计算机问题求解 – 论题1-4

- 基本的算法结构

  课程研讨

  - DH第2章第1、2单元

问题1：控制结构与流程图

- 什么是控制结构？
  为什么需要定义若干种控制结构？

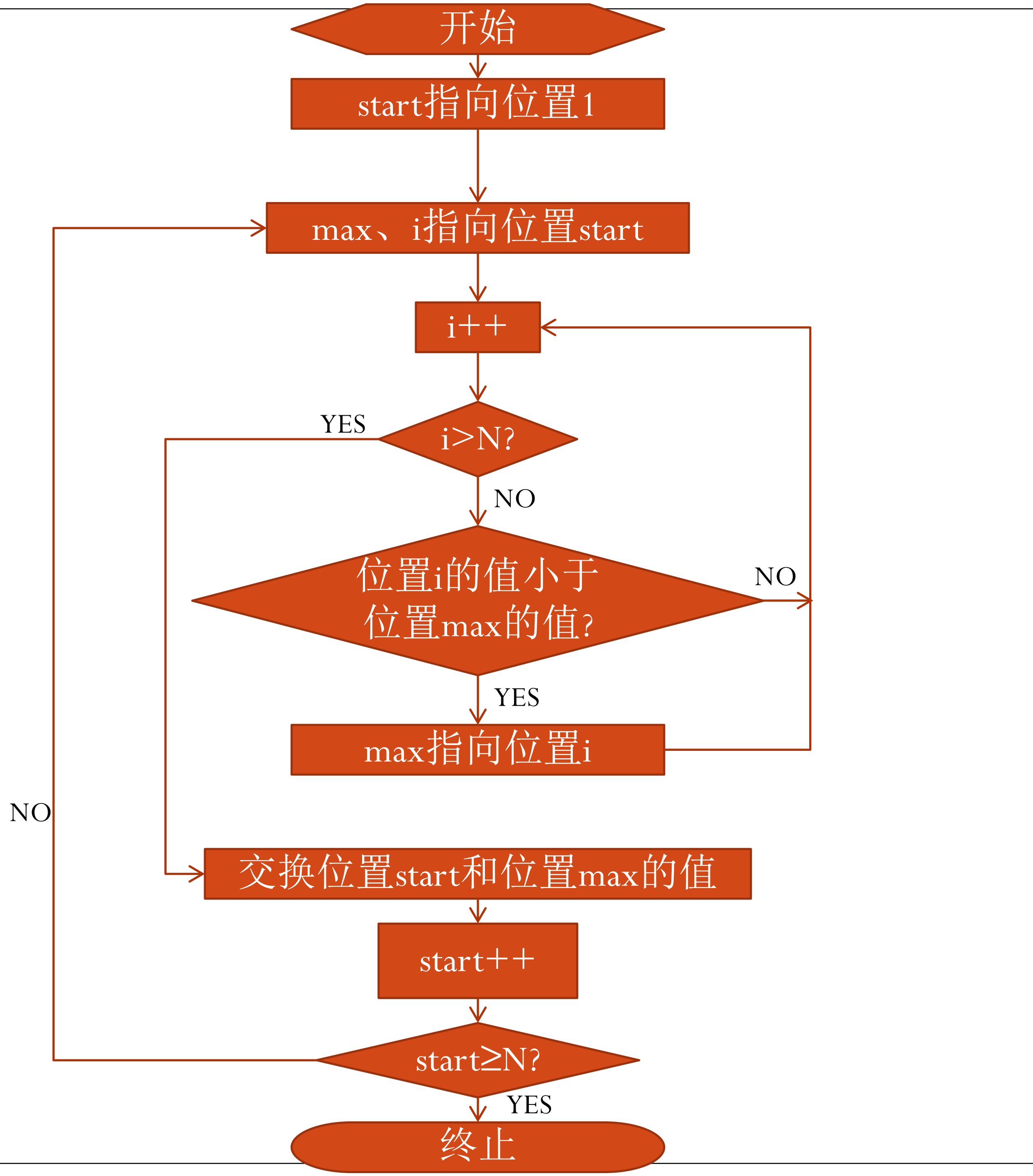
- 你能解释这些控制结构吗？
  - Direct sequencing
  - Conditional branching
  - Bounded iteration
  - Conditional iteration (aka unbounded iteration)
- 它们可以相互替代吗？

