暨南大学本科实验报告专用纸

课程名称 算法分析与设计实验 成绩评定

实验项目名称 众数问题 指导教师 李展

实验项目编号 实验三 实验项目类型 综合性 实验地点

学生姓名 张印祺 学号 2018051948

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2020 年 3 月 25 日下午 ~ 3 月 25 日下午温度 ℃湿度

一、问题描述

给定含有 n 个元素的多重集合 S，每个元素在 S 中出现的次数称为该元素的重数。多重集 S 中重数的最大数称为众数。给定一个多重集合 S，要求找出其中的众数以及其对应的重数。

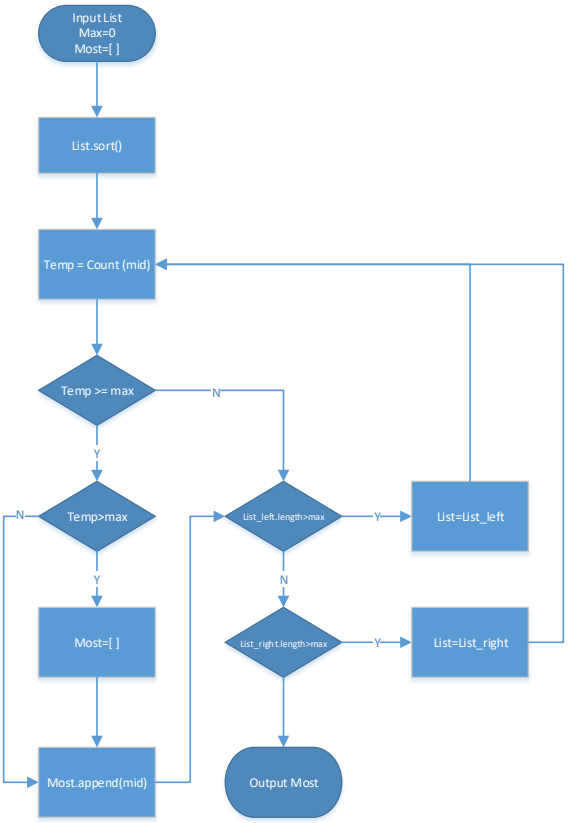
二、算法思路

1、 计算出我们的所选列表的中位数以及其长度将其存入临时变量 _max

2、 使用分治法，若左侧长度大于_max 则在左侧继续选择中位数计算其长度，右侧同样的操作

3、 放弃长度小于 max 的未选序列可以大幅度降低时间复杂度。

三、算法流程



四、测试结果（左列为输入，右为输出）

```
[10, 10, 11, 12, 12, 12, 13, 14, 14, 14, 14, 14, 15, 17, 17, 18, 18, 18, 18, 19, 19, 19, 21, 22, 2
2, 22, 23, 23, 23, 23, 23, 23, 24, 24, 25, 25, 26, 27, 27, 27, 28, 28, 28, 29, 29, 29, 30, 31, 31,
 31, 31, 31, 31, 32, 32, 33, 33, 33, 33, 34, 34, 34, 34, 35, 37, 37, 37, 38, 38, 38, 39, 40, 40, 4
0, 40, 40, 41, 41, 42, 42, 43, 43, 43, 44, 44, 44, 45, 45, 46, 46, 46, 46, 49, 49, 50, 50, 50, 50,
 50]
重数，众数 ([6], [31, 23])
```

五、实验总结

本方法运用到了分治法，将元素进行分块至左右两侧列表大小均小于当前最大数为止，可以大幅度降低平均时间复杂度。

本方法的递推方程式为： $f(n) = f\left(\frac{n}{2}\right) + 1 \rightarrow T(n) = O(n)$，由于之前我们还进行了一次排序算法，所以经过计算封装后的类中的时间复杂度为 $O(nlogn)$。

但是本方法运用了分治的原理，他的最差时间复杂度并不低，如果我们采用 hash 表的方法可以达到平均时间复杂度与最差时间复杂度均为 O(n)。其算法过程如下：

1. 新建一个 dict={}
2. 遍历 list 将 list 的值作为 key，将出现次数作为 dict[key]的值
3. 遍历 dict[key]输出最大数的 key

Hash 表方法代码如下：

```
import random as Rd
PATH = ".\\"
class ModeNum:
    def Creatdata():
        f = open(PATH + "3.ModeNum", 'w', encoding='UTF-8')
        for i in range(99):
            f.write(str(Rd.randint(10,50)) + '\n')
        f.close()

    def Select(dict):
        f1 = open(PATH + "3.ModeNum", 'r', encoding='UTF-8')
        list = f1.readlines()
        for i in range(len(list)):
            if list[i] in dict:
                dict[list[i]] += 1
            else:
                dict[list[i]] = 1

    def Search(dict, result, _max):
        _max = str(max(dict.keys(), key=(lambda x:dict[x])))
        result = ""
        for i in dict:
            if dict[i] == dict[_max]:
```

```
                    result += (str(eval(i)) + " ")
                print("众数：" + result + "重数：" + str(dict[_max]) + "\n")
            def main(self, dict, result, _max):
                ModeNum.Creatdata()
                ModeNum.Select(dict)
                ModeNum.Search(dict, result, _max)

        num = ModeNum()
        num.dict = {}
        num.result = ""
        num._max=""
        num.main(num.dict, num.result, num._max)
```

六、源代码

```
import os
PATH = ".\\"
class ModeNum:
    def Mode(self, most ,_max, a:list, l:int, r:int):
        a.sort()
        Max=_max[0]
        Mosy=most[0]
        mid = a[(l+r)//2]
        end_mid = pr_mid = a.index(mid)
        for i in range(pr_mid, len(a)):      #记录最后结束位
            if a[i] == mid:
                end_mid+=1
            else:break
        if Max == end_mid - pr_mid:
            most.append(mid)
        if Max < end_mid - pr_mid:
            most = [-1]
            Max = end_mid - pr_mid
            Most = mid
            _max[0]=Max
            most[0]=Most
        if pr_mid - l > Max:
            self.Mode( most ,_max, a, l, pr_mid-1)
        if r - end_mid > Max:
            self.Mode(most ,_max,a, end_mid, r, )
        return _max,most

f = open(PATH + "3.ModeNum", 'w', encoding='UTF-8')
for i in range(99):
```

```
    f.write(str(Rd.randint(10,50)) + '\n')
f.close()

f1 = open(PATH + "3.ModeNum", 'r', encoding='UTF-8')
ls=[]
for line in f1.readlines():
    line = eval(line.strip('\n'))
    ls.append(line)
a=ModeNum()
_max = [-1]
most = [-1]
print(ls)
print("重数，众数",a.Mode(most,_max,ls,0,len(ls)))
```