Priority Queue ADT

一个priority queue(P.Q.)， 是一个用来存储 优先元素的ADTT，这些element被称为values

PQ支持任意插入元素，但是移除元素是按优先顺序进行的

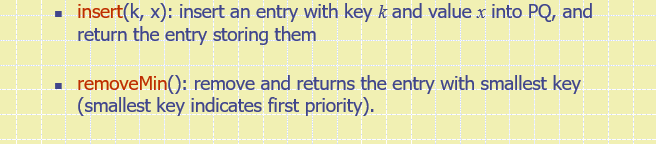
因此，p.q从根本上不同于其他position-based的adt(如stack、队列、Dq等)，这些Position-based对特定position进行操作。

PQ ADT 根据优先级存储元素，不会向用户公开位置

一个key可以用来代表着 value 的优先级，而value这里代表着一个element

PQ里的一个entry实际上就是输入一组（key,value）

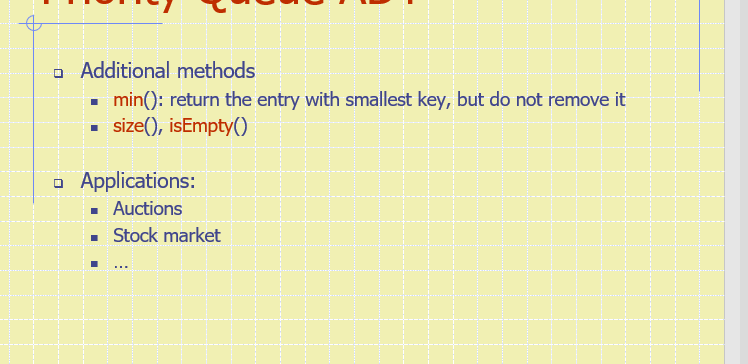
PQ ADT的main method

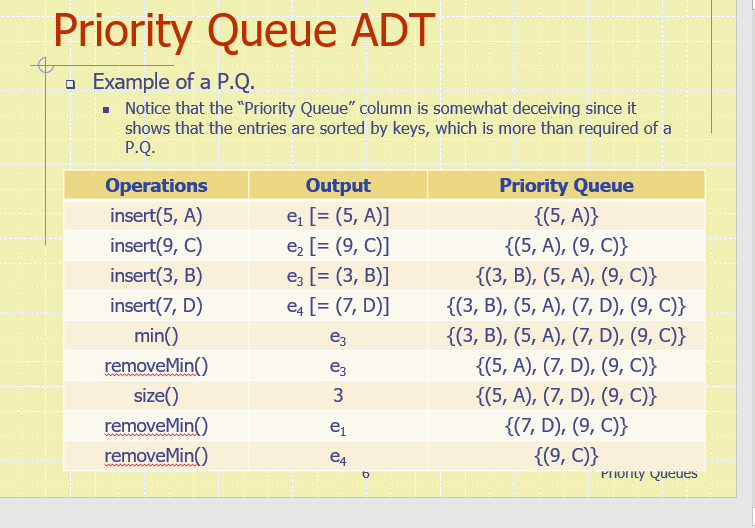


Insert 会插入一个entry条目 with key k 和value x，并会return这个存储（k,x）的entry

remove会remove并且return 拥有最小key的entry(key最小意味着优先级最高)

