

向量

有序向量：二分查找（版本B）

02-D4

和微风匀到一起的光，象冰凉的刀刃儿似的，把宽静的大街切成两半，一半儿黑，一半儿亮。那黑的一半，使人感到阴森，亮的一半使人感到凄凉。

邓俊辉

deng@tsinghua.edu.cn

改进思路

❖ 二分查找中左、右分支转向代价不平衡的问题，也可**直接**解决，比如...

每次迭代仅做**1次**关键码比较；如此，所有分支只有**2个**方向，而不再是**3个**

❖ 同样地，轴点 mi 取作中点，则查找每深入**一层**，问题规模依然会缩减**一半**

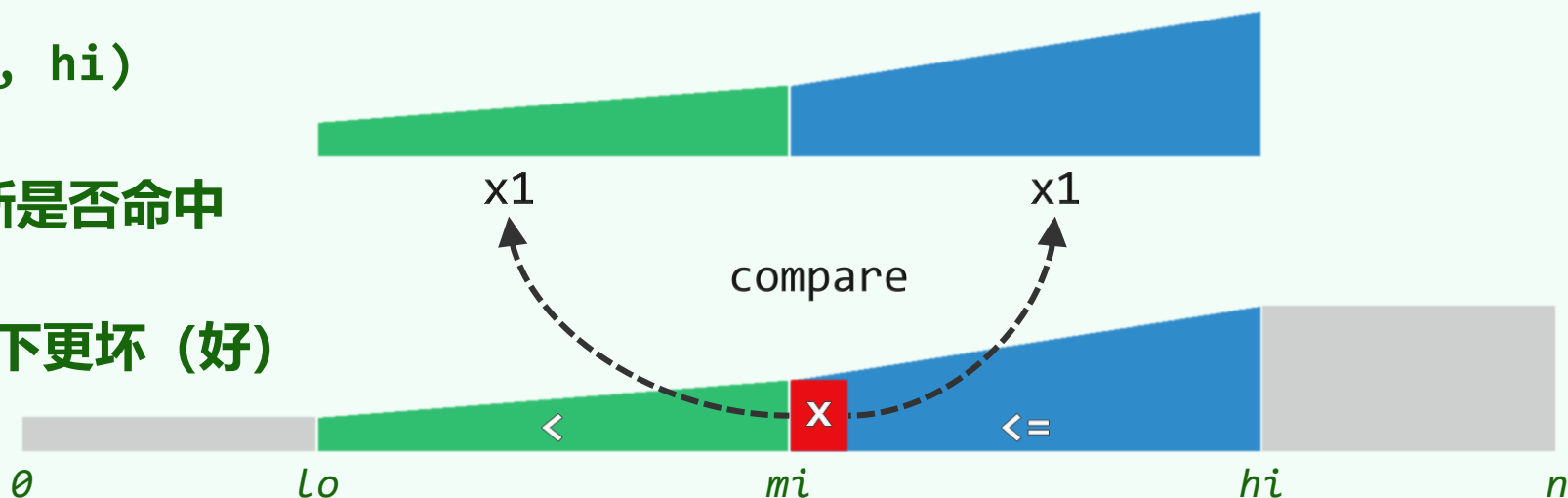
- $e < x$ ：则深入左侧的 $[lo, mi)$

- $x \leq e$ ：则深入右侧的 $[mi, hi)$

❖ 直到 $hi - lo = 1$ ，才明确判断是否命中

❖ 相对于版本A，最好（坏）情况下更坏（好）

整体性能更趋均衡



实现

```
template <typename T>

static Rank binSearch( T * S, T const & e, Rank lo, Rank hi ) {

    while ( 1 < hi - lo ) { //有效查找区间的宽度缩短至1时，算法才终止

        Rank mi = (lo + hi) >> 1; //以中点为轴点，经比较后确定深入[lo, mi)或[mi, hi)

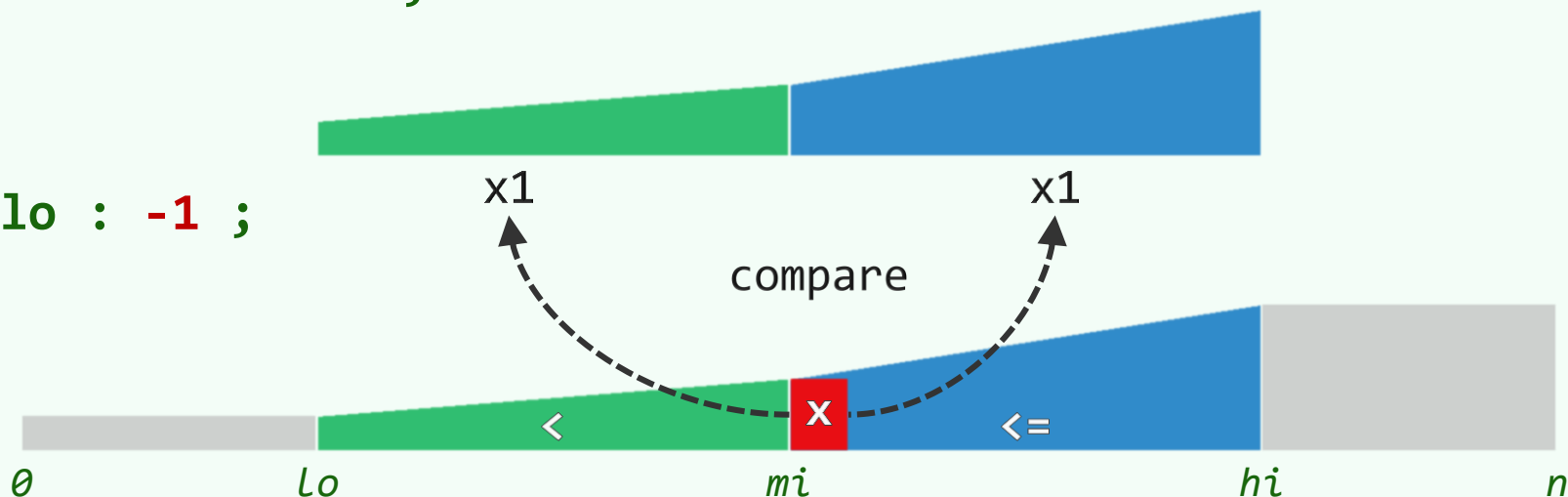
        e < S[mi] ? hi = mi : lo = mi;

    } //出口时hi = lo + 1

    return e == S[lo] ? lo : -1 ;

}
```

