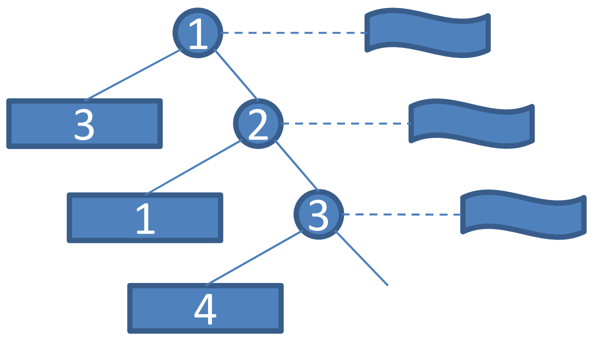
- 作业讲解
  - DH第4章练习1、2、8、9、11、12、13、14

DH第4章练习1

- (a) A[i,2] is the label of his or her manager.
  - if (A[i,1] > A[i,2]) …
  - 这样对吗？

- (b) 树的结构



DH第4章练习2

- 如何遍历一棵树

```
search (Node n) {
  for (int i=0; i<n.childrenNum; i++) {
    search (n.child[i]);
  }
}
CALL search(root);
```

DH第4章练习2a

- 节点深度之和

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

DH第4章练习2b

- 深度为K的节点数

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

DH第4章练习2c

- 是否有偶数深度的叶节点

```
bool answer=false;
search (Node n, int depth) {
  if (n.childrenNum==0 && depth%2==0) answer=true;
  for (int i=0; i<n.childrenNum; i++) {
    search (n.child[i], depth+1);
  }
}
CALL search(root, 0);
```

DH第4章练习11(b)

- 分治方法

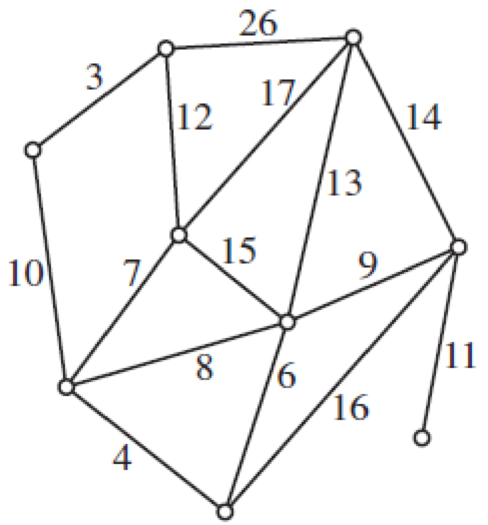
设数组A[1…N], N>2

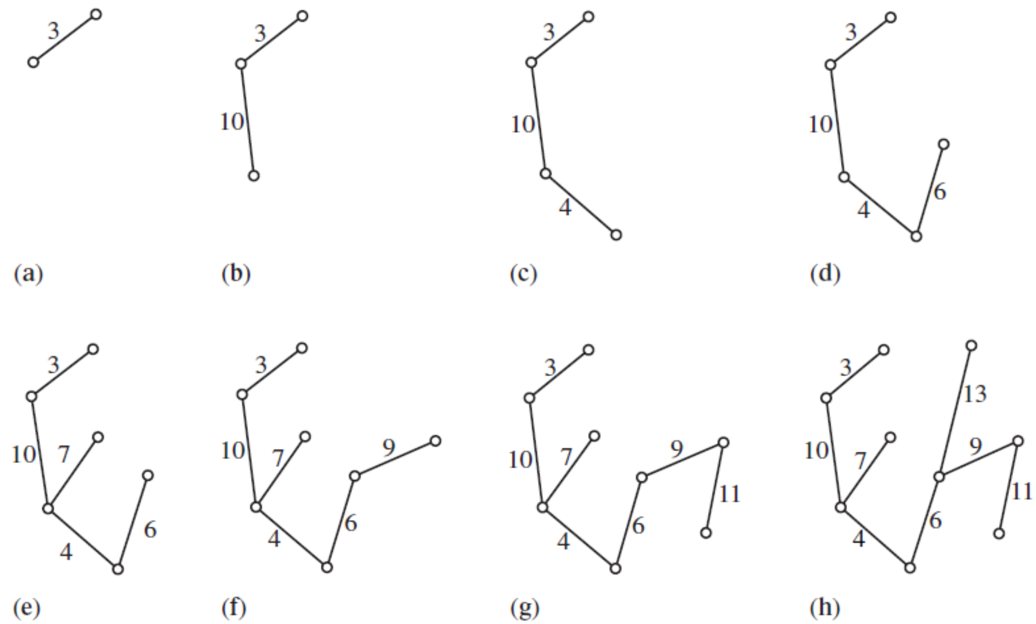
```
Search(A, L, R)
    if(R-L==2)
        if(A[R]>A[L]) swap(A[R], A[L])
        return A[L], A[R]
    else
        M=(L+R)/2
        MAX11, MAX12= Search(A, L, M)
        MAX21, MAX22= Search(A, M+1, R)
        return max(MAX11, MAX21), max(MAX12, MAX22)
```

