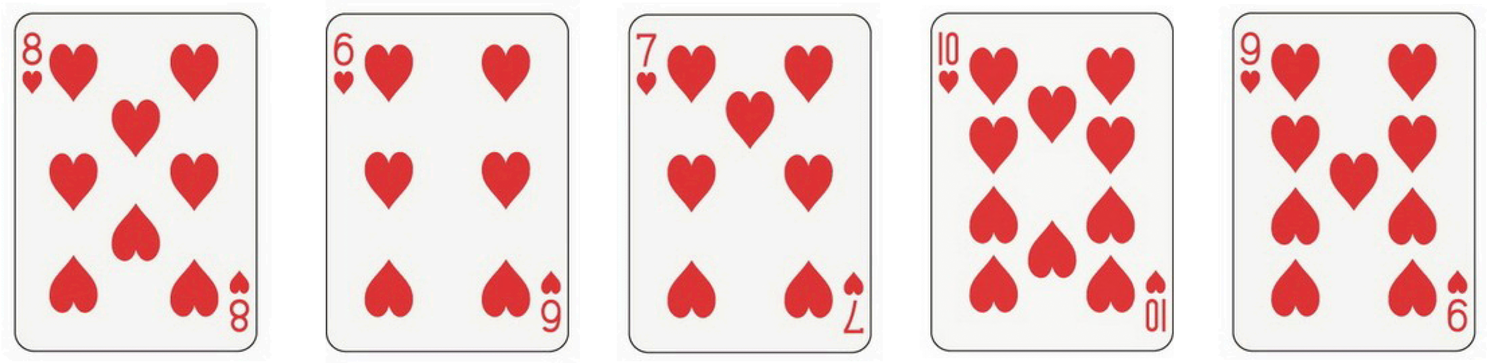
算法与数据结构体系课程

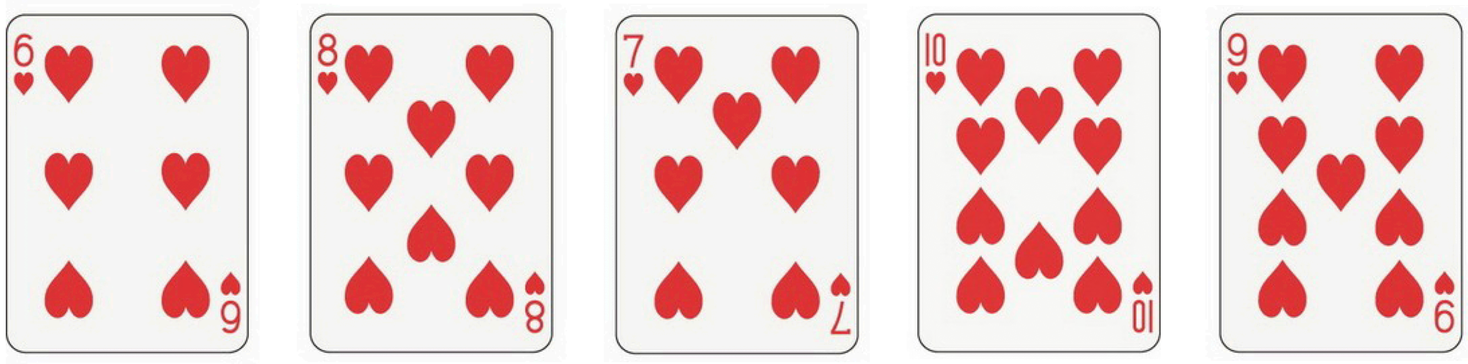
liuyubobobo

插入排序法

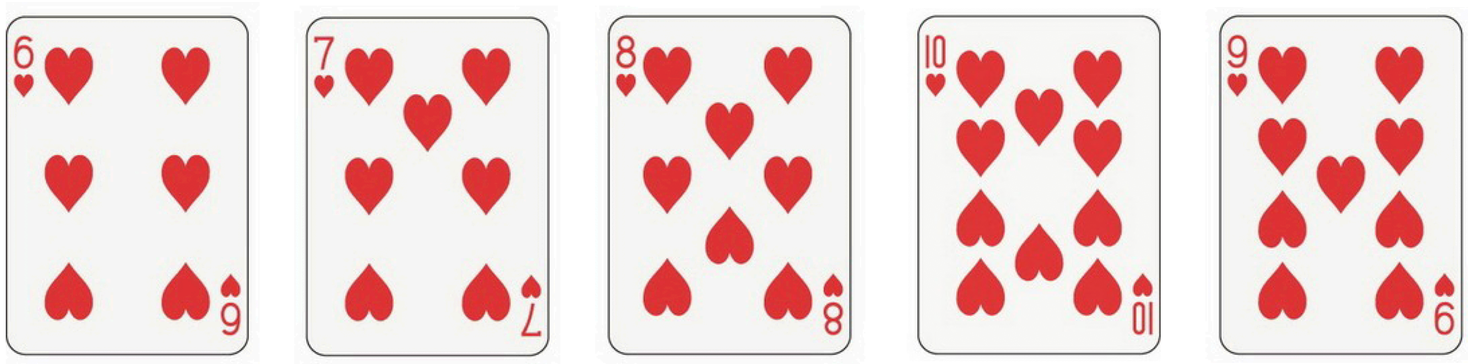
插入排序法



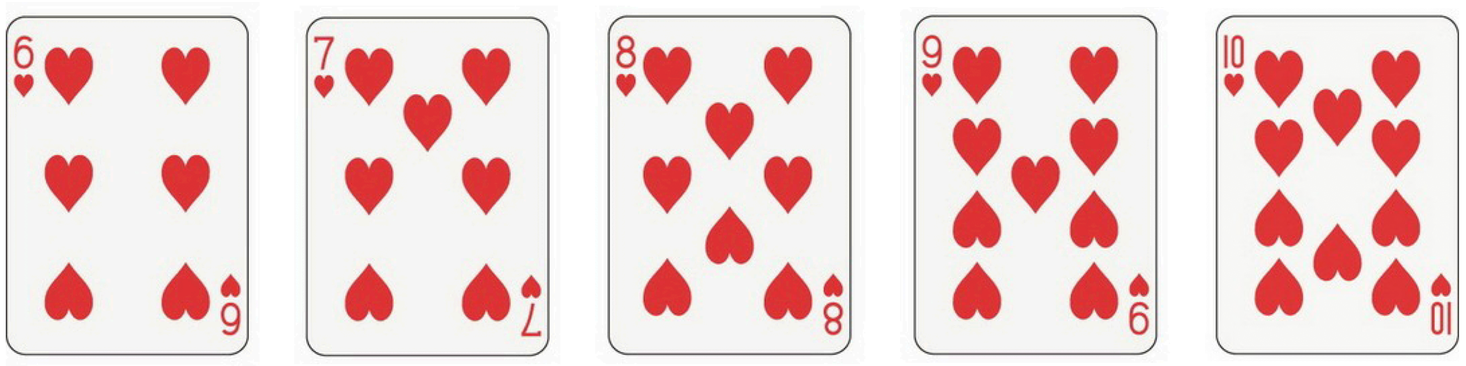
插入排序法



插入排序法



插入排序法

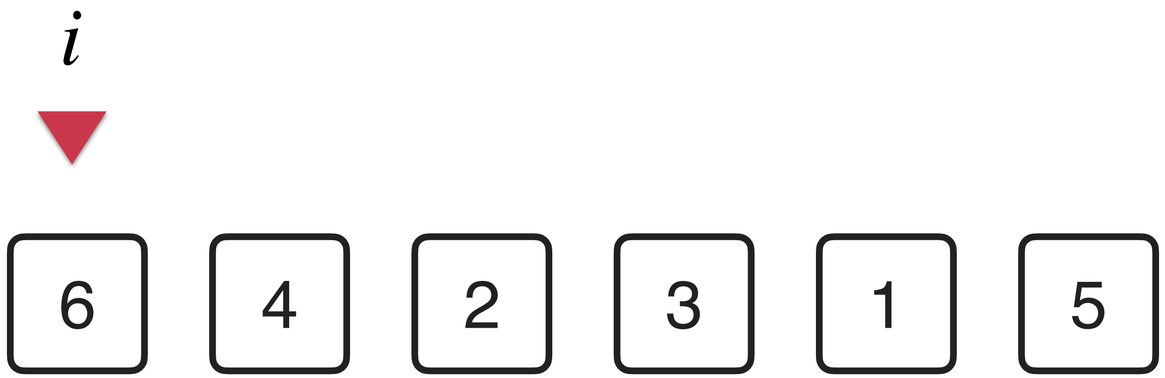


每次处理一张牌，把这张牌插入到前面已经排好序的牌中

插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

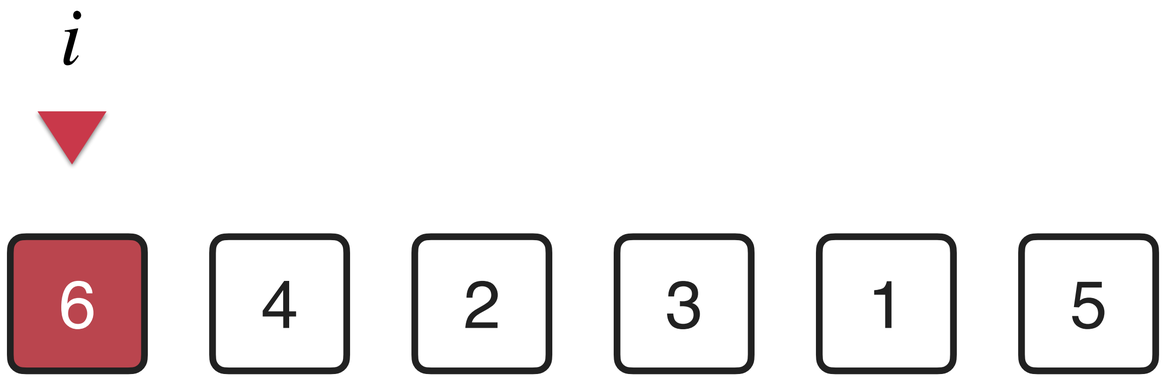
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

