数据结构与算法

Data Structure and Algorithm

XVI. 堆

授课人：Kevin Feng

翻译　：孙 兴

课前回顾

- 数学回顾
- 数组（Array）和数组列表（Array List）
- 递归 vs. 迭代
- 二分法搜索
- 分治法
- 链表
- 栈和队列
- 哈希表
- 树

Overview

概要

- 堆
  - 数据成员（Data Member）
  - 操作（Operations）
- 练习
  - 第K个最大/最小值（Min / Max Kth）
  - 数据流的中位数(Median in Stream)

集合回顾

- 集合是存储数据项的数据类型

data type	key operations	data structure
stack	PUSH, POP	*linked list, resizing array*
queue	ENQUEUE, DEQUEUE	*linked list, resizing array*
priority queue	INSERT, DELETE-MAX	*binary heap*
symbol table	PUT, GET, DELETE	*BST, hash table*
set	ADD, CONTAINS, DELETE	*BST, hash table*

堆

- 集合（Collections）　　插入删除项；删除哪一项？

- 数组列表（ Array List ）　　通过给定索引删除项
- 链表（ Linked List ）　　通过迭代列表删除项
- 栈（ Stack ）　　只能删除最近添加的项
- 队列（ Queue ）　　删除最开始添加的项
- 哈希表（ Hash Table ）　　根据给定关键值删除项

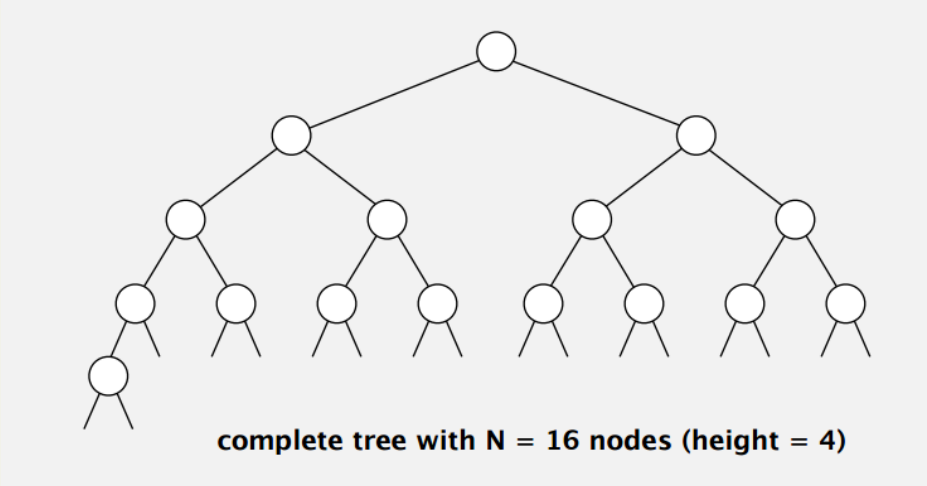
- 堆（ Heap, or Priority queue ）　　删除最大/最小项

堆的应用

- 事件驱动模拟 [Customers in a line, who’s next]
- 图搜索 [Dijkstra's algorithm, Prim's algorithm]
- 操作系统 [load balancing, interrupt handling]
- 计算机网络 [web cache]
- 任何需要得到最小值/最大值/优先级的

