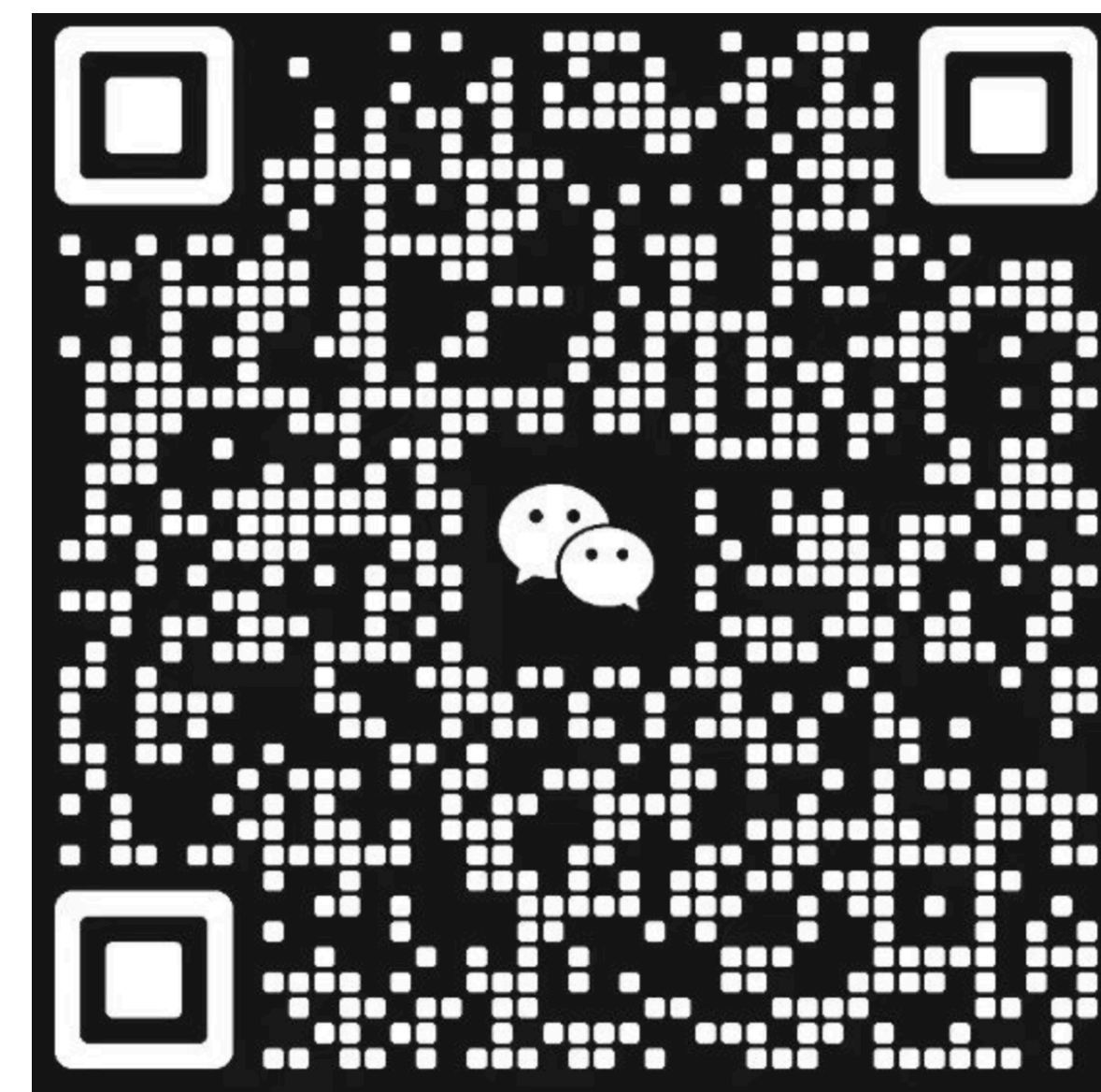
- 作业

第一次使用需要把自己加入小组；
请把自己的小组昵称改成学号

- 多数作业为编程作业，在<http://cssyb.openjudge.cn>在线提交和实时评测

- 少数为实验报告，请提交到教学网（喜欢手写的同学也请拍照提交）

答疑与联系方式



- 助教
 - 冯施源、吴天意：图灵班在读，去年选修本课程的优秀学生
- 答疑：Email、微信群、上机课（无考勤、（暂）无额外内容，主要是答疑）
shaofeng.jiang@pku.edu.cn wuty@stu.pku.edu.cn 2200013060@stu.pku.edu.cn
- 上机课时间地点待定