12B2 插入和上滤

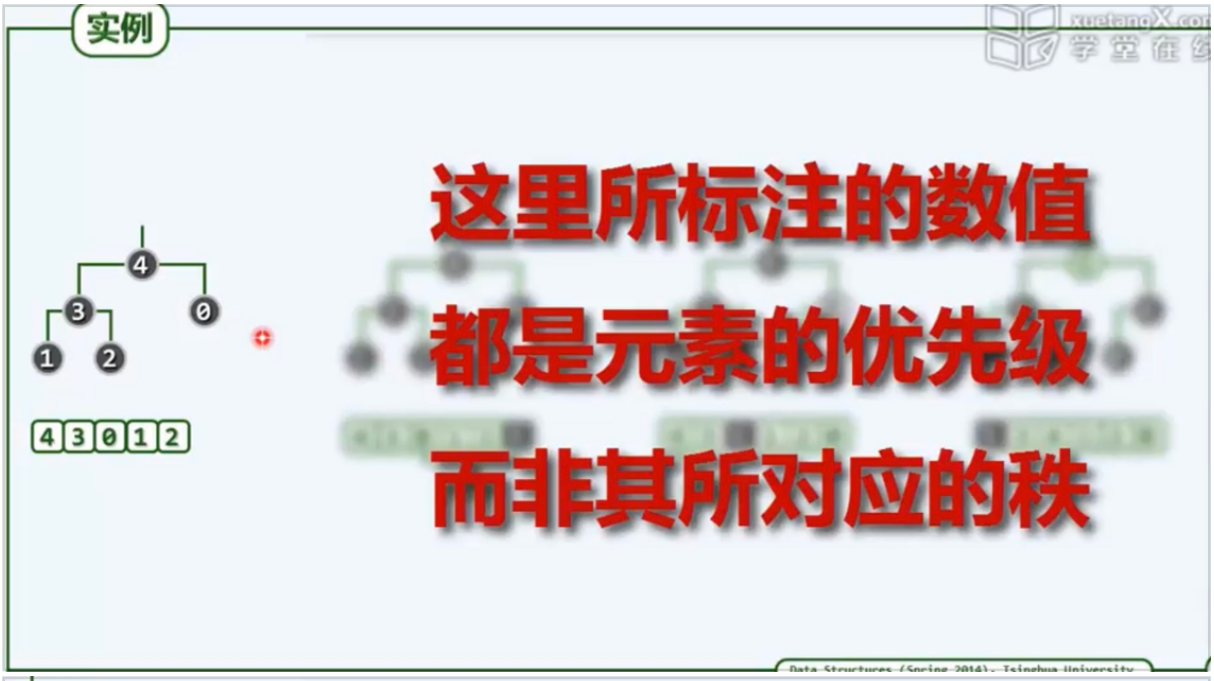
#数据结构邓神

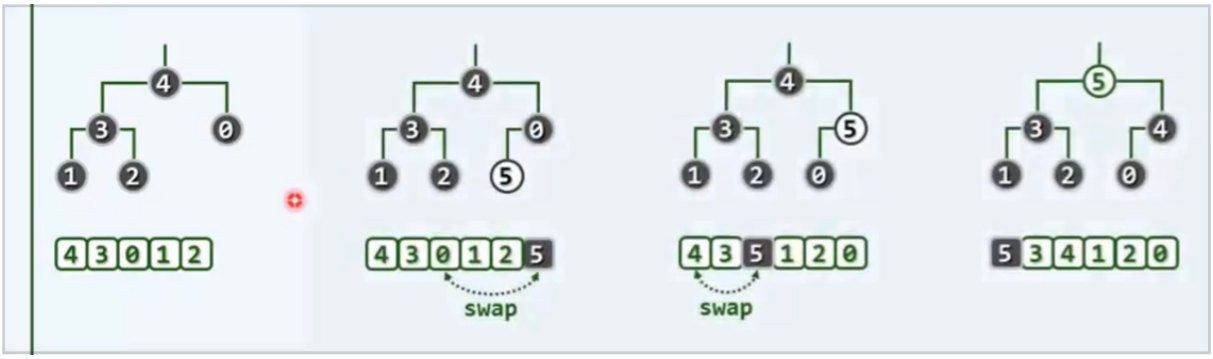
上滤

一旦堆序性被破坏了，就与其父亲节点交换，所以时间复杂度为O(logN)