辅助method





会自动生成一个entry,return的时候只会return这个entry

Total Order Relations 序列关系

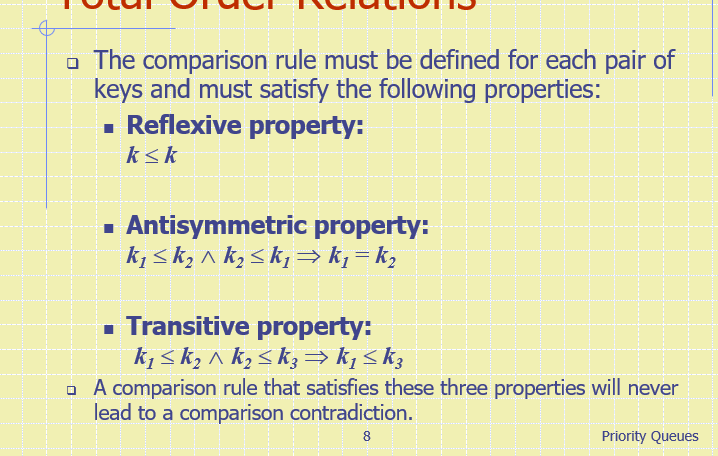
PQ中的key 可以是任意已经有顺序的object

两个不同的entry可以有同一个key

PQ需要一个永远不会自相矛盾的比较规则

比较规则（用<=表示），必须表示一个total order关系

比较规则必须比较每一pair的key并且必须满足一下关系



自反属性，他永远小于等于他自己

k1<=k2并且k2<=k1,那么k1=k2

如果k1<=k2,k2<=k3，那么k1<=k3

PQ有两种特殊的object

entry object: 用来track keys 与value的练习

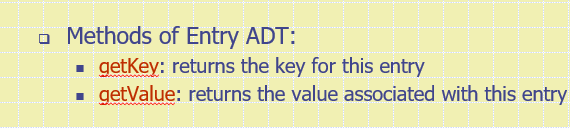
comparator object:用来比较到底哪个是smallest key

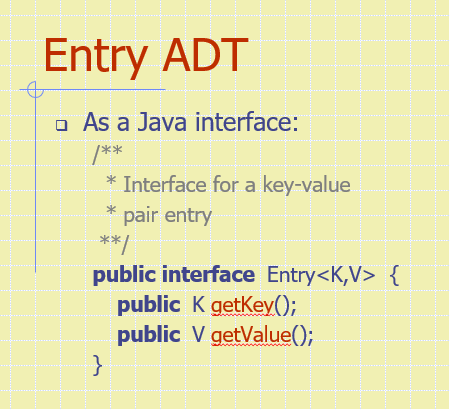
Entry ADT

一个entry是一个key-value pair，换句话说一个entry object是由一个key object和一个value object组成的

PQ存储entry这样他就可以做到有效的insertion与removal

Entry ADT的method





Comparator ADT

需要key之间能够互相比较，写一个通用PQ，可以存储key class里的有良好natural ordering的instance

然而这种方法就是key需要提供更多的信息，比如比较规则，如果比较大小7<21，如果比较字符顺序，7>21，

这个例子需要key他们自己来提供比较规则

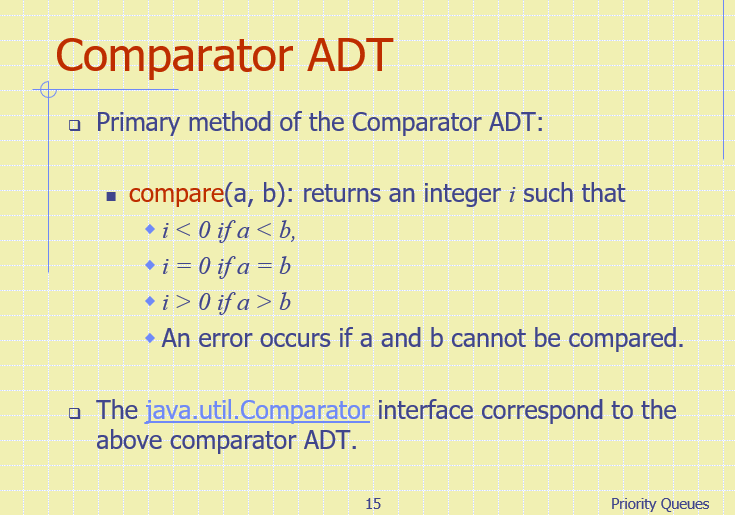
所以作为替代，我们可以用特殊的在key之外的comparator object来提供comparison rules

