

优先级队列

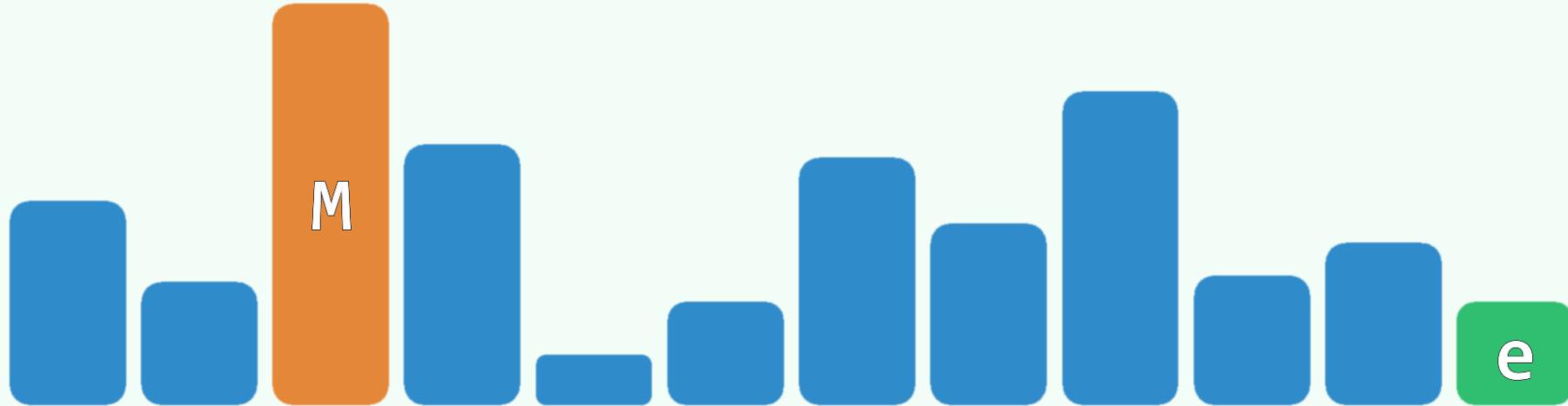
概述：基本实现

12-A2

邓俊辉

deng@tsinghua.edu.cn

# Vector



getMax()	delMax()	insert()
traverse()	remove( traverse() )	insertAsLast(e)
$\Theta(n)$	$\Theta(n) + \Theta(n) = \Theta(n)$	$\Theta(1)$

# Sorted Vector



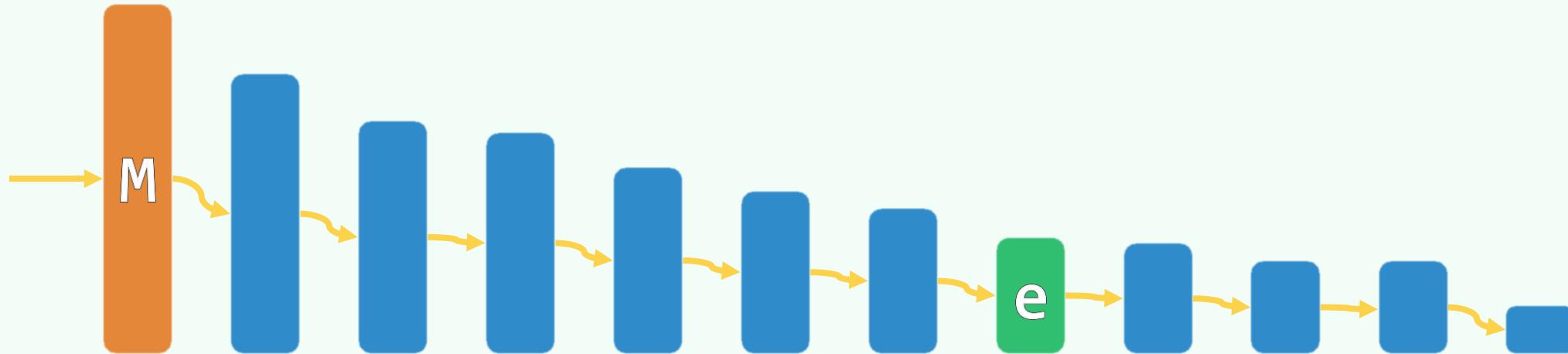
getMax()	delMax()	insert()
$[n - 1]$ $\mathcal{O}(1)$	$\text{remove}(n - 1)$ $\mathcal{O}(1)$	$\text{insert}(1 + \text{search}(e), e)$ $\mathcal{O}(\log n) + \mathcal{O}(n) = \mathcal{O}(n)$

# List



getMax()	delMax()	insert()
traverse()	remove( traverse() )	insertAsFirst(e)
$\Theta(n)$	$\Theta(n) + \Theta(1) = \Theta(n)$	$\Theta(1)$

# Sorted List



getMax()	delMax()	insert()
<b>first()</b> $\mathcal{O}(1)$	<b>remove( first() )</b> $\mathcal{O}(1)$	<b>insertA( search(e), e )</b> $\mathcal{O}(n) + \mathcal{O}(1) = \mathcal{O}(n)$

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

但是，BBST的功能远远超出了PQ的需求...

$$\text{❖ PQ} = \boxed{1 \times \text{insert}()} + \boxed{0.5 \times \text{search}()} + \boxed{0.5 \times \text{remove}()}$$

❖ 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

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

❖ 当然，就最坏情况而言，这类实现方式已属最优——为什么？

# 统一测试

```
❖ template <typename PQ, typename T> void testHeap( int n ) {  
    T* A = new T[ 2 * n / 3 ]; //创建容量为2n/3的数组，并  
    for ( int i = 0; i < 2 * n / 3; i++ ) A[i] = dice( (T) 3 * n ); //随机化  
    PQ heap( A + n / 6, n / 3 ); delete [] A; //Robert Floyd  
    while ( heap.size() < n ) //随机测试  
        if ( dice( 100 ) < 70 ) heap.insert( dice( (T) 3 * n ) ); //70%概率插入  
        else if ( ! heap.empty() ) heap.delMax(); //30%概率删除  
    while ( ! heap.empty() ) heap.delMax(); //清空  
}
```