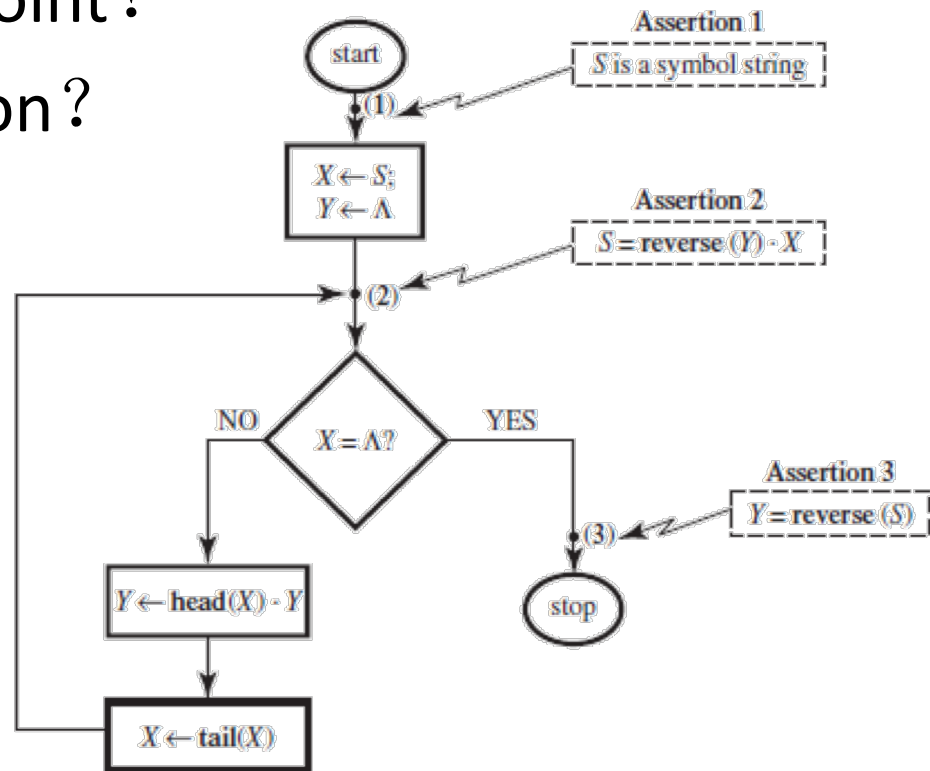- 教材讨论
  - DH第5章

问题1：程序设计中的错误

- 这些错误分别是什么意思？
  你犯过这些错误吗？说说你的教训
  如何避免/纠正这些错误？谈谈你的经验
  - Language error
  - Logical error
    - Semantic error
    - Algorithmic error
    - Run-time error
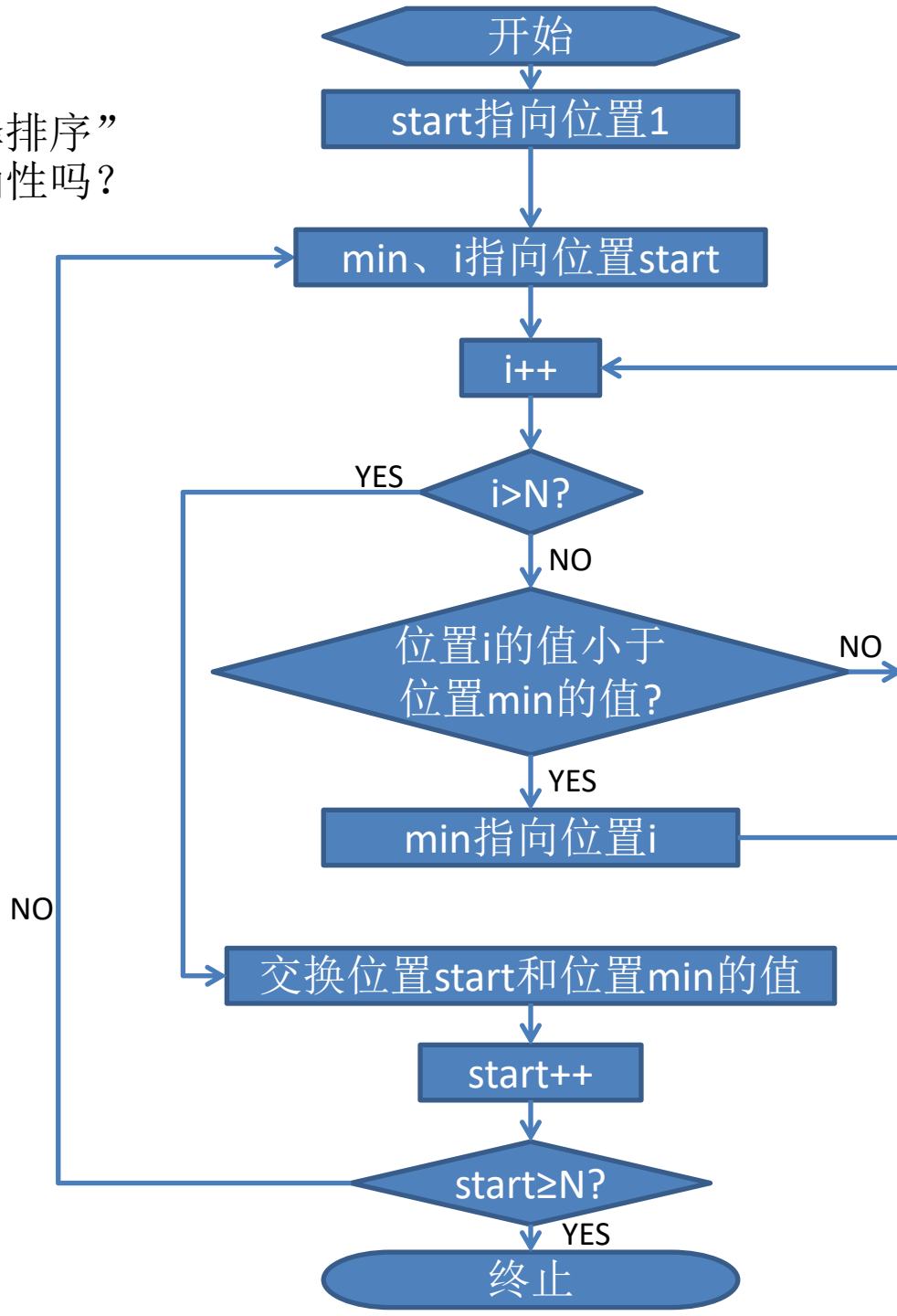   - Infinite loop

问题2：算法的正确性

- 你理解这些重要概念了吗？
  - Partially correct
  - Termination
  - Totally correct

问题3：算法正确性的证明

- 你能结合书上的这个例子，解释一下算法正确性证明的基本方法吗？
  - 在哪设置checkpoint？
  - 如何设置assertion？

你能证明"选择排序"
算法的完全正确性吗?

问题3：算法正确性的证明 (续)

- 通过上述证明过程，你是不是对as-you-go verification有了一些认识？

问题3：算法正确性的证明 (续)

- 你能结合书上的这个例子，解释一下带有递归的算法的正确性证明的基本方法吗？
  - 在哪设置checkpoint？
  - 如何设置assertion？

subroutine **move** N **from** X **to** Y **using** Z:

(1) if N is 1 then output "move X to Y";

(2) otherwise (that is, 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.

Assume that the peg names A, B, and C are associated, in some order, with the variables X, Y, and Z. Then, a terminating execution of the call **move** N **from** X **to** Y **using** Z lists a sequence of ring-moving instructions, which, if started (and followed faithfully) in any legal configuration of the rings and pegs in which at least the N smallest rings are on peg X, correctly moves those N rings from X to Y, possibly using Z as temporary storage. Moreover, the sequence adheres to the rules of the Towers of Hanoi problem, and it leaves all other rings untouched.

你能证明"计算树中节点深度之和"
算法的完全正确性吗？

```
int sum=0;
search (Node n, int depth) {
  sum+=depth;
  for (int i=0; i<n.childrenNum; i++) {
    search (n.child[i], depth+1);
  }
}
```

你能证明"计算树中节点深度之和"
算法的完全正确性吗？

```
int sum=0;

search (Node n, int depth) {

  sum+=depth;

  for (int i=0; i<n.childrenNum; i++) {

    search (n.child[i], depth+1);

  }

}
```

归纳假设：search(n,depth)将且仅将以深度为depth的节点n为根的、
包含节点数不超过N的子树中所有节点的深度累加到sum。
* 如何证明base case？
* 如何递推？