**大作业报告**

姓名： 朱天骋 学号： 20123915

日期： 2022/5/23

1. **题干**

**题目描述：**

假设上海大学通信学院大数据专业每届招收保研学生k人，目前正在陆续提交考核分数（分数介于0-100之间，且无序输入）。现需要设计一个程序，随着每输入一个学生分数，实时显示当前的录取分数线。请设计一个Python程序，实现该功能。

其中k的取值>=1，学生数量没有限制

程序首先需要启动参数获取k值，确定录取人数。然后依次循环输入分数，并实时输出分数线。

**示例：**

score.py 2 #2代表k

>> 100

>> 当前分数线100

>> 93

>> 当前分数线93

>> 98

>> 当前分数线98

>> 35

>> 当前分数线98

**测试用例描述：**

k=10

98,100,32,78,88,19,100,100,88,77,22,67,89,42,95,87

请给出10组数据的输入和输出结果

1. **解答**

**2.1 解题思路**

【请以文字及流程图的方式，描述对题目的解题思路】

程序首先输入一个k，然后建立一个可以放k个数据项的容器。在前k个输入的数据项，我们需要将其按大小存入容器，并返回容器中最小的数据项，而从k+1个输入数据项开始，我们需要将其与容器中的最小值进行比较，若输入的数据项较大，则将容器最小项去除，添加该输入数据项，然后返回容器新的最小项。题目中没有说明如何去终止程序，故我选择在输入负数时停止程序。

值得注意的是，由于我们每输入一个数就可能需要进行一次插入和删除元素，且每次将其插入一个较为有序的容器，那么我偏向于使用完全二叉堆实现或二叉树实现。普通二叉搜索树，每次删除最左边叶子节点，并在随机处生成新的叶子节点，在经过多次数据的插入删除后，原先的根节点的左子树会全部被删除，树会逐渐退化为斜树，而平衡二叉树在判断是否平衡时也需要很高的开销，且编写复杂，故最后选择了完全二叉堆实现。

为了方便实际操作需求，我为二叉堆添加了一个去根节点添加新结点的函数，具体实现为：根节点的值换为新的值，并将其与左右子结点的较小值进行比较，若根节点值小则不动；若根节点值较大则与左右子结点中较小的节点进行交换，并向下继续循环比较直到其变为叶子节点或值小于两个子结点值。

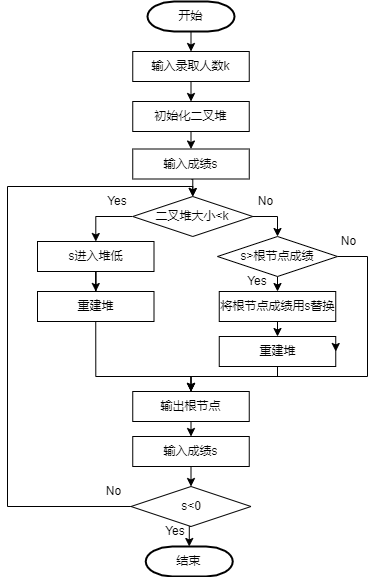


图1 程序流程图

**2.2 算法性能**

【对自己实现的算法进行算法性能分析，分析其复杂度】

本程序中，前k个数需要构建一个k个结点的完全二叉堆，对于每次输入操作，首先将新结点插入末位，时间复杂度为O（log2k），而交换的时间复杂度也为O（log2k）。从第k+1个数起每输入一个新的成绩先与根节点的头结点数值进行比较，如果该成绩大于最低成绩，则需要将头结点的值替换为新的成绩，其时间复杂度为O（1），再将其从上至下下沉交换，时间复杂度是O（log2k），故每次操作的时间复杂度为O（log2k），若执行n次则为O（nlog2k）.