完全二叉树

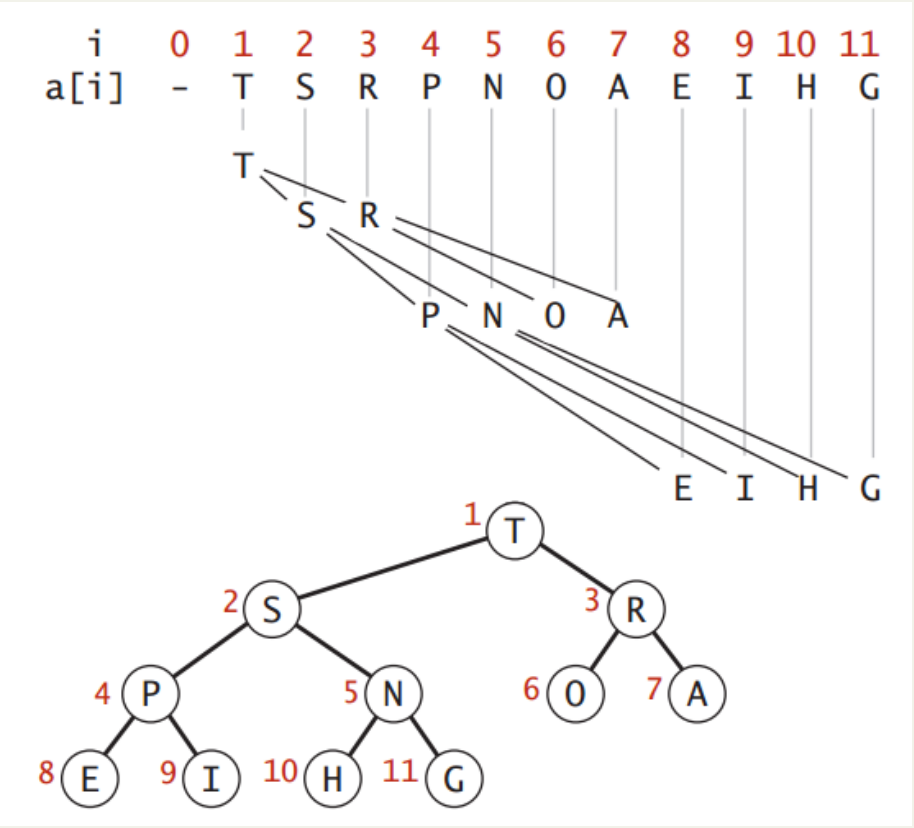
- 二叉树（ Binary tree ）二叉树的左右侧连接为空或者节点
- 完全树（ Complete tree ） 除了最底层，完全对称平衡



- 特性——有N个节点的完全树的层数是
- Pf.只有当N是2的指数是层数才会增加。 Pf. Height increases only when N is a power of 2

