12-C

优先级队列

堆排序

邓俊辉

deng@tsinghua.edu.cn

原理不变，还是逐个选取

❖ 在selectionSort()中

将U替换为H...

❖ J. Williams, 1964

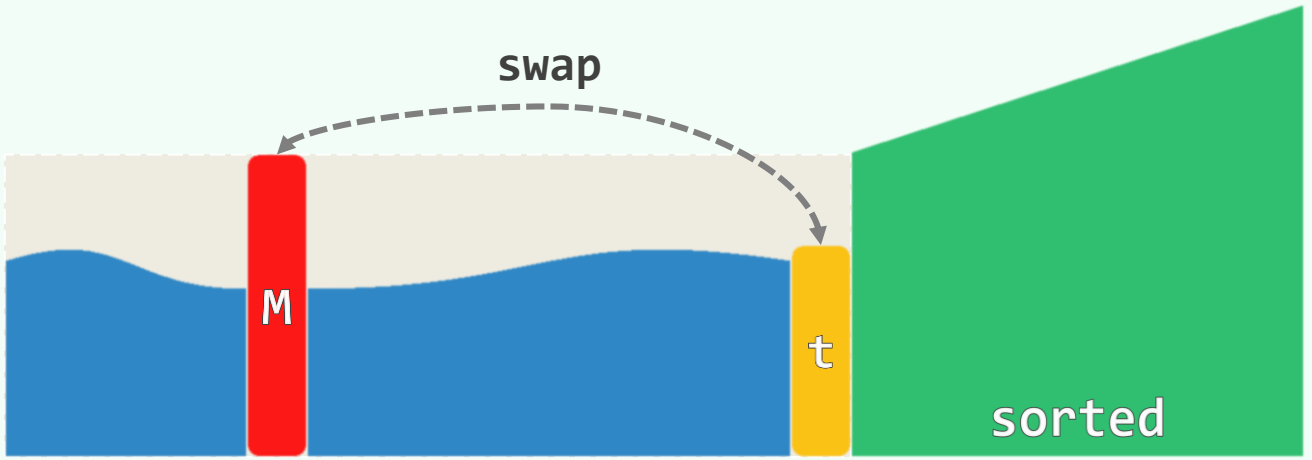
初始化 ： heapify(), $\mathcal{O}(n)$

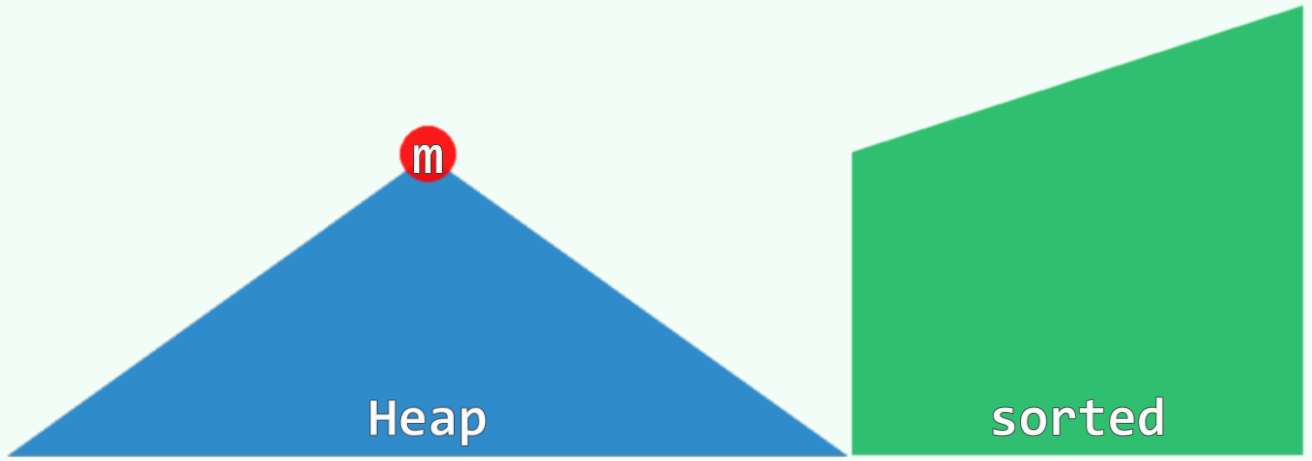
迭代 ： delMax(), $\mathcal{O}(\log n)$

