12-A2

优先级队列

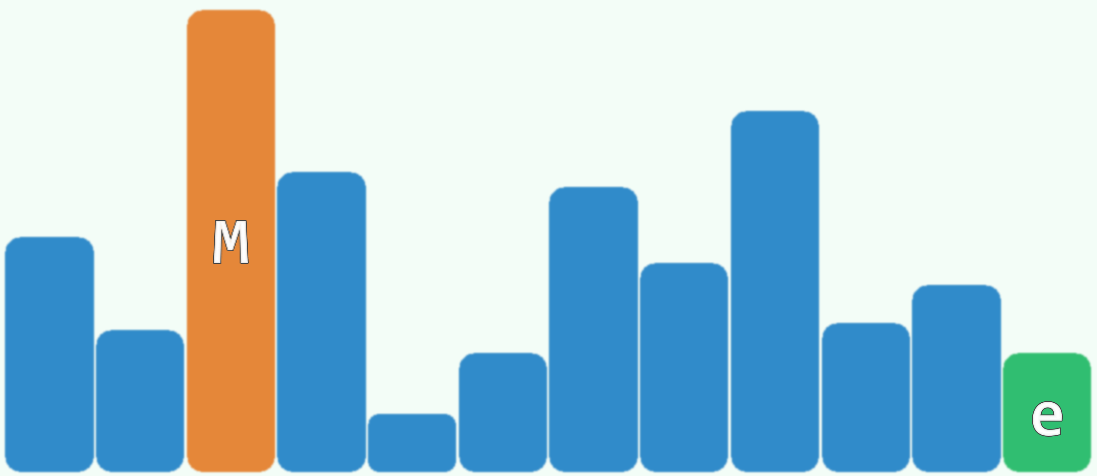
概述：基本实现

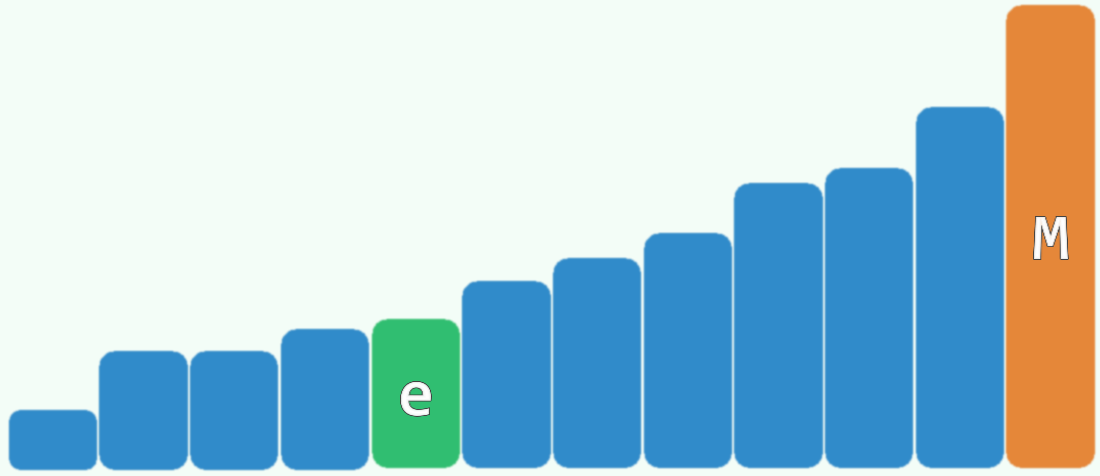
邓俊辉

deng@tsinghua.edu.cn

大兒孔文舉，小兒楊德祖；與子碌碌，莫足數也

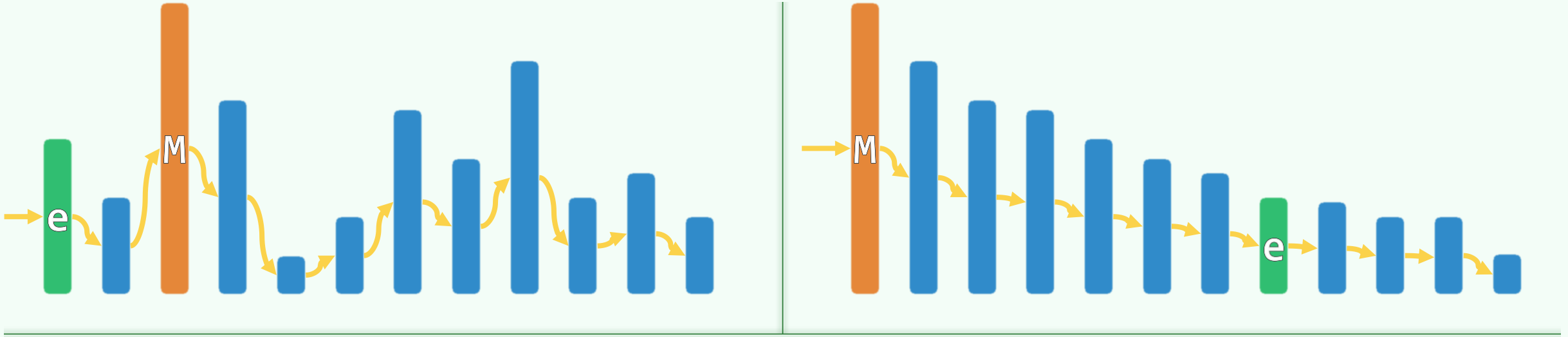
Vector





Vector	getMax()	delMax()	insert()
Unsorted	traverse() $\Theta(n)$	remove(traverse()) $\Theta(n) + \mathcal{O}(n) = \Theta(n)$	insertLast(e) $\mathcal{O}(1)$
Sorted	[n-1] $\mathcal{O}(1)$	remove(n-1) $\mathcal{O}(1)$	insert(1 + search(e), e) $\mathcal{O}(\log n) + \mathcal{O}(n) = \mathcal{O}(n)$

List



List	getMax()	delMax()	insert()
Unsorted	traverse() $\Theta(n)$	remove(traverse()) $\Theta(n) + \mathcal{O}(1) = \Theta(n)$	insertFirst(e) $\mathcal{O}(1)$
Sorted	first() $\mathcal{O}(1)$	remove(first()) $\mathcal{O}(1)$	insert(search(e), e) $\mathcal{O}(n) + \mathcal{O}(1) = \mathcal{O}(n)$

BBST

❖ 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：

PQ = 1 × insert() + 0.5 × search() + 0.5 × 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
}
```