- 多种控制结构之间可以错杂嵌套，你能举一些实际生活中的例子吗？

问题1： 控制结构与流程图 (续)

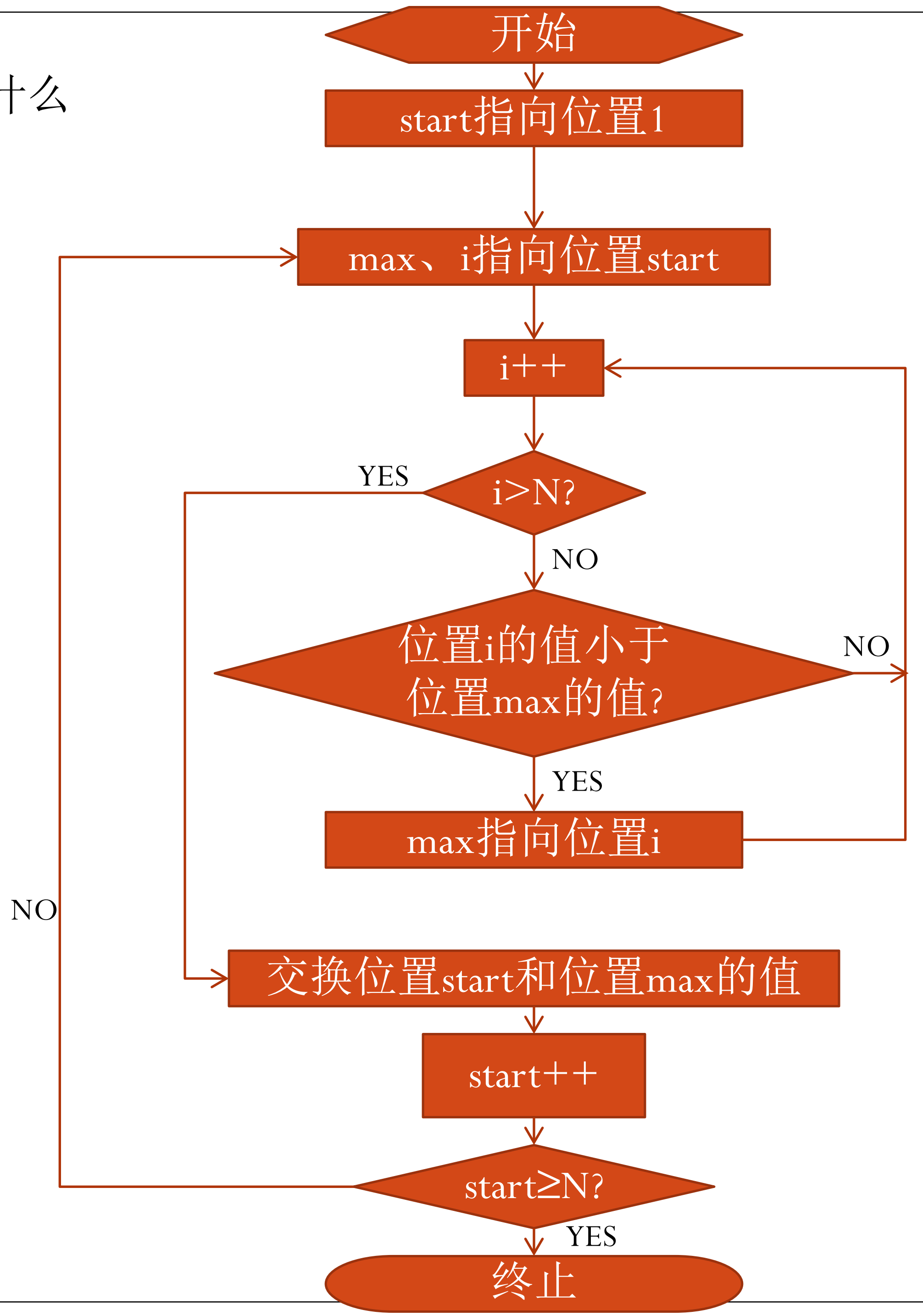
- 你能解释“选择排序”算法吗？
  1. Find the minimum value in the list.
  2. Swap it with the value in the first position.
  3. Repeat the steps above for the remainder of the list (starting at the second position and advancing each time).
- 它用到了哪些控制结构？
- 你能绘制出它的流程图吗？