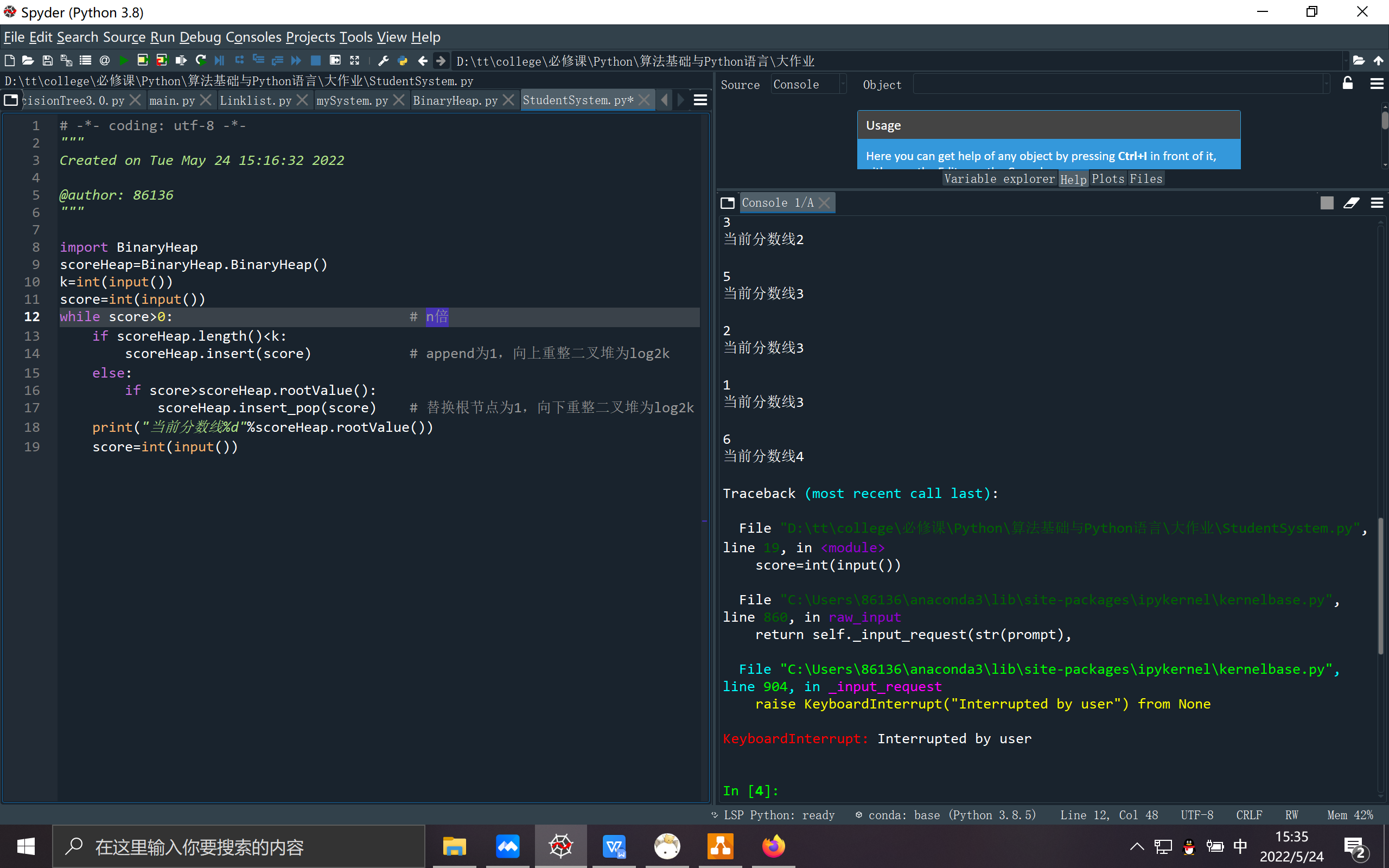