一个comparator比较器里面封装了一个根据两个object的顺序来比较两个object的动作

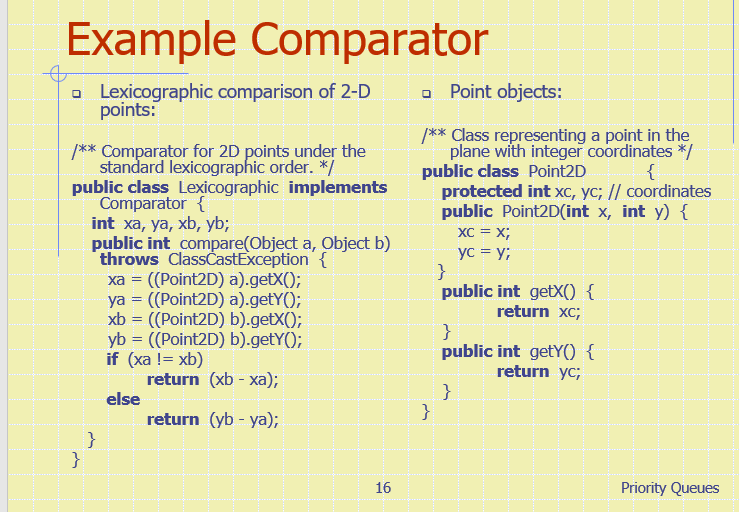
一个通用的PQ用同一个辅助comparator

我们假设和一个PQ在构建时被给了一个comparator object，那么这个PQ就用这个comparator来进行key comparison

总结：COMPARATOR是一个大类，你可以用这个类定义各种各样的规则，其中一个object包含了他的独特比较规则，建立PQ时我们会给他一个特殊object，就相当于给他指定了一个比较规则//例如自然数比较or字符顺序比较