不变性 ： H ≤ S

❖ $\mathcal{O}(n) + n \cdot \mathcal{O}(\log n)$

$= \mathcal{O}(n \cdot \log n)$

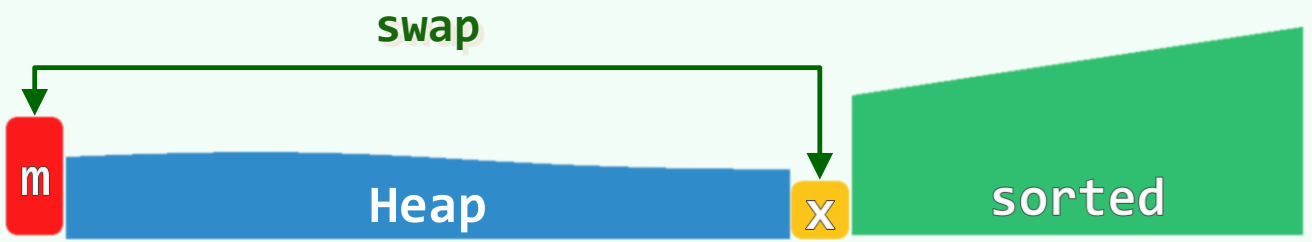




就地

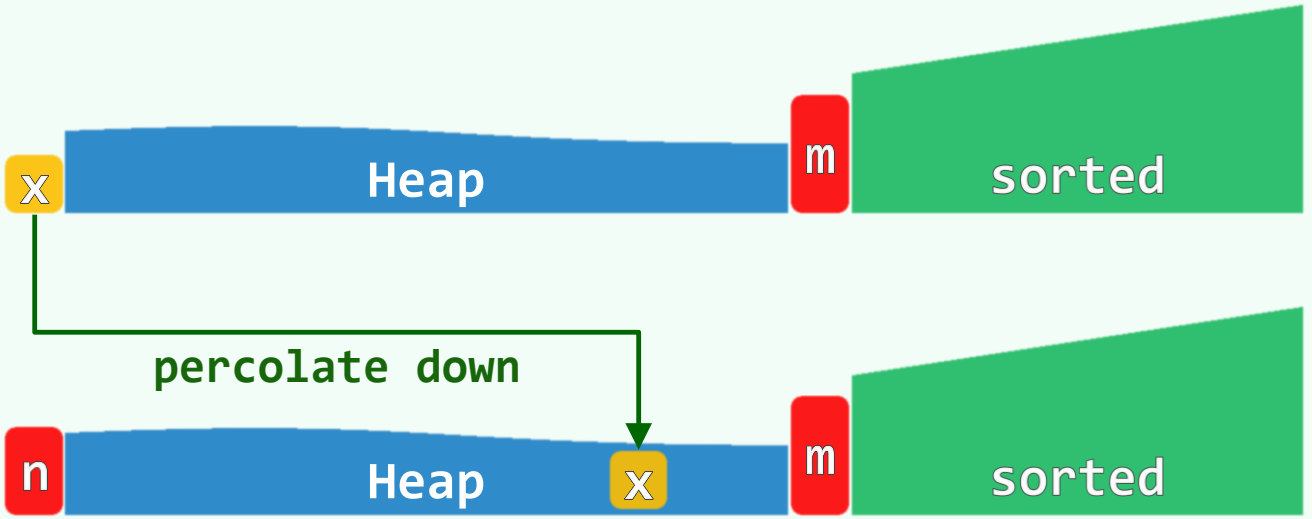
❖ 在物理上

完全二叉堆即是向量



❖ 既然此前有：

- m = H[0]
- x = H[n - 1]

