

2-1

程序的正确性

对应《问题求解》

常见问题求解的方法; 重新审视算法的结构: 从基本结构开始; 继续理解子过程和递归





一. 一些证明的方法

1. 三段论(以前讨论过了)
2. 反证法(proof by contradiction)

问: 为什么反证法的逻辑是对的? (考虑以前的推理规则的理论就可以得到)

3. 数学归纳法

一. 一些证明的方法



中学时候的数学归纳法...

- (1) 证明base case成立
- (2) 证明从 k 到 $k + 1$ 成立
- (3) 证明了对于任意的自然数都成立

问: 能不能证明有关无穷的命题?

为什么不能证明关于 \mathbb{R} 上的命题?



借题发挥

一. 一些证明的方法

看一个“证明”：

1 所有的马都有同样的颜色，我们可以对给定集合中的马匹数量运用归纳法来证明之。理由就是：“如果恰有一匹马，那么它与它自身有相同的颜色，故而基础是显然的。根据归纳法的步骤，假设
有 n 匹马，标号从 1 到 n 。根据归纳假设，标号从 1 到 $n-1$ 的马都有同样的颜色，类似地，标号
从 2 到 n 的马也有同样的颜色。但是，处于中间位置标号从 2 到 $n-1$ 的马，当它们在不同的马
群中时不可能改变颜色，因为这些是马，而不是变色龙。故而依据传递性可知，标号从 1 到 n 的
马也必定有同样的颜色，于是全部 n 匹马都有同样的颜色。证毕。”如果这一推理有误，那么错
在哪儿？

《具体数学：计算机科学基础》第一章热身题1

Why?

一. 一些证明的方法

3. 数学归纳法

(1) 整数的结构与数学归纳法

定义 2.9.1 (Peano 公理). 自然数的 Peano 公理有如下几条:

1. 0 是自然数;
2. 如果 n 是自然数, 则它的后继 S_n 也是自然数;
3. 0 不是任何自然数的后继;
4. 两个自然数相等当且仅当它们的后继相等;

数学归纳法的合理性来自证明对象(\mathbb{Z})结构的定义

一. 一些证明的方法

3. 数学归纳法

(2) 良序原理(well-ordering principle): 自然数集的任意非空子集都有一个最小元.

看上去很显然? 想一想 \mathbb{Z} 和 \mathbb{R} 是不是都满足.

最小的正实数是谁?

好像没有吧...

最小的正整数是谁?

1

一. 一些证明的方法

3. 数学归纳法

(3) 第一类数学归纳法

定理 2.9.1 (第一数学归纳法 (The First Mathematical Induction)). 设 $P(n)$ 是关于自然数的一个性质. 如果

1. $P(0)$ 成立;
2. 对任意自然数 n , 如果 $P(n)$ 成立, 则 $P(n + 1)$ 成立.

那么, $P(n)$ 对所有自然数 n 都成立.

这个是我们高中就学过的...

接下来看个更强大的——

一. 一些证明的方法

3. 数学归纳法

(4) 第二类数学归纳法

定理 2.9.2 (第二数学归纳法 (The Second Mathematical Induction)). 设 $Q(n)$ 是关于自然数的一个性质. 如果

1. $Q(0)$ 成立;
2. 对任意自然数 n , 如果 $Q(0), Q(1), \dots, Q(n)$ 都成立,
则 $Q(n+1)$ 成立.

那么, $Q(n)$ 对所有自然数 n 都成立.

$$\frac{Q(0) \quad \forall n \in \mathbb{N}. \left((Q(0) \wedge \cdots \wedge Q(n)) \rightarrow Q(n+1) \right)}{\forall n \in \mathbb{N}. Q(n)} \quad (\text{第二数学归纳法})$$

$$\left(Q(0) \wedge \forall n \in \mathbb{N}. \left((Q(0) \wedge \cdots \wedge Q(n)) \rightarrow Q(n+1) \right) \right) \rightarrow \forall n \in \mathbb{N}. Q(n).$$

一. 一些证明的方法

3. 数学归纳法

(5) 第一类和第二类数学归纳法是等价的

关键就在于选取的“任意的命题”

引理 1.1.1. 第一类数学归纳法蕴含第二类数学归纳法.

证明. 要证第二类数学归纳法, 也即任给一个命题 F , 若满足 $F(1)$ 及 $(F(1) \wedge F(2) \wedge \cdots \wedge F(n)) \Rightarrow F(n+1)$, 则有 $\forall k \in \mathbb{N}. F(k)$. 那么, 我们可以构造命题 $G(n) := F(1) \wedge F(2) \wedge \cdots \wedge F(n)$. 显然, $G(n) \Rightarrow F(n+1)$, 又有 $G(n) \Rightarrow G(n)$, 则 $G(n) \Rightarrow (F(n+1) \wedge G(n))$, 而后者即为 $G(n+1)$. 故, 命题 G 满足第一类数学归纳法的条件, 所以 $\forall k \in \mathbb{N}. G(k)$ 成立. 而 $G(k) \Rightarrow F(k)$, 故 $\forall k \in \mathbb{N}. F(k)$, 也即第二类数学归纳法成立. \square

引理 1.1.2. 第二类数学归纳法蕴含第一类数学归纳法.

证明. 要证第一类数学归纳法, 也即任给一个命题 F , 若满足 $F(1)$ 及 $F(n) \rightarrow F(n+1)$, 则有 $\forall k \in \mathbb{N}. F(k)$. 显然, F 是满足第二类数学归纳法的条件的 (因为 1 的条件比 2 强), 故根据第二类数学归纳法, $F(k)$ 对所有正整数 k 成立, 也即第一类数学归纳法成立. \square

一. 一些证明的方法

3. 数学归纳法



Blue Eyes:
The Hardest Logic Puzzle in the World



一. 一些证明的方法

3. 数学归纳法

问题 2.9.4. Of the 1000 islanders, it turns out that **100 of them have blue eyes** and **900 of them have brown eyes**, although the islanders are not initially aware of these statistics (each of them can of course only see 999 of the 1000 tribespeople).

One day, a **blue-eyed foreigner** visits to the island and wins the complete trust of the tribe.

One evening, he addresses the entire tribe to thank them for their hospitality.

However, not knowing the customs, the foreigner makes the mistake of mentioning eye color in his address, remarking “**how unusual it is to see another blue-eyed person like myself in this region of the world**”.

What effect, if anything, does this faux pas (失礼) have on the tribe?

假设这个岛屿的人都学习的数理逻辑课程....

看起来什么事情都没有发生?



一. 一些证明的方法

3. 数学归纳法

定理 2.9.3. Suppose that the tribe had $n > 0$ blue-eyed people.

Then n days after the traveller's address, all n blue-eyed people commit suicide.

证明. 考虑采用数学归纳法:

- 基础步骤: $n = 1$.

这个唯一的蓝眼人的内心独白: “你直接念我身份证吧”

- 归纳假设: 有 n 个蓝眼人时, 前 $n - 1$ 天无人自杀, 第 n 天集体自杀.

