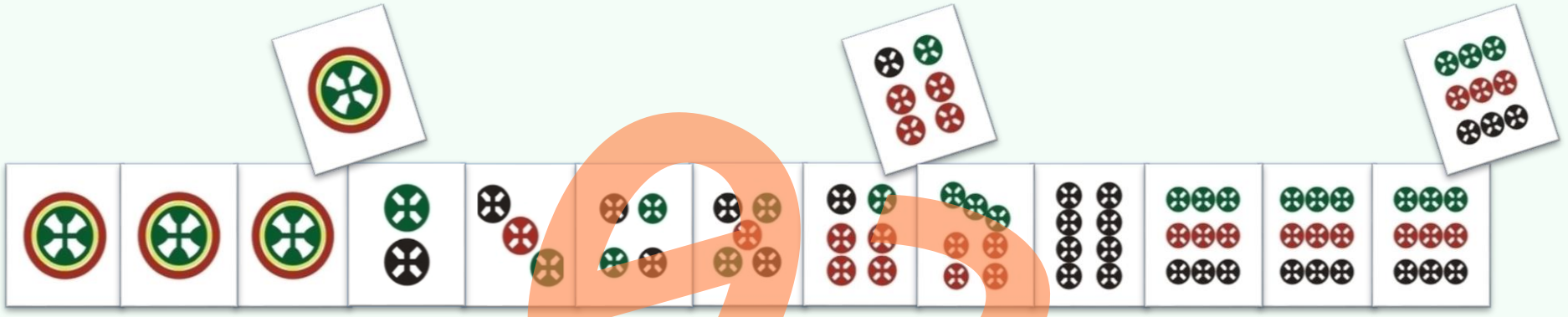


列表

插入排序

一语未了，只见宝玉笑嘻嘻的掮了一枝红梅进来，众丫鬟忙已接过，插入瓶内

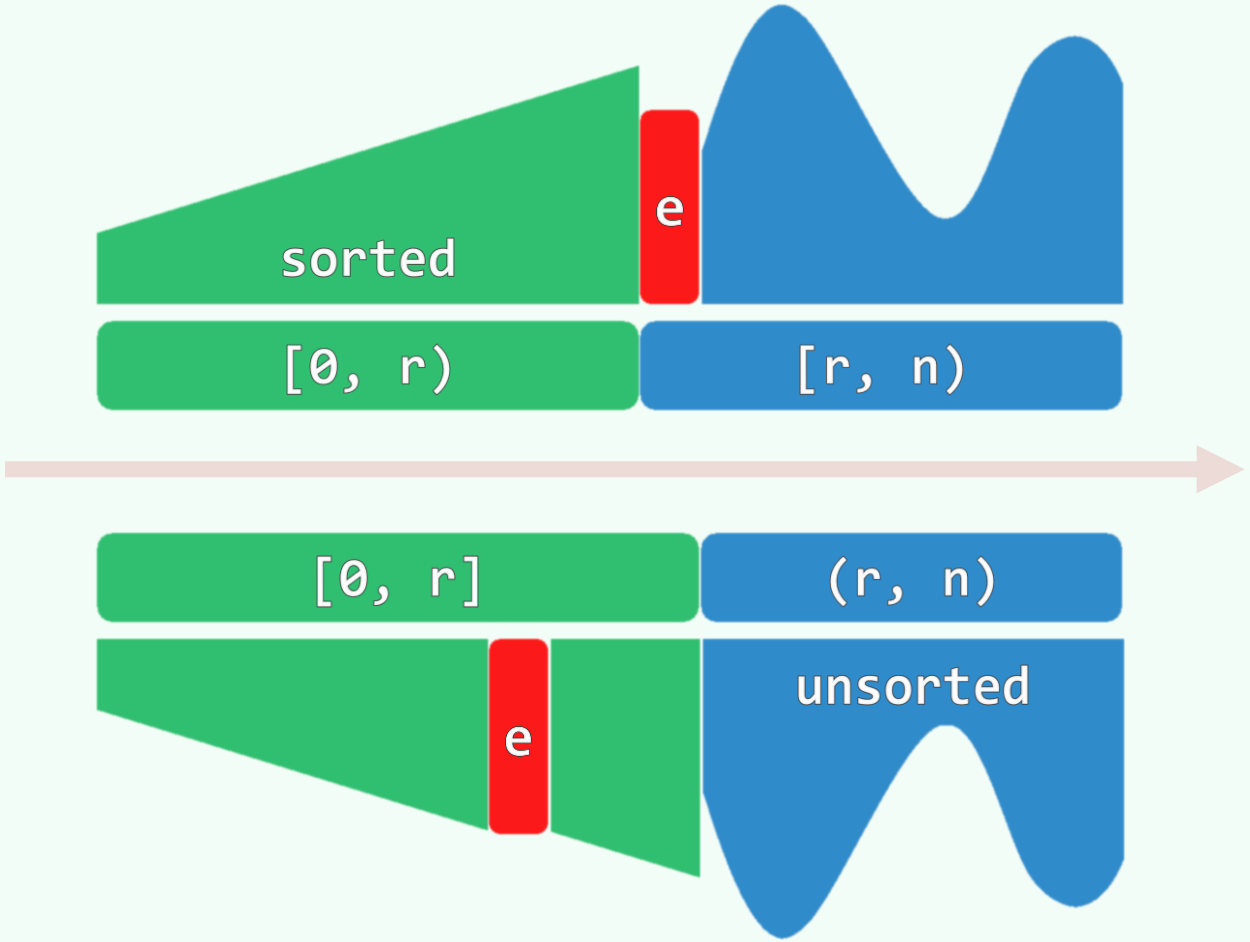
一日与樊、饶、章、蒋太太等看竹廿余周，余又负十余元。后又
看众人打poker，觉无意味，二点始睡

