数据结构 期中考试
中厅讲座

杨乐

专题1 - 时间复杂度分析

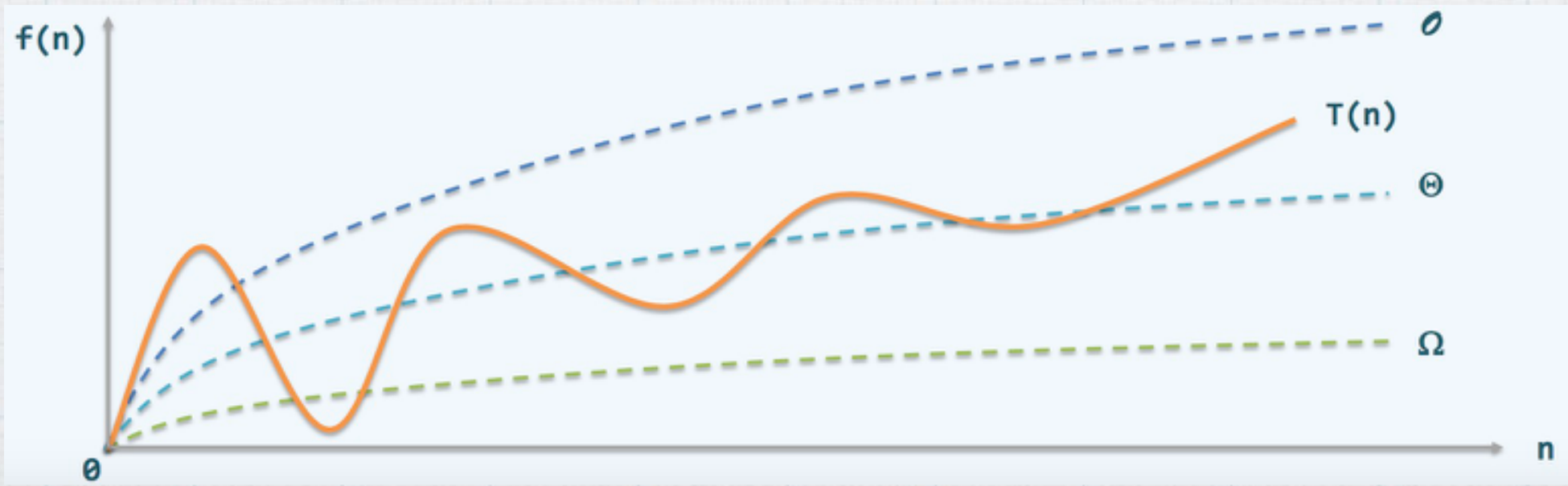
* 三种时间复杂度记号：

$$T(n) = \mathcal{O}(f(n)) \quad iff \quad \exists\, c > 0 \quad s.t. \quad T(n) < c \cdot f(n) \quad \forall\, n \gg 2$$

$$T(n) = \Omega(f(n)) \quad iff \quad \exists\, c > 0 \quad s.t. \quad T(n) > c \cdot f(n) \quad \forall\, n \gg 2$$

$$T(n) = \Theta(f(n)) \quad iff \quad \exists\, c_1 > c_2 > 0 \quad s.t. \quad c_1 \cdot f(n) > T(n) > c_2 \cdot f(n) \quad \forall\, n \gg 2$$

* 图像辅助记忆



P27-48

专题1 - 时间复杂度分析

* 大O记号，T(n) = O(f(n))，“T(n) < f(n)”，上界

* 大Ω记号，T(n) = Ω(f(n))，“T(n) > f(n)”，下界

* 正确 $f(n) = O(g(n))$，当且仅当 $g(n) = \Omega(f(n))$。

P27-48

专题1 - 时间复杂度分析

* 题型1 —— 判断题
* 直接按照“大小比较”代入即可
* 判断时: 找特例!!!
* 运算十分严格, 基本上很少是对的

P27-48

专题1 - 时间复杂度分析

* 判断:

若 $f(n) = O(n^2)$ 且 $g(n) = O(n)$，则下列结论正确的是：注：O 为上界，没有明确究竟是多少；故不能使用除法

A. $f(n)+g(n) = O(n^2)$　　B. $f(n)/g(n) = O(n^2)$　　C. $g(n) = O(f(n))$　　D. $f(n)*g(n) = O(n^3)$

P27~48

专题1 - 时间复杂度分析

* 重点: 时间复杂度计算！！

* 1 循环 — 分层计算，看清步进
* 2 递归 — 嵌套代入，或公式计算
* 3 随机 — 期望，乘上概率

专题1 - 时间复杂度分析

* 1 循环 — 分层计算，看清步进

```
void F(int n)                    //O(    sqrt(n)    )
{
    for (int i = 0, j = 0; i<n; i+=j, j++);
}
```

```
void F(int n)                    //O( nlog(n))
{ //  调和级数
    for (int i=1; i<n; i++)
        for (int j=0; j<n; j+=i);
}
```

```
void F(int n)                    //O(loglogn )
{ //同理第一题：改+1 为乘 2
    for (int i = 1, r = 1; i<n; i<<=r, r<<=1);
}
```

专题1 - 时间复杂度分析

* 2 递归 — 嵌套代入，或公式计算

```
void  F(int  n)                          //O(1.618^(logn))  =
O(n^0.694)
{// 转乘法为加法，fibonacii
      return (n<4) ? n : F(n>>1)+F(n>>2);
}
```

```
int F(int n)                  //O(2^n)
{ // 再回头看 F，  F = G(2, G(2, G(2, ..., G(2, 1)))
      return (n==0) ? 1 : G(2, F(n-1));
}
int G(int n, int m)
{ // 先分析 G, G = O(m), G(n, m) = n * m
      return (m==0) ? 0 : n+G(n, m-1);
}
```

```
int F(int n)                  //O(n^2)
{// 再分析 F, G((n-1)*(n-1)) = O((n-1)^2)
      return G(G(n-1));
}
int G(int n)
{// 先分析 G，  G=O(n), G(n) = n*n
      return (n==0) ? 0 : G(n-1)+2*n-1;
}
```

P27-48

专题1 - 时间复杂度分析

* 3 随机 — 期望，乘上概率

```
void F(int n) //expected-O(log^2n)
{//  级数求和
      for (int i=1; i<n; i++)
            if(0 == rand()%i)
                  for (int j=1; j<n; j<<=1);
}
```

P255-277

专题2 - 排序算法

* 四种排序算法 、 排序理论
* 起泡排序P59：n-1趟扫描、每一趟比较相邻的、把小的放到前面
* 选择排序P255：找n次，每次找最大，放到最后
* 插入排序P270：顺序插入n个数
* 归并排序P277：分治，递归两个子序列进行排序

P255-277

专题2 - 排序算法

* **要点：比较次数、交换次数**

* **选择排序**相对于**起泡排序**的优点：**比较次数不变、交换次数减少**

* **选择排序、起泡排序**：比较次数均为O(n^2)，交换次数不同

P255-277

专题2 - 排序算法

* **要点：最优情况、最差情况**

* **插入排序**相对于**选择排序**的优点：**在最优情况下，比选择排序要好（选择时间是固定的、插入是根据输入规模的）**

* **冒泡排序**在**最优情况**下，也比选择排序要快

P255-277

专题2 - 排序算法

* **要点：最优情况、最差情况**

7.（ 0 ）只要是采用基于比较的排序算法，对任何输入序列都至少需要运行 $\Omega(n \log n)$ 时间。注：插入排序对原本就有序的序列排序，时间是 O(N)（注意题目中所说的是“对任何输入序列”）