- 归纳步骤: 考虑恰有 $n + 1$ 个蓝眼人的情况.

每个蓝眼人都如此推理: 我看到了 n 个蓝眼人, 他们应该在第 n 天集体自杀.

但是, 每个蓝眼人都在等其它 n 个蓝眼人自杀, 因此, 第 n 天无人自杀.

每个蓝眼人继续推理: 一定不止 n 个蓝眼人, 但是我看到的其余人都不是蓝眼.

所以, “小丑竟是我自己”.

□



二. 程序的力量

1. Python库—z3

我们可以用z3库求解一个数独.

<https://shzaiz.github.io/lecture/PythonPSolve/code/1-sd-py.html>

```
zgw - IPython: Users/zgw - python3.10 ~/miniforge3/bin/ipython - 80x24
...:             X[i][j] == instance[i][j])
...:     for i in range(9) for j in range(9) ]
...:
...: s = Solver()                                     # (1)
...: s.add(sudoku_c + instance_c)                     # (2)
...: if s.check() == sat:                            # (3)
...:     m = s.model()                               # (4)
...:     r = [ [ m.evaluate(X[i][j]) for j in range(9) ]      # (5)
...:             for i in range(9) ]
...:     print_matrix(r)                             # (6)
...: else:
...:     print("failed to solve")                   # (7)
...:
[[5, 3, 4, 6, 7, 8, 9, 1, 2],
 [6, 7, 2, 1, 9, 5, 3, 4, 8],
 [1, 9, 8, 3, 4, 2, 5, 6, 7],
 [8, 5, 9, 7, 6, 1, 4, 2, 3],
 [4, 2, 6, 8, 5, 3, 7, 9, 1],
 [7, 1, 3, 9, 2, 4, 8, 5, 6],
 [9, 6, 1, 5, 3, 7, 2, 8, 4],
 [2, 8, 7, 4, 1, 9, 6, 3, 5],
 [3, 4, 5, 2, 8, 6, 1, 7, 9]]
```

In [2]:

三. 程序的基本结构



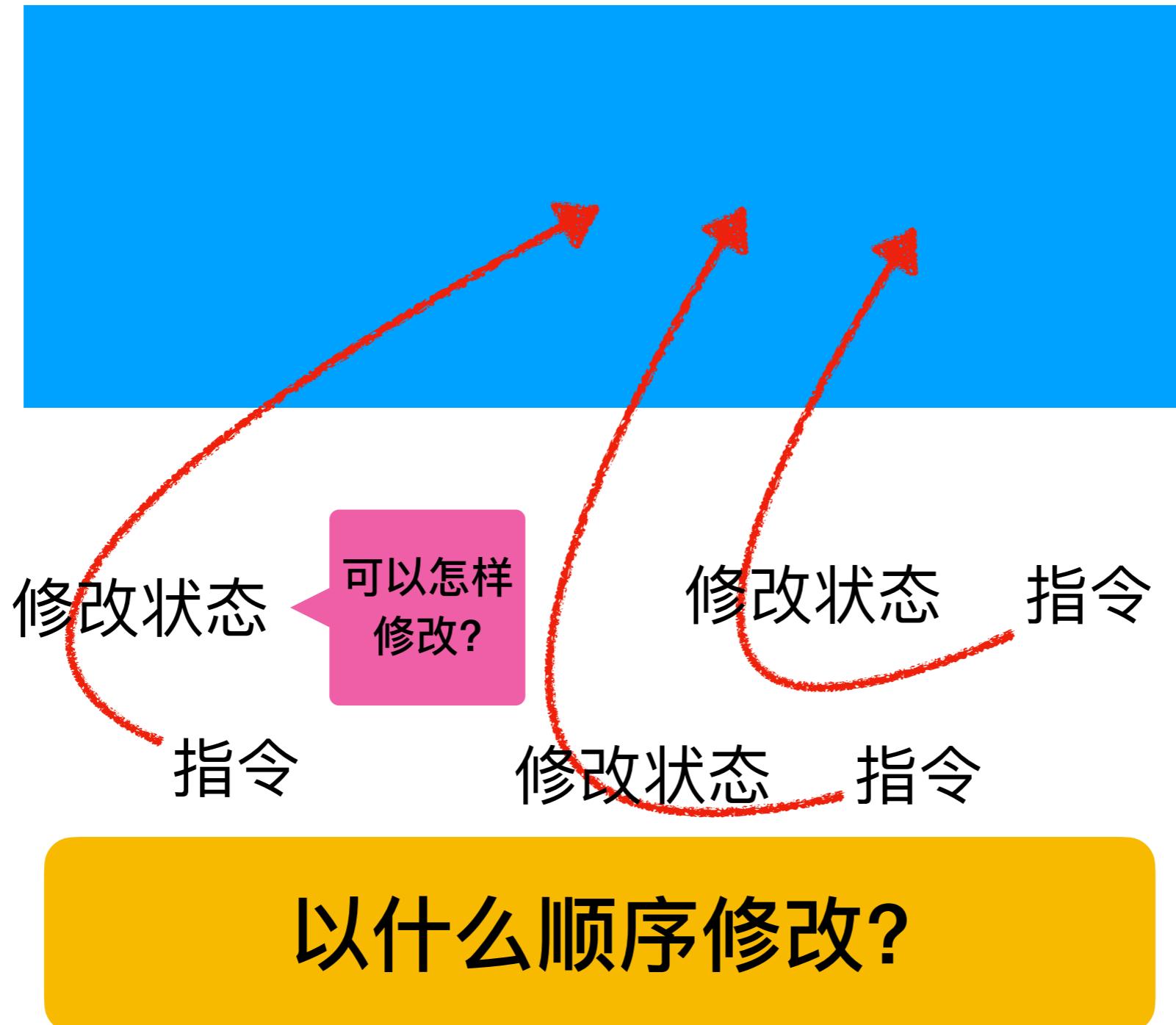
和Coding有什么关系?



借题发挥

三. 程序的基本结构

计算机内部的状态





三. 程序的基本结构

凭什么说我们的算法是对的?
一些滑稽和致命的bug...

- In the early 1960s one of the American spaceships in the Mariner series sent to Venus was lost forever at a cost of millions of dollars, due to a mistake in a flight control computer program.
- In 1981 one of the television stations covering provincial elections in Quebec, Canada, was led by its erroneous computer programs into believing that a small party, originally thought to have no chance at all, was actually leading. This information, and the consequent responses of commentators, were passed on to millions of viewers.
- In a series of incidents between 1985 and 1987, several patients received massive radiation overdoses from Therac-25 radiation-therapy systems; three of them died from resulting complications. The hardware safety interlocks from previous models had been replaced by software safety checks, but all these incidents involved programming mistakes.
- Some years ago, a Danish lady received, around her 107th birthday, a computerized letter from the local school authorities with instructions as to the registration procedure for first grade in elementary school. It turned out that only two digits were allotted for the “age” field in the database.

