Design and Analysis of Algorithms
Course Information

Two ideas changes the world!

Typography

1448, German, Johann Guternberg: print books by putting together movable metallic pieces

- literacy spread $\Rightarrow$ Dark Ages ended $\Rightarrow$ human intellect was liberated $\Rightarrow$ science and technology triumphed $\Rightarrow$ Industrial Revolution happened

imagine a world in which only an elite could read lines

Typography

1448, German, Johann Guternberg: print books by putting together movable metallic pieces

- literacy spread $\Rightarrow$ Dark Ages ended $\Rightarrow$ human intellect was liberated $\Rightarrow$ science and technology triumphed $\Rightarrow$ Industrial Revolution happened

imagine a world in which only an elite could read lines

But others insists that the key development was not typography, but *algorithm*

Algorithm

Origin: decimal system

- 10 symbols $\Rightarrow$ even large numbers can be expressed compactly (invented in India around AD 600)
- basic methods for add, mul, div, even square roots and π (9th century, Arabic, Baghdad, Al-Khwarizmi)

These procedures are precise, unambiguous, mechanical, efficient, correct $\rightsquigarrow$ algorithms

Algorithm

Origin: decimal system

- 10 symbols $\Rightarrow$ even large numbers can be expressed compactly (invented in India around AD 600)
- basic methods for add, mul, div, even square roots and π (9th century, Arabic, Baghdad, Al-Khwarizmi)

These procedures are precise, unambiguous, mechanical, efficient, correct $\rightsquigarrow$ algorithms

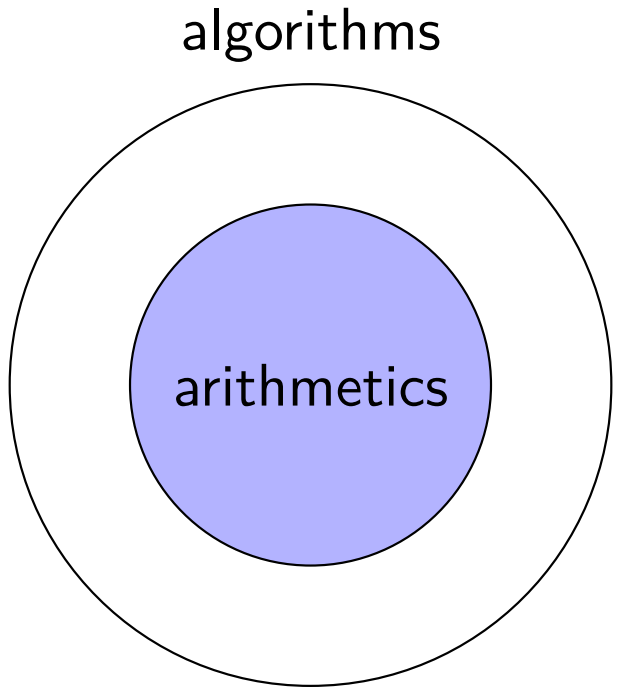
Back to 1448: imaging how to add/mul two Roman numbers: MCDXLVIII+DCCCXII? fingers are not enough

Algorithm Etymology

Spread to Europe around 12th century $\rightarrow$ plays an enormous role in Western civilization (science and technology, commerce and industry)

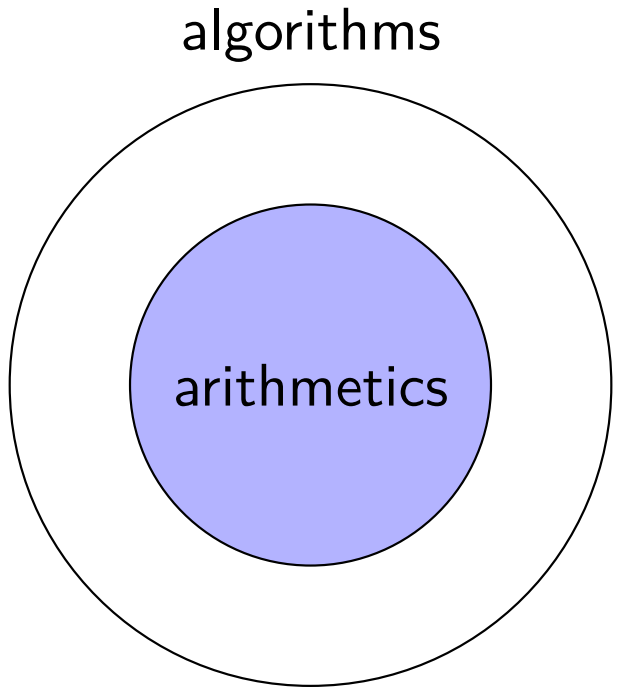
Algorithm Etymology

Spread to Europe around 12th century $\rightarrow$ plays an enormous role in Western civilization (science and technology, commerce and industry)



Algorithm Etymology

Spread to Europe around 12th century $\rightarrow$ plays an enormous role in Western civilization (science and technology, commerce and industry)



Computer era: evolute to embody the positional system and arithmetic unit $\rightsquigarrow$ scientists develop algorithms for all kinds of problems — ultimately change the world

Why Study Algorithms

Internet. Web search, packet routing, distributed file sharing, ...