* 错误：对**插入排序**不成立

P255-277

专题2 - 排序算法

* 要点：时间复杂度分析

* 选择排序：排List，与循环节长度有关

* 每次会减少循环节的总长度1

* 实际上是没什么影响的（lnn/n->0）

P255-277

专题2 - 排序算法

* 要点：时间复杂度分析

* 插入排序：就地算法、在线算法、具有输入敏感性、有最好/最坏情况

* 二分查找？改用向量、用“比较”换“交换”？总体不变

* 平均比较次数分析

P255-277

专题2 - 排序算法

* 要点：时间复杂度分析

* 插入排序：时间复杂度O(N+I)

* 时间取决于逆序对数量（I），输入敏感

* 怎么算逆序对？归并排序！

P255-277

专题2 - 排序算法

* 题目

1. 考察如下问题：任给 12 个互异的整数，且其中 10 个已组织为一个有序序列，现需要插入剩余的两个已完成整体排序。若采用基于比较的算法（CBA），最坏情况下至少需要做几次比较？为什么？

答：8 次。 我们知道对于 CBA,我们可以将其涵盖于一棵比较树里边，而树的每一个节点可以代表一次比较运算，树的分支可以代表算法下一步执行的方向。由此可以推算，树高则可以代表比较的次数。

注：类似排序复杂度分析：总情况数为 11*12 = 132. 故二叉树至少要有 132 个儿子，高度至少为 8.（2^8 = 256 > 132）

P255-277

专题2 - 排序算法

* 题目

2. 向量的插入排序由 n 次迭代完成，逐次插入各元素。为插入第 k 个元素，最坏情况需要做 k 次移动，最好情况则无需移动。从期望的角度来看，无需移动操作的迭代次数平均有多少次？为什么？

假定个元素是等概率独立均匀分布的。

答：logn。调和级数

注：总次数 = 1 + 1/2 + 1/3 + 1.4 + … + 1/n = n log n, 平均为 log n.