三. 程序的基本结构

1. 一些基本结构

(1) 顺序结构

例子: 写一份批判中国地质大学不合理的早起跑操制度

- 拿出一张纸
 - 拿出一根笔
 - 开始写文章
 - 写完公布
- 改变了纸笔的位置
纸上的内容

如果我们不止写一份, 要写好多份呢?

三. 程序的基本结构

1. 一些基本结构

(2) 循环结构:

① 两种策略

- 控制数量: 写他个10份
- 控制条件: 直到我满意了为止

② 对于循环的正确性: 循环不变式

定理 2.10.1 (证明循环不变式的基本方法). (1) Initialization. It is true prior to the first iteration of the loop.

(2) Maintenance. If it is true before an iteration of the loop, it remains true before next iteration.

(3) Termination. The loop terminates, and when it terminates, the invariant – usually along with the reason that the loop terminated–gives us a useful property that helps show that the algorithm is correct.

三. 程序的基本结构

1. 一些基本结构

(2) 循环结构:

② 对于循环的正确性: 循环不变式

例子1. 用一个逐项累加的循环计算 a^*b , 循环不变式可以是: “存放中间结果的量的值= a^* “循环变量的当前值”

例子2. 保证当前最大值是已经扫描过元素中的最大值

例子3. 在世界杯足球赛的第二阶段(小组赛结束后) 进行单循环比赛, 每场必须分出胜负, 赢者进下一轮, 输者回家. 16 个队参加, 赛多少场决出冠军?

答案是 15 场. 因为每一次循环满足不变量: 这一次的值 = 上一次的值-1.

三. 程序的基本结构

1. 一些基本结构

(3) 分支结构 – 这一部分就和逻辑有点关系了

2. 基本结构的组合方式

- 顺序地写
- 嵌套

FOR LOOP 1

...

END LOOP

FOR LOOP2

...

END LOOP

FOR LOOP 1

...

FOR LOOP 2

...

END LOOP

...

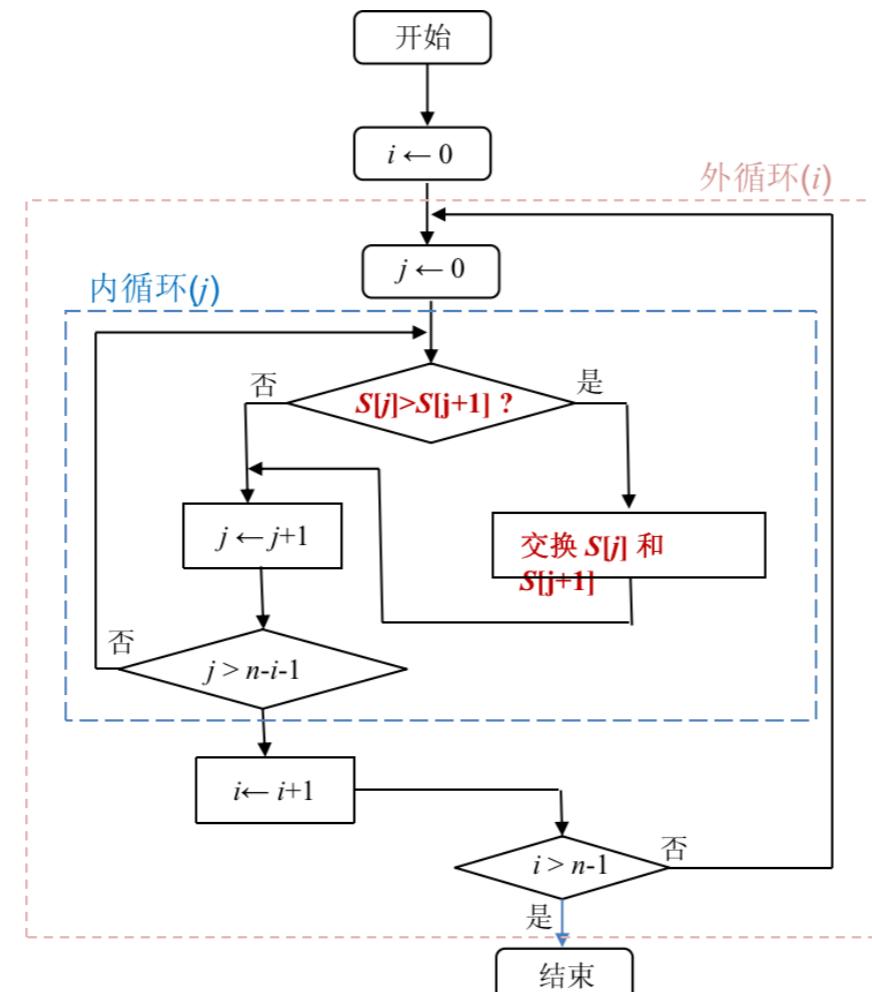
END LOOP

三. 程序的基本结构

2. 基本结构的组合方式

问题 2.10.4. 用上面的语句 (可以画流程图) 及其列举与嵌套, 完成排序问题:

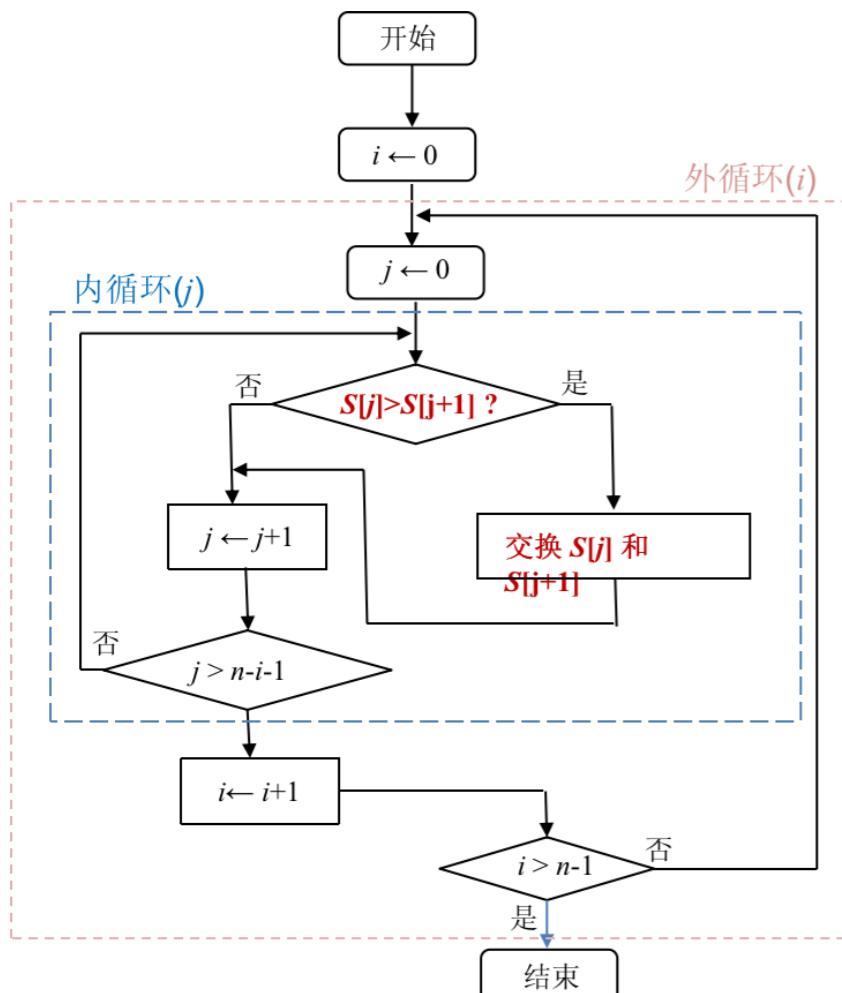
- (1) 输入: 长度为 n 的自然数序列 S (假设其中没有重复元素).
- (2) 输出: 序列 S' , 其元素与 S 完全一样, 但已从小到大排好序.