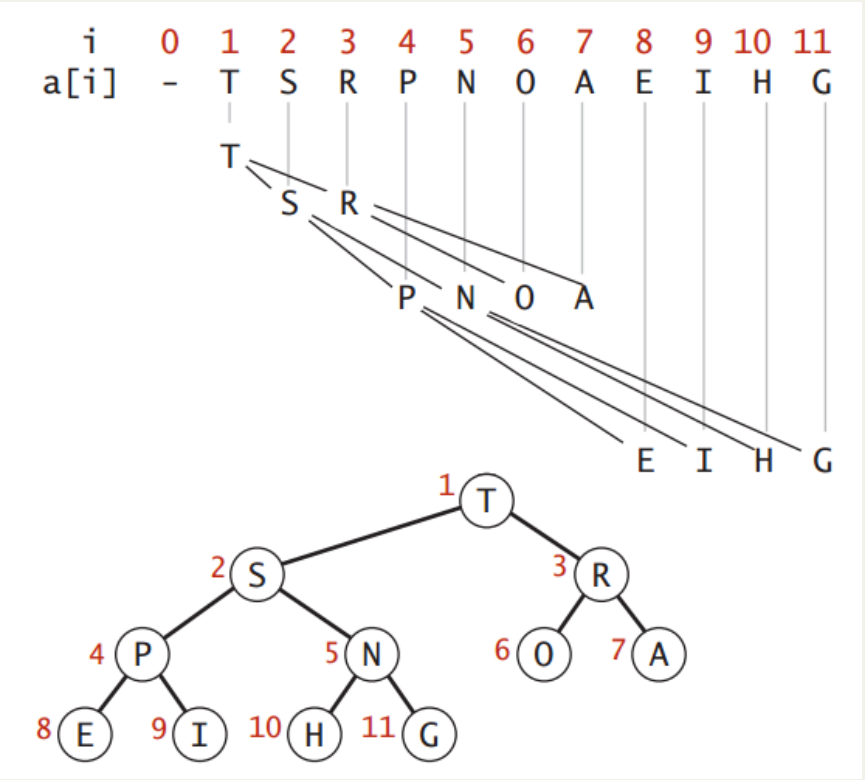
堆的表示

- 一个有序堆完全二叉树的数组表示
- 有序堆二叉树
  - 节点的键值
  - 父节点的键值不小于子节点的键值



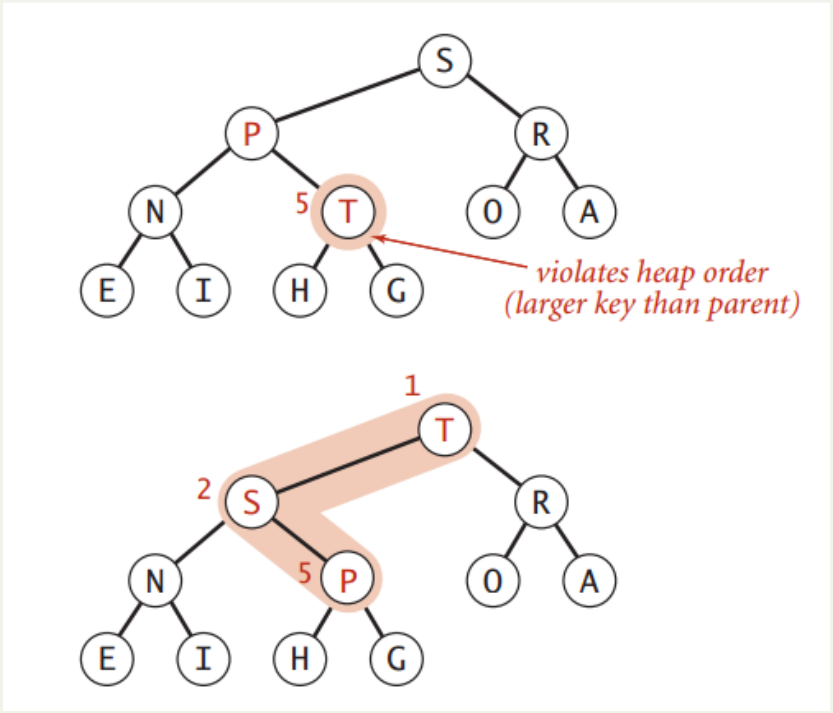
堆的特性

- Proposition: 最大键值是a[1]，同时也是二叉树的根节点
- Proposition:可以使用数组索引做移动
  - 在位置k处节点的父节点的位置是k/2
  - 在位置k处节点的子节点的位置是2k和2k+1