P172-197

专题3 - 二分查找\Fib查找

* 要点：二分查找三个版本

* Fib查找是什么？为什么？怎么做？

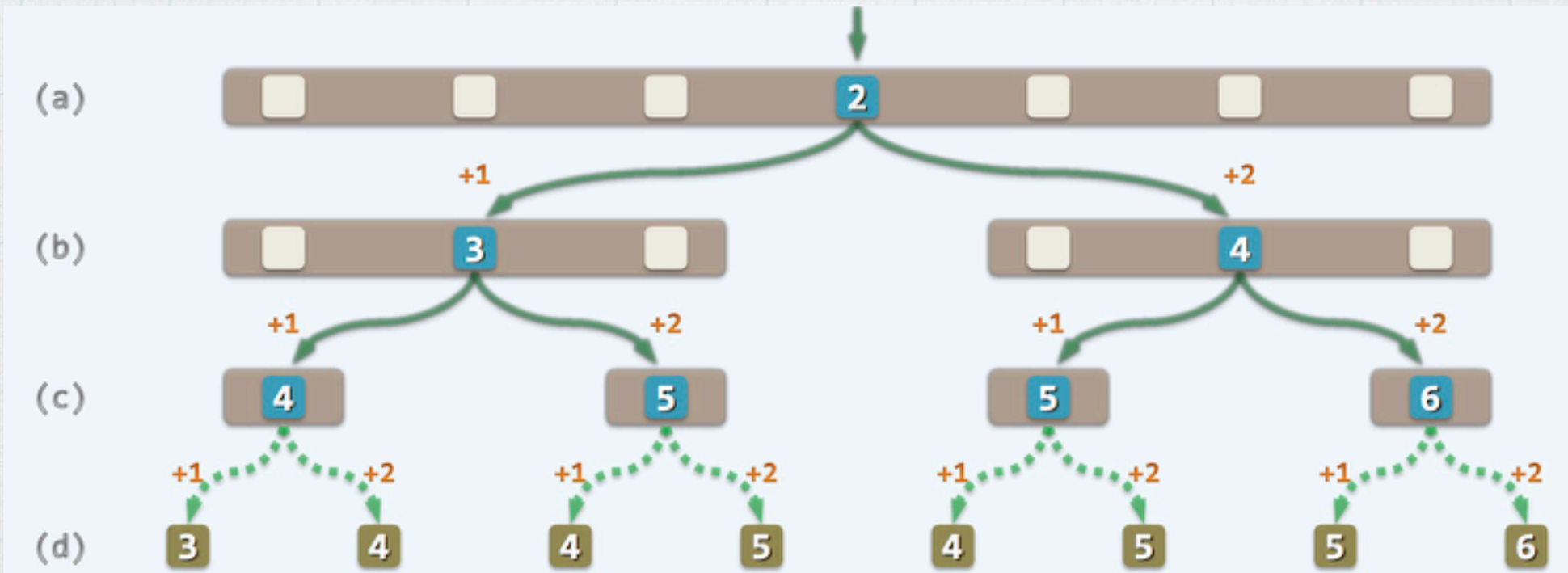
P172-197

专题3 - 二分查找\Fib查找

* 二分查找 —— 基于“向量”非“列表”

* 有序，可“比较”非“比对”