1. 比较坐标点以字典顺序：X小的就大，X相等再比较Y，比如ab在ba的前面，ab在ac的前面



左边的是comparator，他收取了两个点并进行比较X和Y

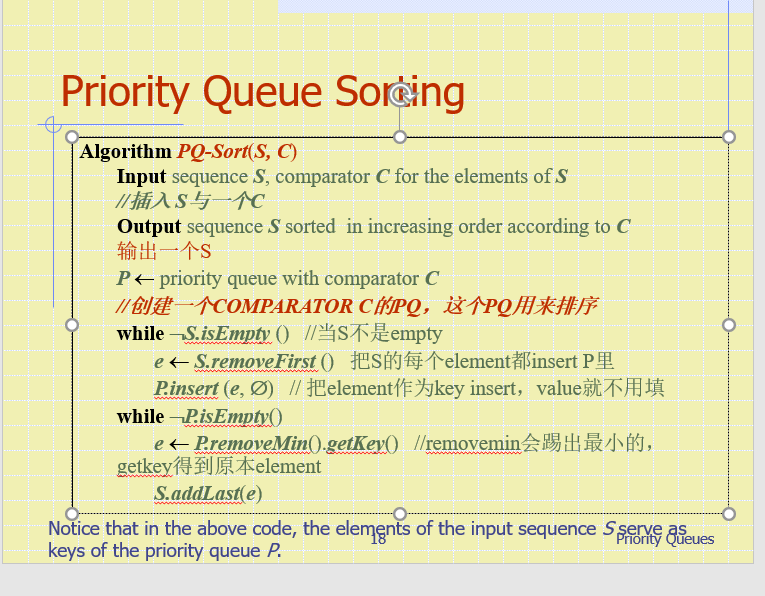
右边的是点的构造器

我们用PQ来给一系列可以比较的element排序

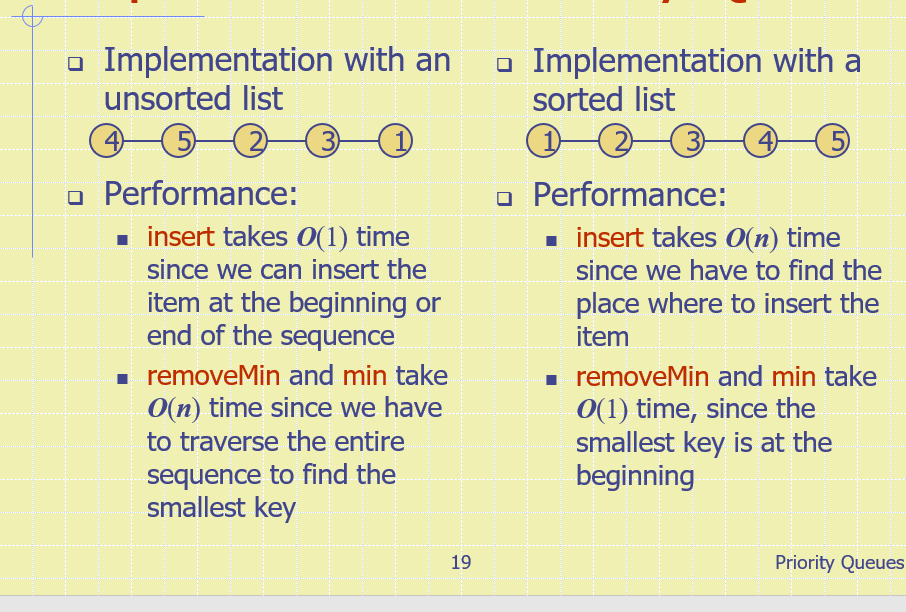
通过一系列insert操作逐个插入元素

通过一系列removeMin操作按排序顺序删除元素

此排序方法的运行时间取决于PQ本身是怎么实现的



