计算机问题求解－论题2-8
- 基本的数据结构

TC-10、MA-2/3/4.1/4.2

Part I
抽象与实现

“动态集合”与“字典结构”

定义在“一组”数据上的典型操作：

- 创建: 定义一个“具体”的“结构”

- 搜索：在“结构”中搜索指定关键字（可能找不到）
- 插入：将给定（比如用指针）的（已初始化的）元素插入结构
- 删除；将结构中指定（比如用指针）的元素删除，保持结构

（以下的操作不改变结构，适用于元素属于全序集的情况）

- 查询最小值：仅对元素属于全序集的情况有效
- 查询最大值：仅对元素属于全序集的情况有效
- 查询制定元素的前驱元素（按关键字排序），可能不存在
- 查询指定元素的后继元素（按关键字排序），可能不存在

问题1：结构体现在那里？

问题2：

你能解释下列任务在计算机中高效实现的关键吗？

1. 你用鼠标连续点击了屏幕几个不同的按钮，计算机总能够一一顺序执行。

2. 你输入计算式2+3*5，计算机会正确的算出结果17，而不是像简单计算器上只能得到25。

“数据结构”：首先是关乎解题，而不是关于“存放”

栈 – “后进先出” (LIFO)

“可用”的操作集：

创建： s=create(): s指向一个新的空栈

查询是否为空：isEmpty(s)

元素进栈：push(s, x): isEmpty(s)=false, top(s)=x

元素出栈：t=pop(s): s为空则出错，t=top(s), 从栈中删除x

注意：

这里的定义并不涉及数据如何存放，当然就不涉及对于具体存放方式，结构的改变如何实现。

在抽象动态集合的基础上增加了一条“栈公理”（甚至不需要“top”有特定含义）

连续执行的一对push-pop不影响原来的结构

显然：从栈被创建，任意“合法”的push/pop序列中pop数不能大于push数

队列 – “后进先出” (FIFO)

“可用”的操作集：

创建： q=create(): q指向一个新的空队列

查询是否为空：isEmpty(q)

元素进队列：enqueue(q, x): isEmpty(q)=false,

如果操作前isEmpty=true, front(q)=x

否则：front(q)保持原值不变

元素出队列：t=dequeue(q): s为空则出错，t=front(q), 从栈中删除front(q),
并按下面的说明修改front(q)

注意：

“队列公理”：（甚至不需要“front”有特定含义）

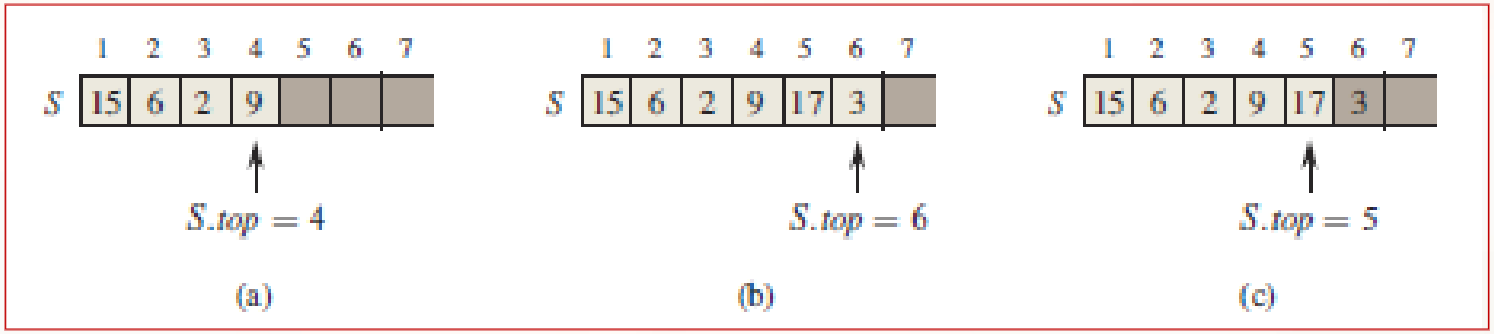
- 从队列被创建后的操作序列中第k个enqueue-dequeue操作对删除不影响整个操作序列的执行结果队列状态