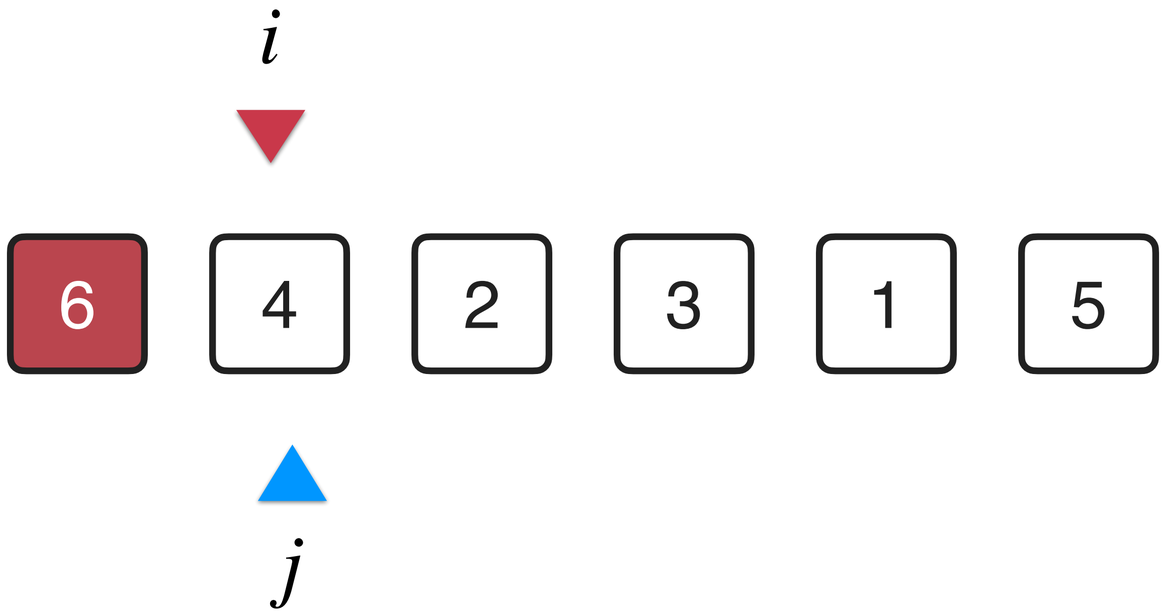
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

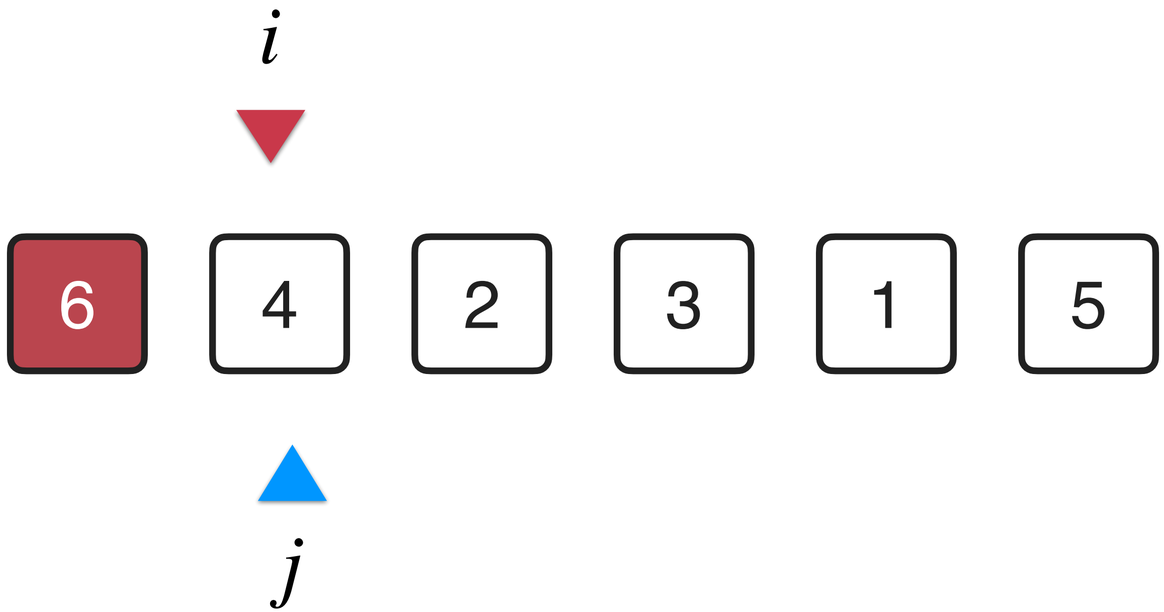
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

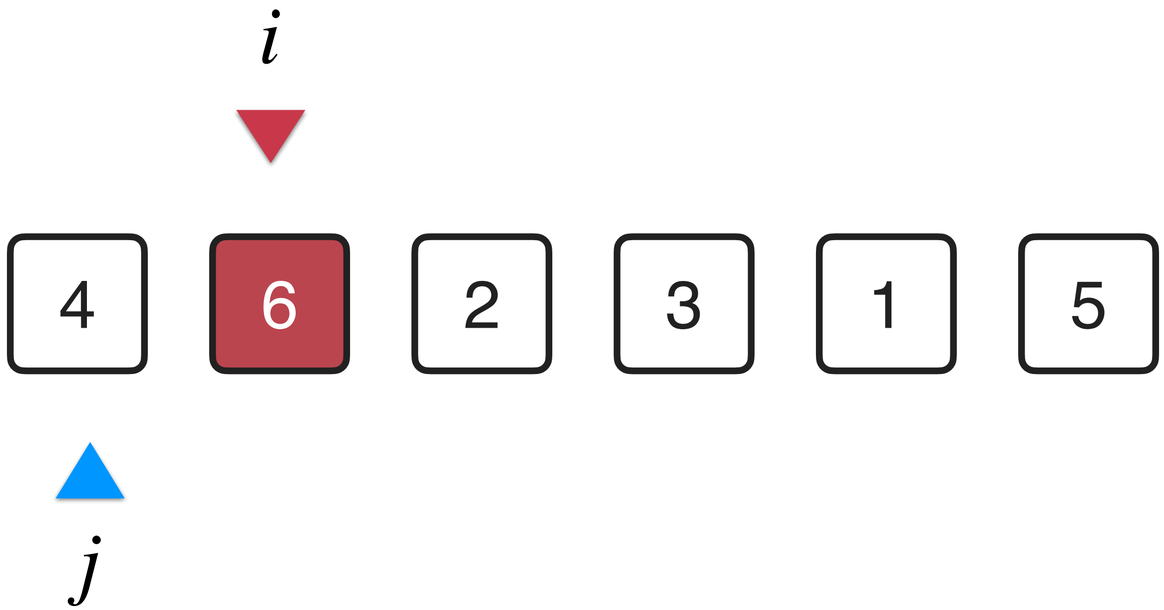
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

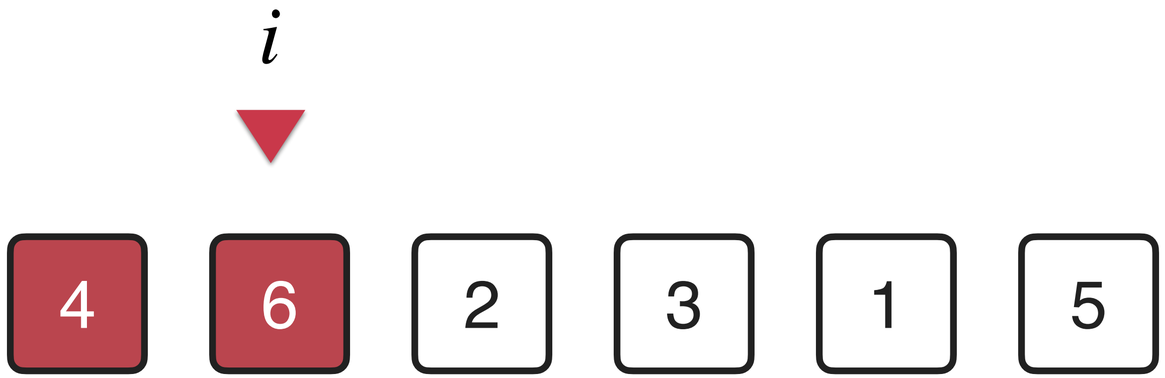
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