基于sequence的pq



左边的是没拍过虚的list，insert只用O(1),插在开始或末尾

RemoveMin需要O（n），应为要遍历整个sequence来找到最小Key

Sorted list

Insert要O（n）因为我们输入的Key要正好插在位置上

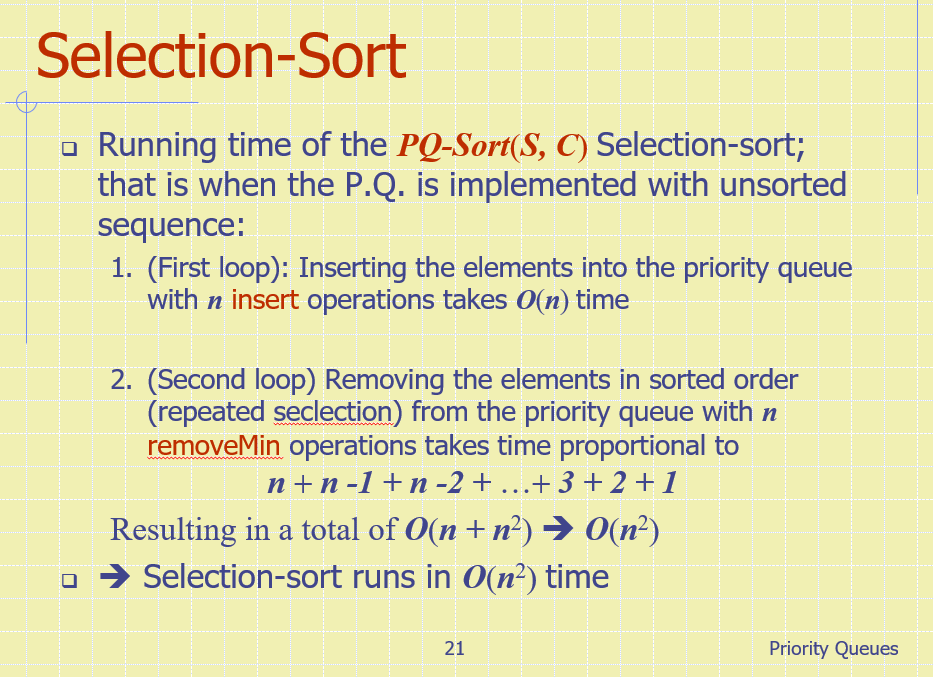
Remove就是O（1）因为最小的就是开头

Selection sort

见文件夹里的图，从0到N-1找到最小数，与Index0 互换

