θ2-D4

2.向量

有序向量

二分查找（版本B）

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

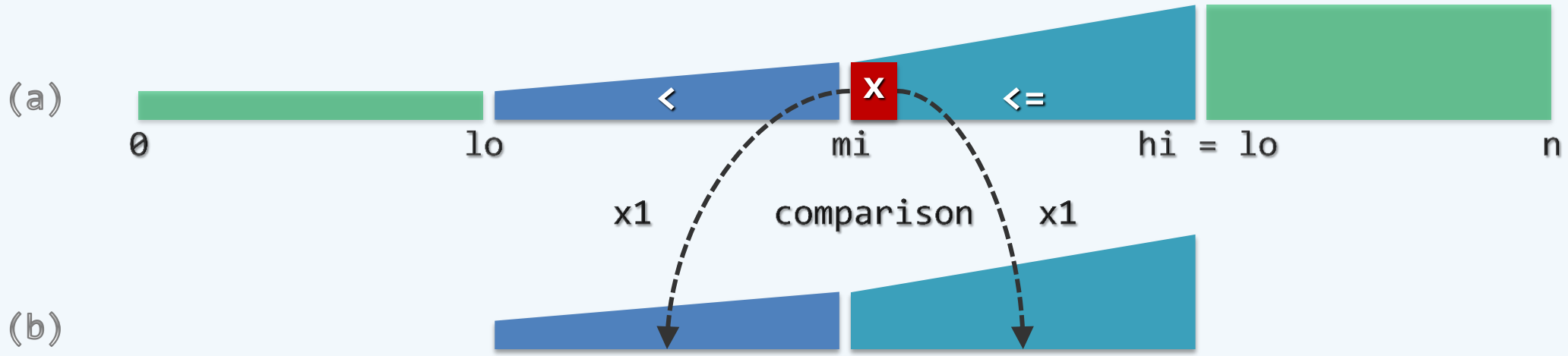
deng@tsinghua.edu.cn

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（或每个递归实例）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



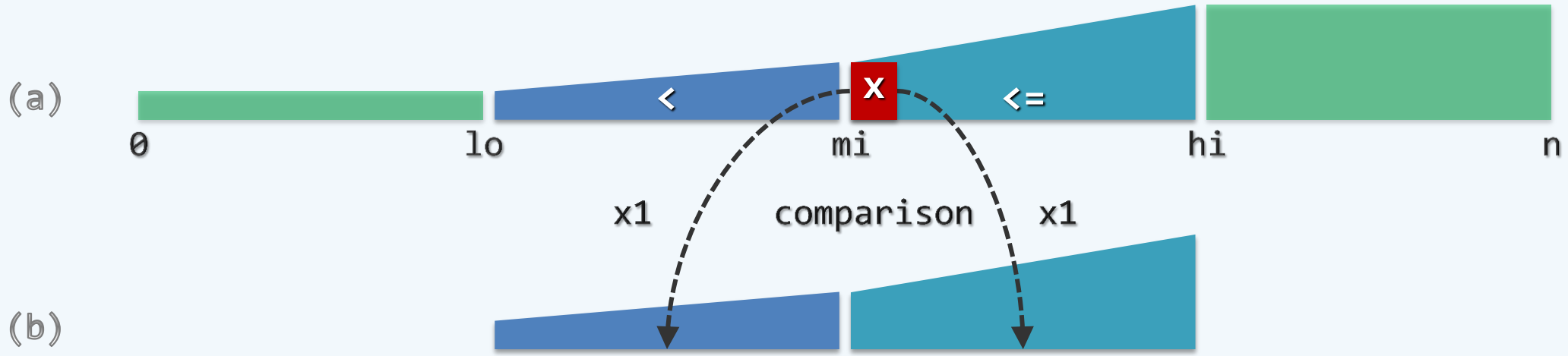
改进思路

❖ 同样地，轴点mi取作中点，则查找每深入一层，问题规模也缩减一半

1） `e < x`：则e若存在必属于左侧子区间S[lo, mi)，故可递归深入

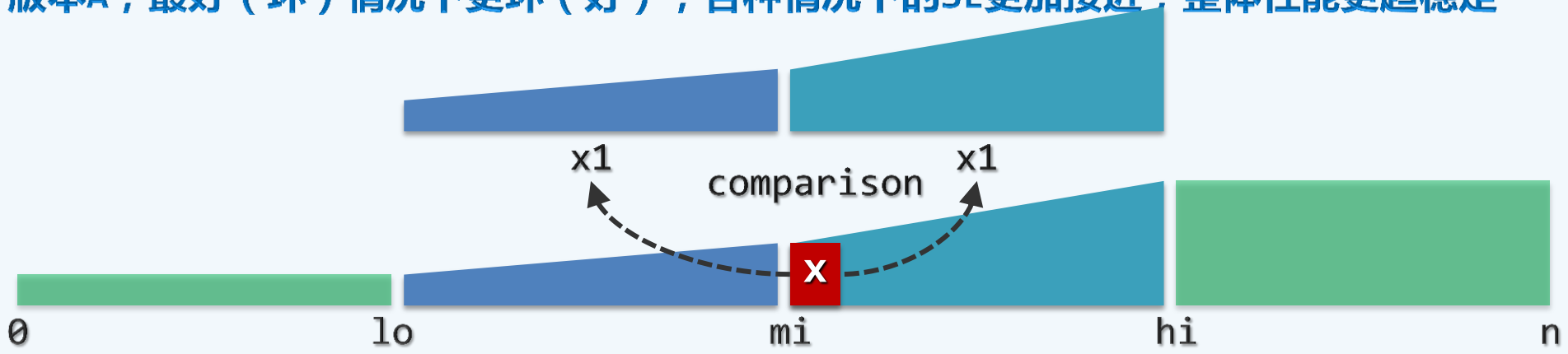
2） `x <= e`：则e若存在必属于右侧子区间S[mi, hi)，亦可递归深入

只有当元素数目hi - lo = 1时，才判断该元素是否命中



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，算法才会终止
       Rank mi = (lo + hi) >> 1; //以中点为轴点，经比较后确定深入
       e < S[mi]  ?  hi = mi  :  lo = mi; //[lo, mi)或[mi, hi)
    } //出口时hi = lo + 1，查找区间仅含一个元素A[lo]
    return e == S[lo]  ?  lo  :  -1 ; //返回命中元素的秩或者-1
  } //相对于版本A，最好（坏）情况下更坏（好）；各种情况下的SL更加接近，整体性能更趋稳定
```



语义约定

❖ 各种特殊情况，如何统一地处置？比如

- 目标元素不存在；或反过来
- 目标元素同时存在多个

❖ 有序向量自身，如何便捷地维护？

比如：`V.insert( 1 + V.search(e), e )`

- 即便失败，也应给出新元素适当的插入位置 //有序性
- 若允许重复元素，则每一组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

语义约定

❖ 约定：search()总是返回不大于e的最后一个元素　　//其后继，即大于e的第一个元素

若 -∞ < e < V[lo]，则返回lo - 1　　//左侧哨兵 = 首元素的前驱

若 V[hi - 1] < e < +∞，则返回hi - 1　　//末元素 = 右侧哨兵的前驱

若 V[k] < e < V[k + 1]，则返回k　　//e可作为V[k]的后继插入

若 V[k] ≤ e < V[k + 1]，亦返回k　　//e可作为V[k]的后继插入，且稳定

❖ 二分查找版本A、版本B及fibSearch()，均未严格兑现这一语义约定

❖ 课后： 对版本B及fibSearch()略作调整，使之符合约定

❖ 有没有更为简明的实现方式？