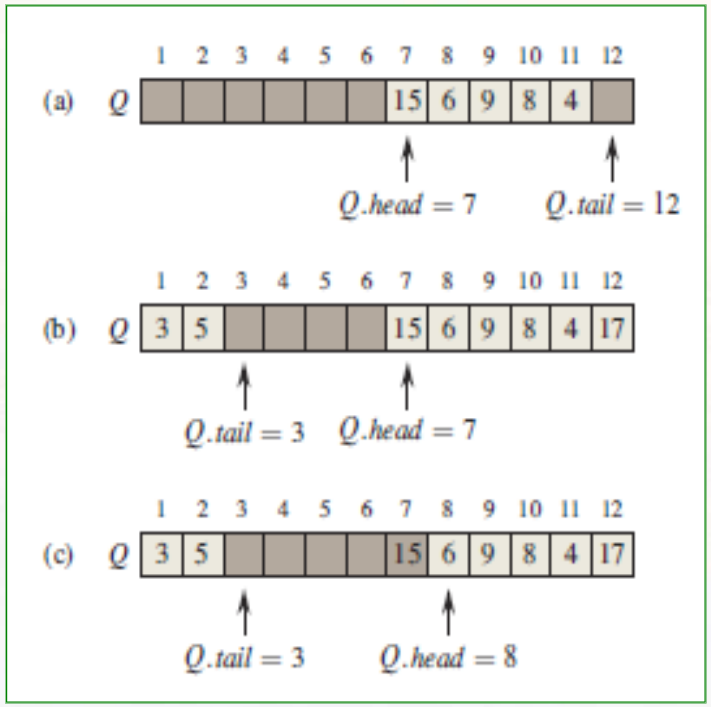
显然：队列被创建后，任意“合法”的操作序列中dequeue数不大于enqueue数，如从该序列中删除所有匹配的enqueue/dequeue操作，余下的一定不含dequeue，此序列对应的front(q)就是从队列中删除元素后新的front(q)

问题3：

什么是数据结构的“实现”(implementation)？程序设计语言如何影响实现方法？

基于指针的实现方法





问题4：

你根据什么区分这两个图各自描述哪种结构？

逻辑结构与数据存放在何处
并没有直接关系。

栈与队列：效率

问题5:

阅读材料上说栈与队列的基本操作代价均为O(1)，是什么意思？

问题6：

栈与队列是否可以“相互”提供实现方式？如何分析其效率？

“实现”本身并没有抽象层次高低的概念？

问题7:

按前面图示的方式，为什么存储区含有n个单元的队列只能放$n-1$个元素？

问题8：

“空”和“满”该如何判别？

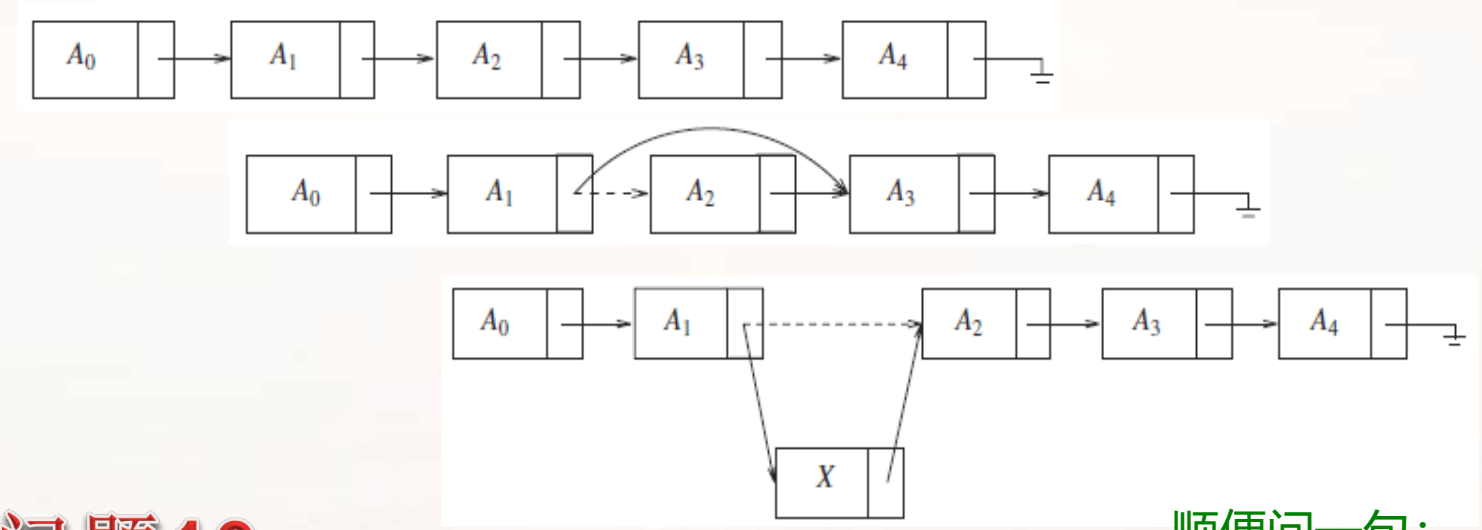
从抽象数据类型与实现角度看，有什么不同?

