10.优先级队列

（d）堆排序

邓俊辉

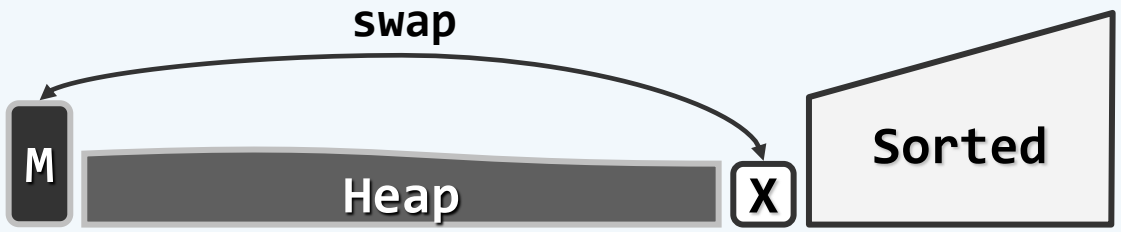
deng@tsinghua.edu.cn

算法

- J. Williams, 1964

  初始化 ： heapify()，O(n) //建堆

  迭代 ： delMax()，O(logn) //取出堆顶并调整复原

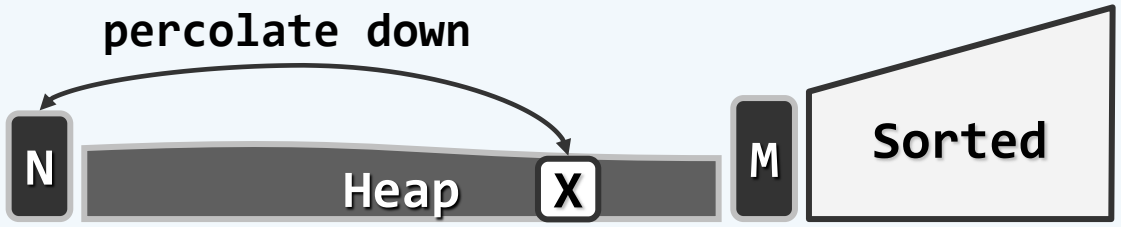


- 等效于常规selectionSort()，正确无疑

- O(n) + n x O(logn) = O(nlogn)