Known as Bubble sort

三. 程序的基本结构

2. 基本结构的组合方式



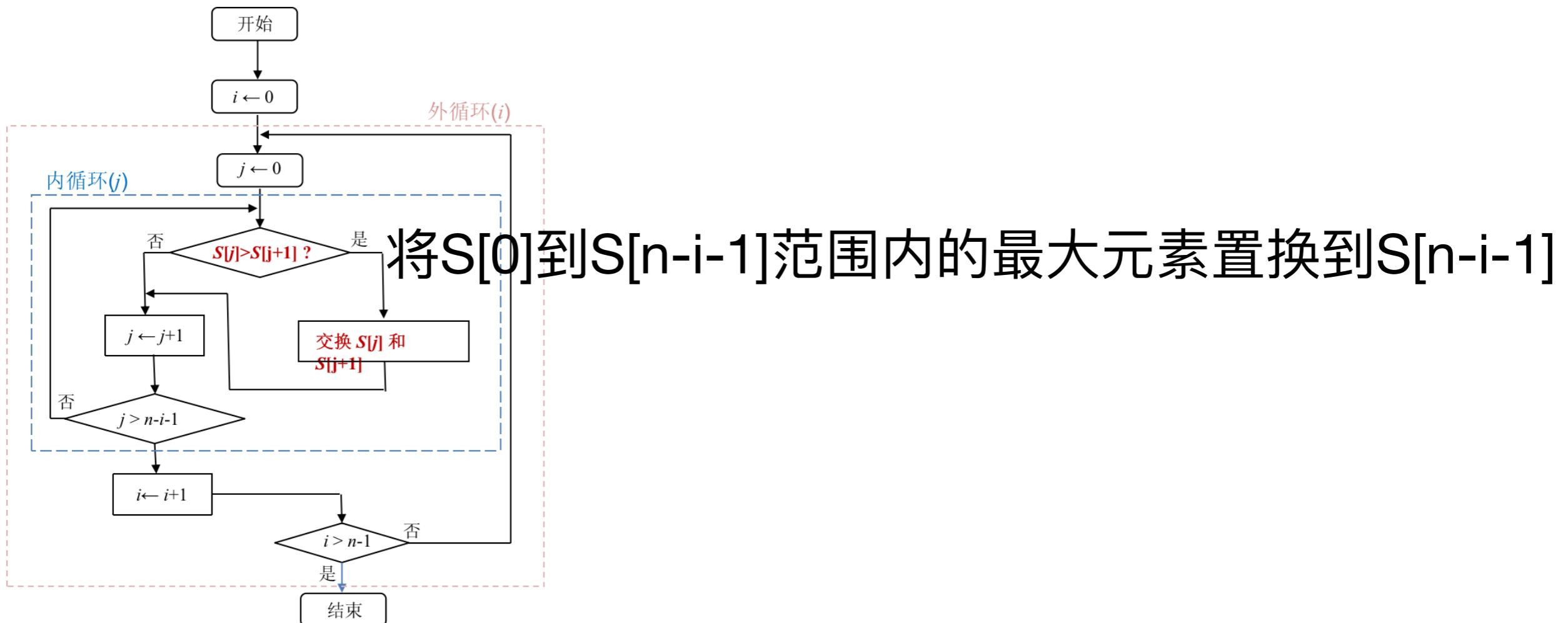
每次 i 循环开始前，序列 S 中地址最高的 i 个位置 (除 $i = 0$ 外，即 $S[n - i]$ 到 $S[n - 1]$) 包含 S 中前 i 个最大的元素，且已从小到大排好序；序列中其它位置上的元素即 S 中其它 $n - i$ 个元素。

下面按照如下的三个方法归纳：

- 当 $i = 0$, 显然;
- 如果当 $i = k > 0$ 时上述命题成立, 即 k 循环开始前, S 中最大的 k 个元素已被置换到 $S[n - k]$ 到 $S[n - 1]$; 在 k 循环中, 指针 j 从位置 0 扫描到位置 $n - k - 1$, 并将此范围内最大元素置换到 $S[n - k - 1]$. 则当 $i = k + 1$ 时, S 中最大的 $k + 1$ 个元素被排好序, 放置在 $S[n - (k + 1)]$ 到 $S[n - 1]$;
- 当 $i = n$, 算法终止, 即 n 循环 (未执行) 前, n 个 S 中的元素排好序, 并放置在 $S[0]$ 到 $S[n - 1]$ 的位置上.