问题9：
栈与队列是线性结构吗？

在数组中插入元素有什么难处？这对于用它来实现“真正”的线性结构有什么影响？

Part II
线性结构与层次结构

链表：内在的顺序结构



问题10：

链表可以提供动态集合的所有操作，你能解释一下吗？

顺便问一句：

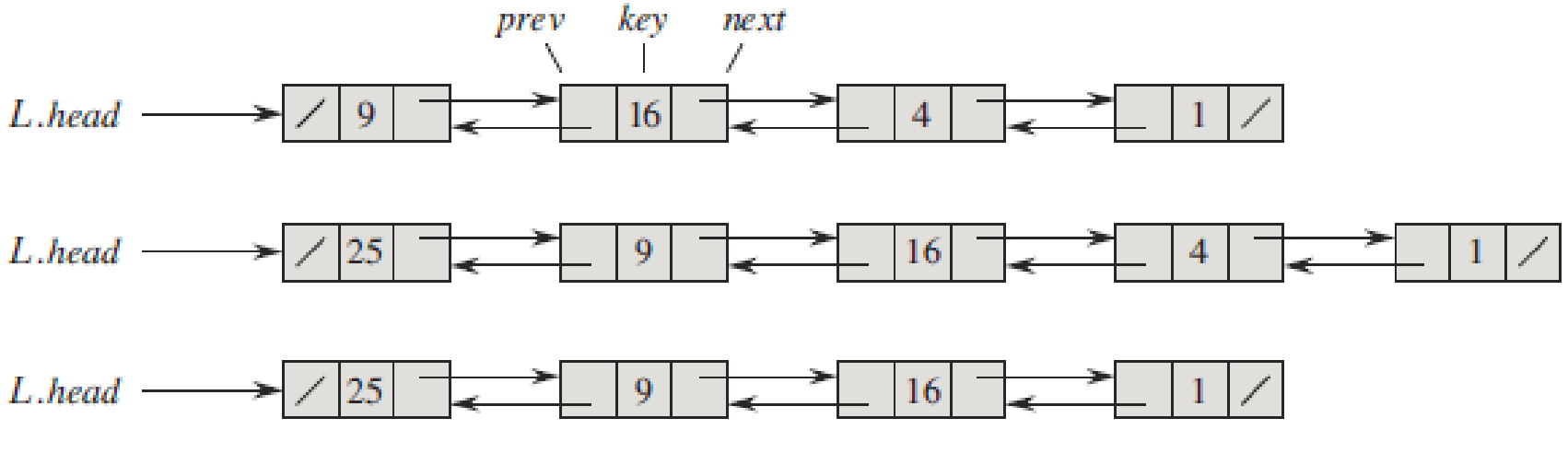
讨论数据结构时“given an object x”和搜索操作中提供的参数 k 有什么不同。对操作代价有何影响？

无序双向链表

这里的无序是对什么而言的?

问题11：

下面的操作哪个代价最大？




```
LIST-SEARCH(L, k)
1  x = L.head
2  while x ≠ NIL and x.key ≠ k
3      x = x.next
4  return x
```