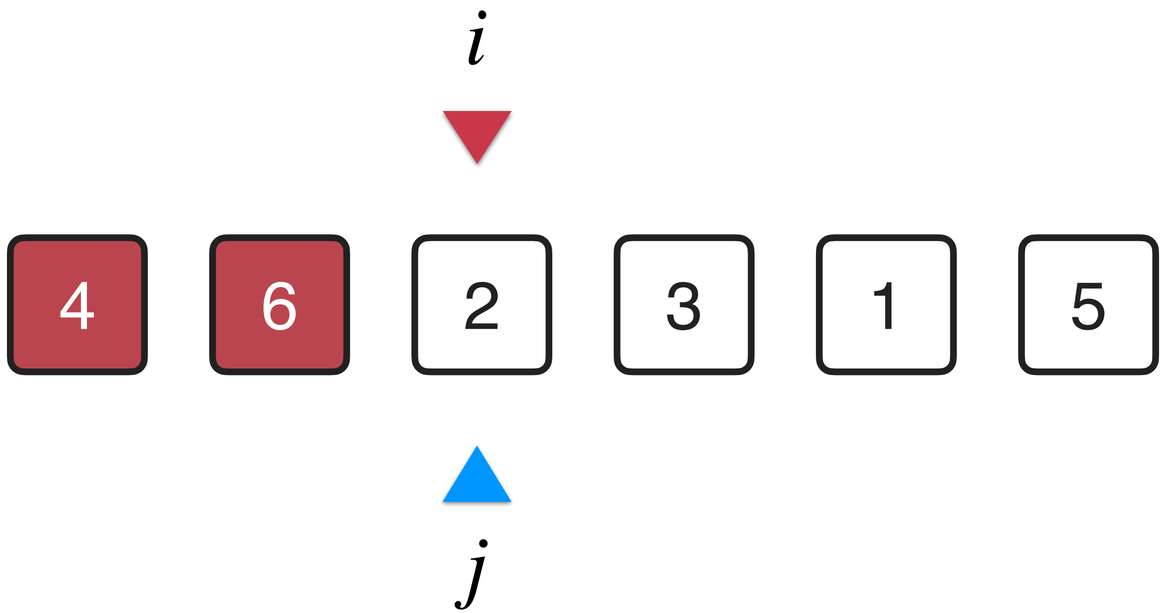
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

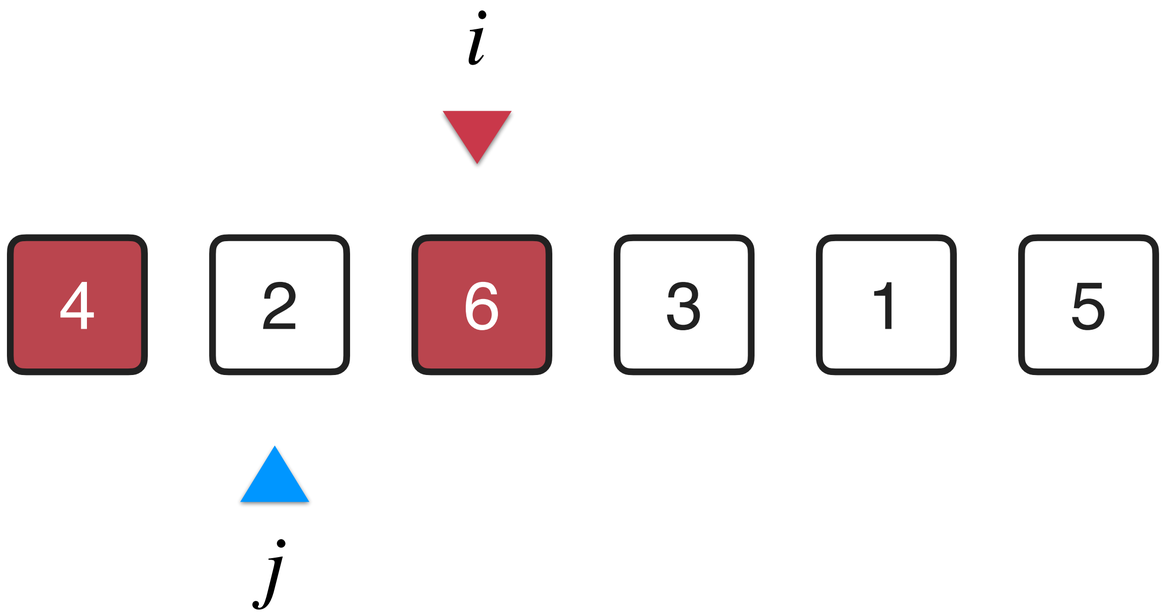
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

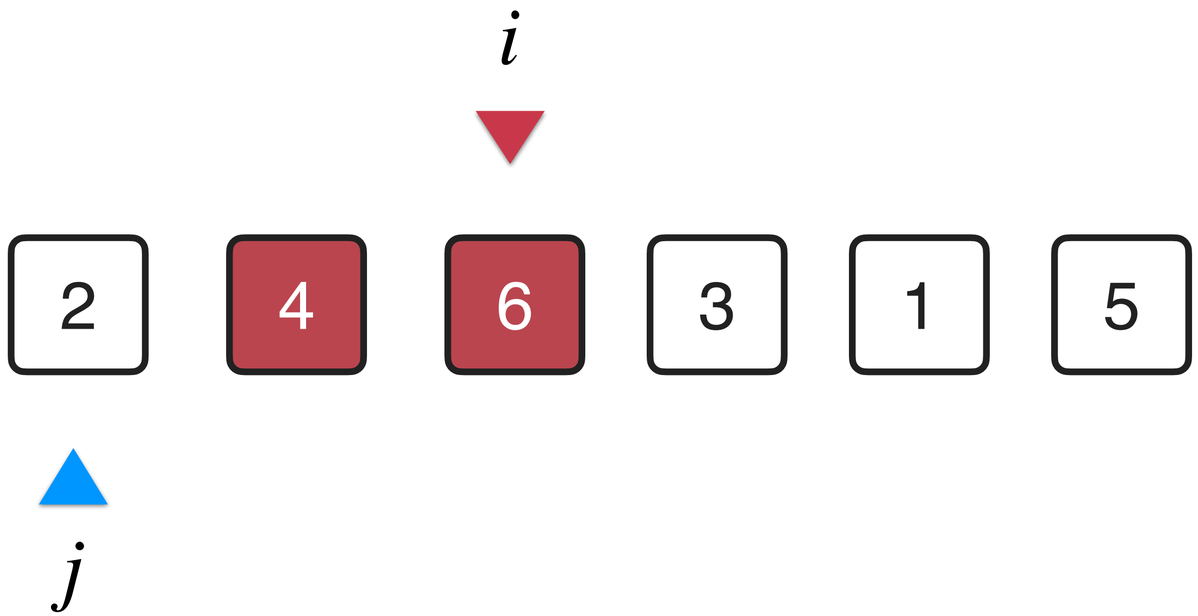
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

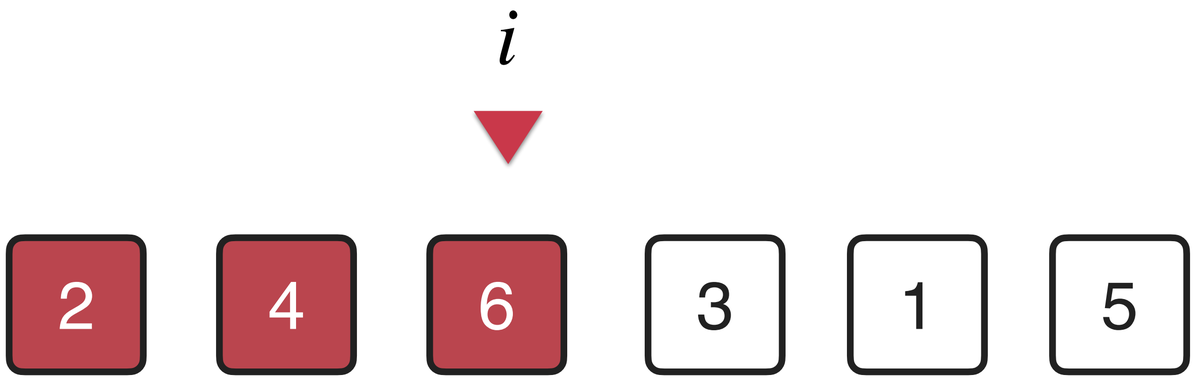
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

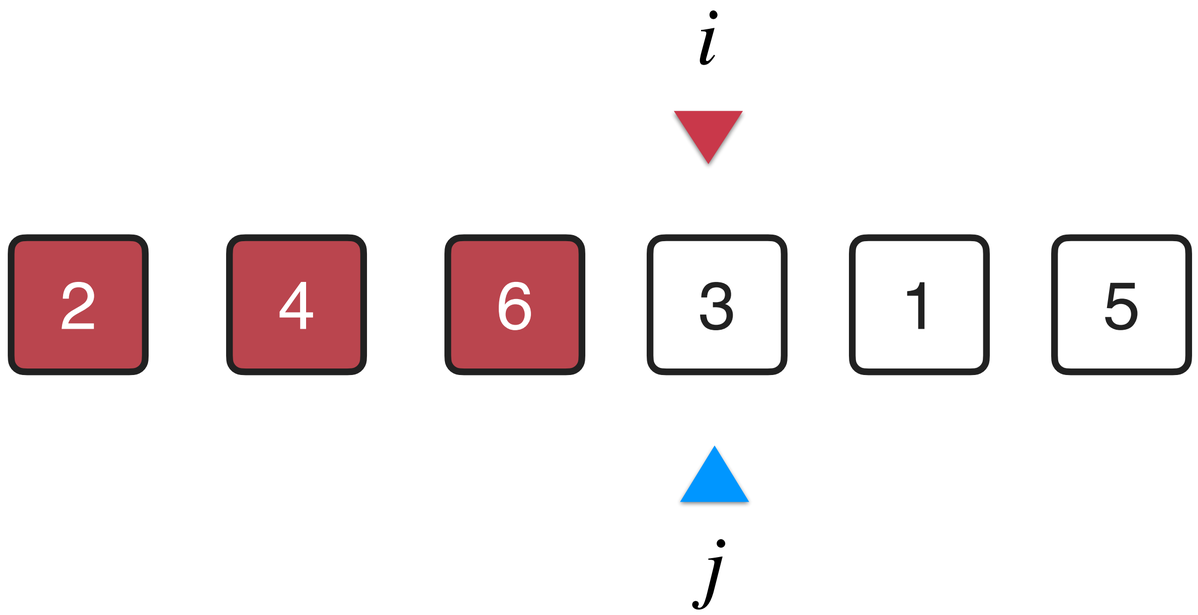
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