* 成功情况n种，失败情况n+1种

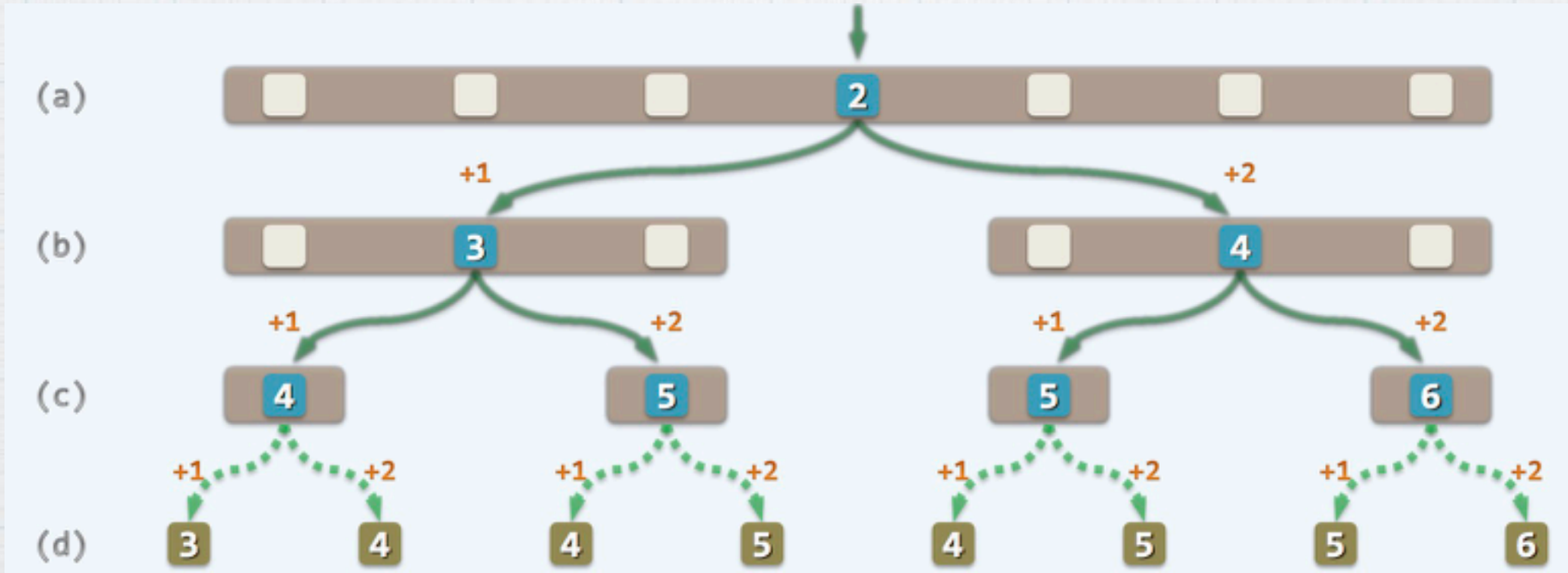


P172-197

专题3 - 二分查找\Fib查找

* 二分查找 ： 比较次数（向左走+1，向右走+2，根=2）
* (平均/最大/最小)成功查找长度
* (平均/最大/最小)失败查找长度

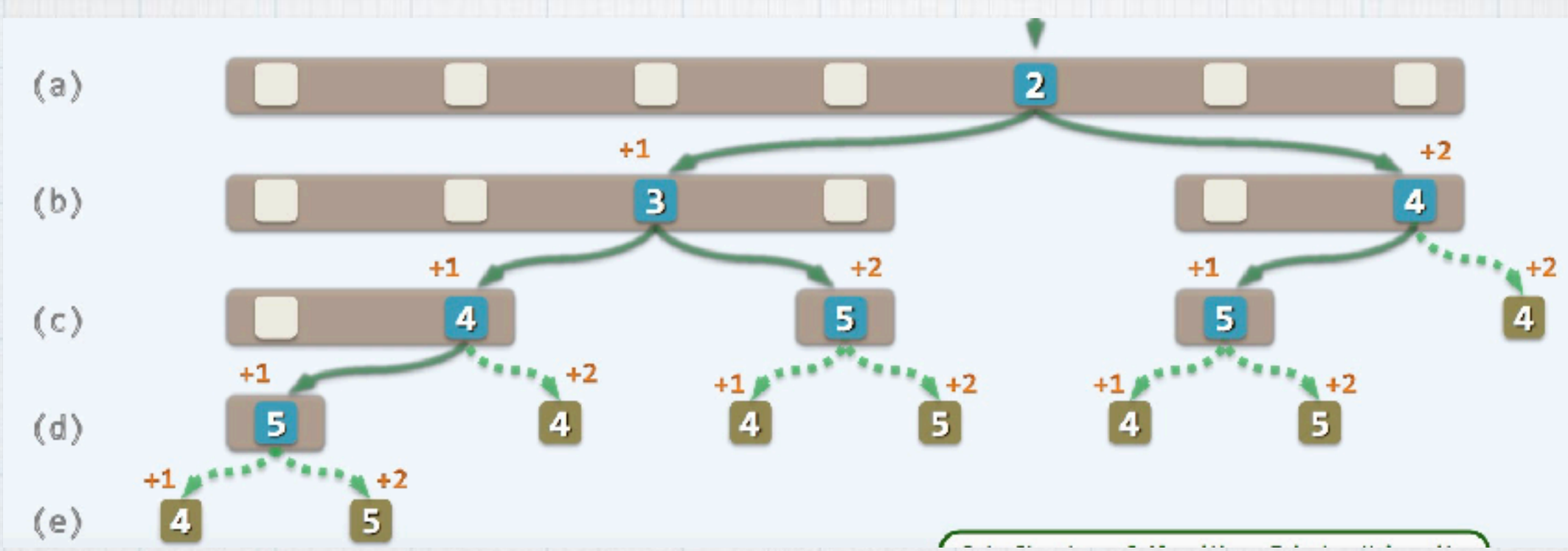
* 至多比较次数



P172-197

专题3 - 二分查找\Fib查找