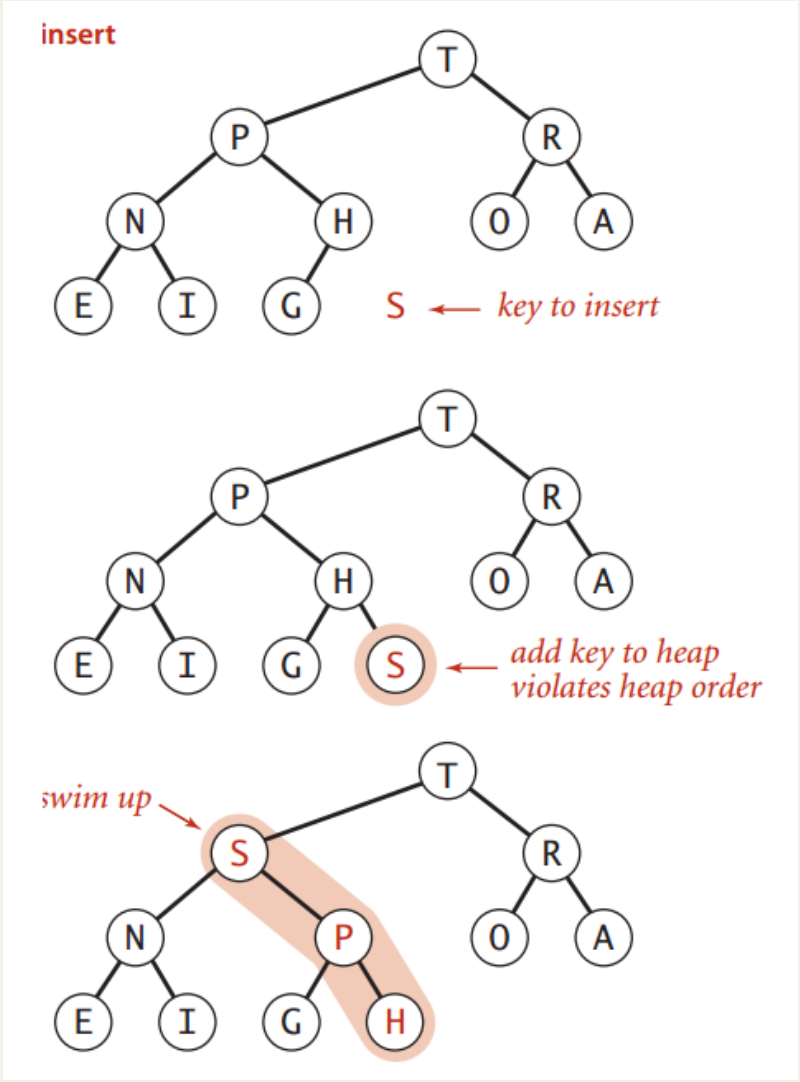


上浮（Promotion）

- 情境：子节点的键值变为比父节点的键值大
- 消除这种违反项：
  - 交换子节点的键和父节点的键
  - 重复这个过程直到堆的顺序恢复正常Repeat until heap order restored



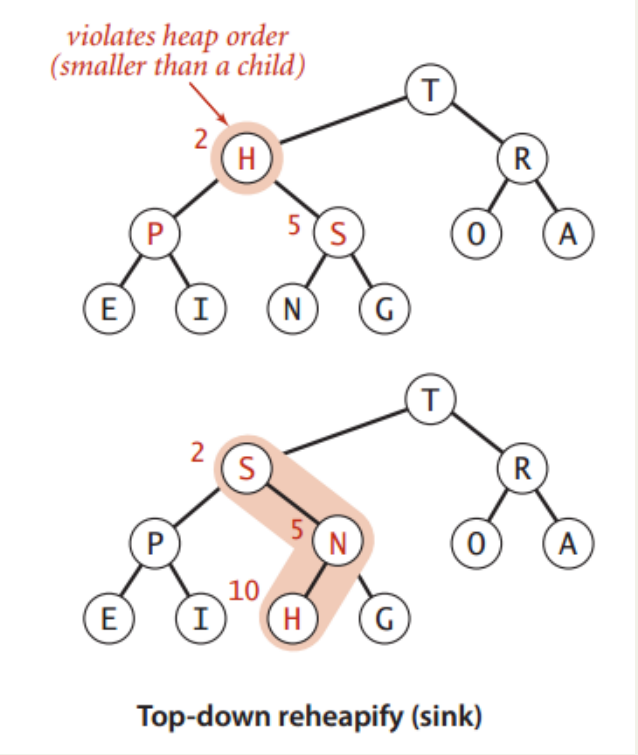
堆的添加




```
def _upheap(self, j):
    parent = self._parent(j)
    if j > 0 and self._data[j] < self._data[parent]:
        self._swap(j, parent)
        self._upheap(parent)
```