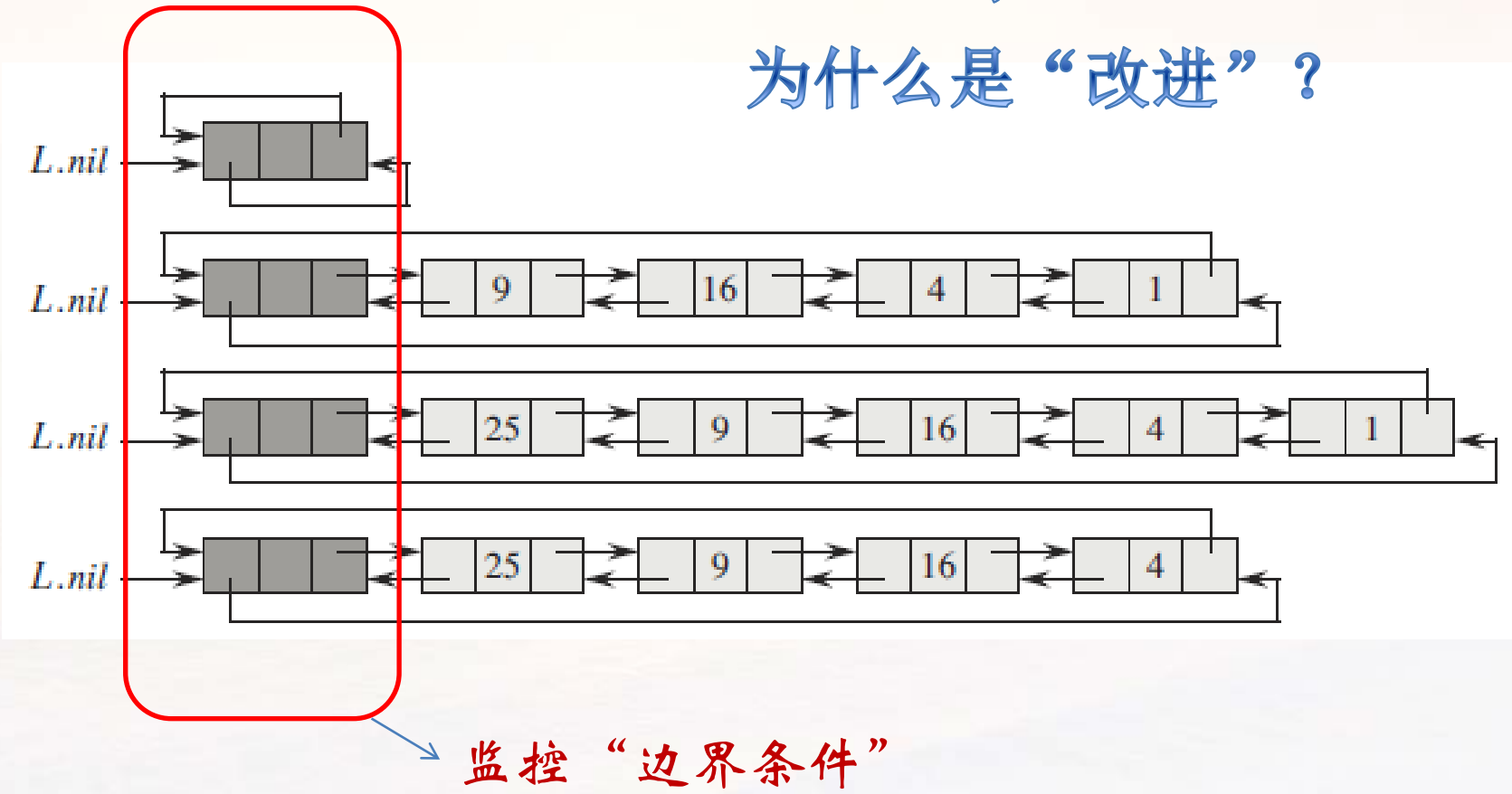
```
LIST-INSERT(L, x)
1  x.next = L.head
2  if L.head ≠ NIL
3      L.head.prev = x
4  L.head = x
5  x.prev = NIL
```

```
LIST-DELETE(L, x)
1  if x.prev ≠ NIL
2      x.prev.next = x.next
3  else L.head = x.next
4  if x.next ≠ NIL
5      x.next.prev = x.prev
```

边界条件

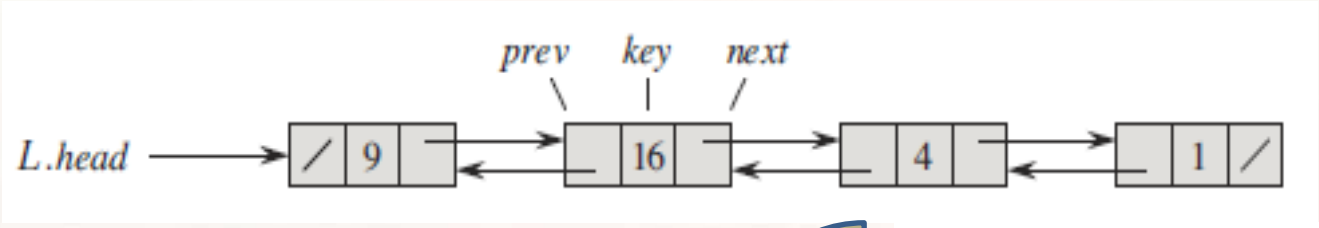
问题12;