人生如此美好，慢慢走，欣赏啊！

邓俊辉

deng@tsinghua.edu.cn

减而治之



- 始终将序列视作两部分：
  - 前缀 S[0, r)：有序
  - 后缀 U[r, n)：待排序
- 初始化：|S| = r = 0
- 反复地，针对e = A[r]
  - 在S中查找适当位置
  - 插入e
  - r ++

实例

								5 2 7 4 6 3 1	
iteration			sorted prefix					e	random suffix
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		

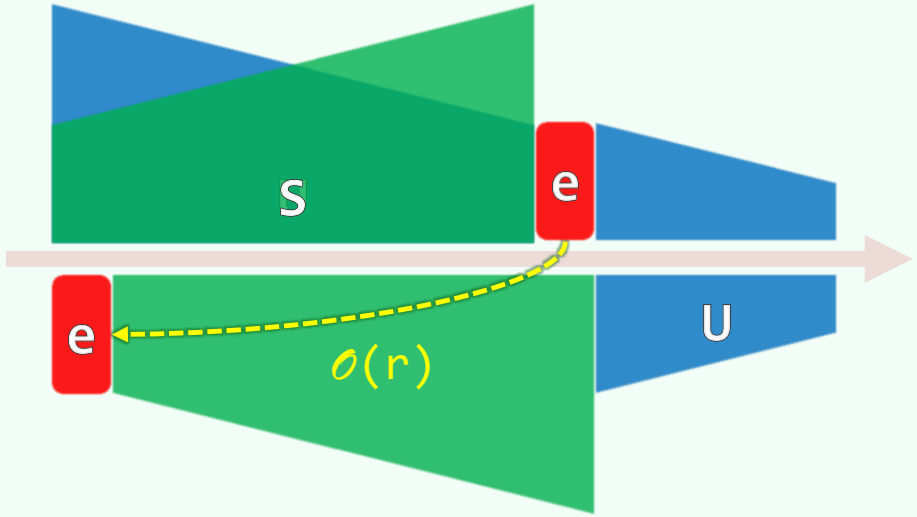
实现

```
template <typename T> void List<T>::insertionSort( ListNodePosi<T> p, Rank n ) {
   for ( Rank r = 0; r < n; r++ ) { //逐一引入各节点，由S_r得到S_{r+1}
      insert( 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
```

❖ 得益于此前约定的search()接口语义，前缀的确总是保持有序，而且稳定

❖ 验证以下情况，体会哨兵的作用：前缀中有元素与p相等；p在前缀中最小/最大；前缀为空；...

