2.向量

（d3）有序向量：Fibonacci查找

邓俊辉

deng@tsinghua.edu.cn

思路及原理

- 二分查找版本A的效率仍有改进余地，因为不难发现

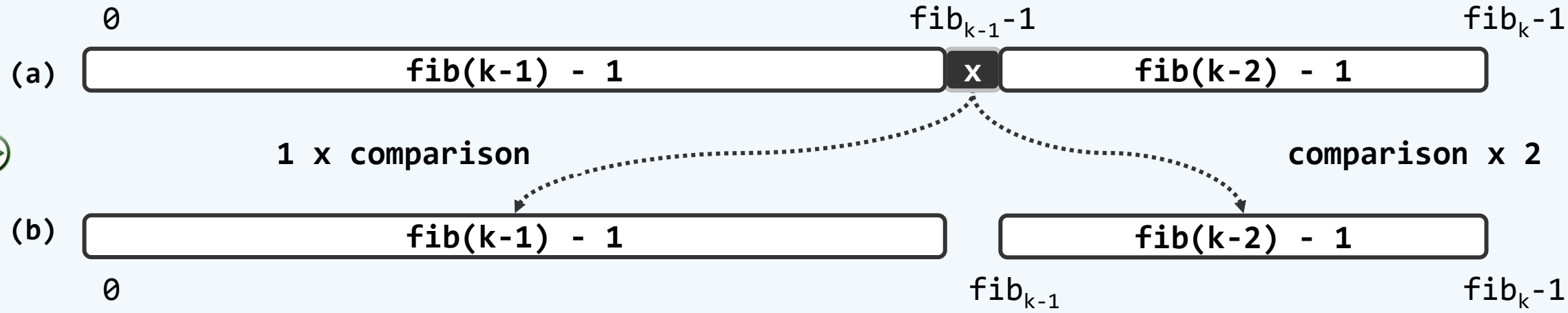
  转向左、右分支前的关键码比较次数不等，而递归深度却相同

- 若能通过递归深度的不均衡，对转向成本的不均衡进行补偿

  平均查找长度应能进一步缩短...

- 比如，若设n = fib(k) - 1，则可取mi = fib(k - 1) - 1

  于是，前、后子向量的长度分别为fib(k - 1) - 1、fib(k - 2) - 1



实现

```
template <typename T> //0 <= lo <= hi <= _size
static Rank fibSearch(T* A, T const & e, Rank lo, Rank hi) {
   Fib fib(hi - lo); //用O(log_Φ n) = O(log_Φ(hi − lo))时间创建Fib数列
   while (lo < hi) {
      while (hi - lo < fib.get()) fib.prev(); //至多迭代几次？
         //通过向前顺序查找，确定形如Fib(k) - 1的轴点（分摊O(1)）
      Rank mi = lo + fib.get() - 1; //按黄金比例切分
      if      (e < A[mi]) hi = mi; //深入前半段[lo, mi)继续查找
      else if (A[mi] < e) lo = mi + 1; //深入后半段(mi, hi)
      else                return mi; //在mi处命中
   }
   return -1; //查找失败
}
```