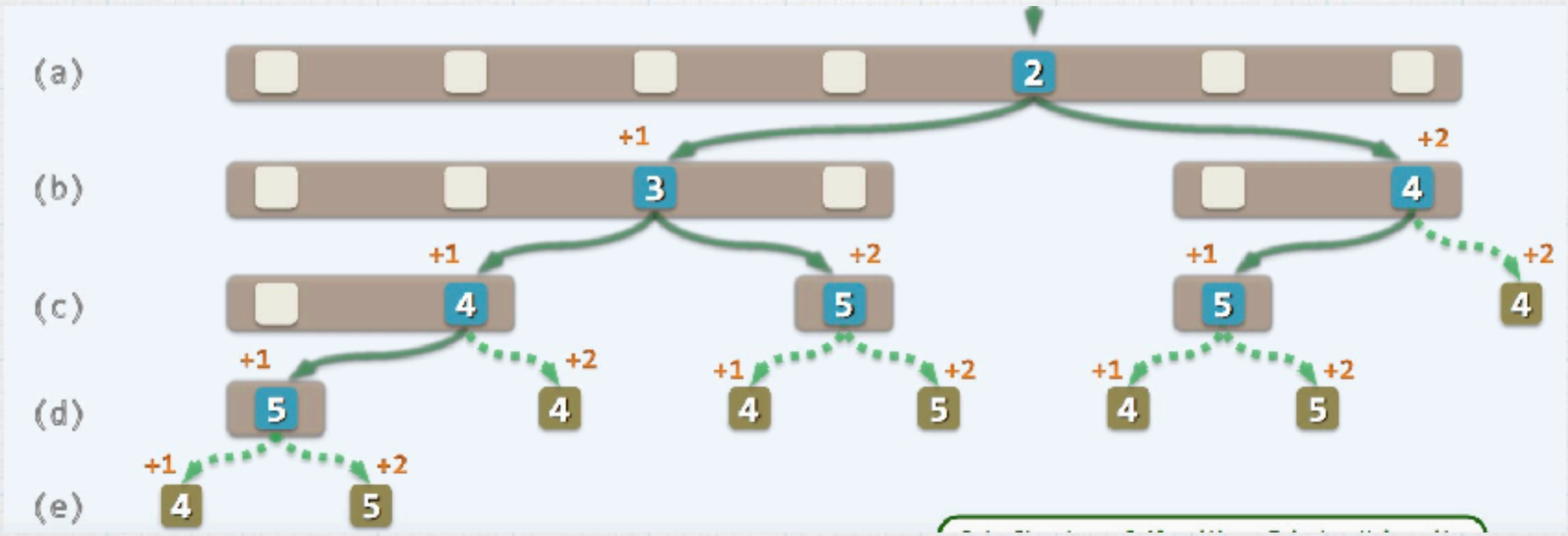
* Fib查找
* 为什么：左右次数差距太大、比较次数不均匀
* 怎么做：把左边区间调大（0.618）



P172-197

专题3 - 二分查找\Fib查找

* Fib查找：比较次数（例子为n=Fib(k)-1,7=Fib(6)-1）
* 平均成功查找长度：【k-2】4
* 平均失败查找长度：【n(k-1)/(n+1)】35/8 = 4.38
* 至多比较次数：【k-1】5