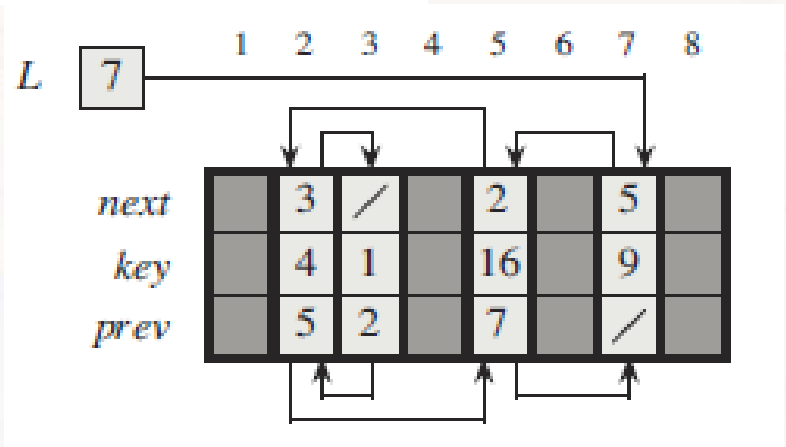
为什么是“改进”？

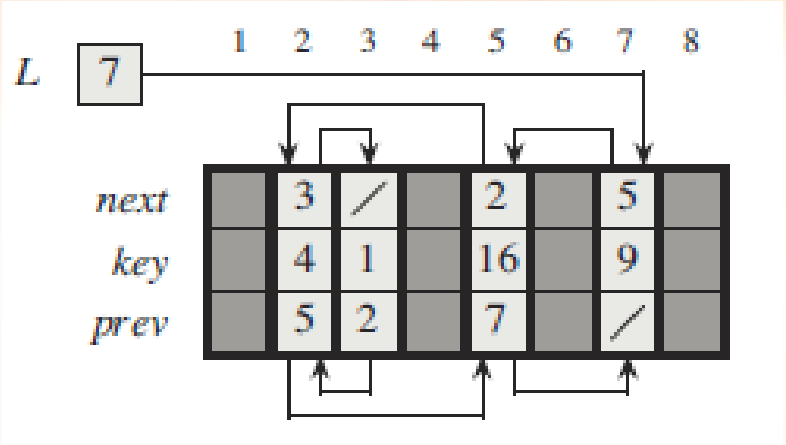


“指针”不一定非得用指针实现

- 通常我们熟悉的第一个“指针”是某种程序设计语言中提供的“语言设施”。
- 在用“指针”方法实现一种数据结构时，所谓的“指针”未必是语言中的指针，可以是实现其目的的其他手段。

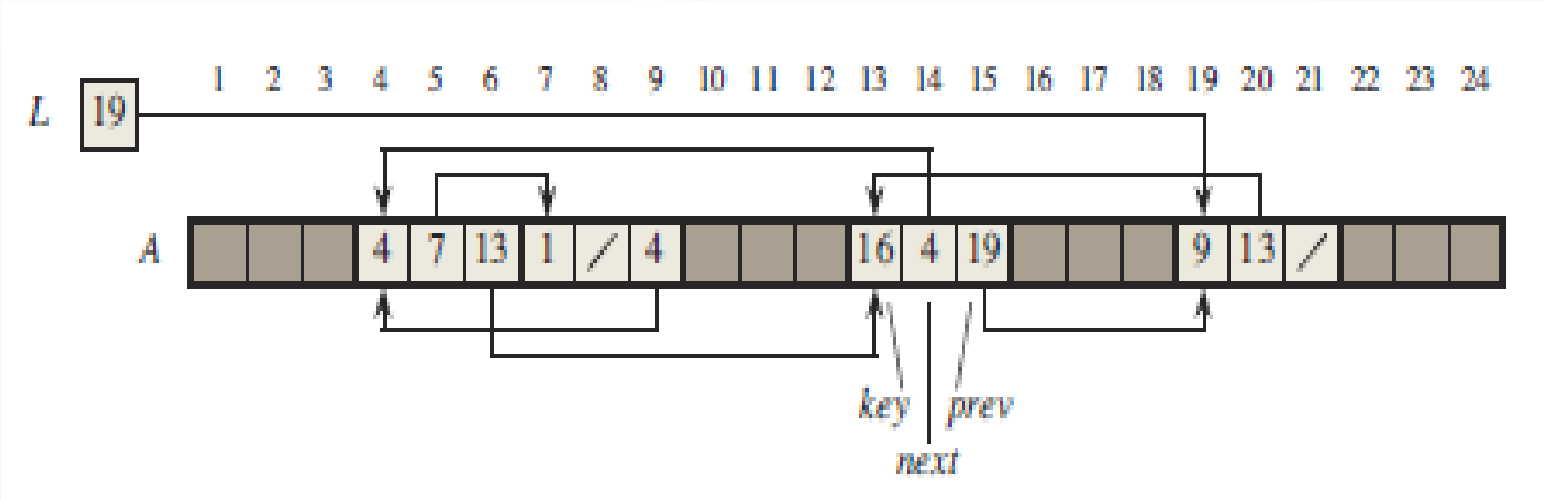






问题13:

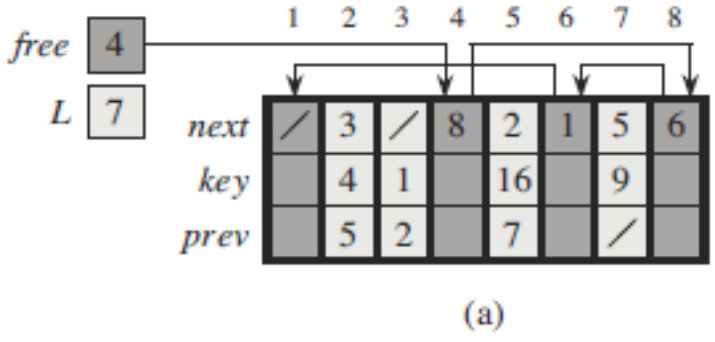
这两个图所示方法有什么不同?