❖ 返回命中处的秩，或失败标志



返回更多信息

❖ 如何统一地处置各种情况？比如

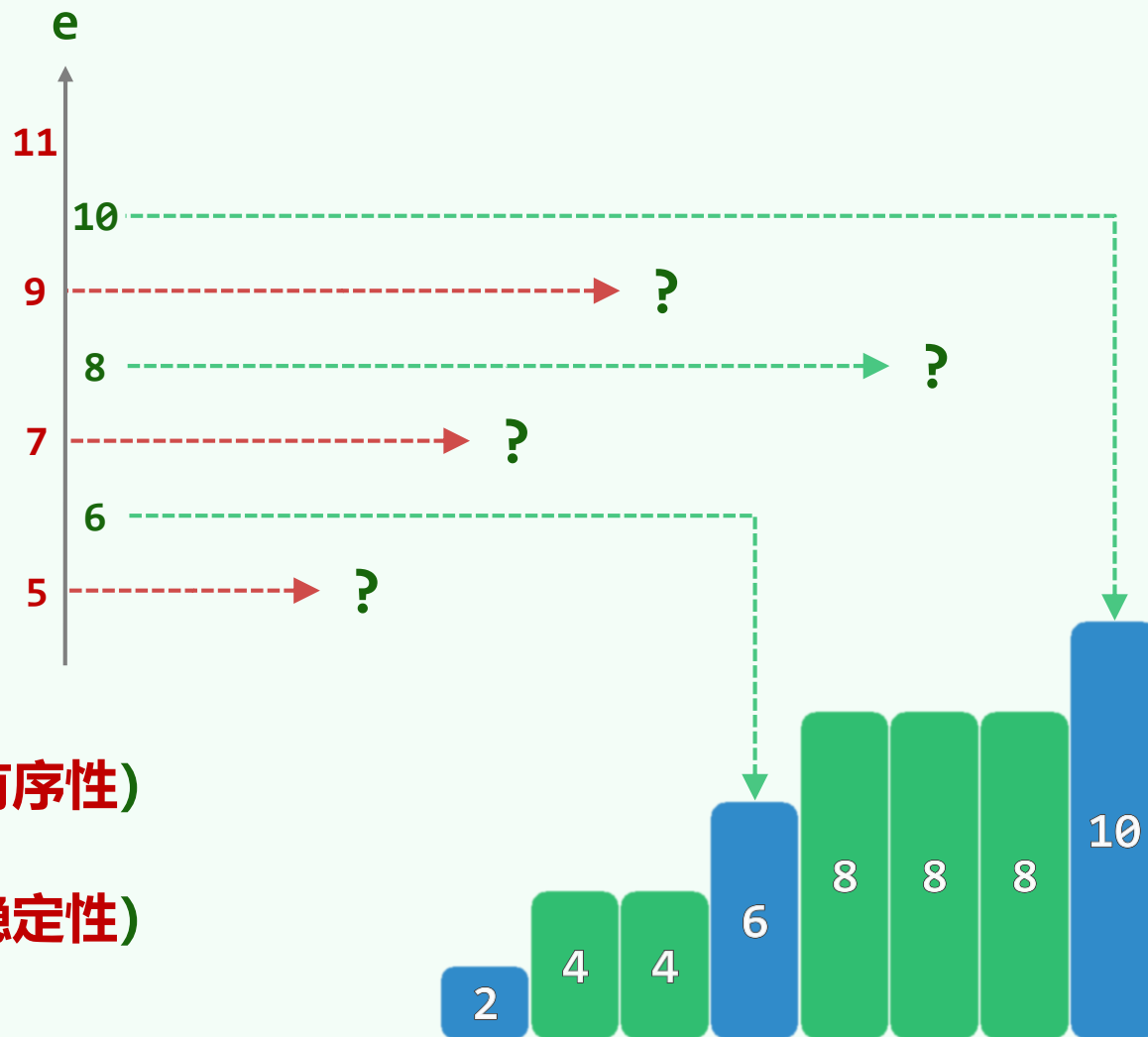
- 目标元素**不存在**；或反过来
- 目标元素同时存在**多个**

❖ 面向未来，有序向量自身又当如何便捷地维护？

比如： `V.insert(1 + V.search(e), e)`

- 即便失败，也应给出新元素可安置的位置（**有序性**）
- 若有重复元素，也需按其插入的次序排列（**稳定性**）

❖ 为此，需要更为精细、明确、简捷地定义 `search()` 的返回值



返回值的语义扩充

❖ 约定总是返回 $m = \text{search}(e) = M-1$

$$-\infty \leq m = \max\{k \mid [k] \leq e\}$$

$$\min\{k \mid e < [k]\} = M \leq +\infty$$

❖ 直接改进版本B:

```
return e == S[lo] ? lo : -1 ;
```

```
return e < S[lo] ? lo-1 : lo ;
```

❖ 虽可行，但不免有些蹩脚

有没有...更为...简明、高明的...实现方式?