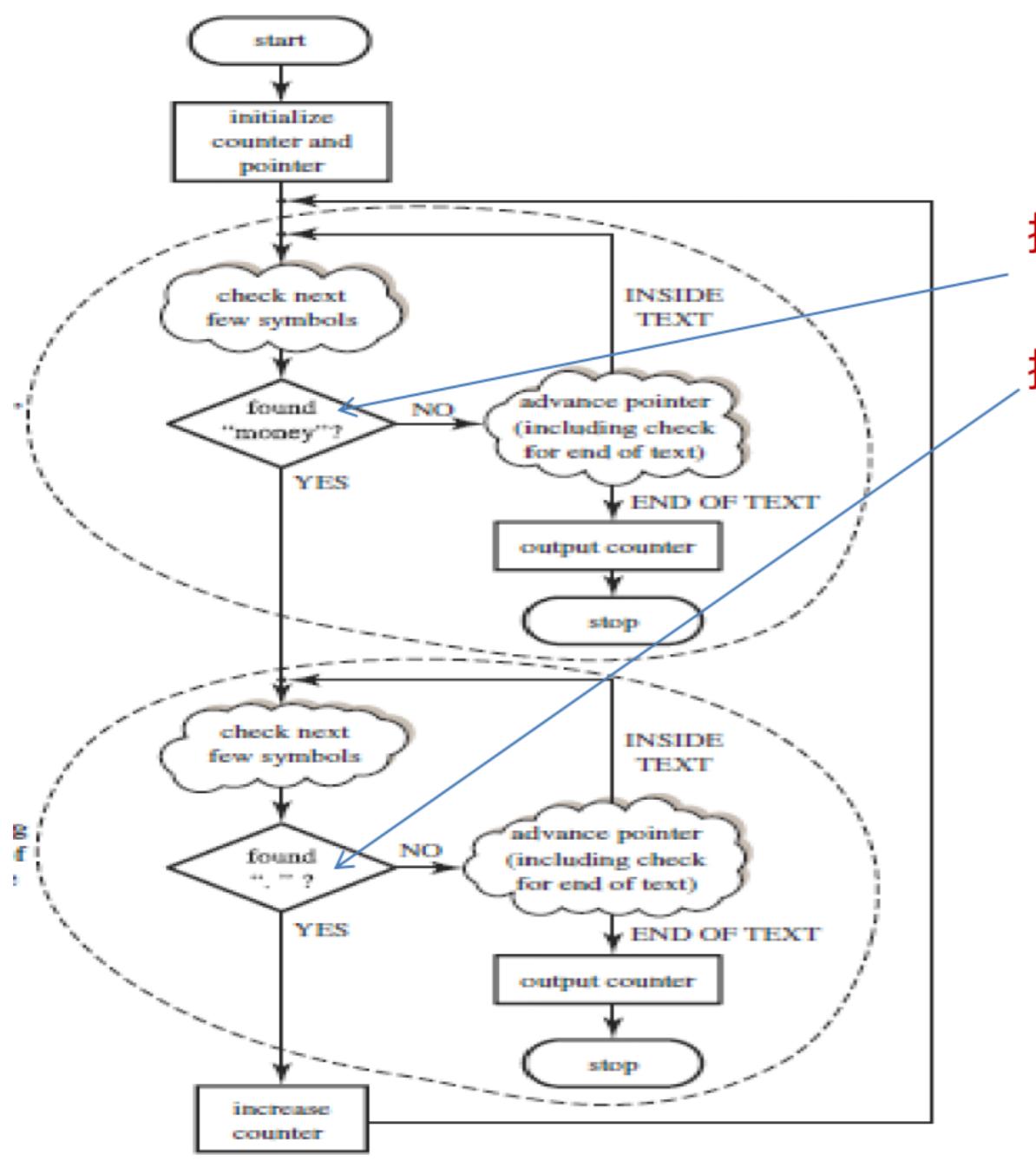
三. 程序的基本结构

3. 子过程 – 过程抽象: 可以不可以把内循环描述为一个问题?



三. 程序的基本结构

3. 子过程 – 过程抽象: 对于一个句子中“money”的个数计数



搜索 “money” 一词

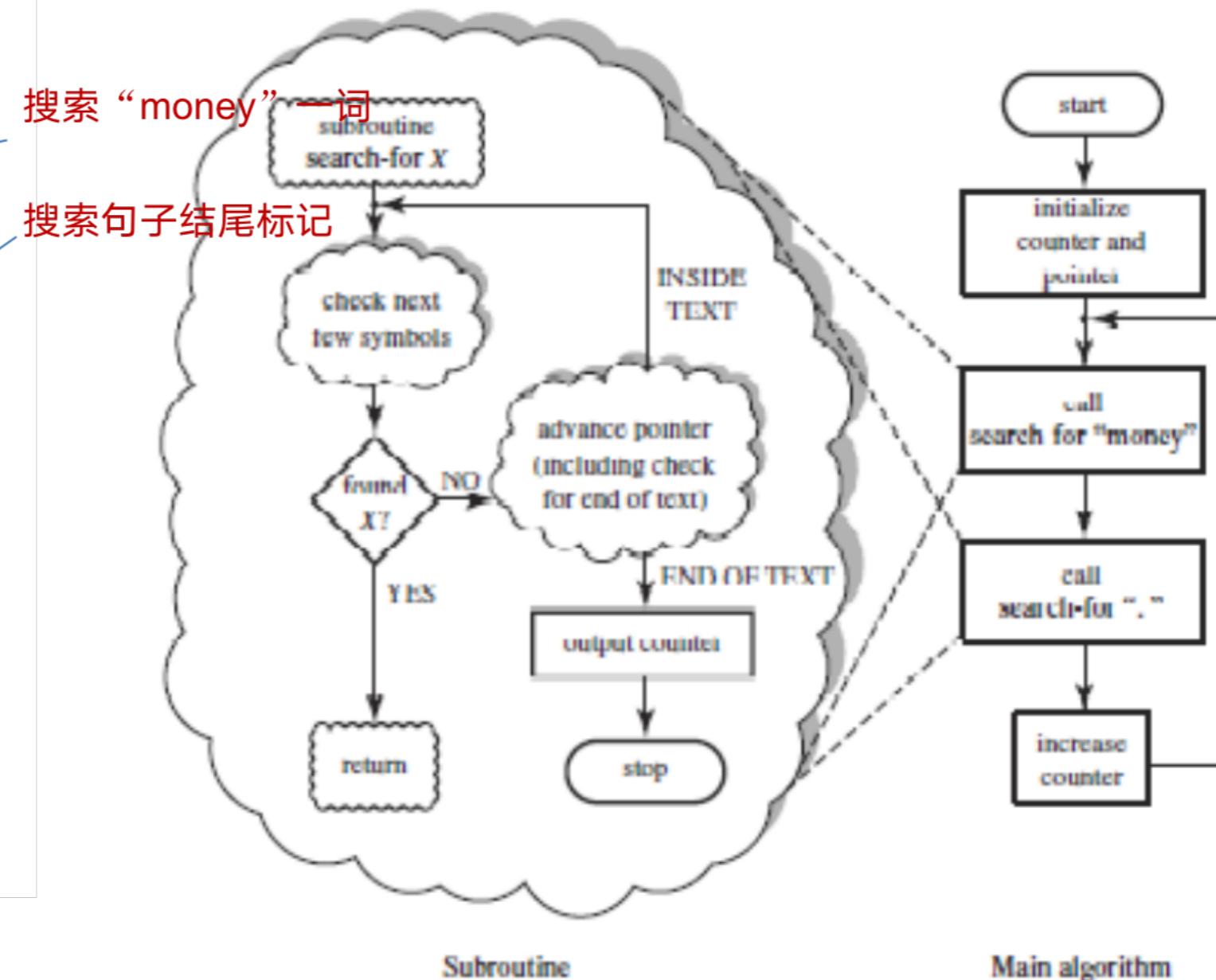
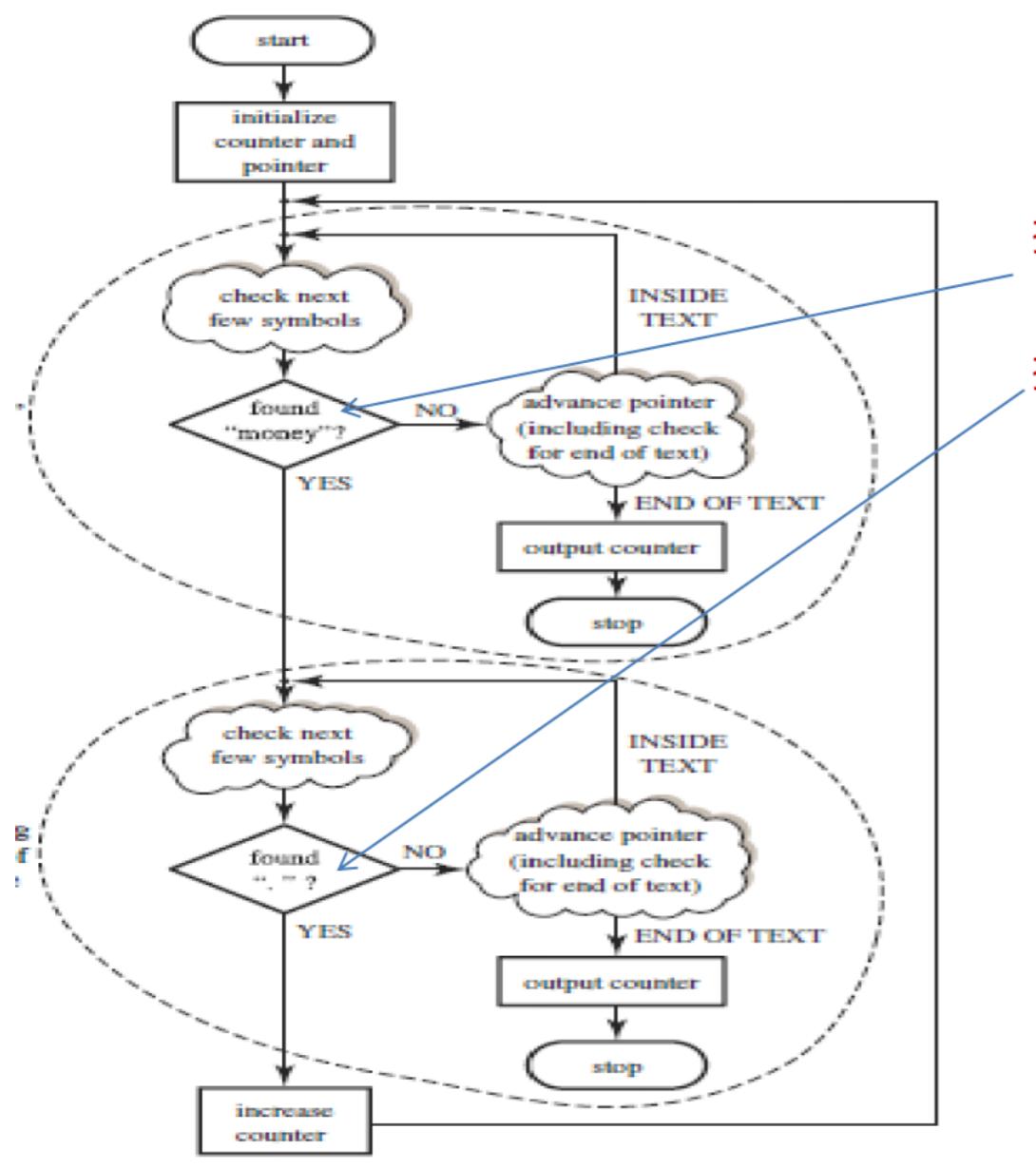
搜索句子结尾标记

