计算机问题求解 – 论题2-4

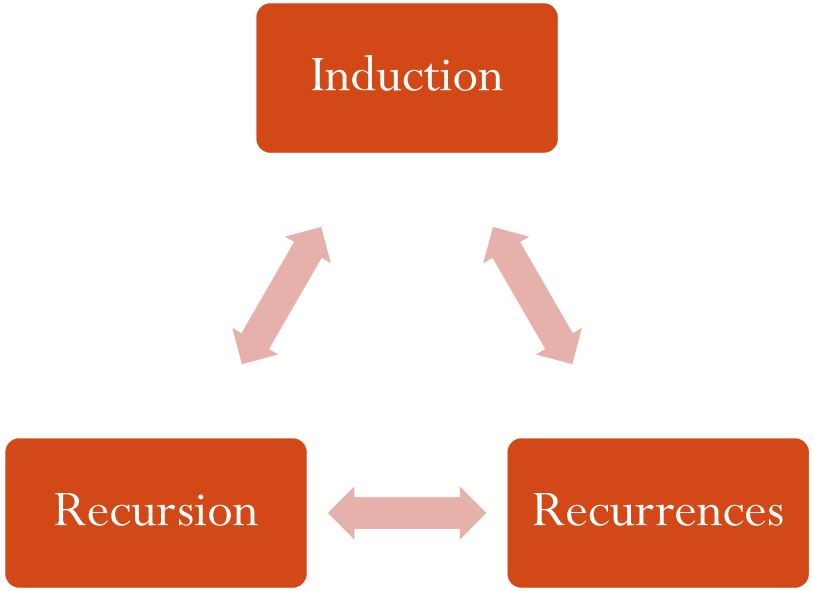
- 递归及其数学基础

课程研讨

- CS第4章4.1-4.4节

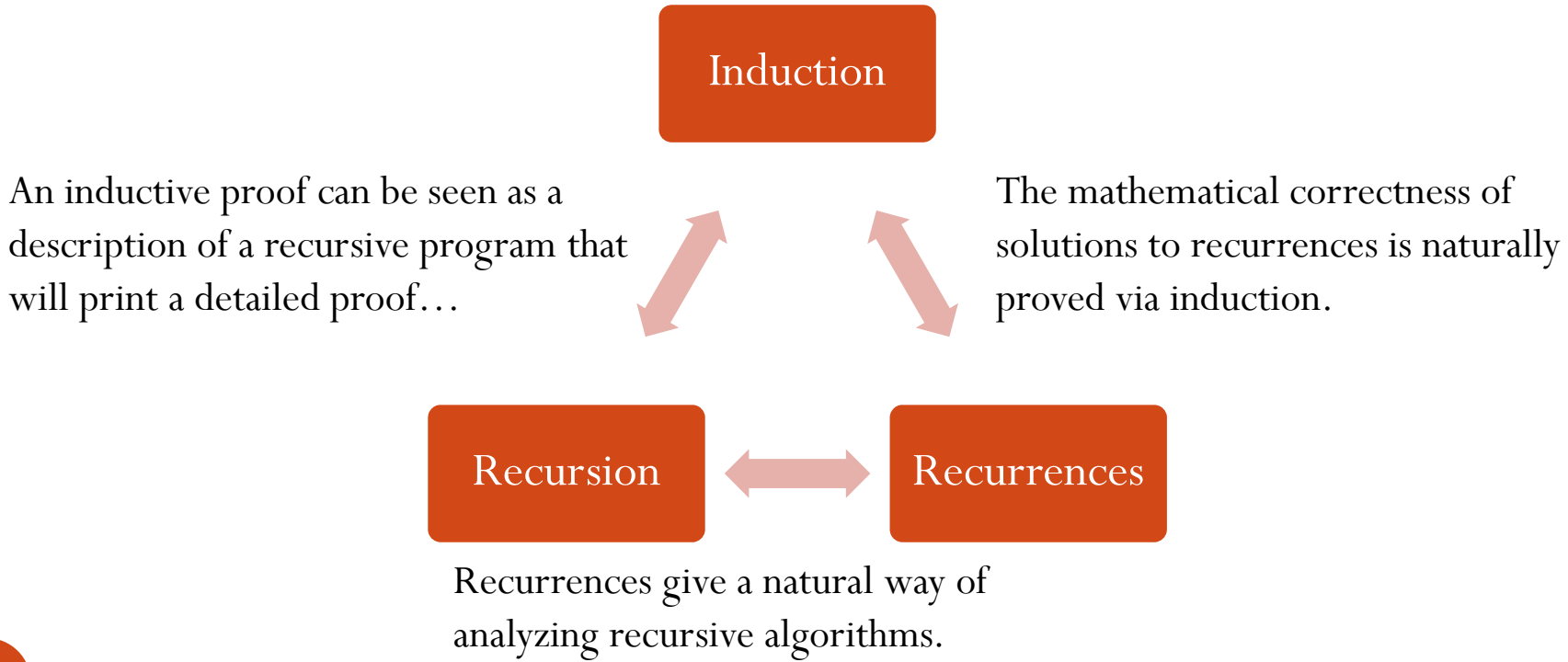
问题1： induction, recursion, recurrences

- 什么是induction、recursion、recurrences？
- 它们两两之间分别有什么联系？



问题1： induction, recursion, recurrences

- 什么是induction、recursion、recurrences？
- 它们两两之间分别有什么联系？



问题2：induction

- 什么是well-ordering principle？
- induction和well-ordering principle之间有什么联系？

- 请利用well-ordering principle（而非induction）证明：

$$1 \cdot 2 + 2 \cdot 3 + \cdots + n(n+1) = \frac{n(n+1)(n+2)}{3}$$

问题2： induction (续)

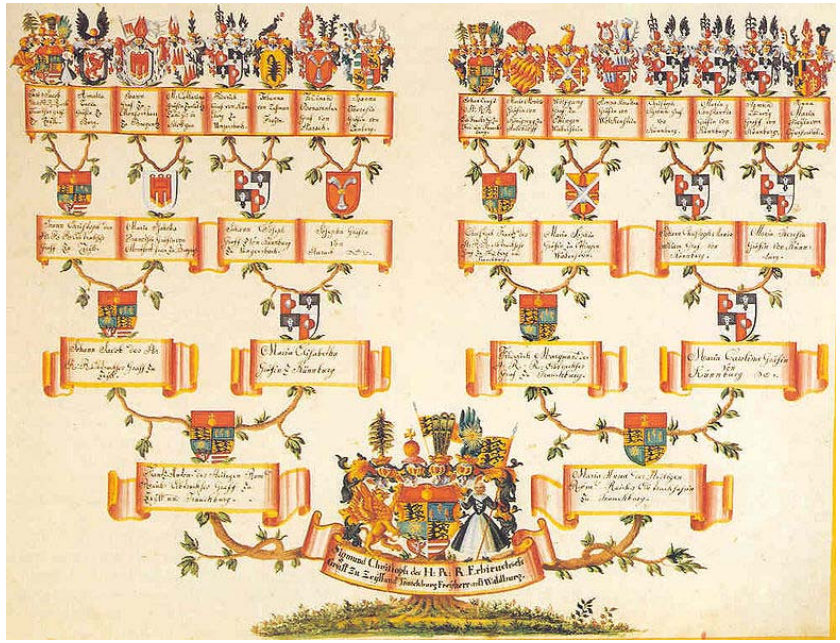
- 你怎么理解build a smaller case to a larger one (bottom-up)和decompose the larger case into smaller ones (top-down)?
- 后者有哪些优势？

问题2：induction (续)

- 你怎么理解build a smaller case to a larger one (bottom-up)和decompose the larger case into smaller ones (top-down)？
- 后者有哪些优势？
  - There are times when this way of thinking is clearly the best way to get a valid proof. Such examples occur throughout computer science.
  - “Building up” small cases into larger ones requires an additional step, namely showing that all larger cases can be created by using the construction given. It is often the case that a “building up” process constructs a proper subset of the possible cases.