Computer graphics. movies, video games, virtual reality, ...

Multimedia. MP3, JPG, DivX, HDTV ...

Artificial Intelligence. face recognition, PS, more AI algorithms

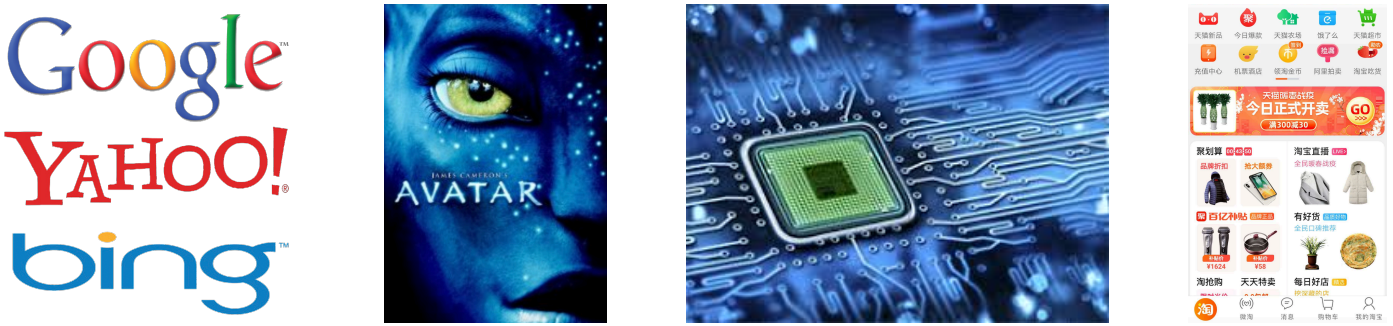
Social networks. recommendations, news feeds, advertisements, ...

Computers. circuit layout, databases, caching, compilers, ...

Biology. human genome project, protein folding, ...

Physics. N-body simulation, particle collision simulation, ...

Importance: Look around you



Algorithms interesting and useful.
We live in the algorithm world!

Cryptographic Algorithms

Most algorithms focus on solving problems efficiently

- make us live in a better world

Good man and bad man live in the same world

- good man need *cryptographic algorithms* to protect them from bad man: enjoying the benefits in a secure manner

Cryptographic algorithms ensure there is no *efficient algorithms* against some problems

Fundamental of Computer Sciences

Algorithm design and analysis

- widespread applications
- fundamental and core part of computer science

Fundamental of Computer Sciences

Algorithm design and analysis

- widespread applications
- fundamental and core part of computer science

Turing Awards: (1966-2005) 50 persons win Turing awards

- algorithm design: 10
- computing and complexity theory: 7

Fundamental of Computer Sciences

Algorithm design and analysis

- widespread applications
- fundamental and core part of computer science

Turing Awards: (1966-2005) 50 persons win Turing awards

- algorithm design: 10
- computing and complexity theory: 7

$\mathcal{P} = \mathcal{NP}$ is one of the most important questions in this century

Contents of This Course

Preliminary about algorithms

- mathematical background
- data structure

Contents of This Course

Preliminary about algorithms

- mathematical background
- data structure

Design paradigm and analysis methods

- divide-and-conquer
- dynamic programming
- greedy strategy
- backtracking and trimming technique

Contents of This Course

Preliminary about algorithms

- mathematical background
- data structure

Design paradigm and analysis methods

- divide-and-conquer
- dynamic programming
- greedy strategy
- backtracking and trimming technique

Advanced topics

- complexity theory
- randomized algorithms
- approximate algorithms

Goal of this Course

Emphasize critical thinking: believe your own reasoning, do not easily repeat or follow

Goal of this Course

Emphasize critical thinking: believe your own reasoning, do not easily repeat or follow

Master problem-solving method

1. abstract and formalize problem
2. solve it efficiently and correctly using algorithms
3. prove its correctness

Goal of this Course

Emphasize critical thinking: believe your own reasoning, do not easily repeat or follow

Master problem-solving method

1. abstract and formalize problem
2. solve it efficiently and correctly using algorithms
3. prove its correctness

Develop rigorous analysis skills

- know how to evaluate the performance of algorithms

Goal of this Course

Emphasize critical thinking: believe your own reasoning, do not easily repeat or follow