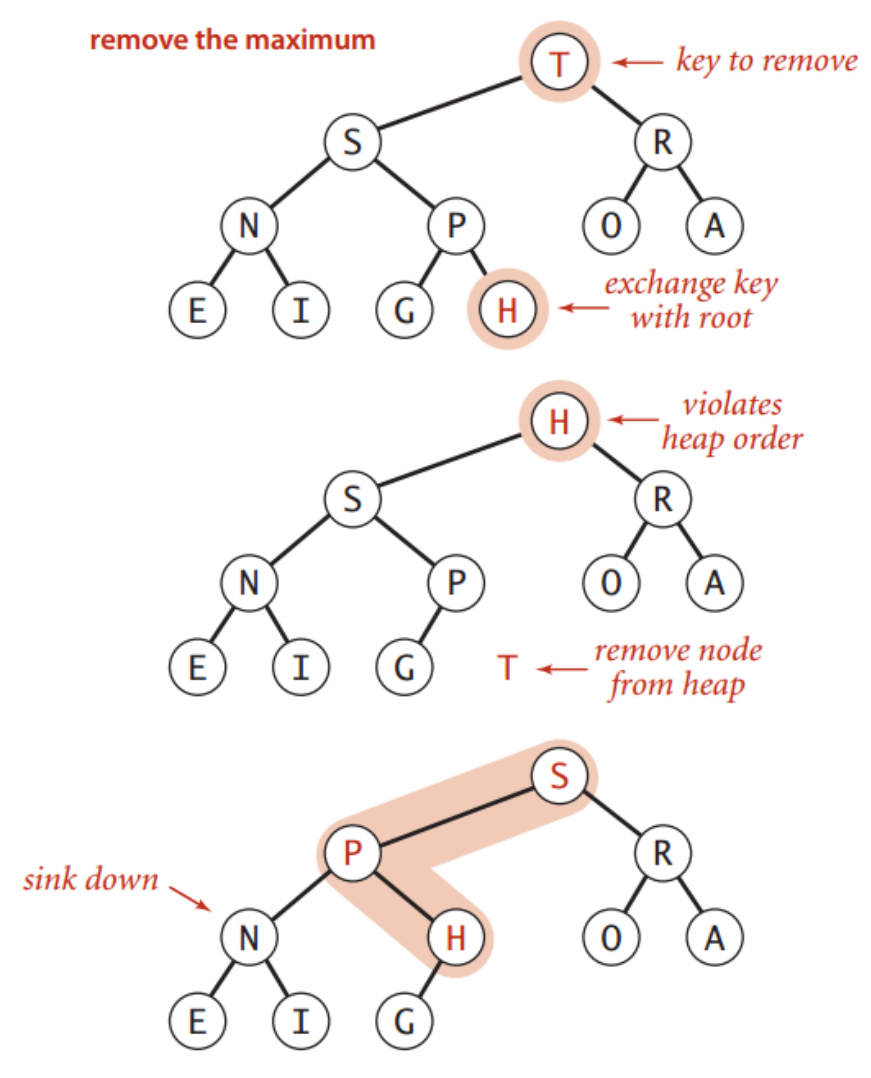
下沉（Demotion）

- 情境：父节点的键值变得比子节点（一个或者2个）的键值还小
- 消除这种违反项：
  - 把父节点的键值和比它大的子节点的键值做交换；
  - 重复这个操作直到堆的顺序恢复正常Repeat until heap order restored



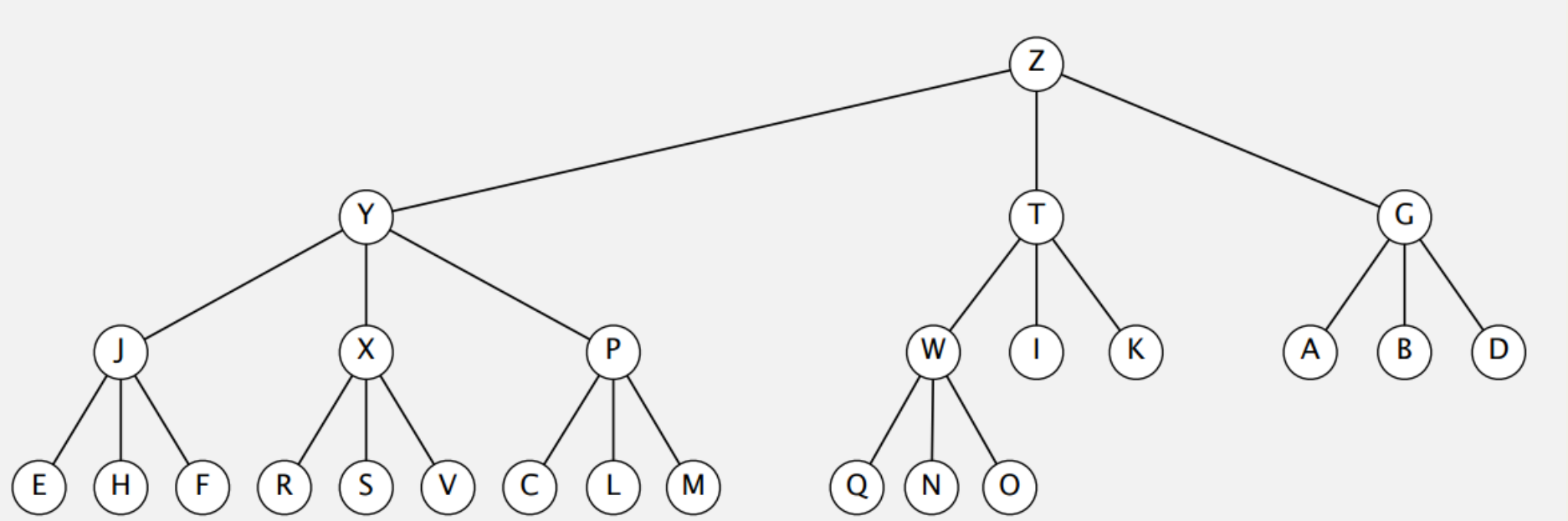
Top-down reheapify (sink)