这样对吗？

DH第4章练习12

- Kruskal算法：反复添加全局的最小边
- Prim算法：从任意点开始，反复添加相邻的最小边
  - 特例：教材中的算法





DH第4章练习13

- 0-1 Knapsack

$$f_m(\hat{c}) = \begin{cases} f_{m-1}(\hat{c}) & \text{for } \hat{c} = 0, \ldots, w_m - 1; \\ \max\ (f_{m-1}(\hat{c}), f_{m-1}(\hat{c} - w_m) + p_m) & \text{for } \hat{c} = w_m, \ldots, c. \end{cases}$$

- Bounded Knapsack

$$f_m(\hat{c}) = \max\{ f_{m-1}(\hat{c} - l w_m) + l p_m : l \text{ integer},\ 0 \le l \le \min(b_m, \lfloor \hat{c}/w_m \rfloor)\}$$

  – 或者：转换成0-1 Knapsack

- 教材讨论
  - DH第5章

问题1：程序设计中的错误

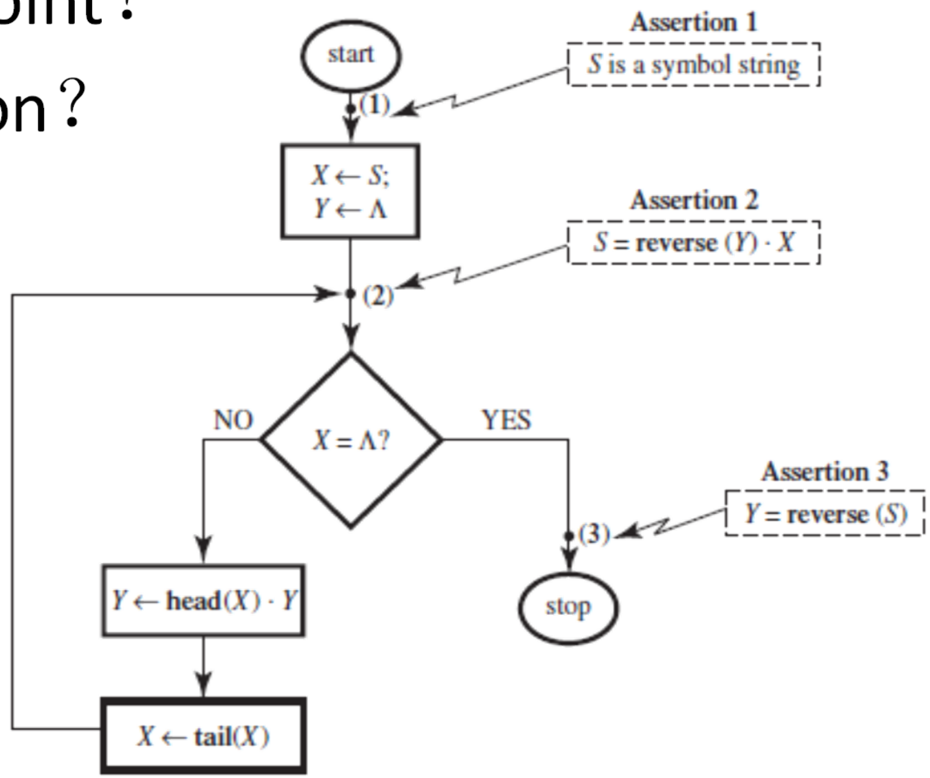
- 这些错误分别是什么意思？
  你犯过这些错误吗？说说你的教训
  如何避免/纠正这些错误？谈谈你的经验
  - Language error
  - Logical error
    - Semantic error
    - Algorithmic error
    - Run-time error
    - Infinite loop