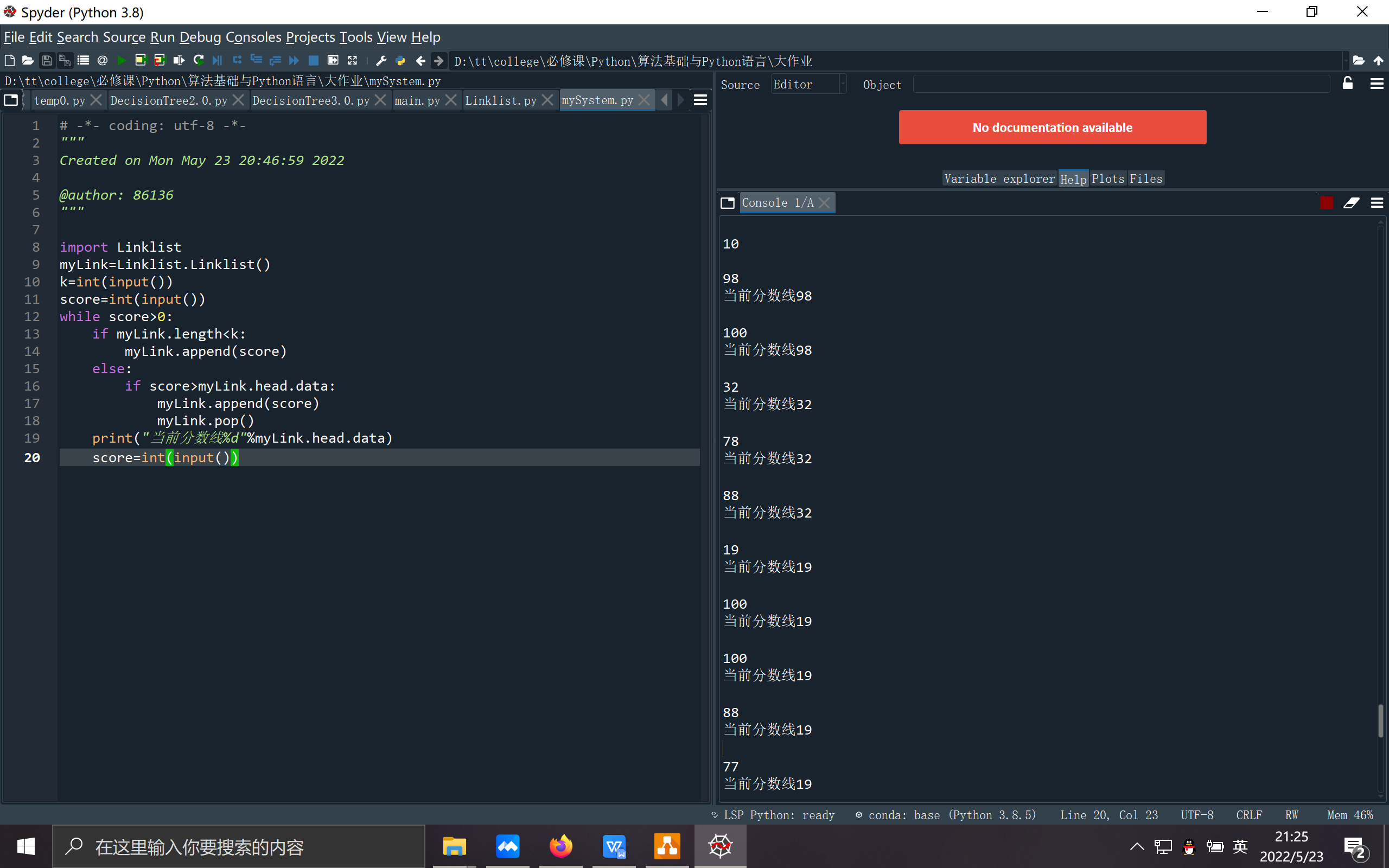