尽管对象不同，动作确是一样的，有可能只描述一次吗？

- 一改全改

三. 程序的基本结构

3. 子过程 – 过程抽象: 对于一个句子中“money”的个数计数



三. 程序的基本结构

4. 递归 – 自己调用自己的子过程

- 从数学归纳法到递归

归纳：

“假如”这个结论对 $k-1$ 是成立的，我试图证明它对 k 也是成立的。

如果我做到了，就可以认为（当然考虑到“奠基”）对任意不小于奠基值的自然数结论都成立。

递归：

“假如”有“人”能帮我解决规模为 $k-1$ 的问题“实例”，我就试图用那个结果来解该问题规模为 k 的“实例”

如果我做到了，就可以认为（当然也得给个“base case”的解法）我解了这个问题。

三. 程序的基本结构

4. 递归 – 自己调用自己的子过程

例子: Tower of Hanoi – 电脑是怎么执行的?



subroutine move N from X to Y using Z :

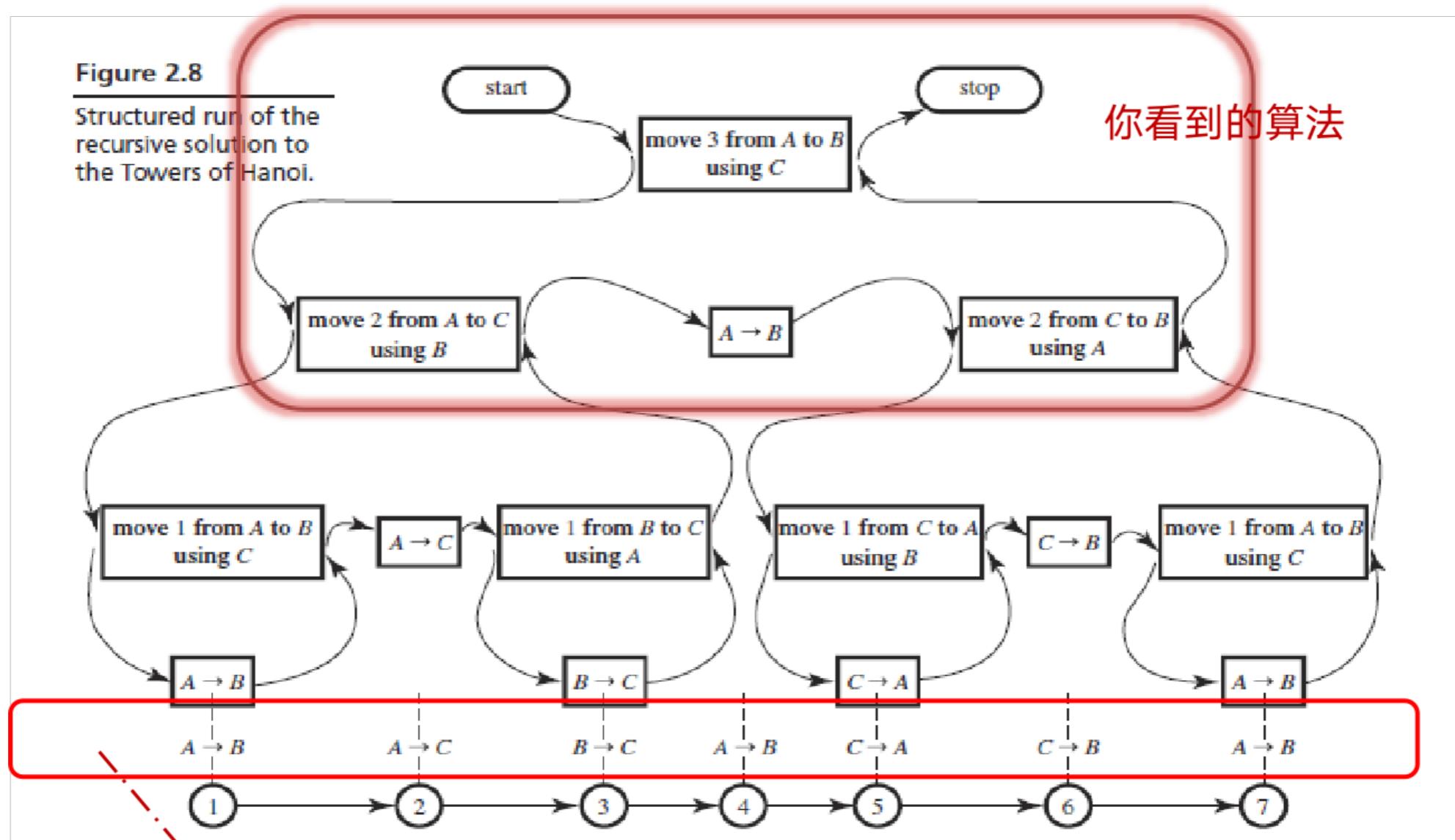
- (1) if N is 1 then output “move X to Y ”;
- (2) otherwise (i.e., if N is greater than 1) do the following:
 - (2.1) call move $N - 1$ from X to Z using Y ;
 - (2.2) output “move X to Y ”;
 - (2.3) call move $N - 1$ from Z to Y using X ;
- (3) **return**.