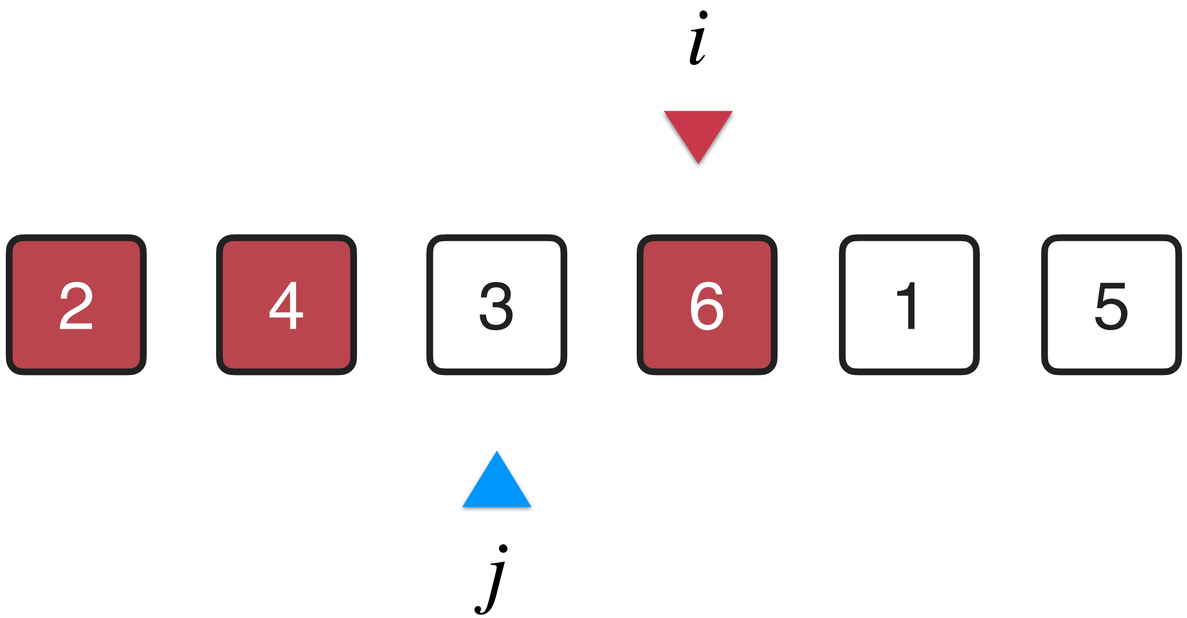
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

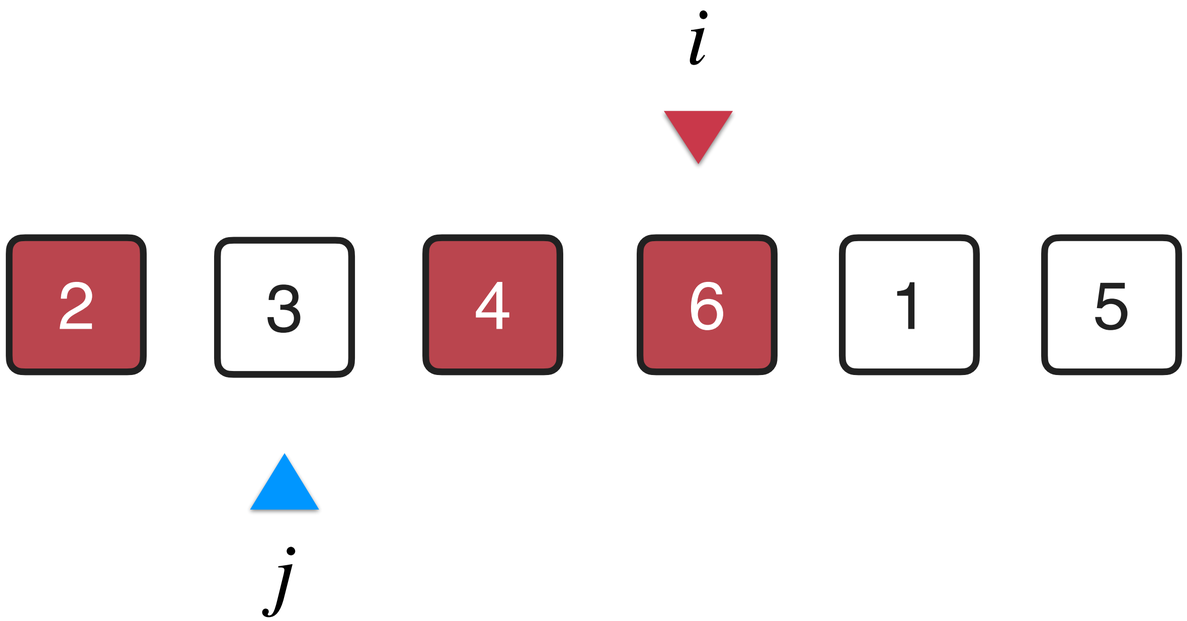
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