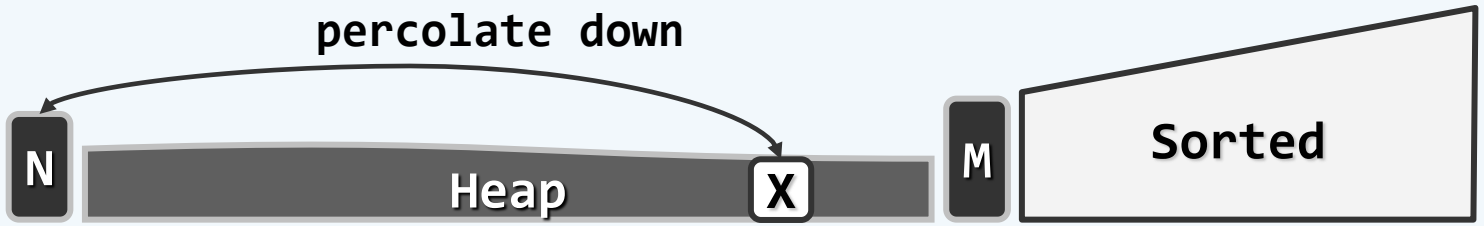
- 若词条已组织为向量

  完全可以就地排序——O(1)附加空间

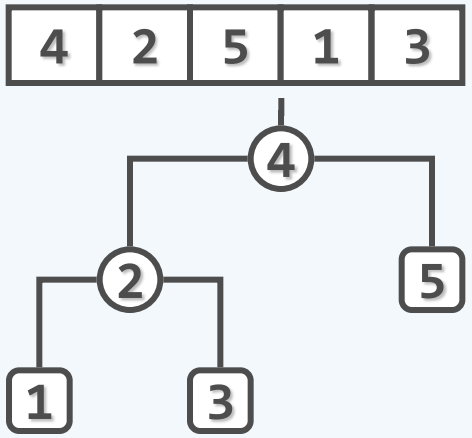


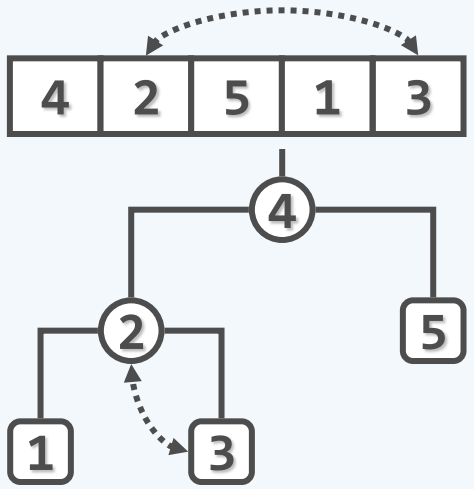
实现

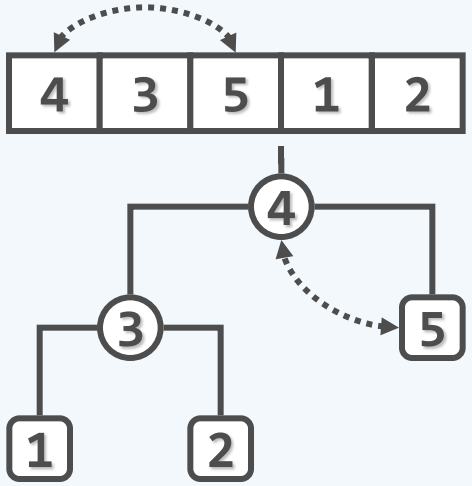
```
template <typename T> //对向量区间[lo, hi)做就地堆排序
void Vector<T>::heapSort( Rank lo, Rank hi ) {
   PQ_ComplHeap<T> H( _elem + lo, hi - lo ); //待排序区间建堆，O(n)
   while ( ! H.empty() ) //反复地摘除最大元并归入已排序的后缀，直至堆空
      _elem[ --hi ] = H.delMax(); //等效于堆顶与末元素对换后下滤
}
```