  - With a top-down approach, there is no question about what our base case or base cases should be. The base cases are the ones where the recursive decomposition no longer works.

问题2： induction (续)

- 你怎么理解structural induction？
- 你能利用数学归纳法证明这个结论吗：
  An ancestor tree extending over *g* generations shows at most 2^g-1 persons. (注意：有些人的祖先信息可能缺失)



问题3： recurrences

- 当主定理不适用于一个递归式时，你还有什么手段？

$$T(n) = \begin{cases} rT(n-1) + g(n) & if\ n > 0 \\ a & if\ n = 0 \end{cases}$$

$$T(n) = 2T(n/4) + \sqrt{n}\ \lg n$$

问题3：recurrences

- 当主定理不适用于一个递归式时，你还有什么手段？

$$T(n) = \begin{cases} rT(n-1) + g(n) & if\ n > 0 \\ a & if\ n = 0 \end{cases}$$

$$T(n) = 2T(n/4) + \sqrt{n}\ \lg n$$

- 通过递推/递归树猜测，通过归纳法证明

问题3：recurrences

- 你能为这个定理的证明绘制recursion tree吗？

Theorem 4.9 *Let a be an integer greater than or equal to 1 and b be a real number greater than 1. Let c be a positive real number and d a nonnegative real number. Given a recurrence of the form*

$$T(n) = \begin{cases} aT(n/b) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

in which n is restricted to be a power of b,

1. *if* $\log_b a < c$, $T(n) = \Theta(n^c)$,
2. *if* $\log_b a = c$, $T(n) = \Theta(n^c \log n)$,
3. *if* $\log_b a > c$, $T(n) = \Theta(n^{\log_b a})$.