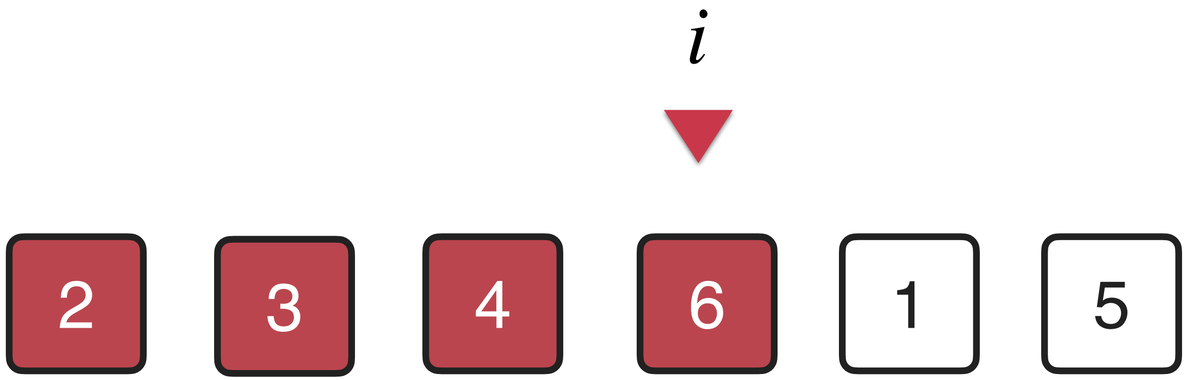
把 arr[i] 放到合适的位置



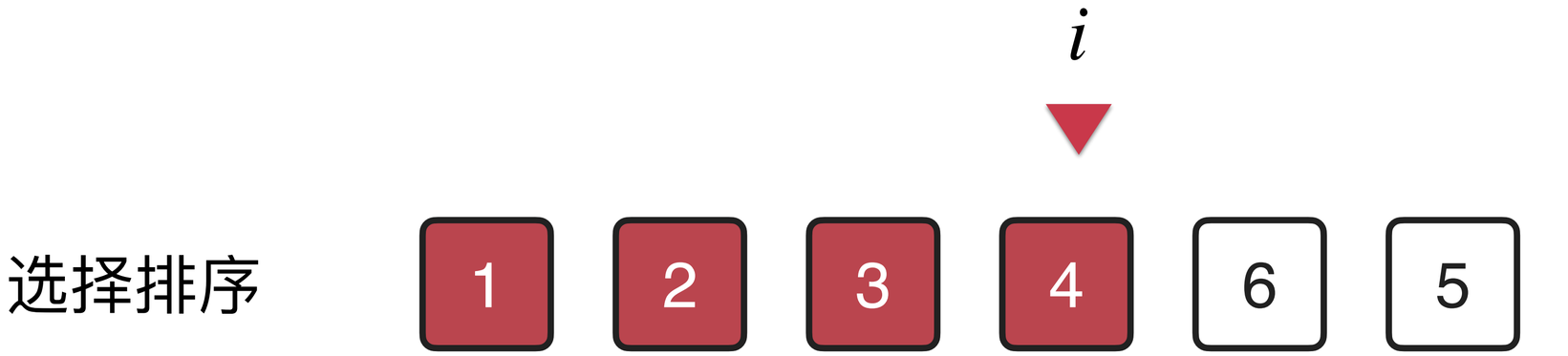
插入排序法

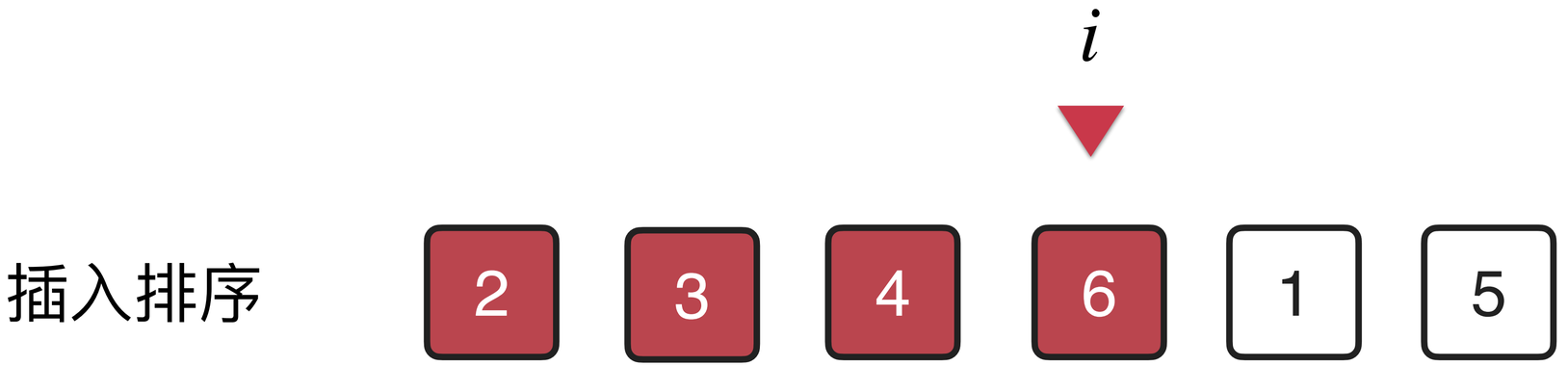
arr[0, i) 已排好序；arr[i…n) 未排序

把 arr[i] 放到合适的位置



插入排序法





实现插入排序法

liuyubobobo

实现插入排序法

实践：实现插入排序法

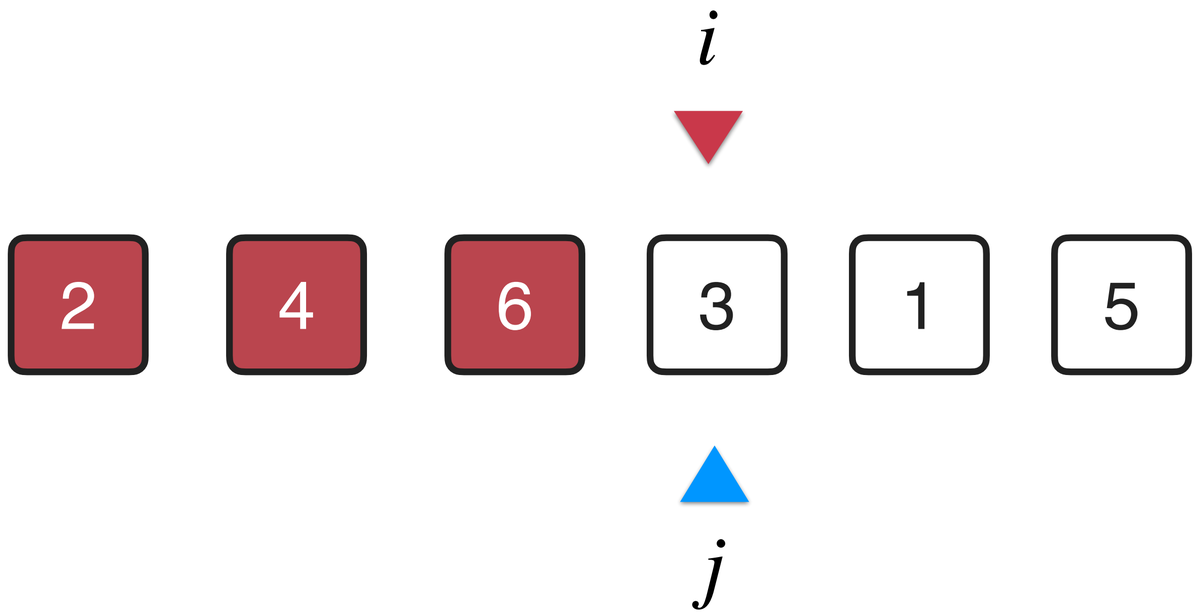
插入排序法的一个小优化

liuyubobobo

插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

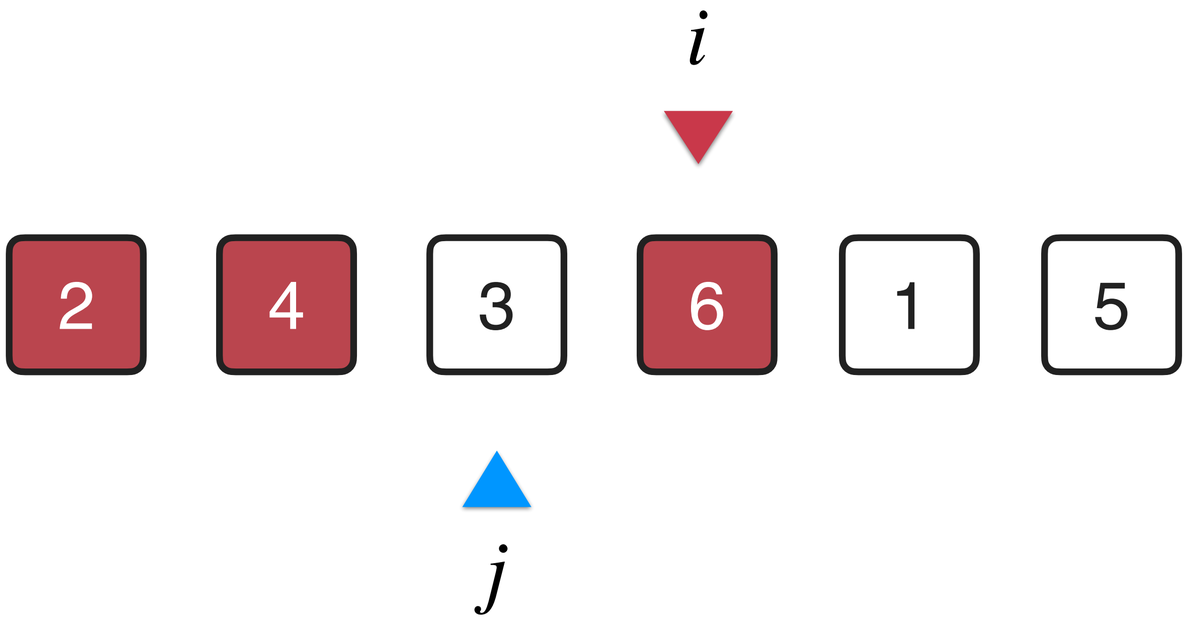
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

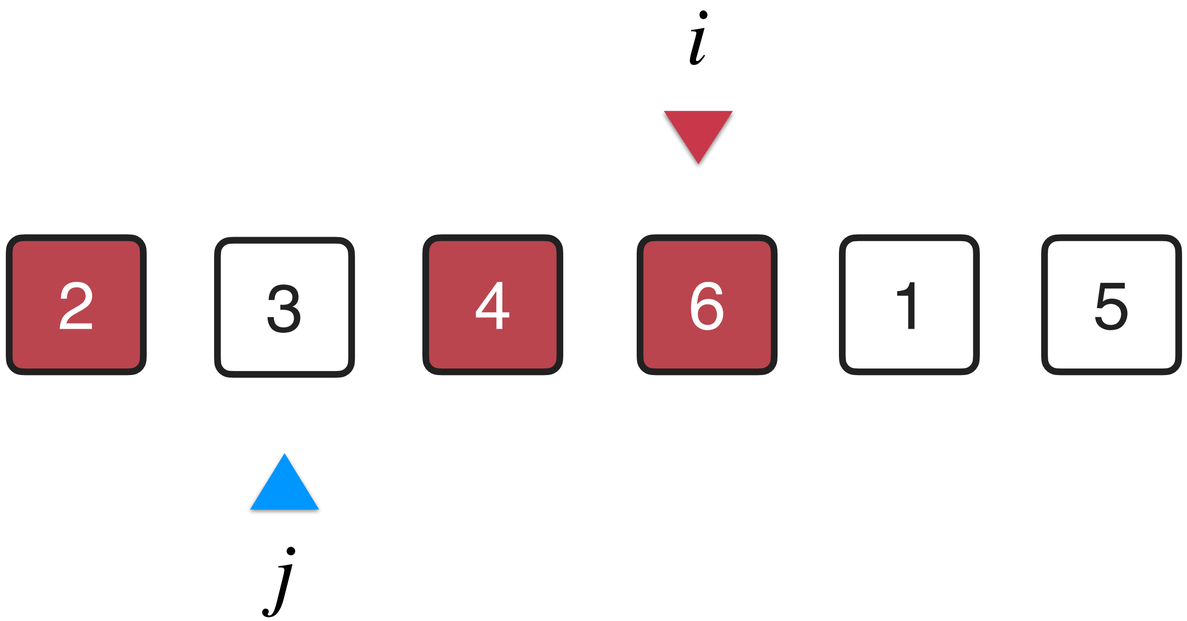
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

