

Algorithms

Xiyuan Yang

2025.10.05

SJTU Semester 2.1 Algorithms: Design and Analysis

Contents

| | |
|------------------------------------|----------|
| 1. Lec1 Introduction | 1 |
| 1.1. Problems and Computation | 1 |
| 1.2. Algorithm | 2 |
| 1.2.1. Definition | 2 |
| 1.2.2. Pseudocode | 2 |
| 1.3. RoadMap | 3 |
| 1.3.1. Divide and Conquer | 3 |
| 1.3.2. Greedy algorithms | 3 |
| 1.3.3. Dynamic Programming | 3 |
| 1.3.4. Back Tracking | 3 |
| 1.3.5. Heuristic Algorithms | 4 |
| 1.4. Correctness of the algorithms | 4 |
| 1.5. 算法正确性证明 | 4 |
| 1.5.1. 数学归纳法 | 4 |
| 1.6. Complexity & Efficiency | 4 |
| 1.7. Asymptotic Notation | 5 |
| 1.8. Model of Computation | 5 |
| 1.9. 系统字节数 | 5 |
| 1.10. 算法时间复杂度证明 | 5 |
| 2. Divide and Conquer | 6 |
| 3. Computational Complexity | 6 |
| 4. Future of Algorithms | 6 |
| 5. Conclusion | 6 |

§1. Lec1 Introduction

§1.1. Problems and Computation

The basis of **AI**:

- Search
- Learning

Definition 1.1.1 (计算问题).

- Given a **input**, I , $x \in I$
- output, O , $y \in O$
- relation: $f : x \rightarrow y$: use the algorithm!
 - we have some boundaries: (s.t.)

Definition 1.1.2 (Problem Domain).

The set of all the problems. $\langle I, O, f \rangle$

Definition 1.1.3 (Problem Instance).

one simple case in the problem domain $\langle I, O, f, x \rangle$

What is algorithm:

- a piece of code
- handling the mapping from x to y

§1.2. Algorithm

§1.2.1. Definition

Definition 1.2.1.1 (Algorithm).

- Fixed length code
- accept input with any length (or we say it can scale up!)
- at finite time terminate

- Natural Language
- pseudocode
- Written Codes

For example, birthday matching problems.

§1.2.2. Pseudocode

- if else end
- foreach end
- init data structure (Use \leftarrow)

```

1 do something
2 do something else
3 while still something to do
4   | do even more
5   | if not done yet then
6   |   | wait a bit
7   |   | resume working
8   | else
9   |   | go home
10  | end
11 end
```

```

1 function BinarySearch(A, x)
```

```
2 low ← 0
3 high ← A.length - 1
4 while low ≤ high
5   mid ← low + floor((high - low) / 2)
6   if A[mid] == x then
7     | return mid
8   else if A[mid] < x then
9     | low ← mid + 1
10  else
11    | high ← mid - 1
12  end
13 end
14 return -1
15 end
```

§1.3. RoadMap

Recordings (RoadMaps).

- 基本概念，算法复杂度和正确性分析
- 分治法
- 排序算法
- 哈希表
- 贪心算法
- 动态规划
- 图搜索算法
- 回溯法
- 分支界限
- 启发式算法

§1.3.1. Divide and Conquer

- Like the merge sort and recursion.
- Split bigger problems into smaller ones.

§1.3.2. Greedy algorithms

- making the **locally optimal choice** at each stage with the hope of finding a global optimum.

§1.3.3. Dynamic Programming

- 最优子结构 optimal sub-structure
 - 这也是和 divide and conquer 算法之间最显著的区别
- 重叠的子问题 overlapping sub-problems
 - 这保证了动态规划的重复利用的部分，也是动态规划的高效性所在（不再重复计算）

§1.3.4. Back Tracking

- a brute-force searching algorithms with pruning.

- Like the DFS algorithm
 - N Queens Problems

§1.3.5. Heuristic Algorithms

- when encountering large solve space
- optimize (or tradeoff) for traditional searching algorithms.
- great for NP-hard problems.

§1.4. Correctness of the algorithms

给定输入-输出组 (x, y) , 给出一个 judger function , 返回一个布尔值是否正确。

Recordings (Judger Functions).

- 一般而言, 算法求解的复杂度是更关注的部分, 算法求解的复杂度会高于算法验证正确性的复杂度
- 但是 Evaluation is also important!

§1.5. 算法正确性证明

§1.5.1. 数学归纳法

归纳法将问题的结构简化为了两个部分的证明:

Definition 1.5.1.1 (数学归纳法).

- 基础情况的证明成立
- 递推关系的证明成立
 - 在递推关系中, 存在“假设”, 相当于多添加了一个前提条件。

Example (Birthday Example).

4.2 生日匹配算法正确性证明

证明: 生日匹配算法的正确性

归纳基础: $k = 0$, 记录中前 k 个学生不包含匹配, 算法正确报告不匹配。

归纳假设: 对于 $k = k_0$ 个学生, 如果前 k_0 个包含匹配, 算法在访问第 $k_0 + 1$ 个学生之前返回匹配。

归纳步骤: 考虑 $k = k_0 + 1$ 的情况

- 如果前 k_0 个包含匹配, 根据归纳假设, 算法已经返回匹配
- 否则前 k_0 个没有匹配, 所以如果前 $k_0 + 1$ 个有匹配, 匹配必须包含第 $k_0 + 1$ 个学生
- 然后算法直接检查第 $k_0 + 1$ 个学生的生日是否存在于前 k_0 个学生中

Figure 1: Demo for the correctness of algorithms for birthday

§1.6. Complexity & Efficiency

时间复杂度的衡量为了摆脱硬件性能的约束和影响, 在衡量算法复杂度的时候, 往往使用原子操作来代表基本的时间步:

- Number of atomic operations.
- 常数开销 $O(1)$ 的操作: 例如加减乘除
 - $O(1)$ 生万物

§1.7. Asymptotic Notation

Definition 1.7.1 (Asymptotic Notation).

- O : Upper Bound
- Ω : Lower Bound
- Θ : 紧界
 - $f(n) = \Theta(g(n))$ 表示 $f(n)$ 和 $g(n)$ 的增长速度相同。
- Polynomial Complexity: $O(n^k)$
- Exponential Complexity: $O(k^n)$
 - X-hard problems

§1.8. Model of Computation

上述 $O(1)$ 生万物的计算模型基于 WordRAM 计算模型:

- 整数运算
 - 浮点数? 理论上不是, 但是基本上是。
- 逻辑运算
- 位运算
- 内存访问 (给定地址的特定内存块的读取和写入)

上述的运算都为 $O(1)$ 的时间复杂度。

§1.9. 系统字节数

32 位系统和 64 位系统标定的是内存地址的长度

- 32 位系统: 4GB
- 64 位系统: 16EB
 - 保证给 16 EB 的内存寻址, 在 $O(1)$ 的时间复杂度进行存址

§1.10. 算法时间复杂度证明

记:

$$H_n = \sum_{i=1}^n \frac{1}{i}$$

求证:

$$H(N) = \Theta(\log N)$$

使用积分不等式:

$$\int_0^1 \frac{1}{x} dx \leq \sum_{i=1}^n \frac{1}{i} \leq \int_0^1 \frac{1}{x} dx + 1$$

§2. Divide and Conquer

§3. Computational Complexity

Lecture notes for MIT 6.006 Lecture 19: Complexity

§4. Future of Algorithms

§5. Conclusion