Master problem-solving method

1. abstract and formalize problem
2. solve it efficiently and correctly using algorithms
3. prove its correctness

Develop rigorous analysis skills

- know how to evaluate the performance of algorithms

Tips

- theory: think rigorously and keep ask yourself why
- practice: implement algorithms using your favorite programming languages

Course Website

Syllabus

Office hours

Assignments

Lecture slides

...

https://yuchen1024.github.io/teaching/SDU/2020_spring_algorithms/2020_spring_algorithms.html

References and Resources

Textbooks

- Algorithms. Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani. The McGraw-Hill Companies,2008.
- 算法设计与分析 (第二版). 屈婉玲, 刘田, 张立昂, 王捍贫. 清华大学出版社,2016.2.

Online resources

- leetcode
- online judging system: ZOJ, POJ

Assignments

- electronic submission
- graded for correctness, clarity, conciseness, rigor, and efficiency
- recommendation: using LaTeX template for writing solutions
- no collaboration, no Google

课程回顾 (1/4): 引论与算法分析技术

算法的起源、发展与学习算法的意义

算法分析: 时、空复杂度为指标定量分析算法优劣

- 算法的伪码表示
- 最优、最坏、平均复杂度 (以排序算法为例)
- 函数的渐近复杂度
- 函数阶的计算: 递归方程求解技术 (迭代、递归树、主定理)
- 常见的函数阶

课程回顾 (2/4): 算法设计技术

分治

- 典型问题: 排序选择类、计算几何类、数值计算类 (快速幂、整数乘、矩阵乘、多项式乘)
- 设计: 子问题的划分与子问题解的合并
- 分析: 递归方程求解

贪心

- 典型问题 (最优子结构): 最优调度、最优装载、Huffman Coding、单源最短路径、最小生成树
- 设计: 选择贪心策略 (试错: 构造反例说明贪心策略非最优)
- 证明: 数学归纳法证明贪心解是最优解 (对问题的规模或算法的步骤进行归纳)

课程回顾 (3/4): 算法设计技术

动态规划

- 典型问题 (最优子结构 + 重叠子问题): 字符串类 (最长递增子序列、最大公共子序列、最小编辑距离)、矩阵链、最大子段和、背包类
- 设计: 找出 DAG, 列出递推方程, 设计备忘录和标记函数
- 分析: 根据递推方程或者备忘录分析时空复杂度

分治、贪心、动态规划设计范式的适用范围: 原始问题具有 (最优) 子结构性质 $\Rightarrow$ 原始问题可分解为子问题

搜索技术: 求解优化类问题的通用技术 (将解空间建模为树)

- 暴力搜索
- 回溯 (Domino 性质: 根据优化类问题的约束条件及时停止局部搜索并回溯)
- 分支定界 (引入界和估计函数, 在约束条件的基础上加入更精细的判断)

课程回顾 (4/4): 复杂性理论初步与随机算法

复杂性理论

- 计算模型: Turing Machine (deterministic poly-time and non-deterministic poly-time)
- 重要的复杂性类: $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-hard, $\mathcal{NP}$-complete
- 更多的复杂性类: $\mathcal{BPP}$ (降低错误的方法和为什么能降低、衍生复杂性类、与其他复杂性类的关系)

随机算法初步

- 概率论初步: 期望线性性质的应用
- Miller-Rabin 素性检测算法
- Schwartz-Zippel 引理及其应用: 多项式恒等测试、矩阵运算结果测试

Exams

总成绩 $=0.2\times$ 平时成绩 $+0.2\times$ 课后作业 $+0.6\times$ 考试成绩

考试日期: 2020.06.24 上午 8:30 ~ 10:30

考试形式: 线上闭卷 (不允许查阅资料和搜索)

- 考试开始前 5 分钟: 微信群中推送试卷电子版
- 腾讯会议打开摄像头进行答题
- 考试完成延长 5 分钟对试卷进行拍照, 将电子版发送至助教老师邮箱

Figure: 结合同学意见和学校要求

考试题型与范围

选择题 (送分题): $5 \times 2 = 10$

- 考察渐近复杂度、经典算法和复杂性的常识

简答题 (又是送分题): $15 \times 2 = 30$

- 渐近复杂度或递归方程求解
- 复杂性类的基本概念

算法设计与分析题 (还是送分题): $20 \times 3 = 60$

- 分治、贪心、动态规划

评判标准: 根据算法的正确性和效率评分.

解答需包含以下四部分: (1) 算法思想; (2) 分治算法需写出递归方程, 贪心法需给出正确性证明, 动态规划算法需写出递推方程和标记函数; (3) 伪代码; (4) 时间复杂度分析

回溯和随机算法不做重点考察