P172-197

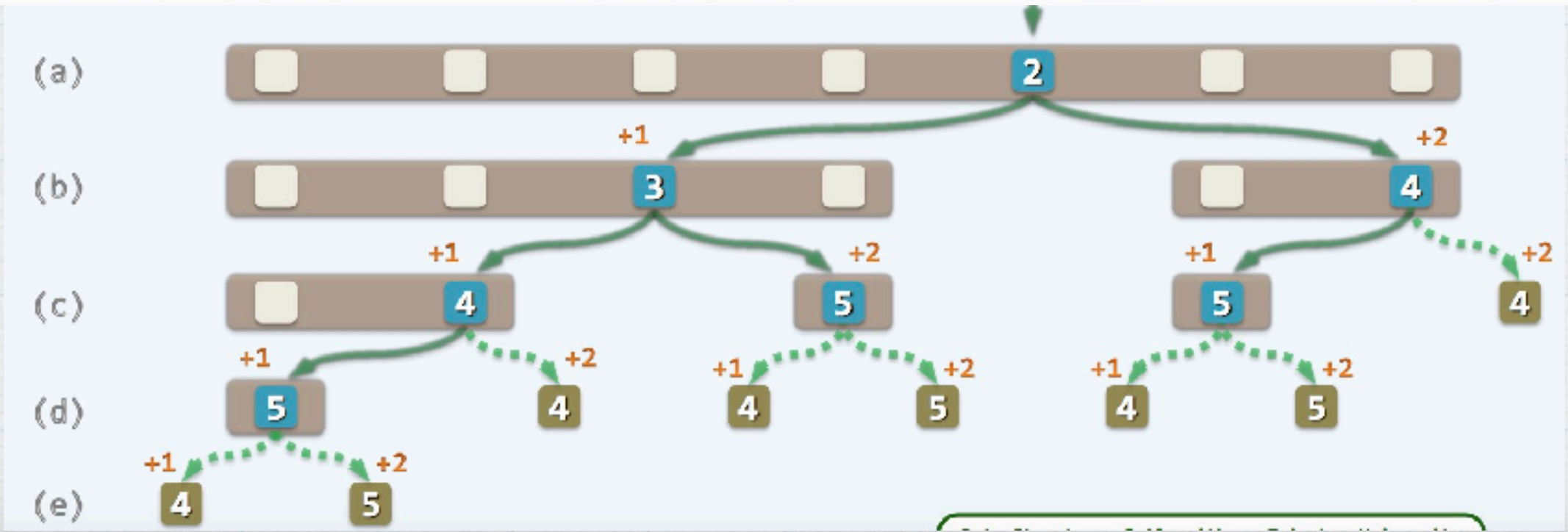
专题3 - 二分查找\Fib查找

* Fib查找：比较次数（例子为n=Fib(k)-1,7=Fib(6)-1）

3.（ B ）对长度为 $n = Fib(k) - 1$ 的有向序列做 Fibonacci 查找。若个元素的数值等概率独立均匀分布，且平均成功查找长度为 L，则失败平均查找长度为：（举例子 or 习题解析 P46）

注：P184：平均成功查找长度 L=k-2，平均失败查找长度 n(k-1)/(n+1) = n(L+1)/(n+1)，比较次数至多为 k-1.

A. n(L-1)/(n-1)　　B. n(L+1)/(n+1)　　C. (n-1)L/n　　D. (n+1)L/n



P172-197

专题3 - 二分查找\Fib查找

* 二分查找三个版本

* 版本A：正常，比较+比对（3分支）

* 版本B/C：只比较（2分支）

* 好(坏)的情况会更坏(好)

P172-197

专题3 - 二分查找\Fib查找

* 题目

3.（ 0 ）若借助二分法查找确定每个元素的插入位置，向量的插入排序只需时间$O(n\log n)$

时间。注：单次查找是 logN，但单次插入则不一定（最好 O(1)，最坏 O(N)），故插入排序还是 O(N^2)的

8.（ 0 ）对于同一有序向量，每次折半查找绝不会慢于顺序查找。注：每次找第一个，顺序 O(1)，折半 O(logN)