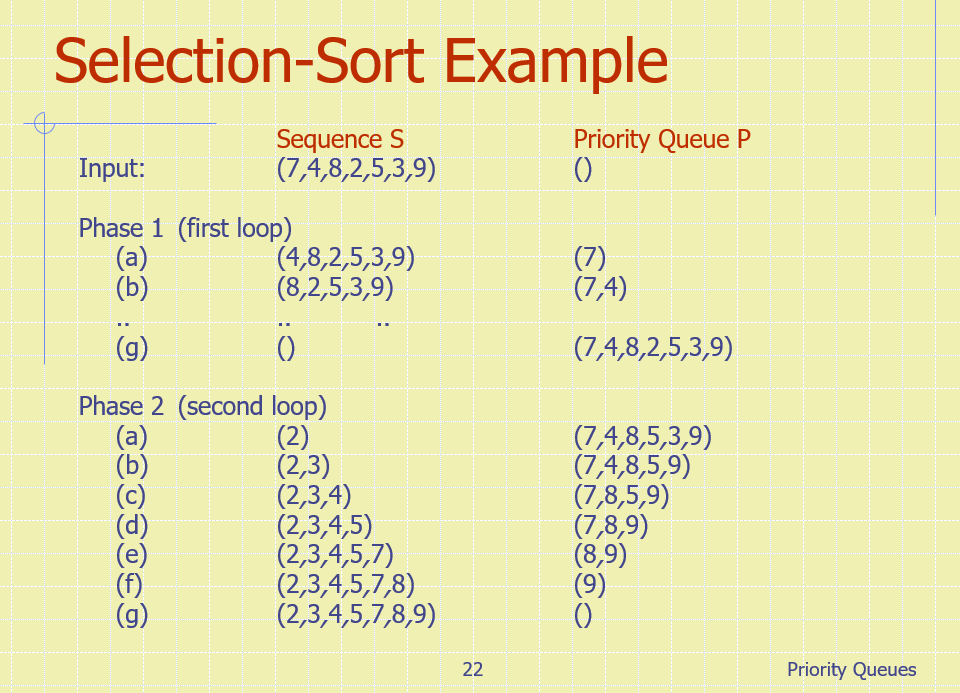
从1到N-1找到最小数，与INDEX1互换

一直循环



PQ SORT的selection sort running time当pq由一个Unsorted sequence表示时，

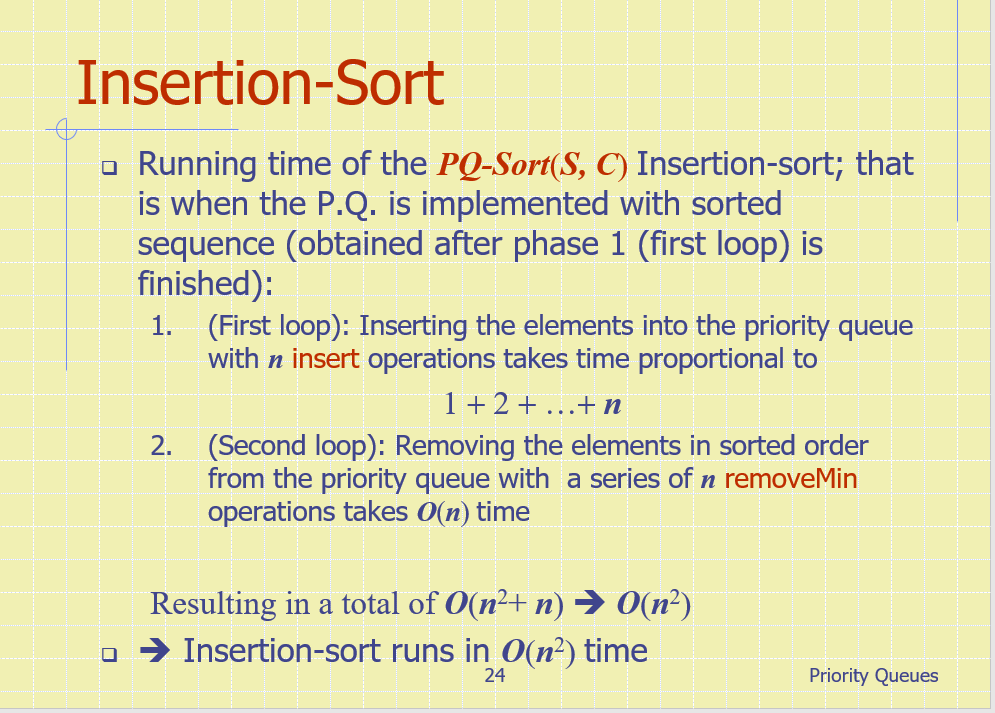
1. 第一个循环，把所有元素全部insert进PQ中，每个是O（1），一共是O（n）
2. 第二个循环，REMOVE 最小的从PQ中，然后放回S，每一步需要n,n-1//见上一页纸，总共需要n^2
3. 所以需要O（n^2）