和数学归纳法不同的是这里有4个
参数

三. 程序的基本结构

4. 递归 – 自己调用自己的子过程

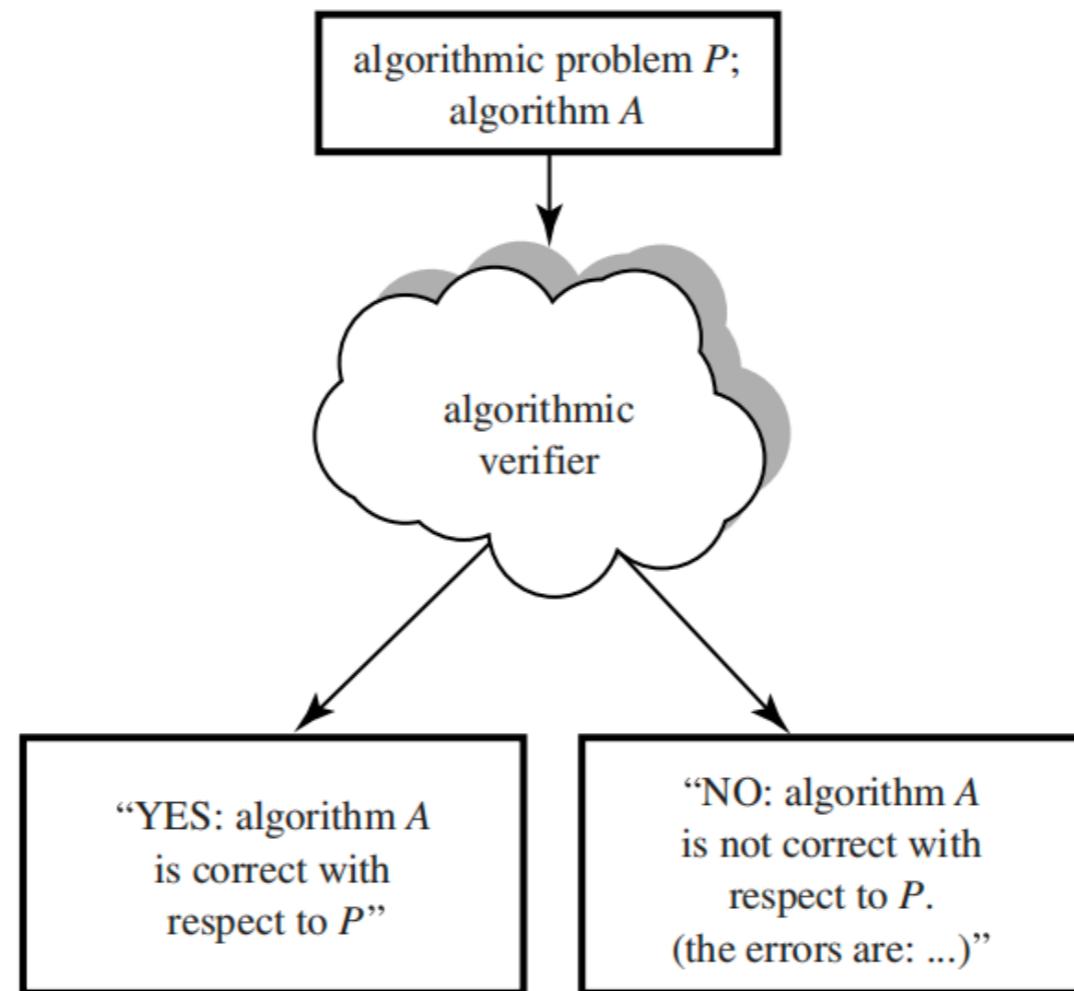
例子: Tower of Hanoi – 电脑是怎么执行的?



这才是“真实”的执行过程!