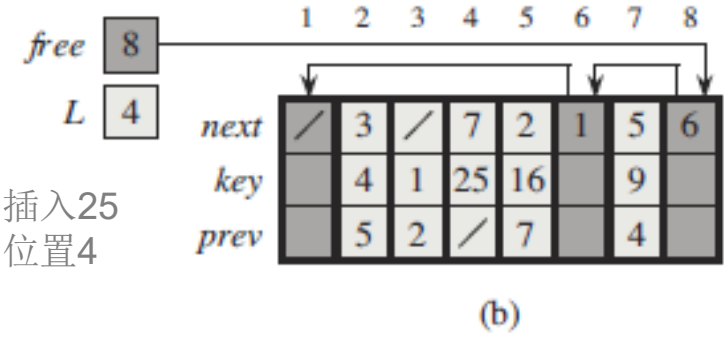
顺便学会两个英语单词：**homogeneous/heterogeneous**

动态结构的存储管理

- 基本原理：在一个数组空间维护两个链表，“实际数据”和“可用空间”。“此消彼长”

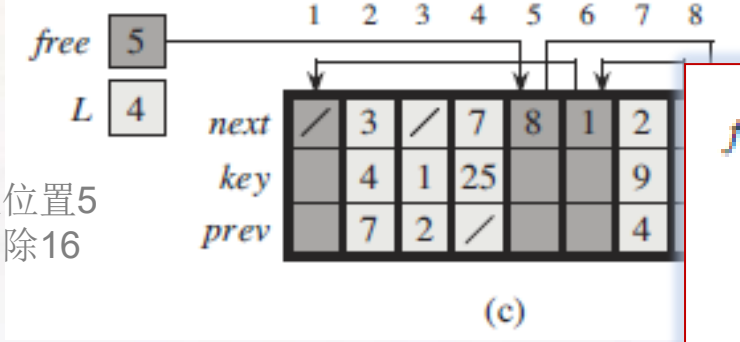


(a)



插入25
位置4

(b)

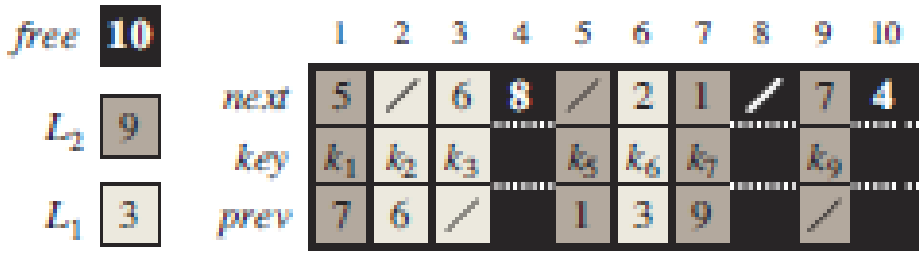


从位置5
删除16

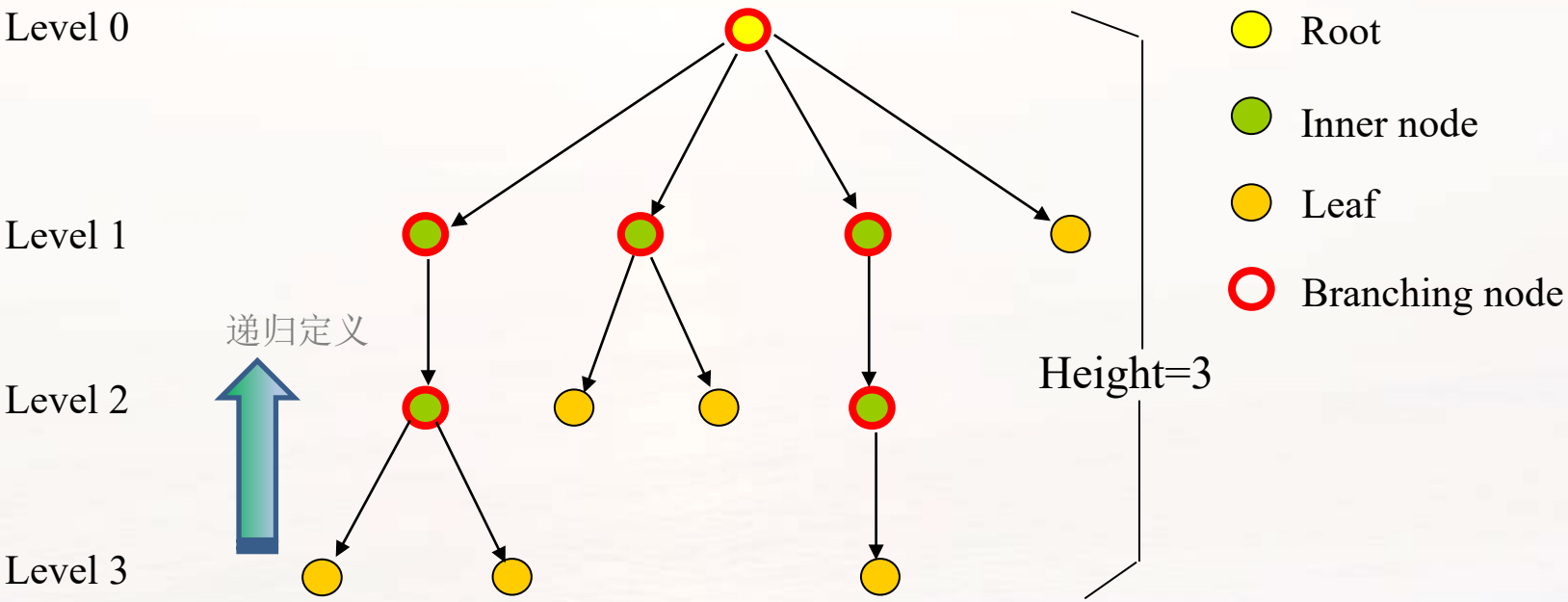
(c)