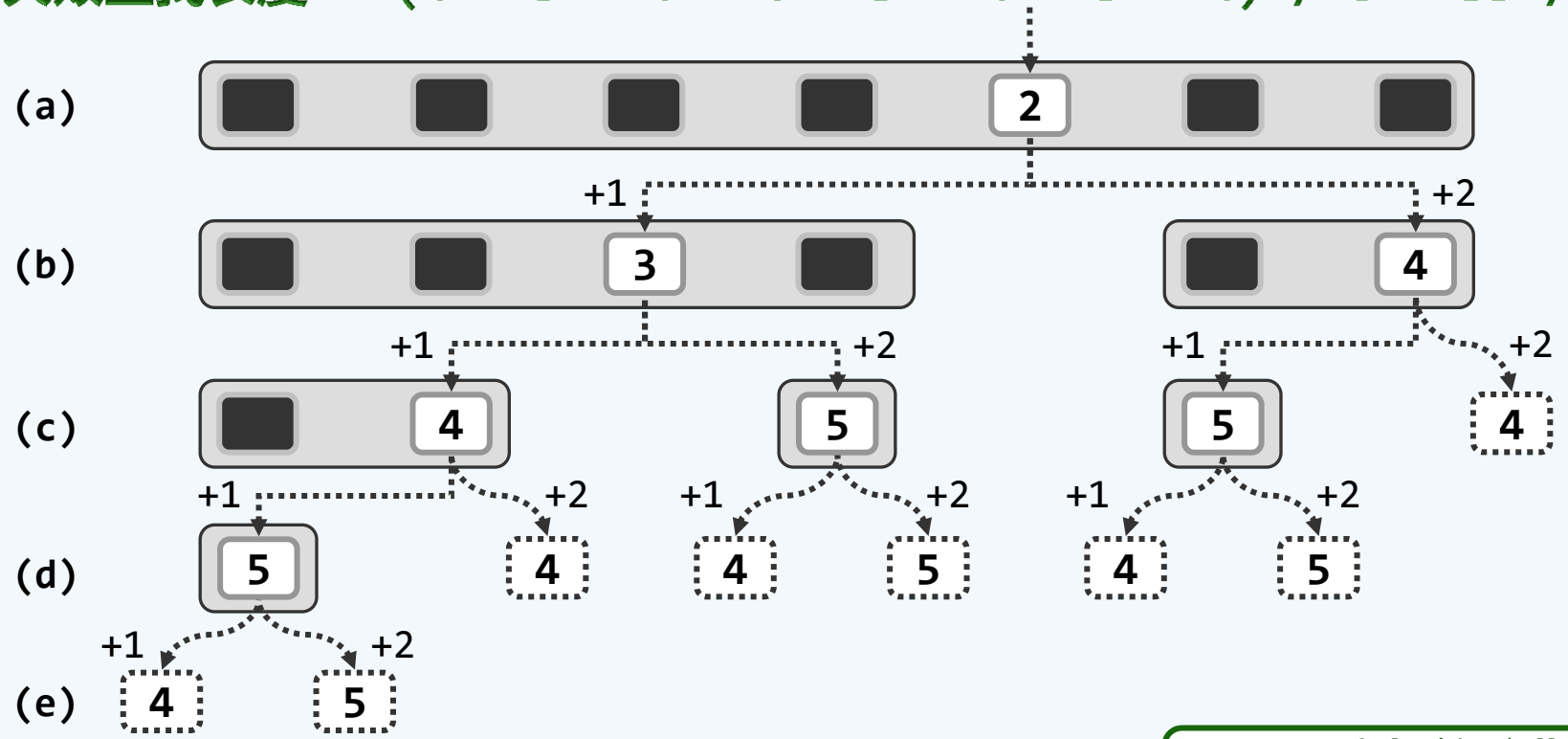
查找长度

❖ Fibonacci查找的ASL，（在常系数的意义上）优于二分查找　　//详见教材、习题解析

❖ 仍以n = fib(6) - 1 = 7为例，在等概率情况下

平均成功查找长度 = (5 + 4 + 3 + 5 + 2 + 5 + 4) / 7 = 28/7 = 4.00

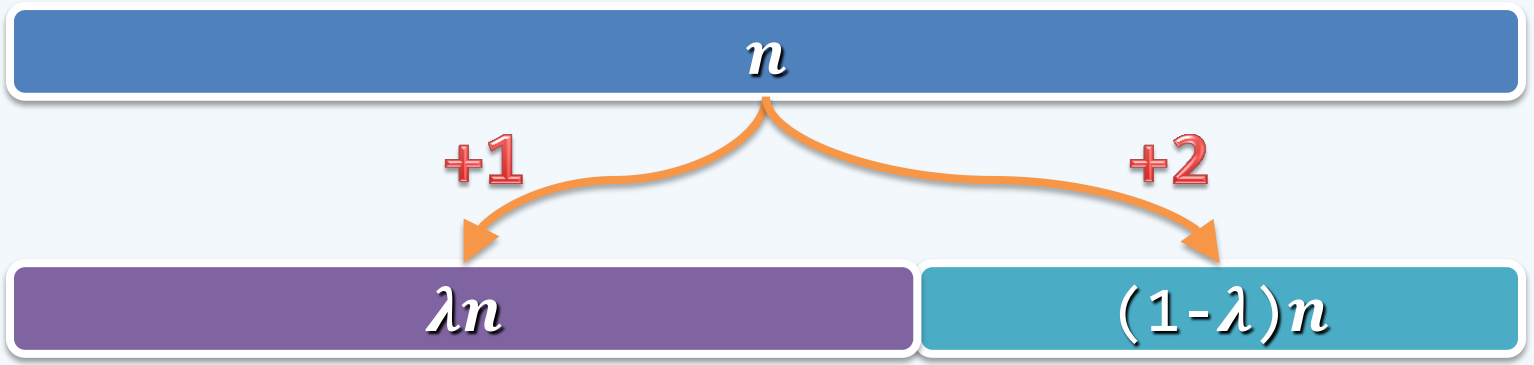
平均失败查找长度 = (4 + 5 + 4 + 4 + 5 + 4 + 5 + 4) / 8 = 35 / 8 = 4.38



ϕ = 0.6180339...

❖ 通用策略：对于任何的A[0, n)，总是选取A[λn]作为轴点，0 ≤ λ < 1

比如：二分查找对应于λ = 0.5，Fibonacci查找对应于λ = ϕ = 0.6180339...



❖ 在[0, 1)内，λ如何取值才能达到最优？设平均查找长度为$\alpha(\lambda)\cdot log_2 n$，何时$\alpha(\lambda)$最小？

❖ 递推式：$\alpha(\lambda)\cdot log_2 n = \lambda\cdot[1+\alpha(\lambda)\cdot log_2(\lambda n)] + (1-\lambda)\cdot\left[2+\alpha(\lambda)\cdot log_2\left((1-\lambda)n\right)\right]$

❖ 整理后：$\dfrac{-ln2}{\alpha(\lambda)} = \dfrac{\lambda\cdot ln\lambda+(1-\lambda)\cdot ln(1-\lambda)}{2-\lambda}$，当λ = ϕ时，$\alpha(\lambda) = 1.440420...$达到最小

课后

- fibSearch()的内层while循环，至多能够连续执行几次？
- 改进本节所给的实现，使Fibonacci查找严格符合search()接口