

12A2 基本实现

#数据结构邓神

Efficiency and Cost

向量

Vector

| getMax() | delMax() | insert() |
|-------------|--|-------------------|
| traverse() | remove(traverse()) | insertAsLast(e) ✓ |
| $\Theta(n)$ | $\Theta(n) + \mathcal{O}(n) = \Theta(n)$ | $\mathcal{O}(1)$ |

|Vector::insert(r, e)| = $\mathcal{O}(n - r)$

时间复杂度太高

有序向量

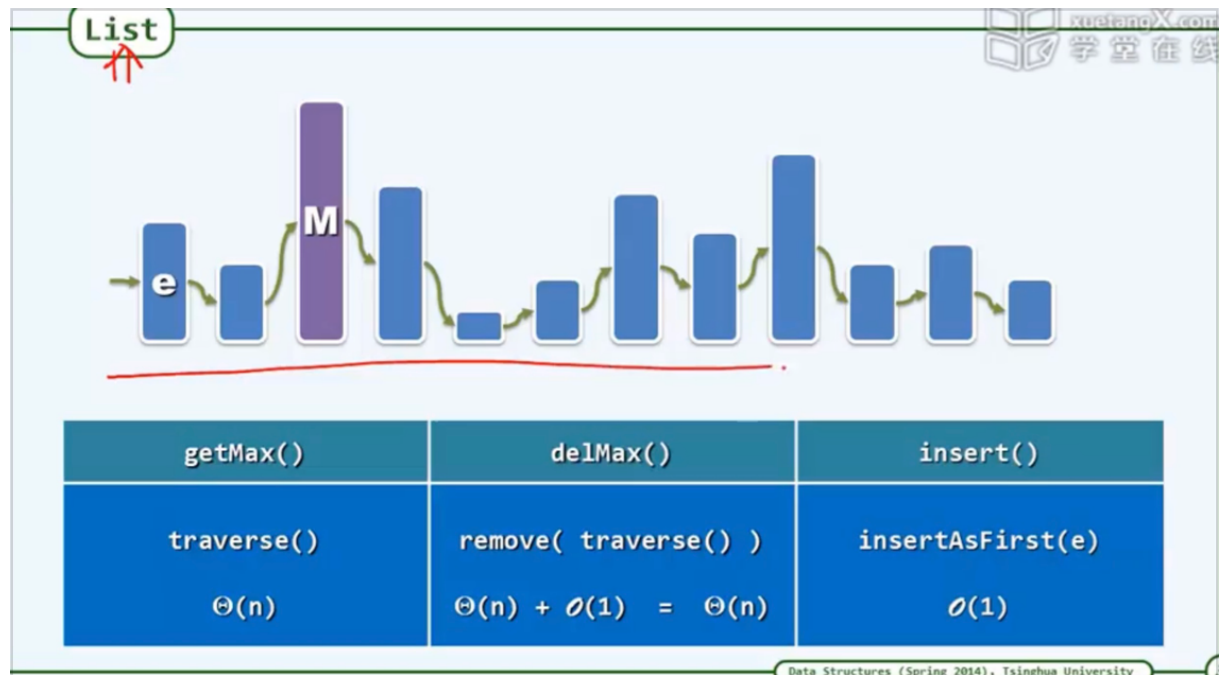
Sorted Vector

| getMax() | delMax() | insert() |
|------------------|-------------------|---|
| $[n - 1]$ | remove($n - 1$) | insert(1 + search(e), e) |
| $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(\log n) + \mathcal{O}(n) = \mathcal{O}(n)$ |

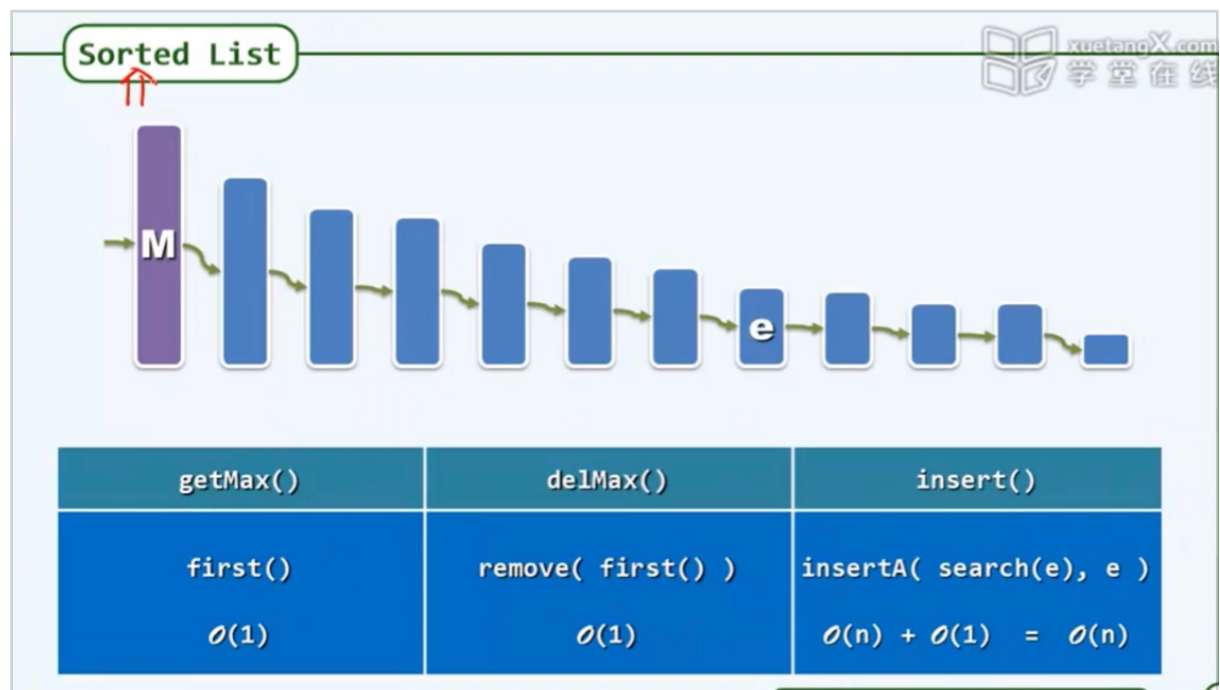
不可避免的还存在 $O(n)$

时间复杂度还是过高!

List



Sort List



BBST 平衡二叉搜索树 杀鸡用牛刀

无论是 ACL, Splay, 还是 Red-black 三个接口的效率都只需要 $O(\log n)$

而且只需要稍做优化, `getMax()`接口可以轻松达到 $O(1)$

但是 BBST 的功能远远超出了PQ的需求

$PQ = 1 * \text{insert}() + 0.5 * \text{search}() + 0.5 * \text{remove}();$

太过于复杂了

如果只需要查找极值元，则不必维护所有元素之间的全序关系，便序就可以了

BBST xuetaangX.com 学堂在线

❖ AVL、Splay、Red-black：三个接口均只需 $O(\log n)$ 时间

但是，**BBST** 的功能远远超出了 PQ 的需求...

❖ $PQ = 1 \times \text{insert}() + 0.5 \times \text{search}() + 0.5 \times \text{remove}() = \frac{2}{3} \times \text{BBST}$

❖ 若只需查找极值元，则不必维护所有元素之间的全序关系，偏序足矣

❖ 因此有理由相信，存在某种更为简单、维护成本更低的实现方式

使得各功能接口 时间复杂度依然为 $O(\log n)$ ，而且 实际效率更高