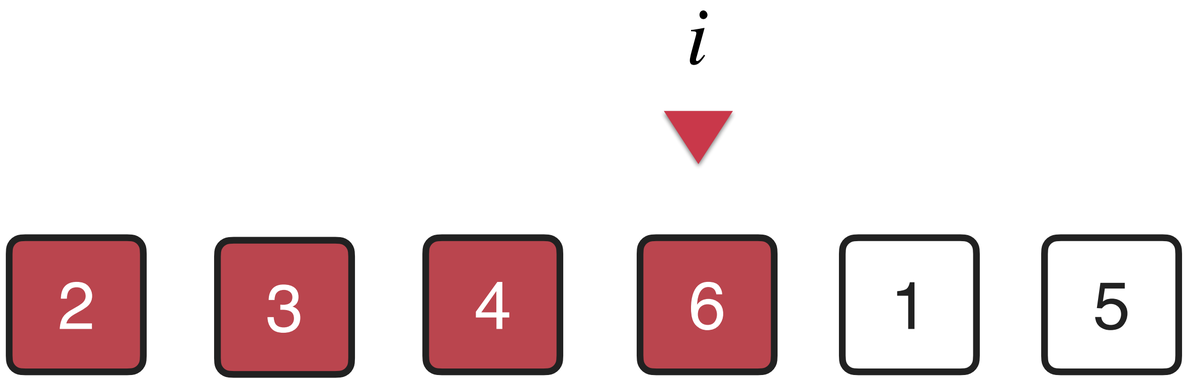
把 arr[i] 放到合适的位置



插入排序法

arr[0, i) 已排好序；arr[i…n) 未排序

把 arr[i] 放到合适的位置

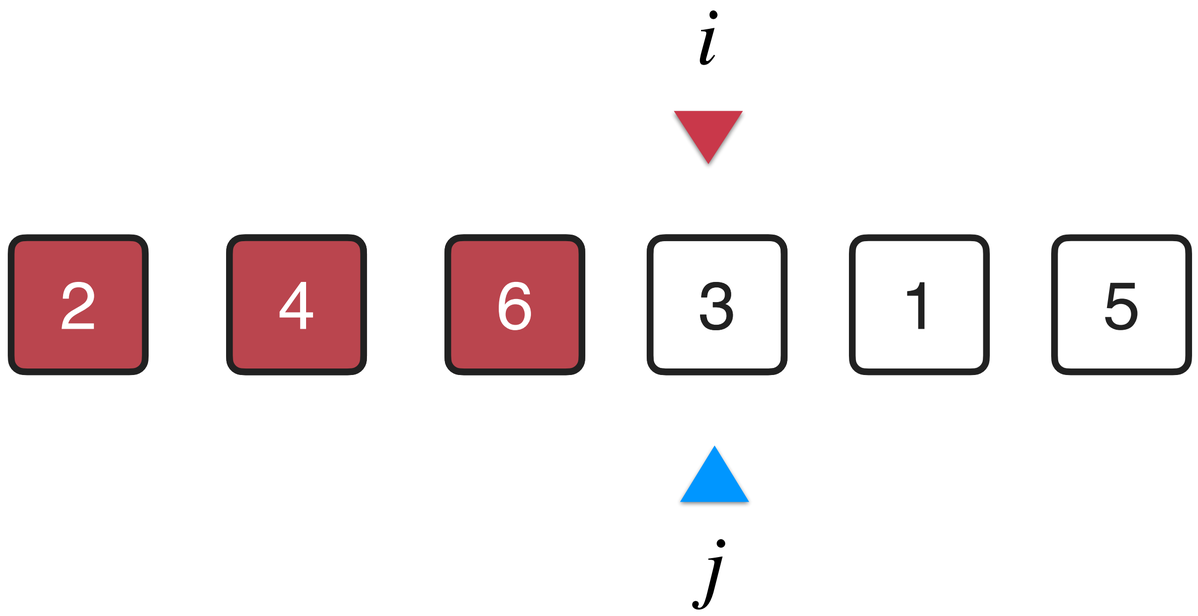


每次交换是三次操作

插入排序法的一个小优化

arr[0, i) 已排好序；arr[i…n) 未排序

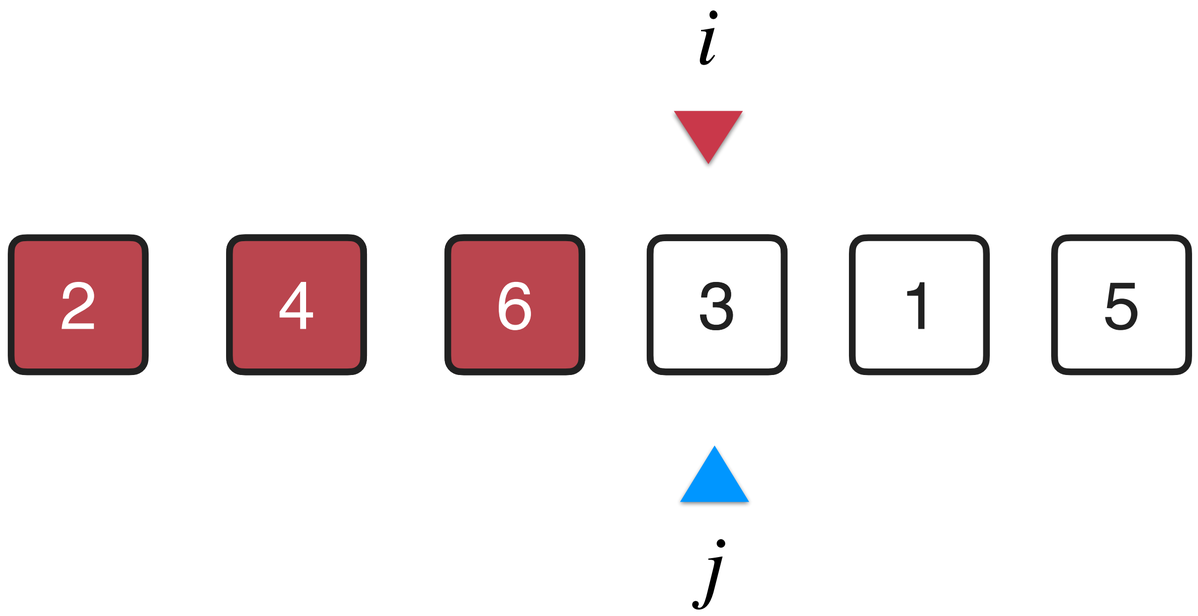
把 arr[i] 放到合适的位置



插入排序法的一个小优化

arr[0, i) 已排好序；arr[i…n) 未排序

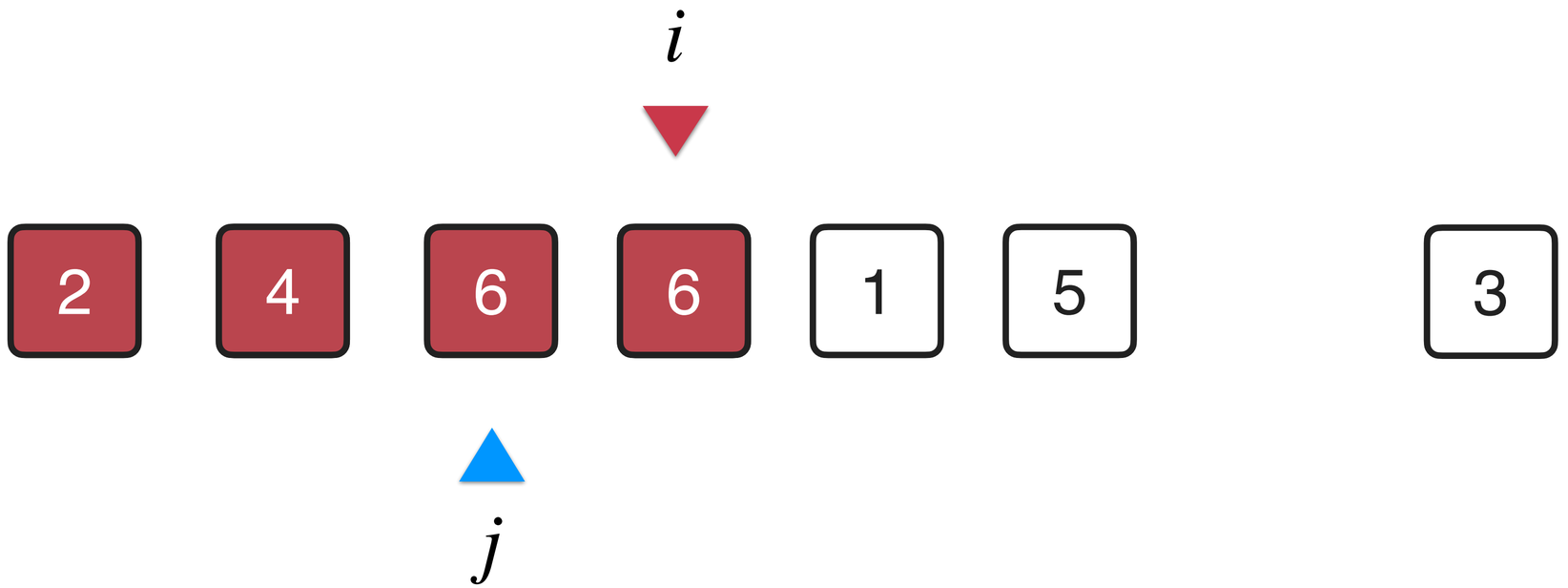
把 arr[i] 放到合适的位置



插入排序法的一个小优化

arr[0, i) 已排好序；arr[i…n) 未排序

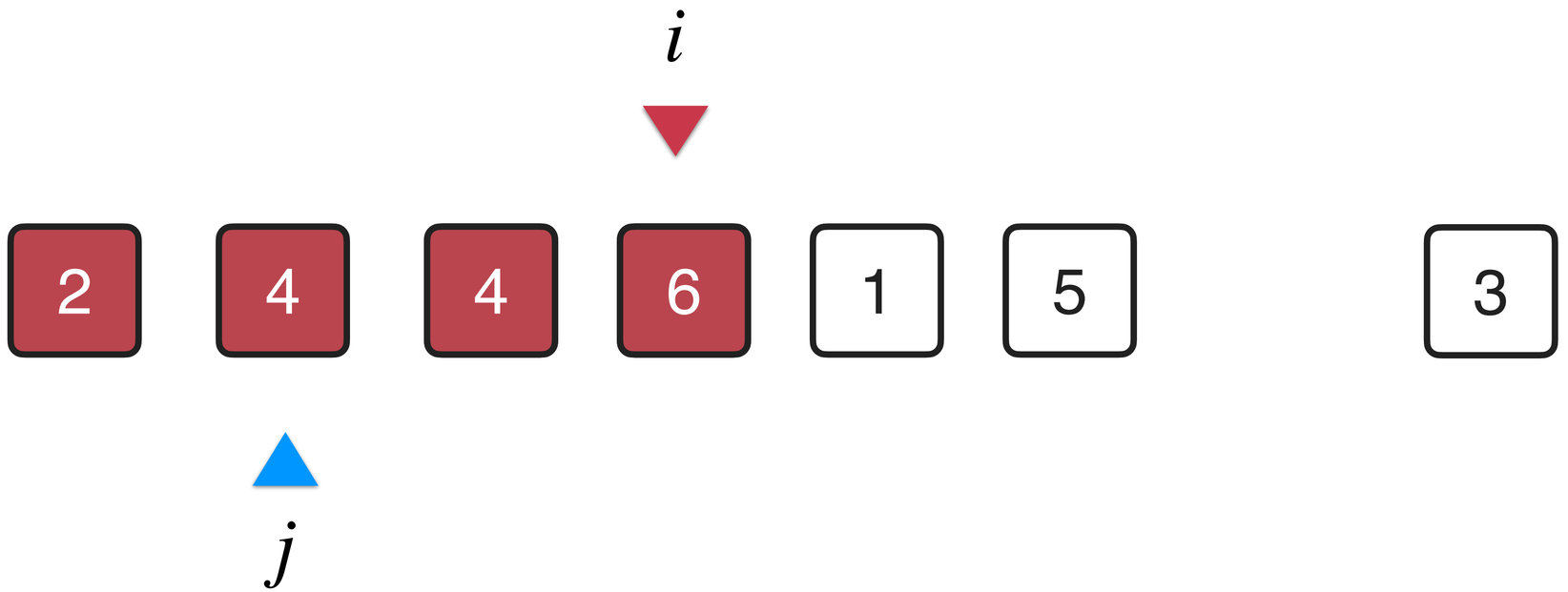
把 arr[i] 放到合适的位置



插入排序法的一个小优化

arr[0, i) 已排好序；arr[i…n) 未排序

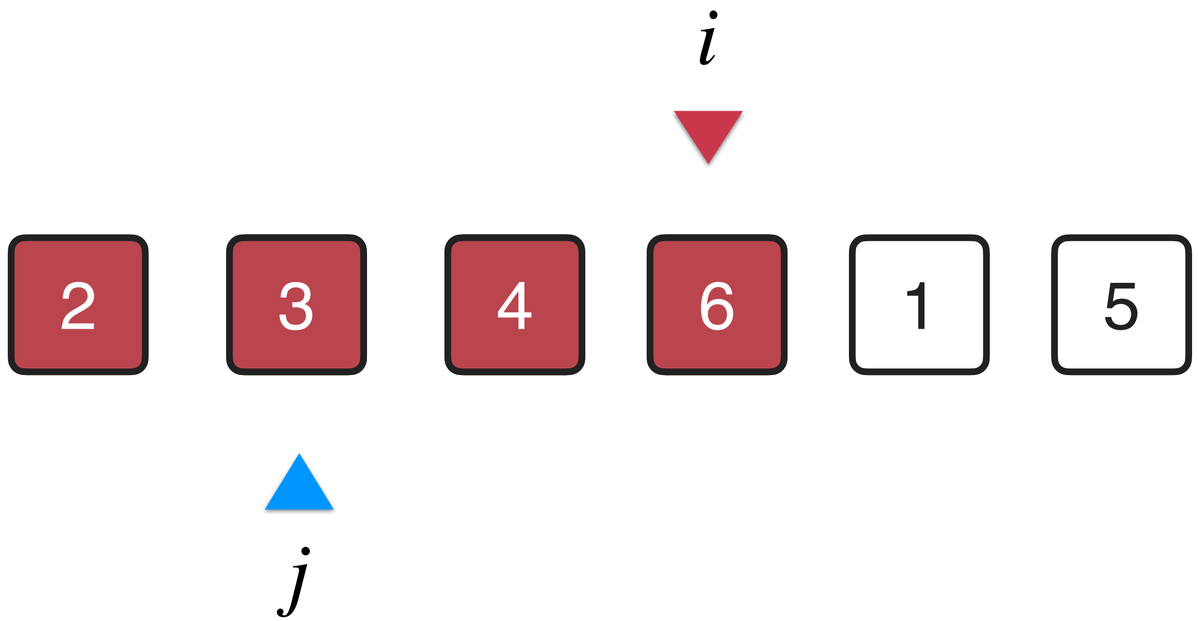