问题4：recurrences and selection

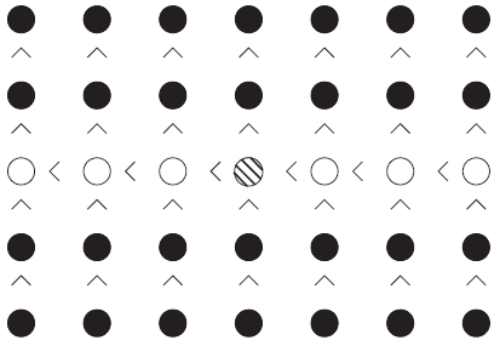
- 第4.6节与前几节有什么联系？

```
Select1(A,i,n)
(selects the ith smallest element in set A, where n = |A| )
(1)  if (n = 1)
(2)       return the one item in A
(3)  else
(4)       p = MagicMiddle(A)
(5)       Let H be the set of elements greater than p
(6)       Let L be the set of elements less than or equal to p
(7)       if (i ≤ |L|)
(8)            Return Select1(L, i, |L|)
(9)       else
(10)           Return Select1(H, i − |L|, |H|).
```

$$T(n) \leq T(n/b) + cn$$

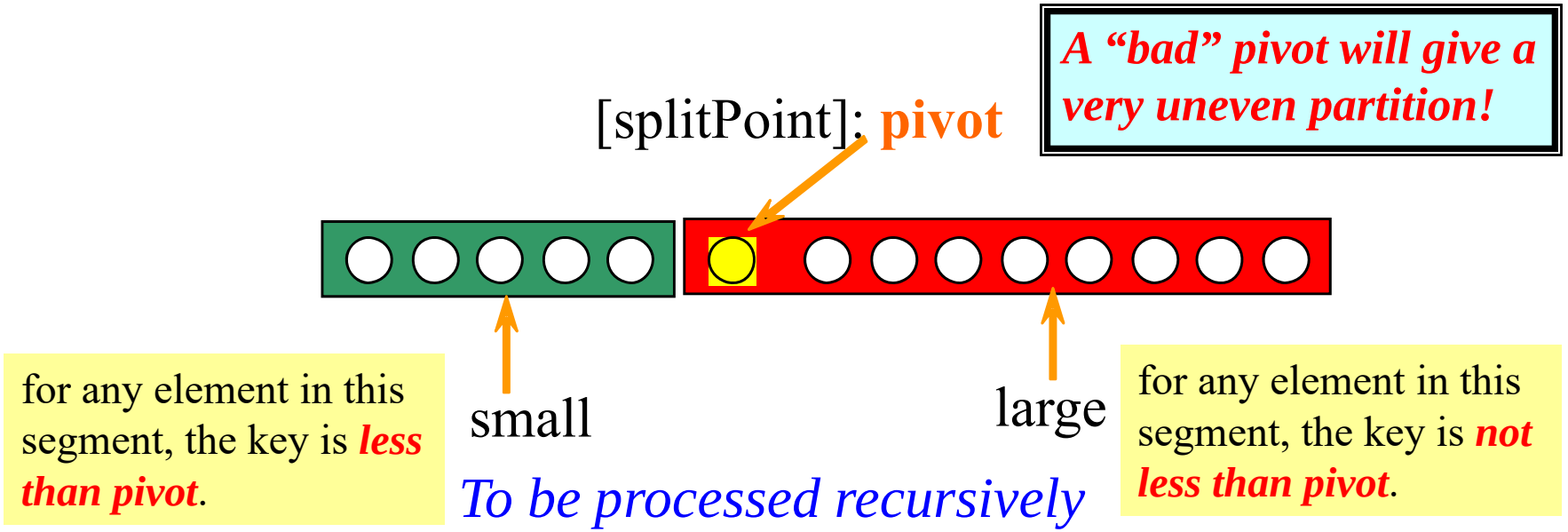
- 你能解释MagicMiddle的基本思路吗？

```
MagicMiddle(A)
(1)  Let n = |A|
(2)  if (n < 60)
(3)       use sorting to return the median of A
(4)  else
(5)       Break A into k = n/5 groups of size 5, G_1, ..., G_k
(6)       for i = 1 to k
(7)            find m_i, the median of G_i (by sorting)
(8)       Let M = {m_1, ..., m_k}
(9)       return Select1 (M, ⌈k/2⌉, k).
```



Partitioning: Larger and Smaller

- Dividing the array to be considered into two subsets: “small” and “large”, the one with more elements will be processed recursively.



补充：找最大和最小值

- 给定n个元素，如何找到最大或最小值？
  - 关键操作？
  - 需要执行多少次？

- 给定n个元素，如何找到最大和最小值？
  - 关键操作？
  - 需要执行多少次？