最好 & 最坏

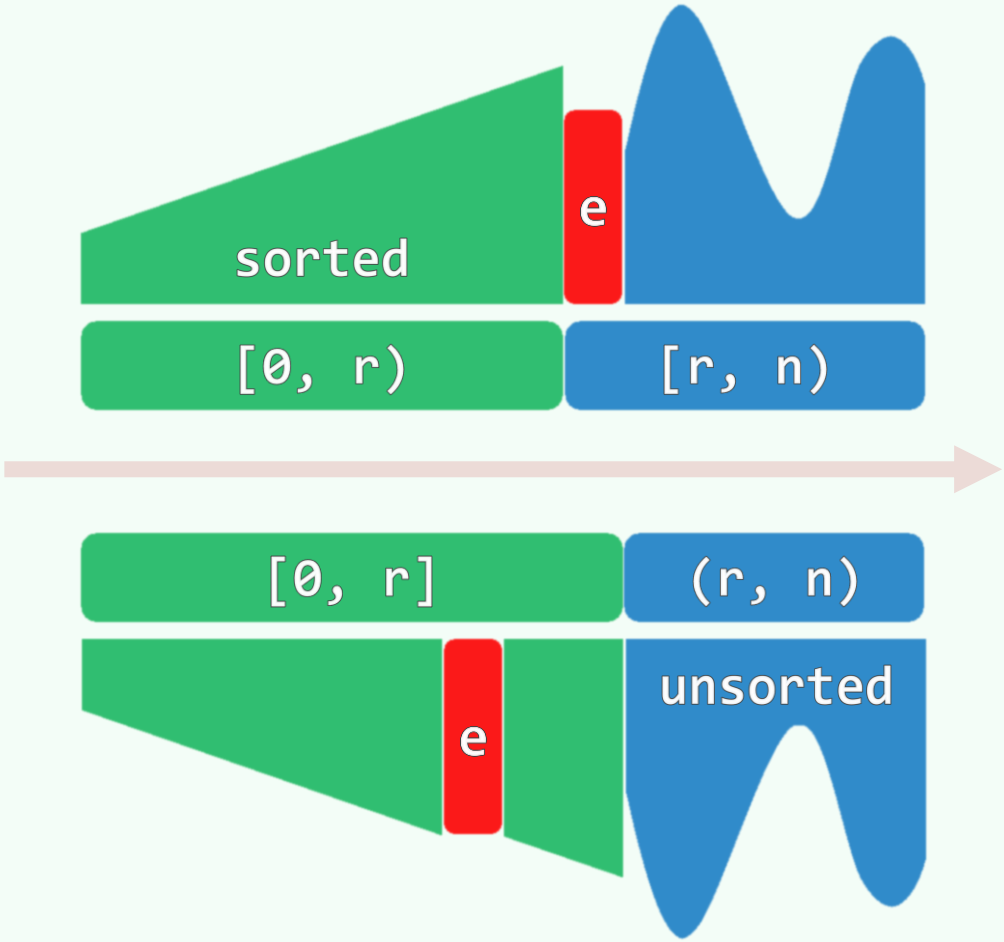


- **最好**：$\mathcal{O}(n)$ //比如，已经有序，相当于**验证**
- **最坏**：$\mathcal{O}(n^2)$ //比如，完全逆序
- **有实质的差异！**

  Selectionsort呢?
- **主要消耗来自查找**

  为何不...改用binSearch()?

平均 ~ 期望



❖ 若知道插入每个元素所需的期望时间

其总和即时总体的期望成本

❖ 后向分析：当前前缀[0,r]中，谁是最后插入的?