问题14：

顺便问一下：

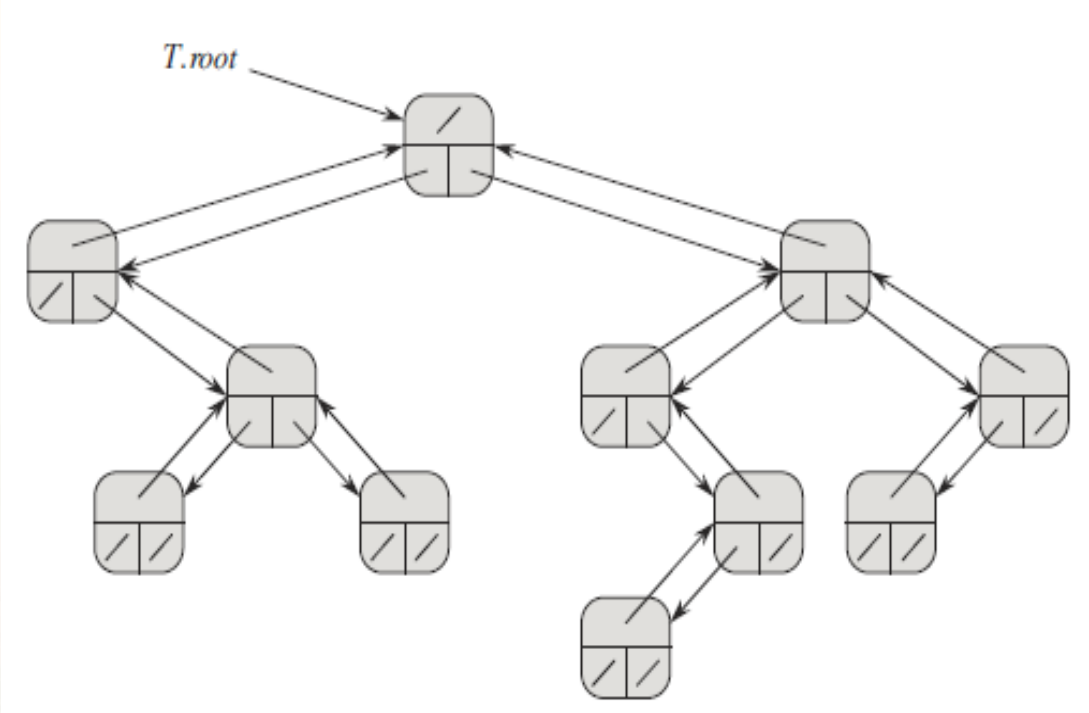


根树: 从逻辑层次到指针实现



如果任意结点最多有两个子结点, 则该根树成为二叉树(binary tree), 显然用指针实现链表的方法很容易扩展到二叉树。

用指针实现 Binary Tree

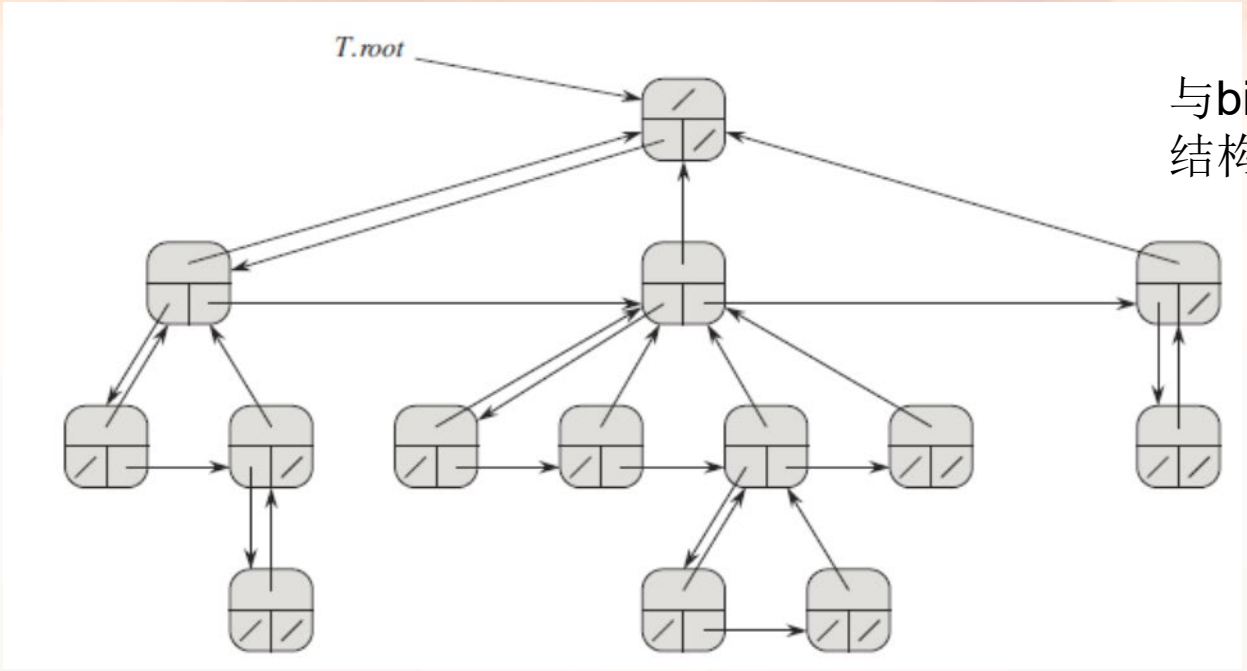


问题15：

你能与链表的指针实现做个比较吗？