Insertion sort

图在文件夹里

1. 从input sequence中移除一个element
2. 把这个element插入sorted 的PQ中
3. PQ从左到右remove



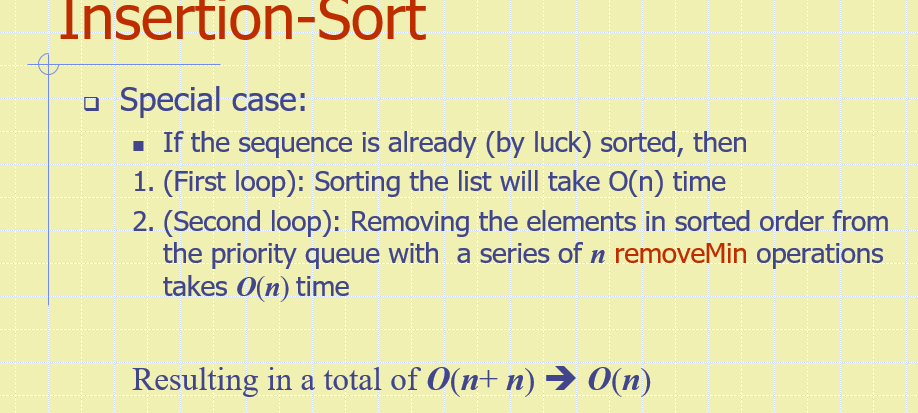
插入因为要比较哪里插入，所以是O（n）,1+2+3…

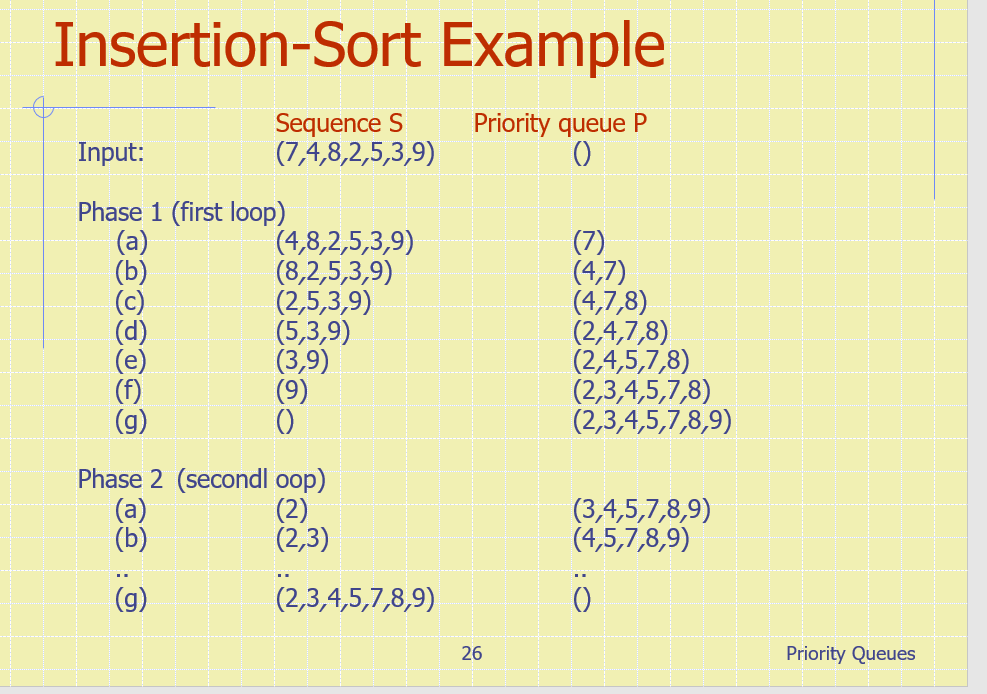
第二步，因为已经是sorted了·，每一次O(1),总共O（n）

总共O(N^2)

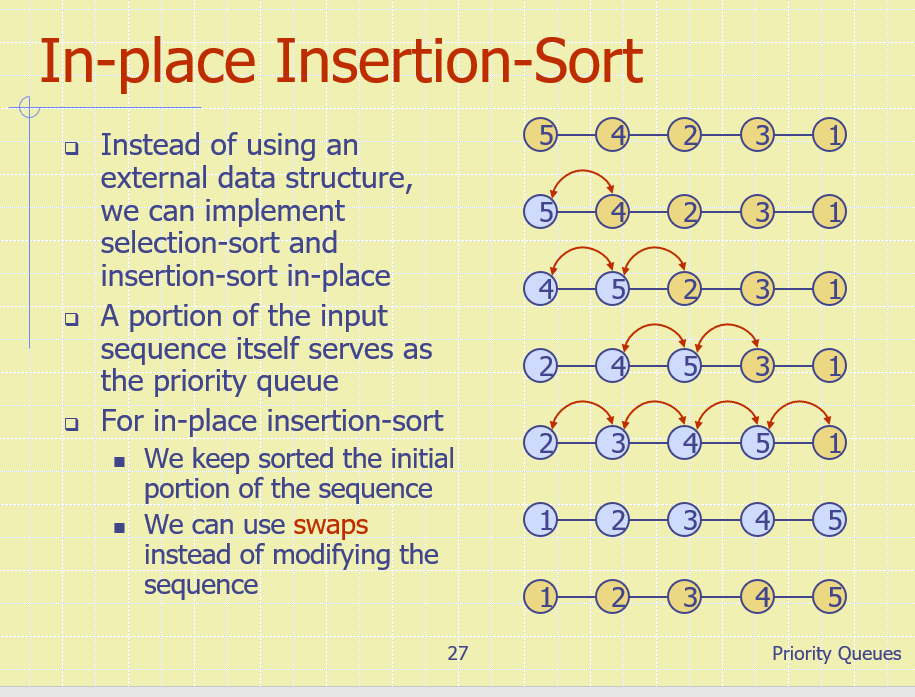
特殊情况：如果sequence已经sorted好了

1. sort每一次花O（1）,总共花O(N)
2. 因为已经是sorted了·，每一次O(1),总共O（n）





我们可以 内置 insertion sort or selection sort



只要把输入的sequence当做priority queue，然后用swap