问题2：算法的正确性

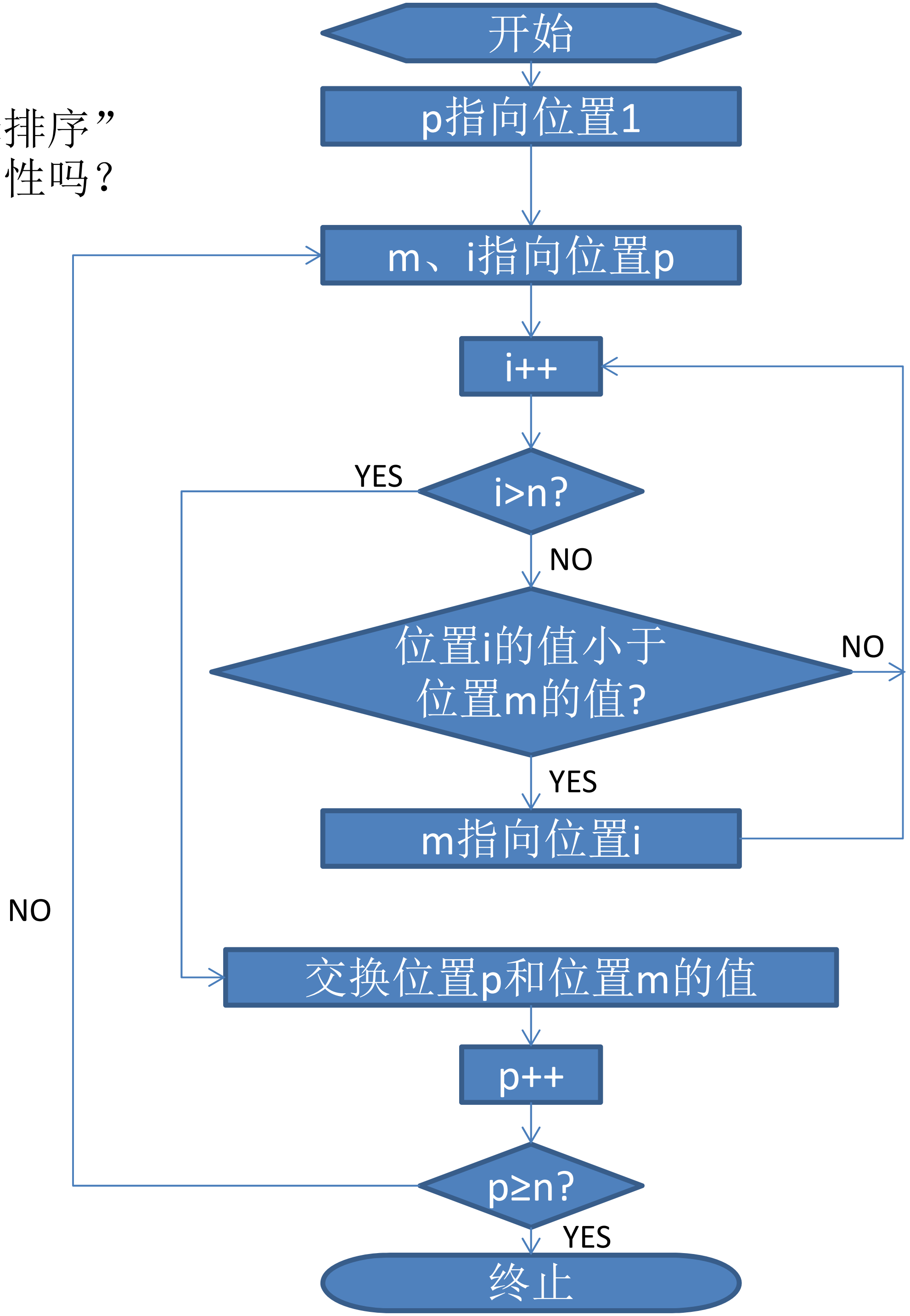
- 你理解这些概念了吗？
  - Partially correct
  - Termination
  - Totally correct

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);
  }
}
```