

3. 列表

(e) 插入排序

一语未了，只见宝玉笑嘻嘻的掬了一枝红梅进来，
众丫鬟忙已接过，插入瓶内。

邓俊辉

deng@tsinghua.edu.cn

构思

❖ 始终将序列看成两部分：

Sorted + Unsorted

$L[0, r)$ + $L[r, n)$

❖ 【初始化】

$|S| = r = 0$ //空序列无所谓有序

❖ 【迭代】：关注并处理 $e = L[r]$

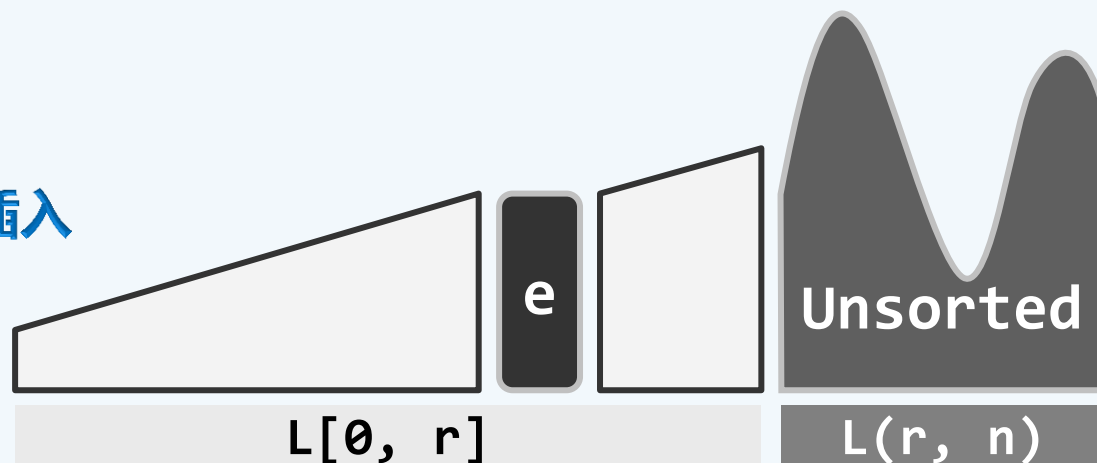
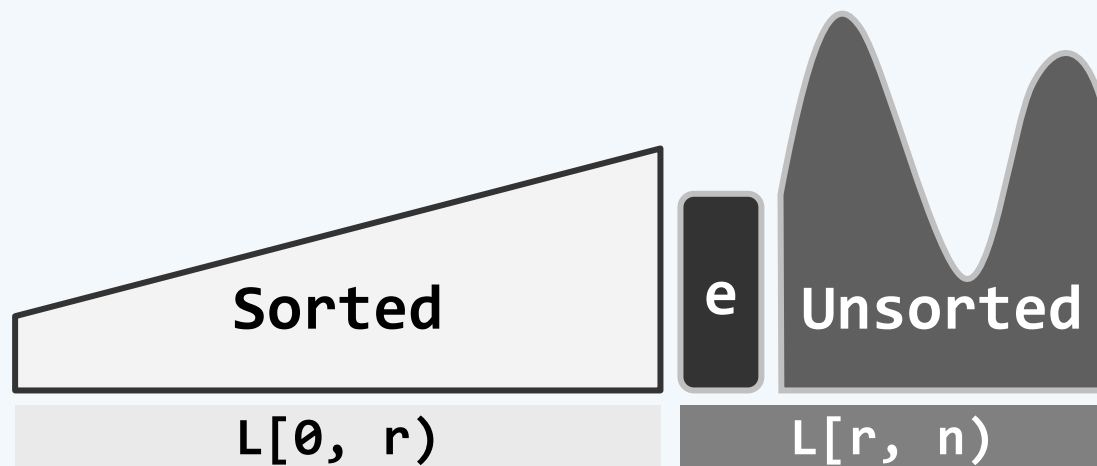
在S中确定适当位置 //有序序列的查找

插入e，得到有序的 $L[0, r]$ //有序序列的插入

❖ 【不变性】

随着r的递增， $L[0, r)$ 始终有序

直到 $r = n$ ，L即整体有序



实例

迭代轮次	前缀有序子序列	当前元素	后缀无序子序列
-1	^	^	5 2 7 4 6 3 1
0	^	5	2 7 4 6 3 1
1	(5)	2	7 4 6 3 1
2	(2) 5	7	4 6 3 1
3	2 5 (7)	4	6 3 1
4	2 (4) 5 7	6	3 1
5	2 4 5 (6) 7	3	1
6	2 (3) 4 5 6 7	1	^
7	(1) 2 3 4 5 6 7	^	^

实现

//对列表中起始于位置p的连续n个元素做插入排序, $\text{valid}(p) \ \&\& \ \text{rank}(p) + n \leq \text{size}$

```
template <typename T> void List<T>::insertionSort(Posi(T) p, int n) {
```

```
    for (int r = 0; r < n; r++) { //逐一引入各节点, 由 $s_r$ 得到 $s_{r+1}$ 
```

```
        insertA( search( p->data, r, p ), p->data ); //查找 + 插入
```

```
        p = p->succ; remove( p->pred ); //转向下一节点
```

```
    } //n次迭代, 每次 $O(r + 1)$ 
```

```
} //仅使用 $O(1)$ 辅助空间, 属于就地算法
```

❖ 紧邻于search()接口返回的位置之后插入当前节点, 总是保持有序

❖ 验证各种情况下的正确性, 体会哨兵节点的作用:

s_r 中含有/不含与p相等的元素; s_r 中的元素均严格小于/大于p

性能

❖ 最好情况：完全（或几乎）有序

每次迭代，只需1次比较，0次交换

累计 $O(n)$ 时间！

❖ 最坏情况：完全（或几乎）逆序

第 k 次迭代，需 $O(k)$ 次比较，1次交换

累计 $O(n^2)$ 时间！

//改用向量呢？稍后分析

❖ 一般情况：包含 I 个逆序对

//inversion，不必相邻

第 k 次迭代，只需 $O(I_k)$ 次比较，1次交换

//为什么？

累计 $O(n + I)$ ！ //逆序对各需单位时间，input-sensitive，对Shellsort至关重要

❖ 平均而言呢？

//当然，首先需要假定具体的随机分布...

平均性能

❖ 假定：各元素的取值遵守均匀、独立分布

于是：平均要做多少次元素比较？



❖ 考查： $L[r]$ 刚插入完成的那一时刻（穿越？）



试问：此时的有序前缀 $L[0, r]$ 中，哪个元素是此前的 $L[r]$ ？

❖ 观察：其中的 $r + 1$ 个元素均有可能，且概率均等于 $1/(r + 1)$

❖ 因此，在刚完成的这次迭代中，为引入 $s[r]$ 所花费时间的数学期望为

$$[r + (r - 1) + \dots + 3 + 2 + 1 + 0] / (r + 1) + 1 = r/2 + 1$$

❖ 于是，总体时间的数学期望 = $[0 + 1 + \dots + (n - 1)] / 2 + 1 = O(n^2)$

❖ 再问：在 n 次迭代中，平均有多少次无需交换呢？

//习题[3-10]

若改用向量...

- ❖ 借助二分查找之类的算法，**查找效率**可优化至 $O(\log k)$ ！
- ❖ 然而，**总体性能**会否因此相应地提高？

逆序对

❖ 习题[3-11]

若共含 I 个逆序对 `inversion` , 则关键码比较次数为 $\mathcal{O}(I)$, 运行时间为 $\mathcal{O}(n + I)$