把 arr[i] 放到合适的位置



插入排序法的一个小优化

arr[0, i) 已排好序；arr[i…n) 未排序

把 arr[i] 放到合适的位置



实现插入排序法的优化

实践：实现插入排序法的优化

插入排序法的重要特性

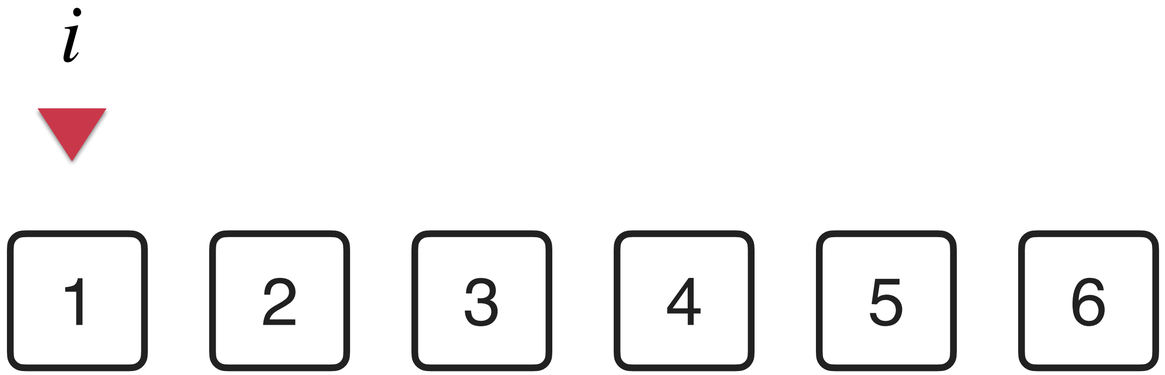
liuyubobobo

插入排序法的重要特性

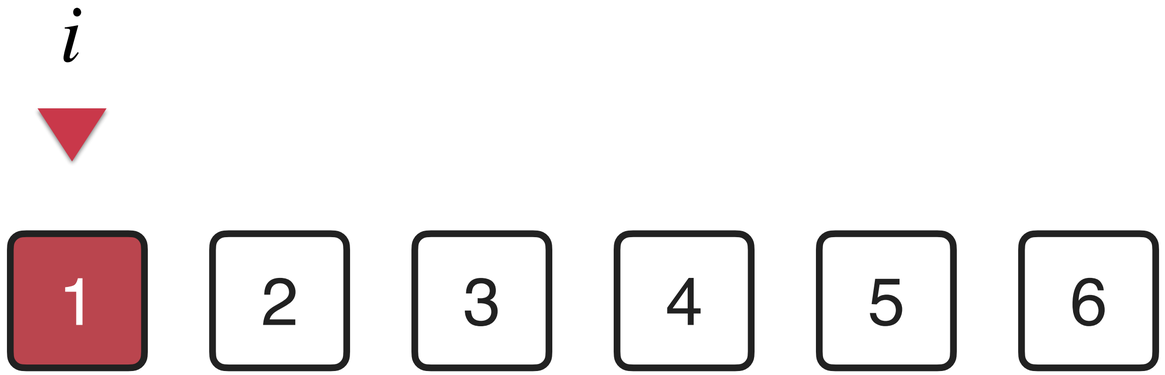
```java
public static <E extends Comparable<E>> void sort(E[] arr){

    for(int i = 0; i < arr.length; i ++){

        // 将 arr[i] 插入到合适的位置
        E t = arr[i];
        int j;
        for(j = i; j - 1 >= 0 && arr[j - 1].compareTo(t) > 0; j --){
            arr[j] = arr[j - 1];
        }
        arr[j] = t;
    }
}
```

O(n^2)