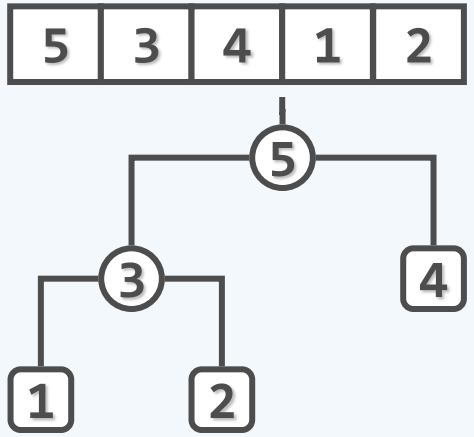


实例：建堆

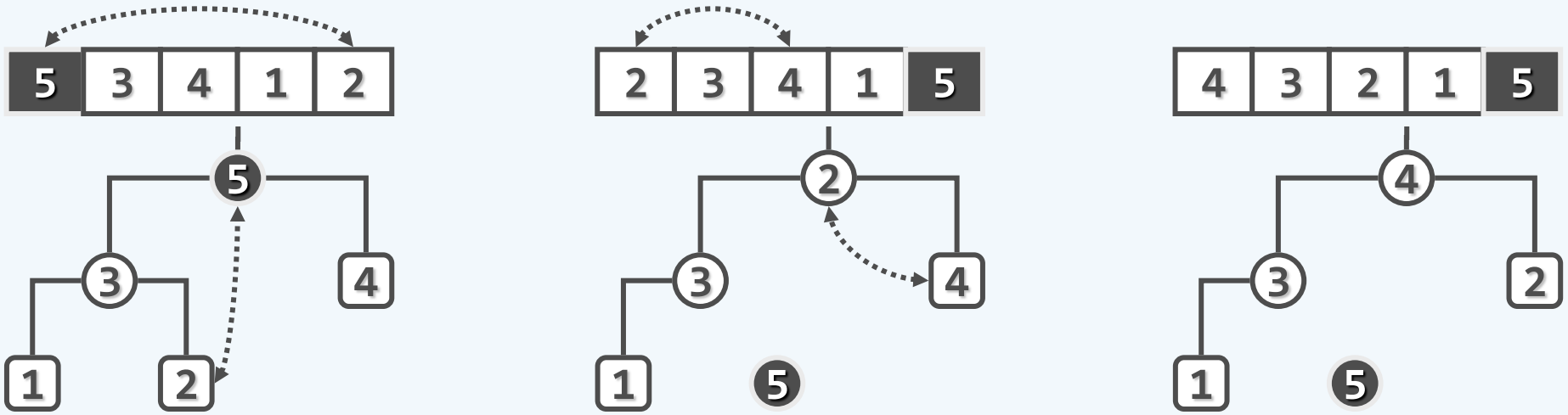


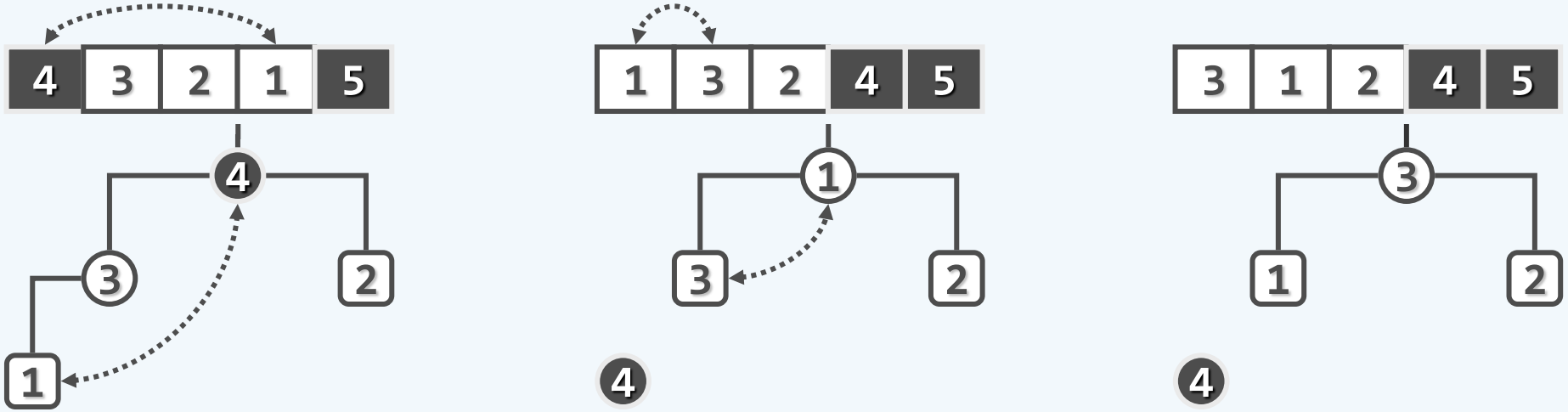




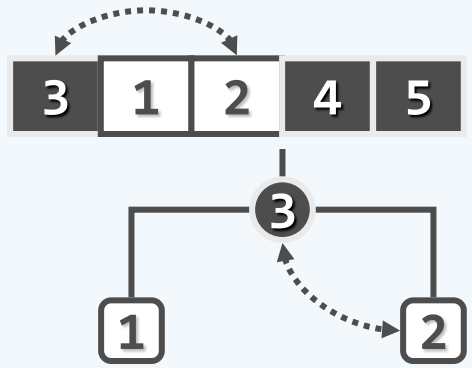


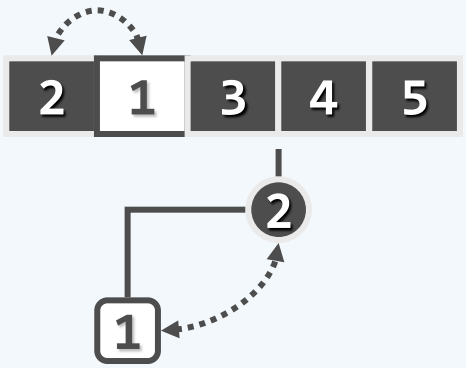
实例：选取+调整

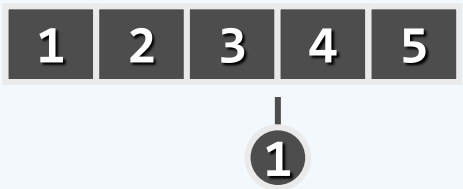

5 3 4 1 2
5
3
4
1
2
2 3 4 1 5
2
3
4
1
5
4 3 2 1 5
4
3
2
1
5

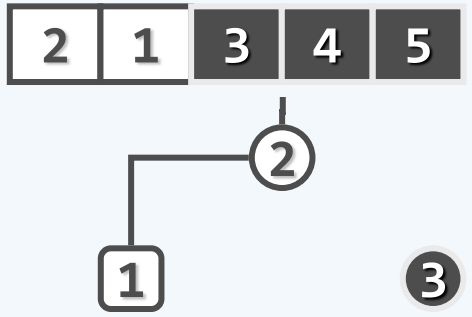

4 3 2 1 5
4
3
2
1
1 3 2 4 5
1
3
2
4
3 1 2 4 5
3
1
2
4

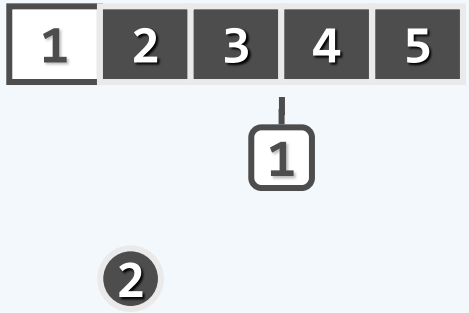
实例：选取+调整











综合评价

- 易于理解，便于实现 //完全基于二叉堆结构及其操作接口

  快速高效 //尤其适用于大规模数据

  可就地运转

  不需全排序即可找出前k个词条 //O(klogn)的selection算法

- 不稳定 //为什么？可否克服？

- 权衡： 采用就地策略，是否值得？

  固然可以节省一定的空间

  但对换操作因此须涉及两个完整的词条，操作的单位成本增加