24
12
78
14
26
8
69
46



24
12
78
14
26
8
69
46

绘制流程图有什么
好处和坏处？

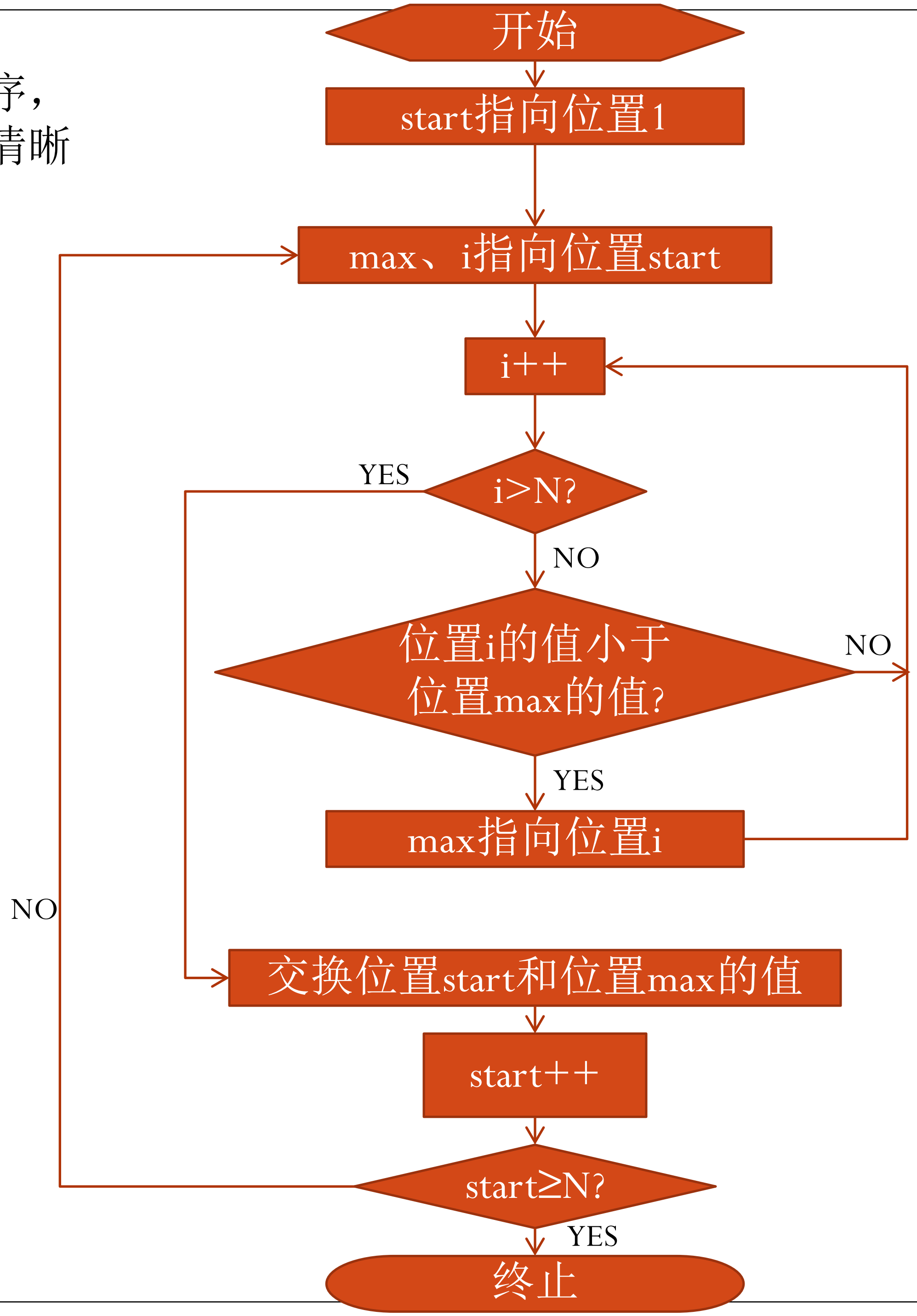


24
12
78
14
26
8
69
46

问题2：子程序

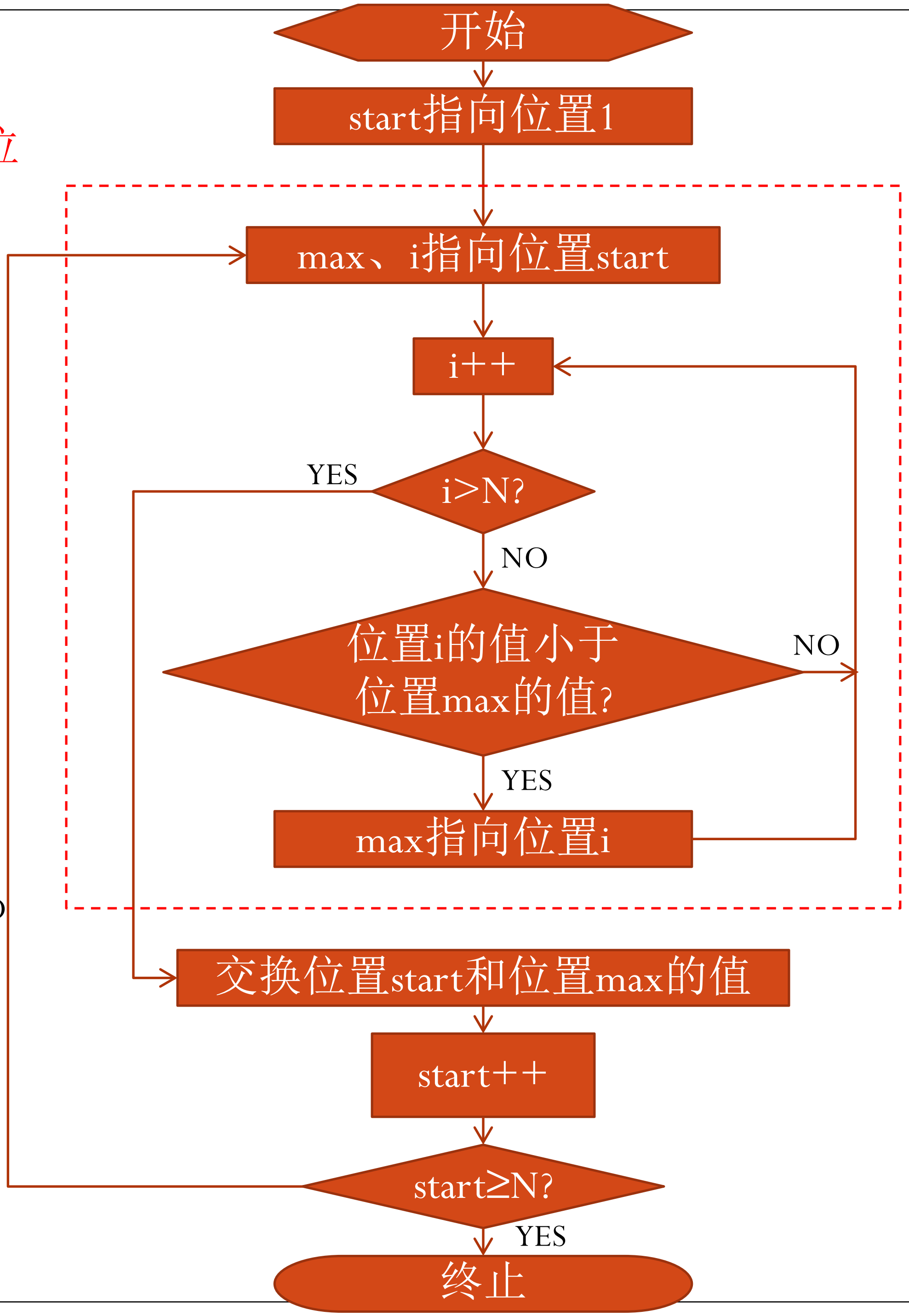
- 什么是子程序？
  什么是子程序的参数？它有什么作用？

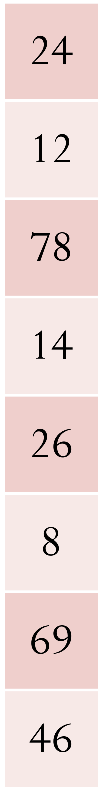
你能借助子程序，
绘制一种更加清晰
的流程图吗？



24
12
78
14
26
8
69
46

找到位置start到N中的最小值，其位置记作max





问题2：子程序 (续)