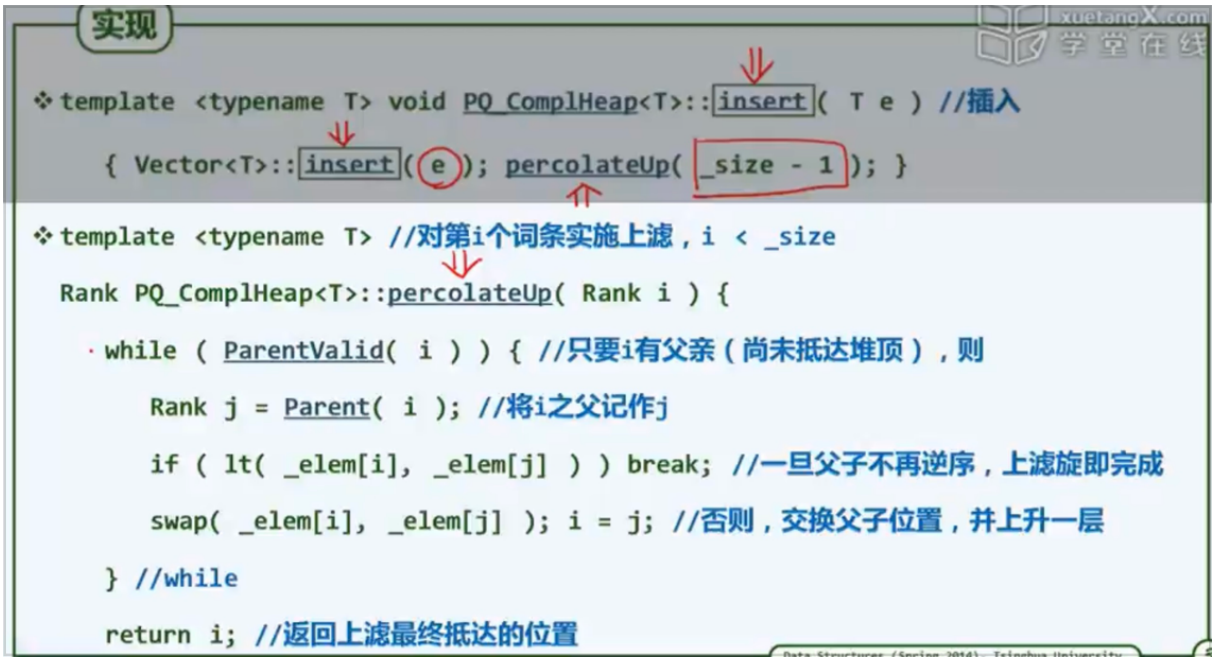
不一定一次就完全解决，但是也没有关系，最多也就是到ROOT节点

实例





实现




```
void insert(T e){
    _elem.push_back(e);
    percolateUp(_elem.size() - 1);
```

```
}
bool ParentVaild(const Rank& i){
    const Rank ROOT = 0;
    return ROOT != i;
}
Rank percolateUp(Rank i){
    while(ParentVaild(i)){
        Rank j = Parent(i);
        if(_elem[i] <= _elem[j]){
            break;
        }
        swap(_elem[i],_elem[j]);
        i = j;
    }
    return i;
}
```

效率

完全二叉树是理想平衡的二叉树
树高可以完美控制在 logn
所有的迭代可以完美控制在 logn
就渐进的意义而言 很好
但是就常系数的意义而言还是不行

但是 swap 函数需要三次交换操作
所以可能会多达 3logN

我们可以对新插入的节点做一个备份，先去判断是否上移动，直到无序上移动在赋值进去
3logN ~ logN+2

大小比较操作也可以改进