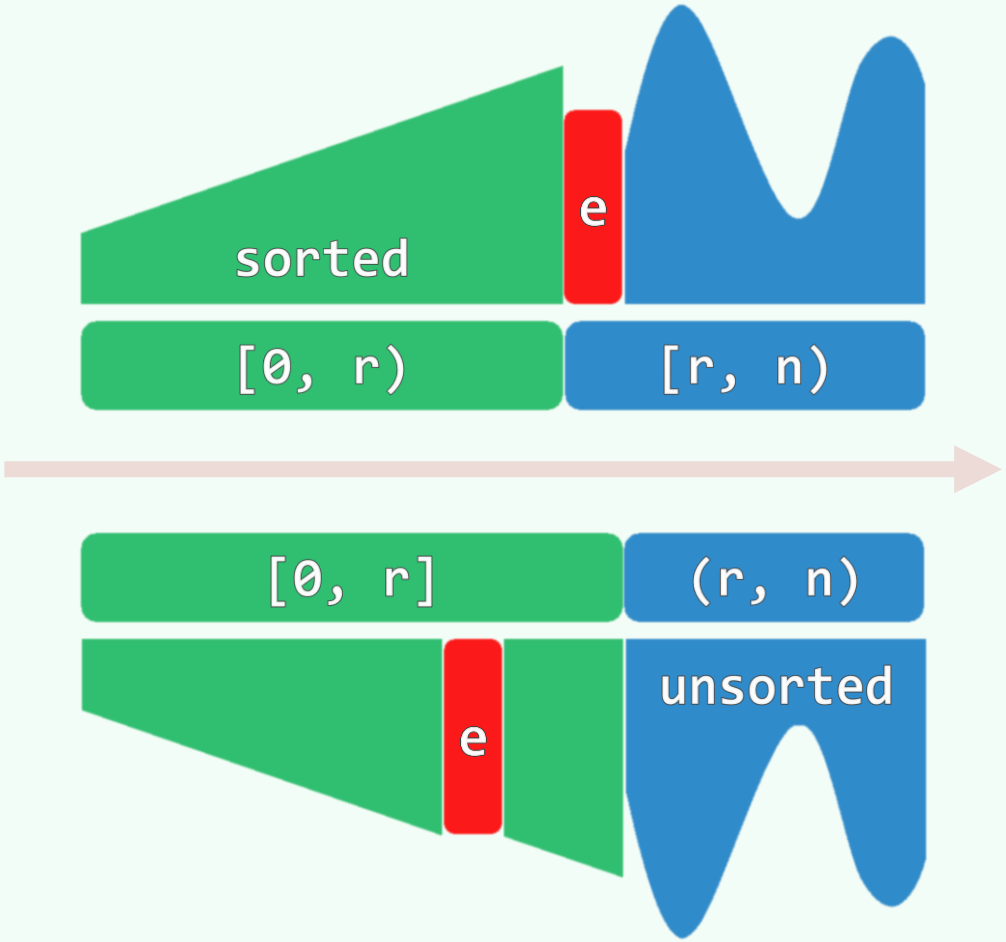
❖ 无非r+1种情况，且概率均等

数学期望： $1+\sum_{k=0}^{r} k/(r+1) = \underline{1+r/2}$

❖ 故总和为： $\sum_{r=0}^{n-1} \underline{(1+r/2)} = \mathcal{O}(n^2)$

❖ 有的元素可能无需移动，特判并节省?

输入敏感



❖ 从 $\mathcal{O}(n)$ 到 $\mathcal{O}(n^2)$，中间情况如何?

❖ 有序/乱序的程度

可简明度量，而且

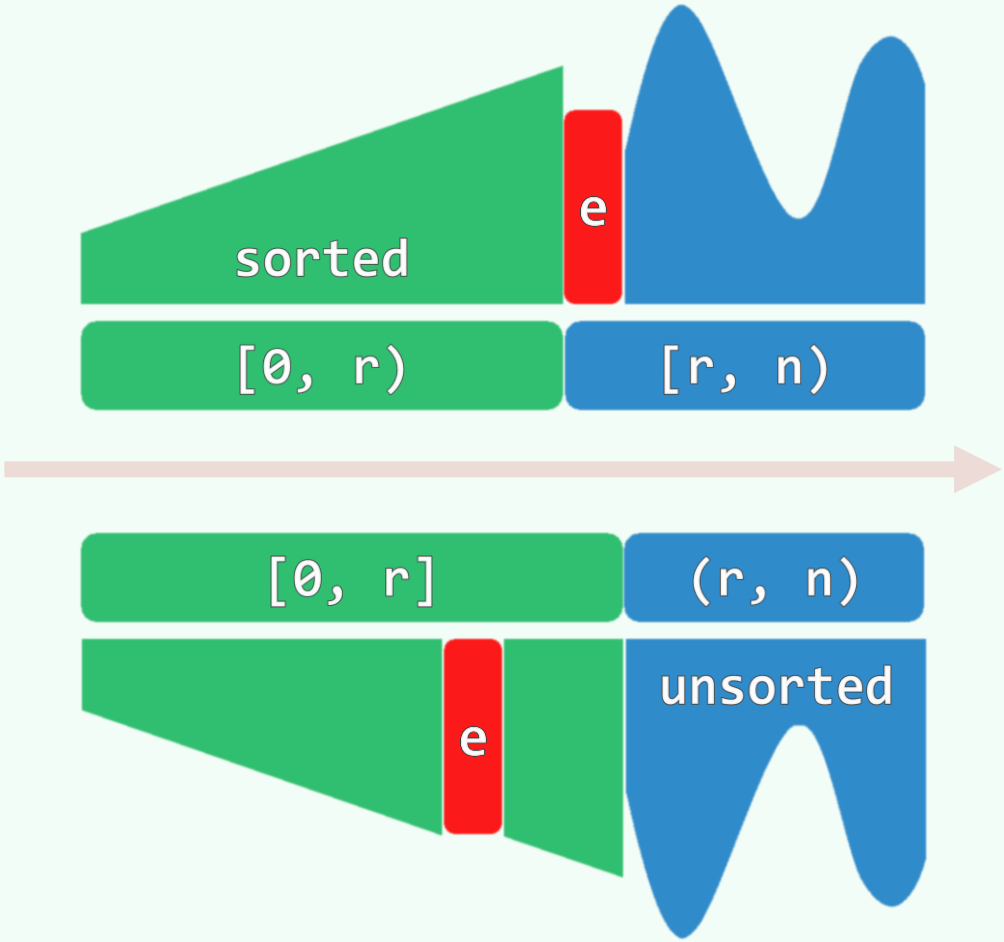
与时间成本之成正比...

❖ 输入敏感性/input-sensitivity

// 量出制入，丰俭由人

（稍后详解）

在线



❖ 日常牌戏中，为何你我他都用它?

❖ 改用其他算法...

发牌完毕，才能各自开始理牌

❖ Insertionsort

可以边发边理；发完即理好

❖ online：在数据完全就绪之前，即可开始计算!

// 何必一味追求结果，不妨充分享受过程