- 使用子程序有什么好处和坏处？

问题2：子程序 (续)

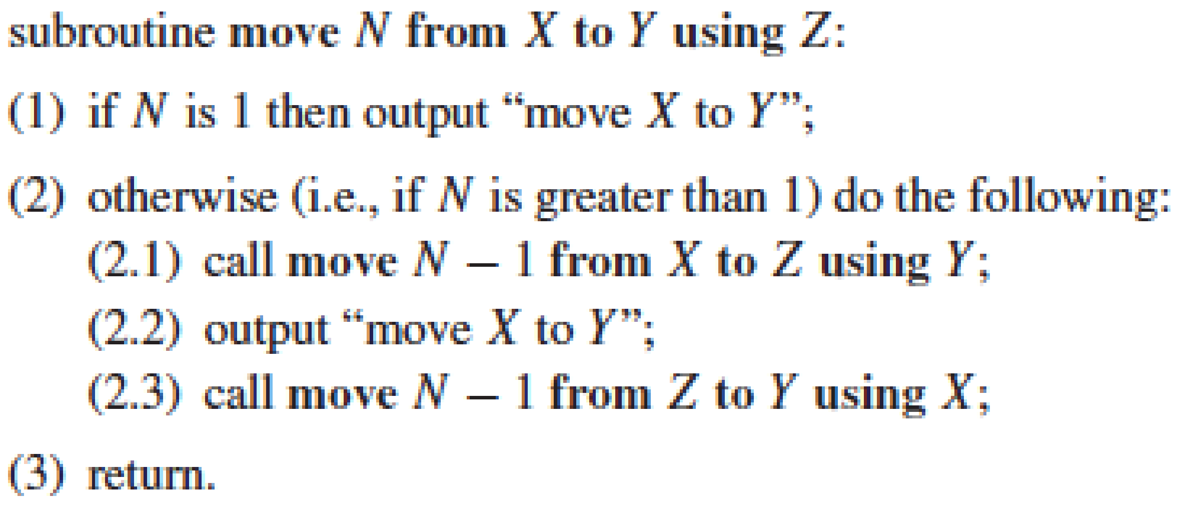
- 使用子程序有什么好处和坏处？
  - Subroutines can be very economical as far as the size of an algorithm is concerned.
  - Subroutines not only shorten algorithms but also make them clear and well structured.
  - All that the user of the subroutine has to know is what it does, but not how it does.
  - Using subroutines, it is possible to develop a complex algorithm gradually step by step.

问题2：子程序 (续)

- 使用子程序有什么好处和坏处？（维基百科）
  - Advantages
    - Decomposing a complex programming task into simpler steps: this is one of the two main tools of structured programming, along with data structures
    - Reducing duplicate code within a program
    - Enabling reuse of code across multiple programs
    - Dividing a large programming task among various programmers, or various stages of a project
    - Hiding implementation details from users of the subroutine
    - Improving traceability (i.e. most languages offer ways to obtain the call trace which includes the names of the involved subroutines and perhaps even more information such as file names and line numbers); by not decomposing the code into subroutines, debugging would be impaired severely
  - Disadvantages
    - Invoking a subroutine (versus using in-line code) imposes some computational overhead in the call mechanism. The subroutine typically requires standard housekeeping code – both at entry to, and exit from, the function (function prologue and epilogue – usually saving general purpose registers and return address as a minimum).

问题3：递归

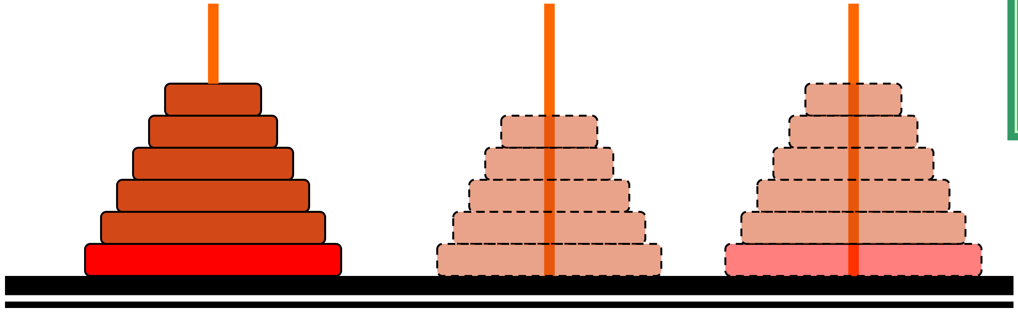
- 什么是递归？
- 你能举一些实际生活中的例子吗？



```
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.
```