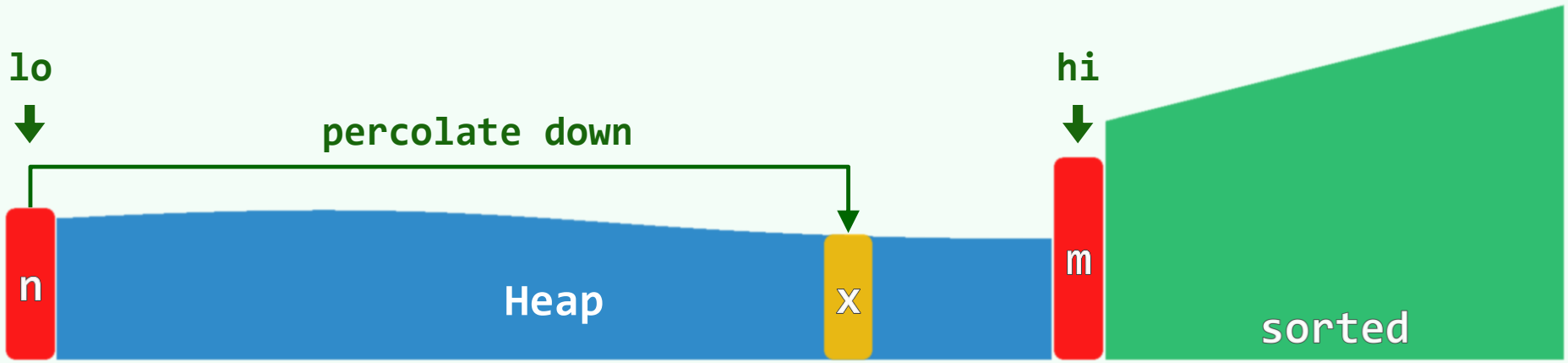


不妨随即就：

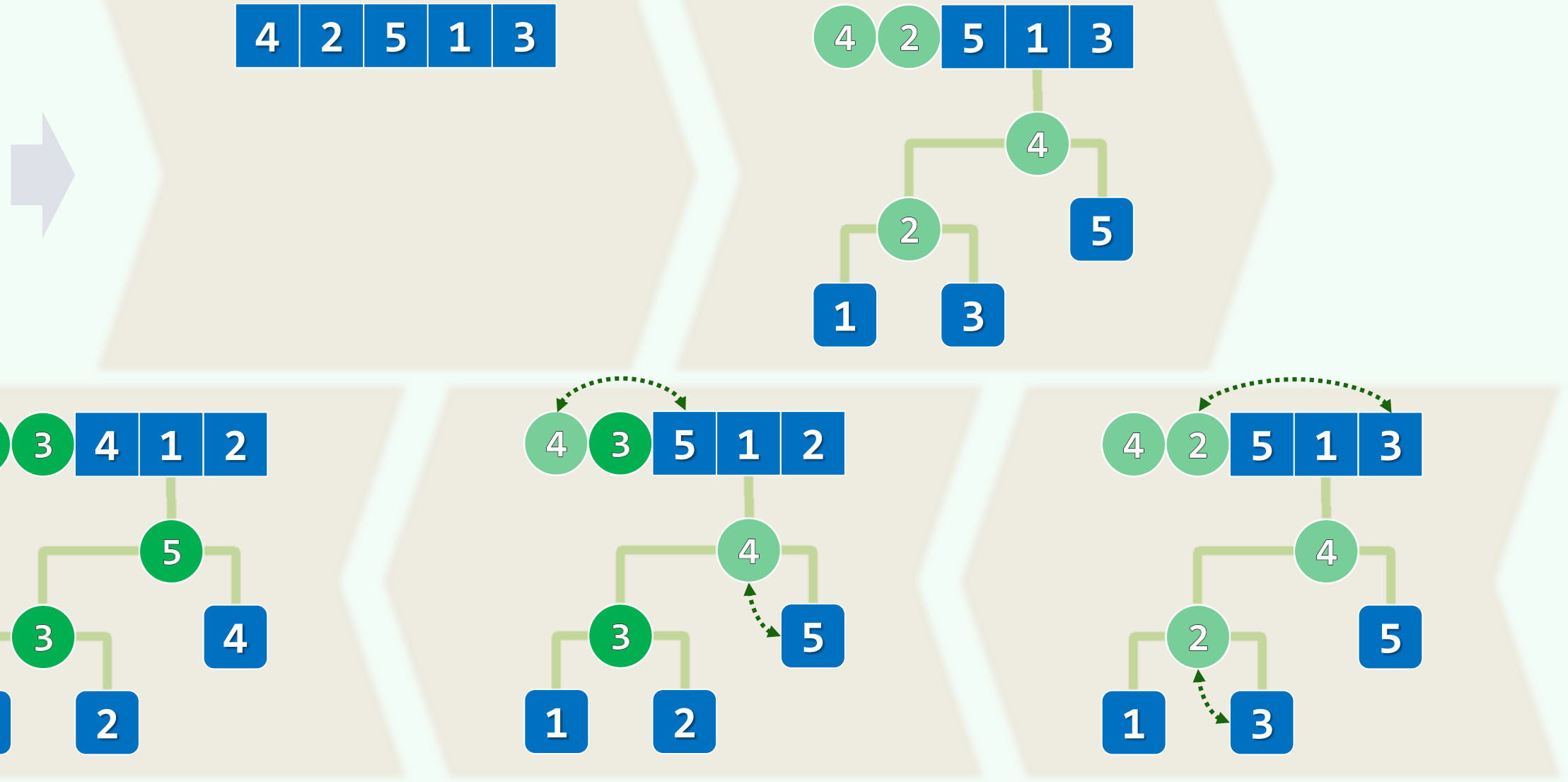
- swap(m , x) = H.insert(x) + S.insert(m)

实现

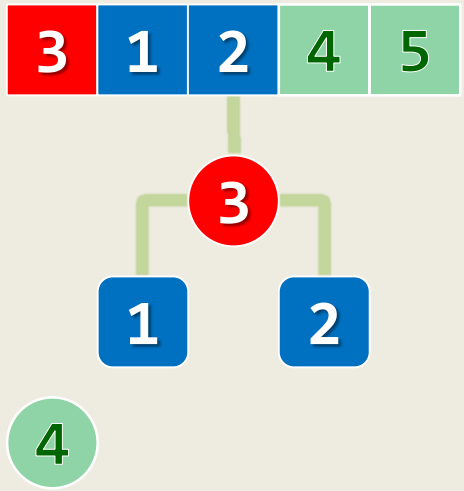
```
template <typename T> void Vector<T>::heapSort( Rank lo, Rank hi ) { //就地堆排序

   T* A = _elem + lo; Rank n = hi - lo; heapify( A , n ); //待排序区间建堆，O(n)

   while ( 0 < --n ) //反复地摘除最大元并归入已排序的后缀，直至堆空

      { swap( A[0], A[n] ); percolateDown( A, n, 0 ); } //堆顶与末元素对换后下滤

}
```