图2 时间复杂度分析图

**2.3 解题答案**

【将程序的运行命令行，包含参数。程序运行的结果进行记录，并保留实际运行的截图】

在首先输入10后，依次输入98，100，32，78，88，19，100，100，88，77，22，67，89，42，95，87后效果如下图所示：



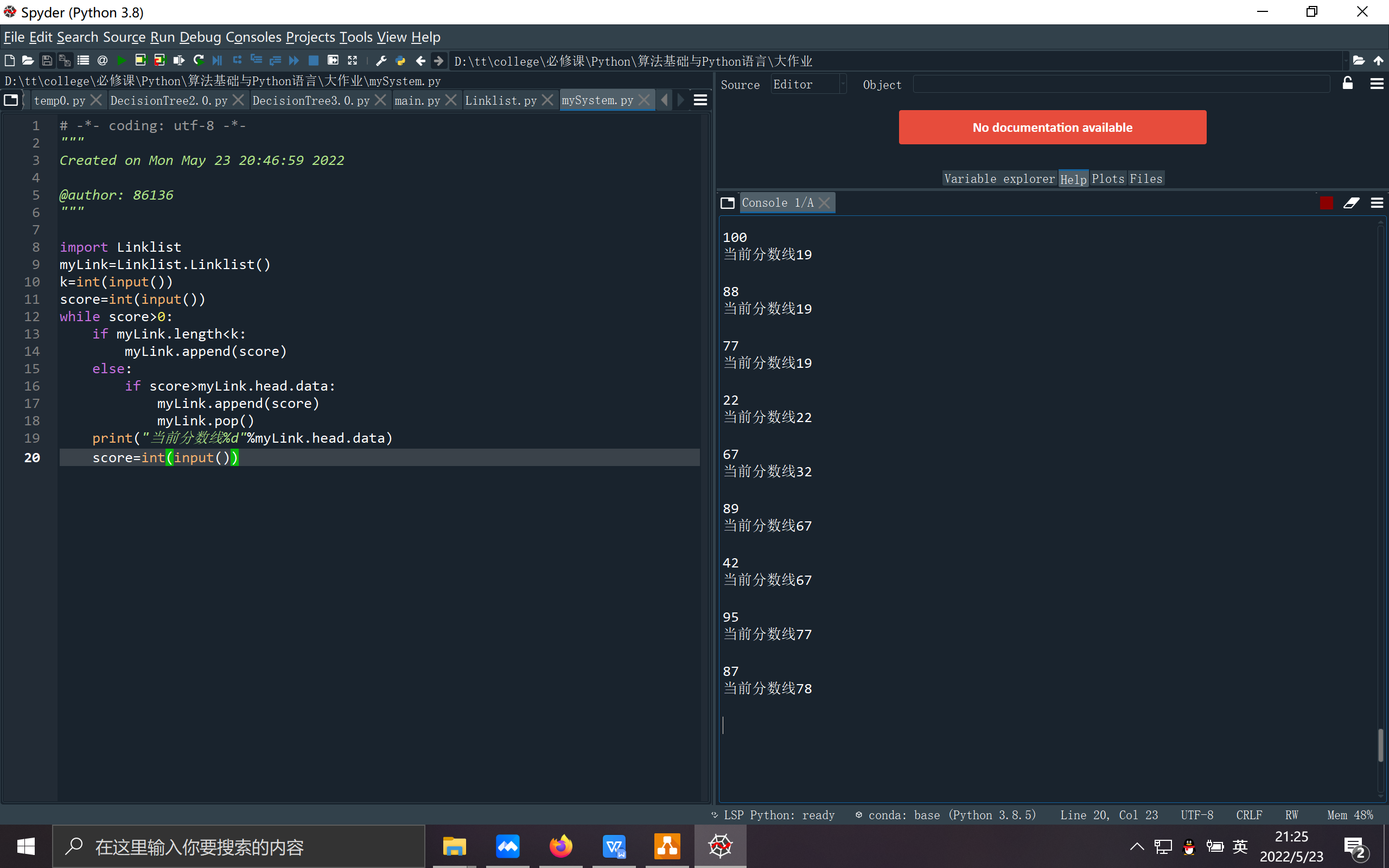


图3 结果展示图

**2.4 程序源码**

【如果涉及自定义数据结构，请在文档中包含，不同文件需要包含在不同源码段内。同时提交报告时也请附加对应源码包】

程序源码A文件名BinaryHeap.py

源码详细

class BinaryHeap:

def \_\_init\_\_(self):

self.heapList=[0]

self.size=0

def length(self):

return self.size

def rootValue(self):

if len(self.heapList) ==1:

print("二叉堆没有元素")

return

else:

return self.heapList[1]

def show(self):

print(self.heapList)

def insert(self,newkey):

self.heapList.append(newkey)

self.size+=1

self.percUp(self.size)

def percUp(self,i):

while i//2>0:

if self.heapList[i]<self.heapList[i//2]:

self.heapList[i],self.heapList[i//2]=self.heapList[i//2],

self.heapList[i]

i=i//2

def insert\_pop(self,newkey):

self.heapList[1]=newkey

self.rootDown()

def rootDown(self):

i=1

while i<=self.size/2:

if i==self.size/2:#正好位于只有一个子节点的节点处

if self.heapList[i]>self.heapList[2\*i]:

self.heapList[i],self.heapList[2\*i]=self.heapList[2\*i],self.heapList[i]

break

if self.heapList[i]>min(self.heapList[2\*i],self.heapList[2\*i+1]):

if self.heapList[2\*i]>self.heapList[2\*i+1]:

self.heapList[i],self.heapList[2\*i+1]=self.heapList[2\*i+1],self.heapList[i]

i=2\*i+1

else:

self.heapList[i],self.heapList[2\*i]=self.heapList[2\*i],self.heapList[i]

i=2\*i

程序源码B文件名 StudentSystem.py

源码详细

import BinaryHeap

scoreHeap=BinaryHeap.BinaryHeap()

k=int(input())

score=int(input())

while score>0:

if scoreHeap.length()<k:

scoreHeap.insert(score)

else:

if score>scoreHeap.rootValue():

scoreHeap.insert\_pop(score)

print("当前分数线%d"%scoreHeap.rootValue())

score=int(input())