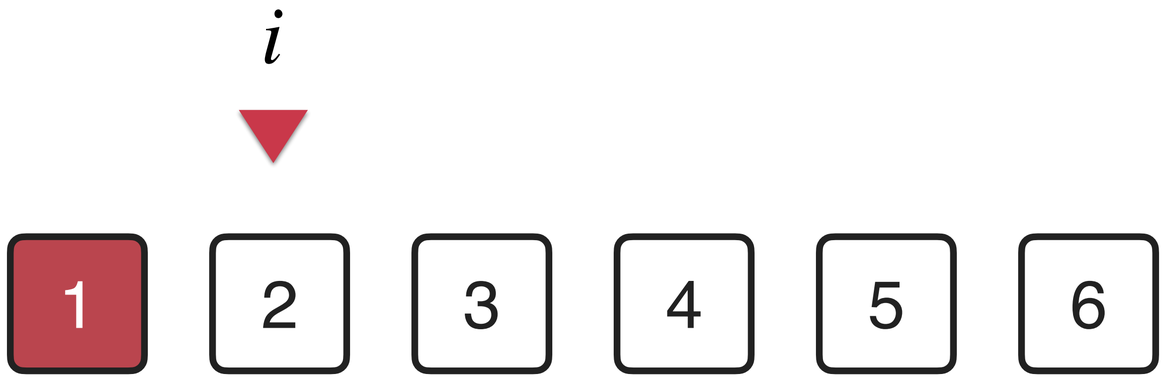
插入排序法的重要特性



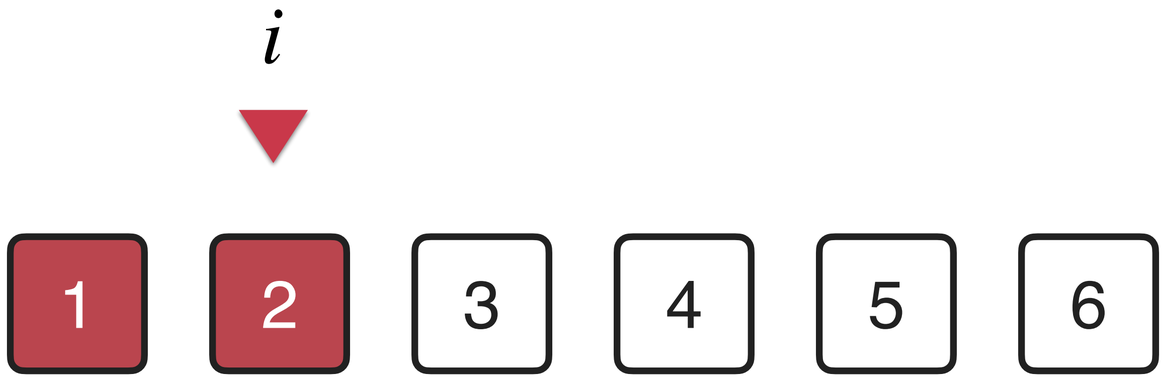
插入排序法的重要特性



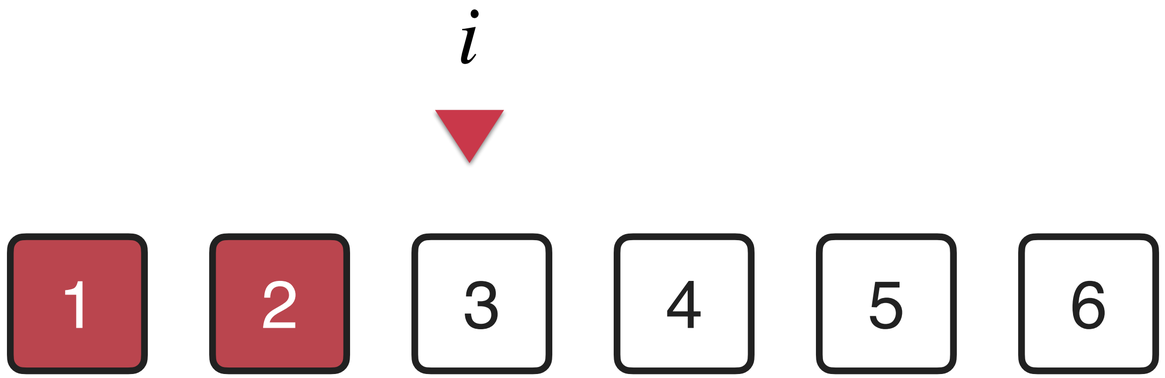
插入排序法的重要特性



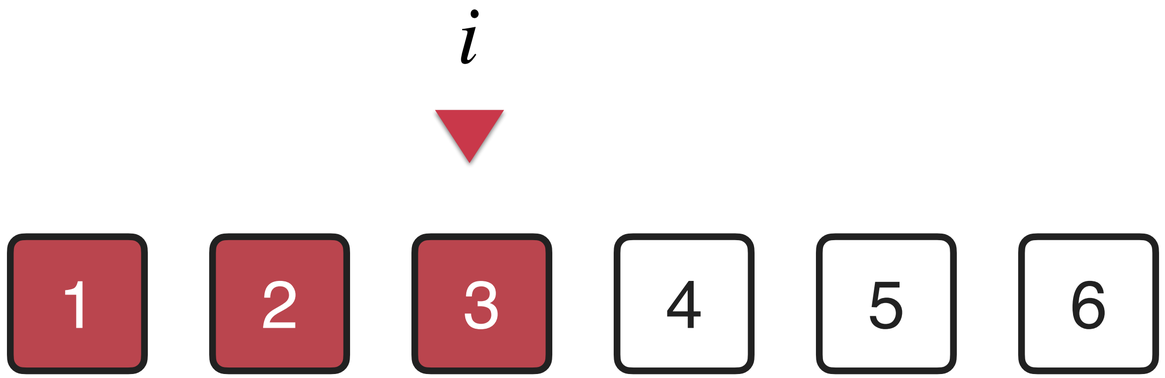
插入排序法的重要特性



插入排序法的重要特性



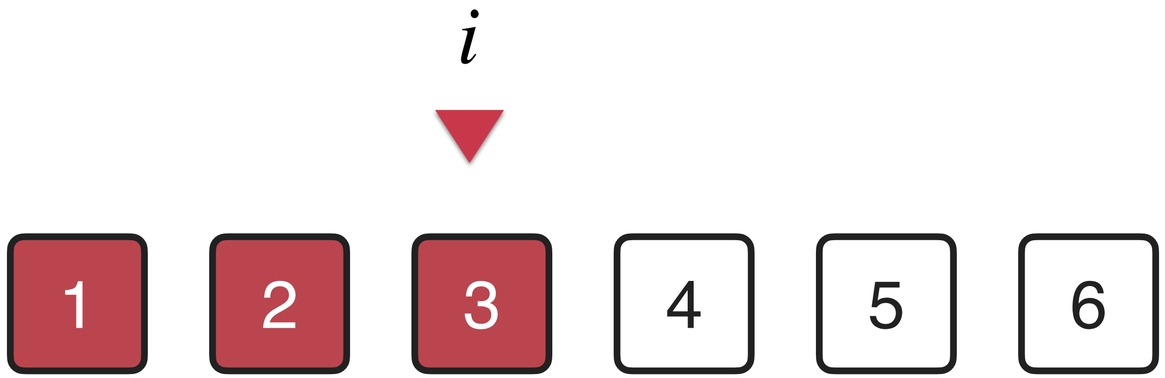
插入排序法的重要特性



插入排序法的重要特性

对于有序数组，插入排序的复杂度是 O(n) 的

整体，插入排序的复杂度依然是 O(n^2) 的



对比，选择排序的复杂度永远是 O(n^2) 的

插入排序法的重要特性

实践：验证插入排序法的特性

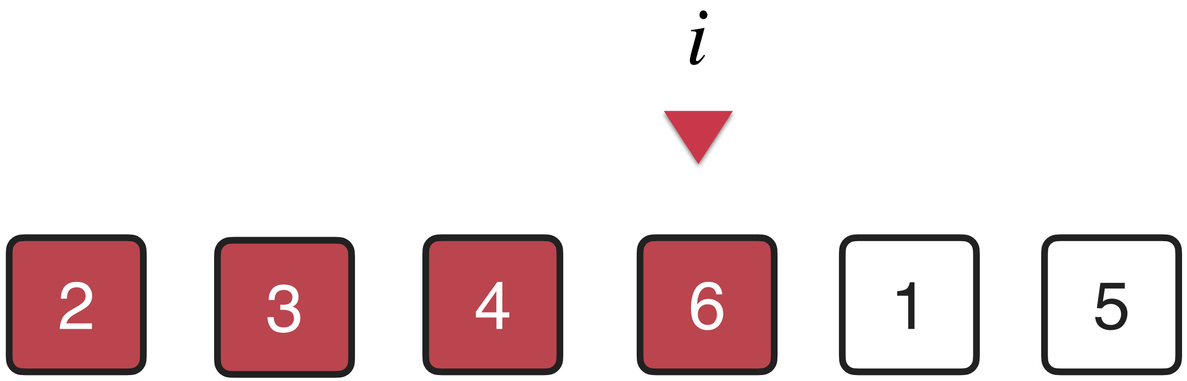
作业：换个方式实现插入排序法?

liuyubobobo

换个方式实现插入排序法?

arr[0, i) 已排好序；arr[i…n) 未排序

arr[0, i) 未排序；arr[i…n) 已排序



换个方式实现插入排序法?

arr[0, i) 未排序；arr[i…n) 已排序　　循环不变量

解析：换个方式实现插入排序法

liuyubobobo

本章小结

liuyubobobo

本章小结

选择排序法

插入排序法

循环不变量

均是 O(n^2) 算法

对于完全有序的数组，插入排序成为 O(n) 的算法

其他

欢迎大家关注我的个人公众号：是不是很酷

算法与数据结构体系课程

liuyubobobo