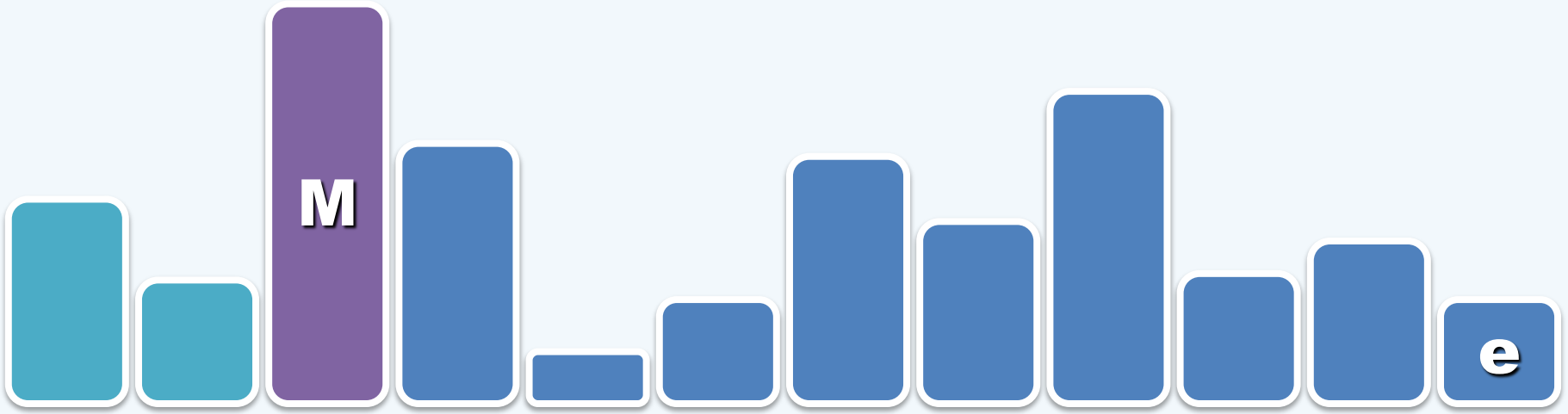
10.优先级队列

（a2）基本实现

邓俊辉

deng@tsinghua.edu.cn

Vector



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

Sorted Vector

getMax()	delMax()	insert()
[n – 1]	remove(n – 1)	insert(1 + search(e), e)
$O(1)$	$O(1)$	$O(\log n) + O(n) = O(n)$

getMax()	delMax()	insert()
traverse() Θ(n)	remove(traverse()) Θ(n) + O(1) = Θ(n)	insertAsFirst(e) O(1)

Sorted List

getMax()	delMax()	insert()
first() $O(1)$	remove(first()) $O(1)$	insertA(search(e), e) $O(n) + O(1) = O(n)$

BBST

- 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 × insert() + 0.5 × search() + 0.5 × remove() = $\frac{2}{3}$ × 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**

- 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
  }
```