四. 程序的正确和等价

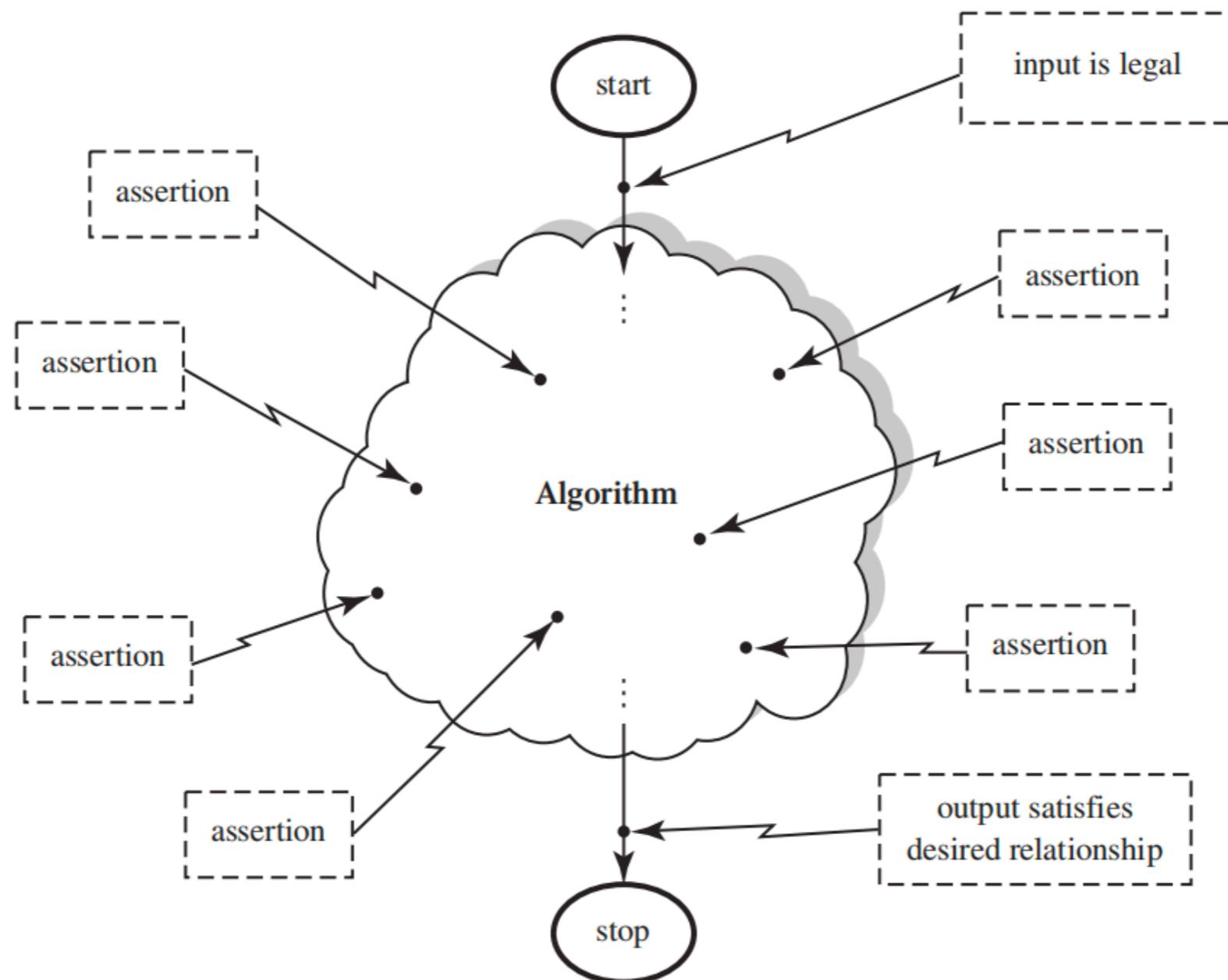
思维实验: 是不是有一个算法可以验证所有的算法?



很遗憾, 这样的算法不存在(可以规约到图灵停机问题)

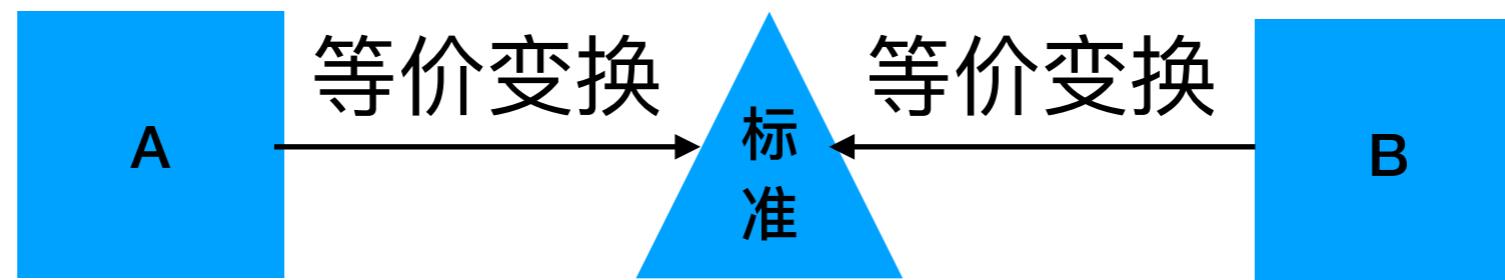
四. 程序的正确和等价

退而求其次: 验证所有的“不变量”是满足的



四. 程序的正确和等价

理想:



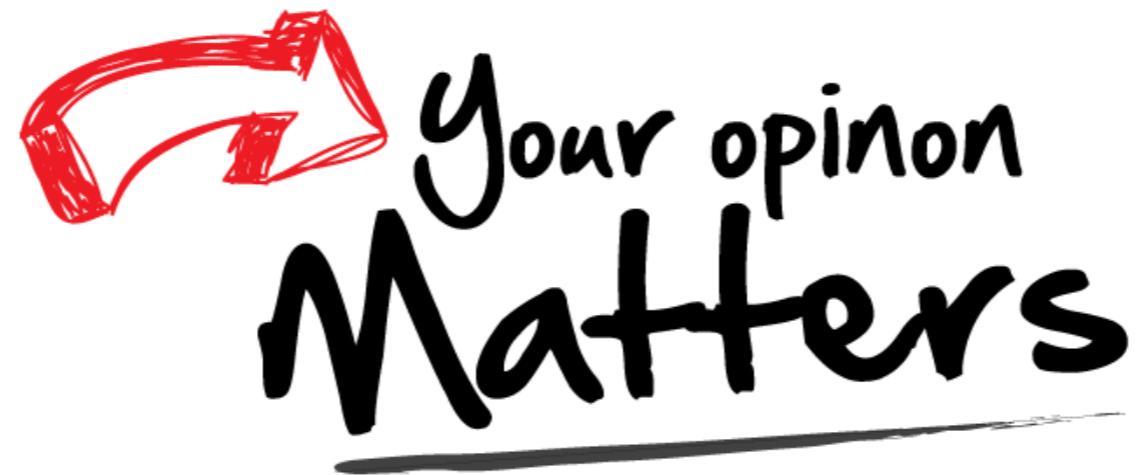
One of the issues relevant to both synthesis and equivalence proofs is that of **program transformations**, in which parts of an algorithm or program are transformed in ways that preserve equivalence. For example, we might be interested in such transformations in order to make an algorithm comply with the rules of some programming language (for example, replacing recursion by iteration when the language does not allow recursion) or for efficiency purposes, as illustrated in Chapter 6.

Research in the fields of correctness and logics of algorithms blends nicely with research on the semantics of programming languages. It is impossible to prove anything about a program without a rigorous and unambiguous meaning for that program. The more complex the languages we use, the more difficult it is to provide them with semantics, and the more difficult it is to devise methods of proof and to investigate them using algorithmic logics. This is another reason why functional languages are more amenable to proofs of correctness than imperative languages.

参考文章和课件

1. 魏恒峰《离散数学2020》 数学归纳法
2. 南京大学《计算机问题求解2022》 论题1-4:基本的算法结构
3. *Algorithmtics: The Spirit of Computing* by D.H.
4. *Concrete Mathematics: Foundation of Computer Science* by Graham, Knuth, Patashnic

Thank You!



QQ: 2095728218

Email: micoael@qq.com

(学校) gwzhang@cug.edu.cn