堆-删除最大值




```
def _downheap(self, j):
    if self._has_left(j):
        left = self._left(j)
        small_child = left
        if self._has_right(j):
            right = self._right(j)
            if self._data[right] < self._data[left]:
                small_child = right
        if self._data[small_child] < self._data[j]:
            self._swap(j, small_child)
            self._downheap(small_child)
```

D-ary 堆



3-way heap

复杂度（cost summary）

Implementation	Insert	Del Max	Max
Unordered Array	*1*	*N*	*N*
Sort	*N*	*1*	*1*
Binary Heap	*log N*	*log N*	*1*
D-ary Heap	$\log_d N$	$d \log_d N$	*1*

更多考虑项

- 下溢和上溢
  - 下溢：如果从空堆中删除，会得到异常
  - 上溢：调整数组大小
- 最小堆
- 更多操作
  - 删除任意项
  - 改变其中一项的优先级
  - 可以有效地实现交换

练习一

- 数组中第k个最大的元素
  - 在一个没有排序的数组中找到第k个最大元素。
- 前K个最高频的词汇
  - 给定一个非空的单词列表，返回k个最高频的词。
  - 你给出的结果应该按照高频到低频的顺序给出。如果两个词有相同的频率，则按照字母表顺序，先给出字母表排序靠前的。

练习二

- 丑数(Ugly Number)
  - 丑数是质因数只包含2, 3, 5的正整数。例如，1, 2, 3, 4, 5, 6, 8, 9, 10, 12是前10个丑数。

- 找到加和值最小的k对数
  - 给你两个整数数组，分别是nums1和nums2和一个整数k。这两个数组已按照升序排序。
  - 定义一个数对（u, v），这个数中的两个元素分别取自上面两个数组。
  - 找到k个数对(u1, v1), (u2, v2) . . . (uk, vk) ，满足他们的加和值最小。

练习三

- 合并k个有序列表
  - 合并k个有序链表，返回合并后的一个列表。分析并描述它的复杂度。
- 从数据流中找到中位数
  - 中位数是有序整数数列中的中间的值。如果数组的大小是偶数，则不存在中位数。所以中位数是两个中间的数据的均值。
  - 例如：
    - [2,3,4]，中位数是 3
    - [2,3]，中位数是 (2 + 3) / 2 = 2.5
  - 设计一个可以支持以下两种操作的数据结构
    - void addNum(int num) - 从数据流中添加一个整数到这个数据结构中
    - double findMedian() - 返回目前已有的所有元素的中位数

练习四

- 管理你的项目（IPO）
  - 给定你几个项目。对于每个项目i，需要有一个净利润Pi和一个最低资金Ci才能启动这个项目。最开始时，你有总资金W。当你完成一个项目后，你会得到这个项目对应的净利润，这笔净利润会加入你的总资金。
  - 总之，从给定的项目中最多选取k个项目使得你最后的总资金最多，并且输出你最后的资金值。
  - 输入：k=2，W=0，利润=[1,2,3]，资金=[0,1,1].
  - 输出：4
  - 补充说明：因为你的初始资金为0，所以只能选取项目0作为开始项目。当完成这个项目后，你会得到利润值1，相应的你的资金会变成1.有了资金1，你既可以做项目1，也可以做项目2. 因为最多只能选取2个项目，所以要选项目2来获得更多的资金。因此，最后输出的最大资金为0+1+3=4.

回顾

- 堆

数据结构与算法

XVI. 堆

结束