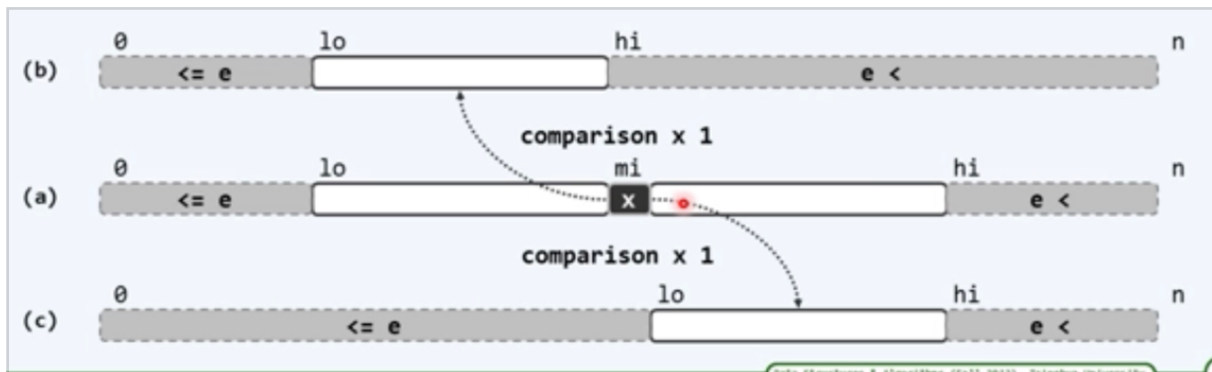


02D5-2 正确性 & 02D6-1 插值查找 - 原理 & 02D6-2 实例

#数据结构邓神

正确性证明：



为什么这个算法不需要判断 mid 这个点

可以通过两个点来证明

1. 不变形: $A[0, lo) \leq e < A[hi, n]$ // $A[hi]$ 总是大于 e 的最小者

开始的时候 $lo = 0$ 且 $hi = n$, $A[0, lo) = A[hi, n]$ 成立 (两个区间都是空的, 显然成立)

不妨假设不变形持续保持到 如上图所示的 a 情况

我们会发现无论是像左还是向右 上面提出的不变形依然会存在。

2. 单调性: 显然 每次元素都会减少

所以最后应该返回 $lo-1$ 因为在前面的算法始终在查找大于 e 的第一个元素

有序向量-插值查找-原理

假设: 已知有序向量中各元素随机分布的规律

比如: 均匀且独立的随机分布

我们就可以获得比二分查找更高的效率

原理与算法

❖ 假设：已知有序向量中各元素随机分布的规律

比如：均匀且独立的随机分布

$O(\log n)$

❖ 于是： $[lo, hi)$ 内各元素应大致按照线性趋势增长

$$\frac{mi - lo}{hi - lo} \approx \frac{e - A[lo]}{A[hi] - A[lo]}$$

❖ 因此：通过猜测轴点 mi ，可以极大地提高收敛速度

$$mi \approx lo + (hi - lo) \cdot \frac{e - A[lo]}{A[hi] - A[lo]}$$

这样就可以使得收敛的速度大大的加快

简单原理

❖ 例如：在英文词典中

binary大致位于2/26处

search大致位于19/26处

类似于人类查字典的原理

实例

实例

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5	10	12	14	26	31	38	39	42	46	49	51	54	59	72	79	82	86	92

❖ $e = 50$

$lo = 0, hi = 18$ 插值: $mi = 0 + (18 - 0) * (50 - 5) / (92 - 5) \approx 9.3$

取: $mi = 9$

比较: $A[9] = 46 < e$

❖ $lo = 10, hi = 18$ 插值: $mi = 10 + (18 - 10) * (50 - 49) / (92 - 49) \approx 10.2$

取: $mi = 10$

比较: $A[10] = 49 < e$

❖ $lo = 11, hi = 18$ 插值: $mi = 11 + (18 - 11) * (50 - 51) / (92 - 51) \approx 10.8$

取: $mi = 10 < lo$

查找完成 (NOT_FOUND)

我们会发现在最后一次查找的时候 直接变为负数，完成了查找

性能

最坏情况: $O(lo - hi) = O(n)$ // 在某些情况下，插值查找效率会大大下降

最好情况: 一次就命中 $O(1)$

一般情况: 也就是平均查找成本?