问题16：

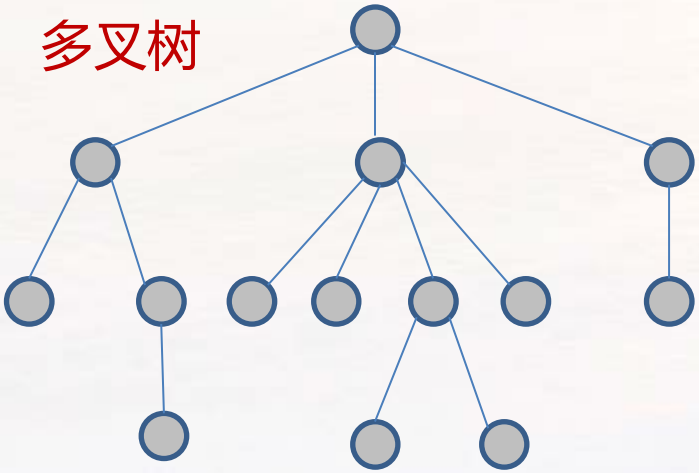
如果每个结点的子结点不限于一个很小的常量（例如2），用指针实现根树结构有什么困难？该如何解决？



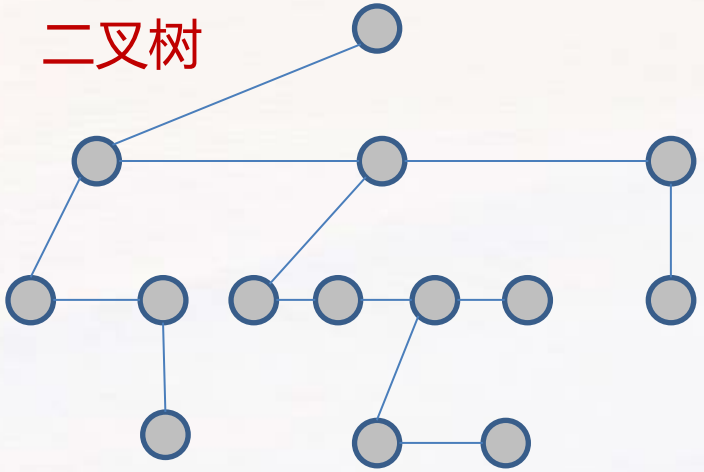
与binary tree的指针结构有什么不同？

注意:
此过程
不可逆!

多叉树



二叉树



这是数学意义下的变换，与指针结构并不对应