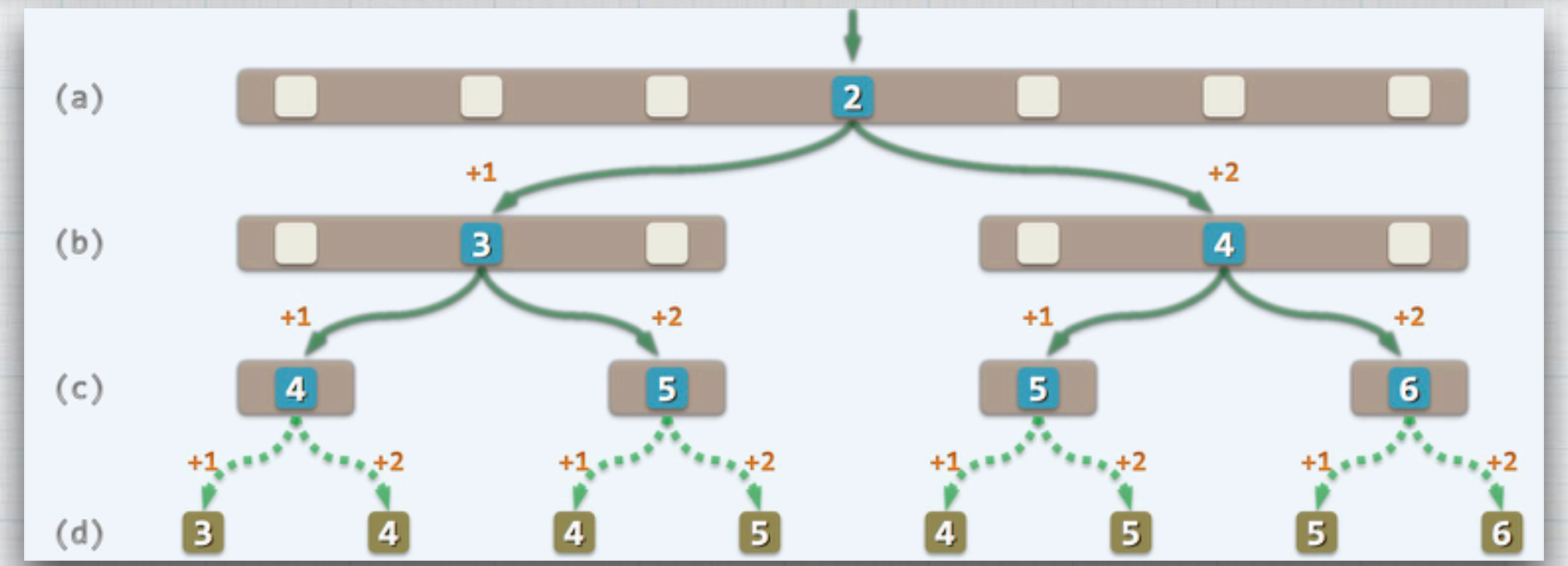
P172-197

专题3 - 二分查找\Fib查找

3. 现有一长度为 **15** 的有序向量 **A[0…14]**，个元素被成功查找的概率如下：

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$P_i(\sum =$	1/128	1/128	1/32	1/8	1/8	1/32	1/16	1/16	1/128	1/64	1/16	1/4	3/16	1/128	1/64

若采用二分查找算法，试计算该结构的平均成功查找长度。



P172-197

专题3 - 二分查找\Fib查找

* 题目

3. 现有一长度为 15 的有序向量 A[0...14]，个元素被成功查找的概率如下：

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$P_i(\sum =$	1/128	1/128	1/32	1/8	1/8	1/32	1/16	1/16	1/128	1/64	1/16	1/4	3/16	1/128	1/64

若采用二分查找算法，试计算该结构的平均成功查找长度。

L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Length	5	4	6	3	6	5	7	2	6	5	7	4	7	6	8

平均查找长度= 5/128 + 4/128 + 6/32 + 3/8 + 6/8 + 5/32 + 7/16 + 2/16 + 6/128 + 5/64 +

7/16 + 4/4 + 7/16*3 + 6/128 + 8/64 = 659/128

P345

专题4 - 栈与RPN

* 栈混洗：卡特兰数
* RPN：栈的一个重要应用

P345

专题4 - 栈与RPN

* RPN：题目
* RPN中各操作数的相对次序，与原中缀表达式完全一致。（正确）

4. 考察表达式求值算法。算法执行过程中的某时刻，若操作符栈中的括号多达 2010 个，则此时栈的规模（含栈底的'\n'）至多可能多达？试说明理由，并示范性地画出当时栈中的内容。

答：4 * 2010 + 1 + 4 = 8045(考虑乘方操作后)、（注意+1，有阶乘操作！）

栈中内容：\0 + * ^ (+ * ^ (+ * ^ ... (+ * ^ !