Thinking Recursively (1)

- Towers of Hanoi
  - How many moves are need to move all the disks to the third peg by moving only one at a time and never placing a disk on top of a smaller one.



$$T(1) = 1$$
$$T(n) = 2T(n-1) + 1$$

Solution of Towers of Hanoi

$T(n) = 2T(n-1) + 1$

$2T(n-1) = 4T(n-2) + 2$

$4T(n-2) = 8T(n-3) + 4$

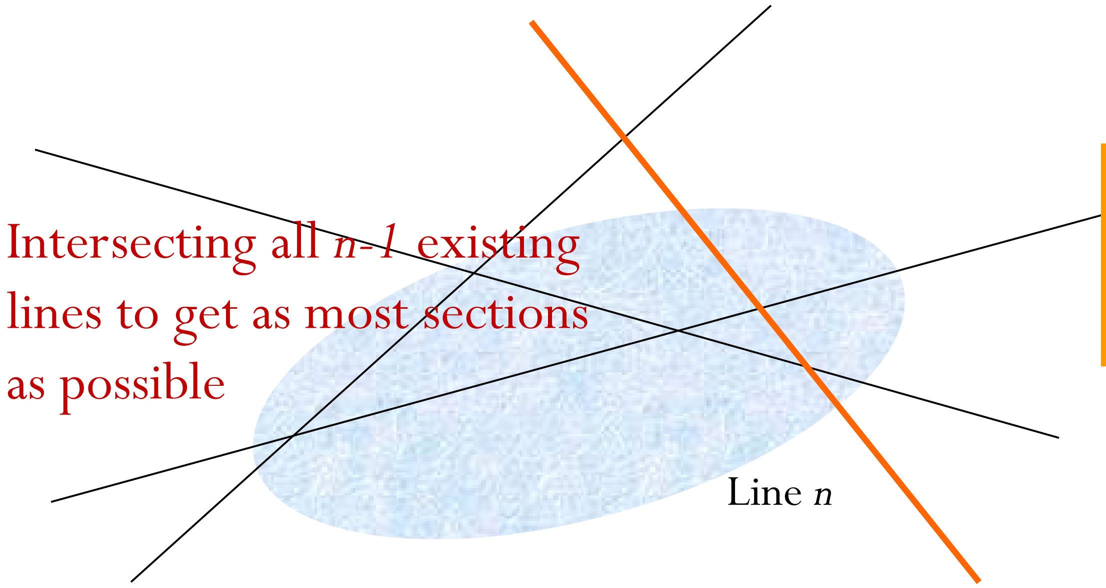
……..

$2^{n-2}T(2) = 2^{n-1}T(1) + 2^{n-2}$

$$\Longrightarrow \quad \boldsymbol{T(n) = 2^n - 1}$$

Thinking Recursively (2)

- Cutting the plane
  - How many sections can be generated at most by *n* straight lines with infinite length.


Intersecting all *n-1* existing lines to get as most sections as possible
Line *n*

$$L(0) = 1$$
$$L(n) = L(n-1) + n$$

Solution of Cutting the Plane

$$\begin{aligned} L(n) &= L(n-1)+n \\ &= L(n-2)+(n-1)+n \\ &= L(n-3)+(n-2)+(n-1)+n \\ &= \ldots\ldots \\ &= L(0)+1+2+\ldots\ldots+(n-2)+(n-1)+n \end{aligned}$$

$$\boxed{L(n) = n(n+1)/2 + 1}$$

上台阶

- 有10个台阶，一次可以上1个台阶或2个台阶，有多少种上台阶的方法？

$$f(n)=f(n-1)+f(n-2)$$

问题3：递归 (续)

- 你能写出解决以下问题的递归子程序吗？
  - Calculate the factorial of a natural number
  - Search a dictionary for a particular word
  - Traverse a filesystem

问题3：递归 (续)

- 既然所有递归子程序都能改写为循环，那为什么还需要用递归？
- 反之，所有循环都能改写为递归子程序吗？
  你能不能举个例子？