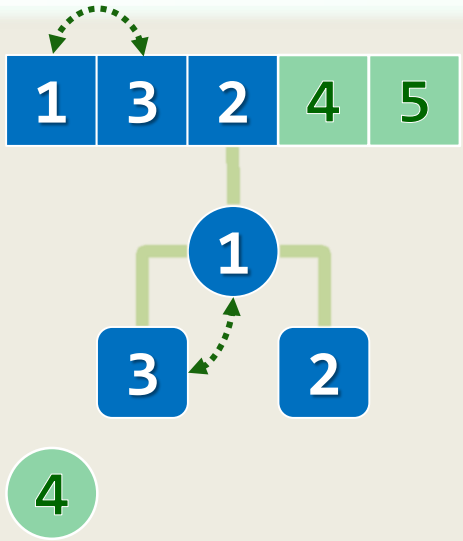


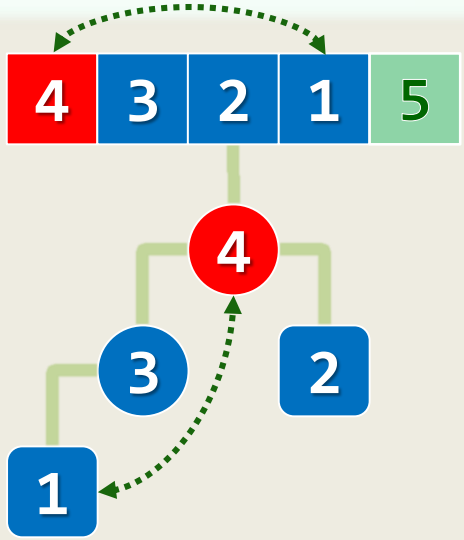
实例：建堆

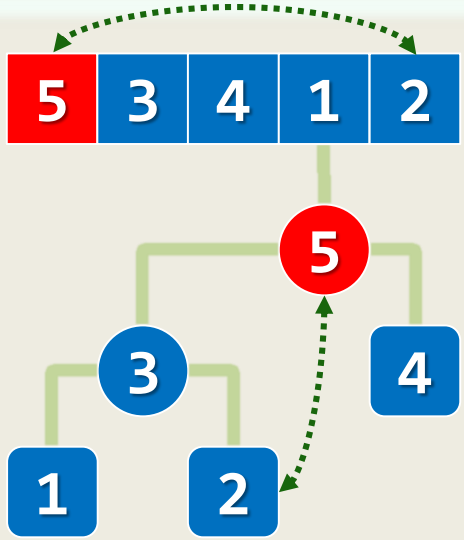


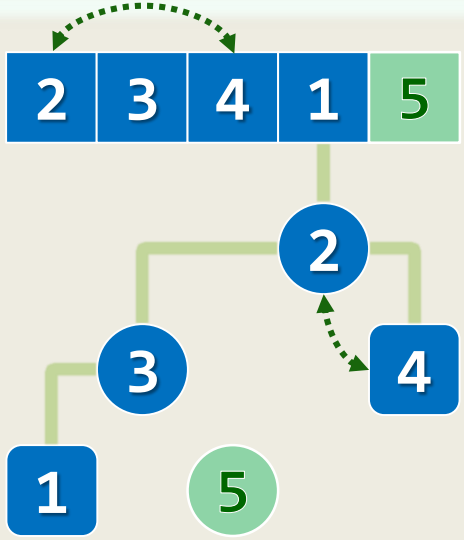
实例：选取 + 调整（1/2）

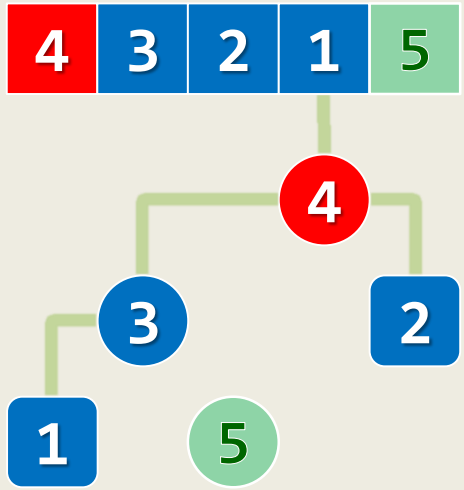

3 1 2 4 5
3
1 2
4


1 3 2 4 5
1
3 2
4


4 3 2 1 5
4
3 2
1


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

实例：选取 + 调整（2/2）