# 2-1 众数问题

## 问题描述

对于给定的由n个自然数组成的多重集S，计算S的众数及其重数。

**输入数据由文件名为input.txt的文本文件提供**。文件的第1行为多重集S中元素个数n；在接下来的n行中,每行有一个自然数。

**将计算结果输出到文件 output.txt。**输出文件有2行，第1行是众数，第2行是重数。

## 算法实现

### 算法一：利用unordered\_map容器

#### 思想

利用对应键位分别存储数字和出现次数，最后找到最大值就好

#### 代码（略）

#include<iostream>

#include<fstream>

#include<string>

#include<unordered\_map>

using namespace std;

int main()

{

unordered\_map <int, int> hashmap;

fstream fin;

fin.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-1\\Debug\\input.txt", ios::in);

if (!fin)

{

cout << "can't open it!" << endl;

return 0;

}

string s;

getline(fin,s);

while (fin)

{

getline(fin,s);

int n = atoi(s.c\_str());

if (hashmap.find(n) != hashmap.end())

{

auto i = hashmap.find(n);

i->second = i->second + 1;

}

else

{

hashmap.insert(std::make\_pair(n, 1));

}

}

fin.close();

fstream fout;

fout.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-1\\Debug\\ouput.txt", ios::out);

int max = 0;

int maxnum = 0;

for (auto i = hashmap.begin(); i != hashmap.end(); i++)

{

if (max < i->second)

{

max = i->second;

maxnum = i->first;

}

}

fout << maxnum << endl << max;

fout.close();

return 0;

}

### 算法二：递归：

#### 思想

首先利用sort对数组进行排序，根据分治法思想，分解子问题；计算数组的中位数，将数组分为左右两部分；若左边数组的个数大于中位数的个数，则递归左边数组，同理右边相同；newl记录数组第一个等于中位数的元素位置（即左边数组个数），newr记录第一个（从left开始）不等于中位数的元素位置（右边数组个数），记录right-left记录众数的重数。经过多次递归，所求的中位数就是众数。

#### 代码

#include<iostream>

#include<fstream>

#include<string>

#include<algorithm>

using namespace std;

void split(int a[], int orrl,int orrr, int &r, int &l)

{

int midnum = (orrr- orrl) / 2 + orrl;

int mid = a[midnum];//找到中位数,如果是偶数取到左边那个

for (l = midnum - 1; l > orrl; l--)

{

if (a[l] != mid)//当左边的位数不为中位数

{

l = l + 1;

break;

}

}

for (r = midnum + 1; r < orrr; r++)

{

if (a[r] > mid)

break;

}

}

void getMaxnum(int a[], int &num, int &maxnum, int l, int r)

{

int newl=0, newr = 0;

split(a, l, r, newr, newl);

if (newr - newl > maxnum)//此时的中位数是计数最多的

{

maxnum = newr - newl;

num = a[(r - l) / 2 + l];

}

if (newl + 1 > maxnum)

{

getMaxnum(a, num, maxnum, 0, newl);

}

if (r - newr + 1 > maxnum)

{

getMaxnum(a, num, maxnum, newr, r);

}

}

int main()

{

fstream fin;

fin.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-1\\Debug\\input.txt", ios::in);

if (!fin)

{

cout << "can't open it!" << endl;

return 0;

}

int i = 0;

string s;

getline(fin, s);

int size = atoi(s.c\_str());

int \*a = new int[size+1];

int num = 0;//num为众数

int maxnum = 0;//maxnum为重数

while (fin)

{

getline(fin,s);

a[i] = atoi(s.c\_str());

i++;

}

fin.close();

sort(a, a+size);//升序排序

getMaxnum(a, num, maxnum, 0, size - 1);

fstream fout;

fout.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-1\\Debug\\ouput.txt", ios::out);

fout << num << endl << maxnum ;

fout.close();

}

## 分析结果

### 测试文档1：

Input.txt 20 9 9 9 1 2 3 4 5 6 8 9 8 98 98 110 98 8 98 98 98

output.txt 98 6

### 测试文档2：

Input.txt 50 18 67 63 26 93 59 99 43 36 2 88 93 19 25 20 81 32 18 34 55 20 24 9 30 75 88

93 19 27 90 50 4 6 44 33 49 79 14 25 39 11 18 11 18 81 92 96 11 58 30

output.txt 18 4

# 半数单集问题

## 问题描述：

给定一个自然数n，由n开始可以依次产生半数集set(n)中的数如下：

(1) n ∈set(n)；

(2) 在n的左边加上一个自然数，但该自然数不能超过最近添加的数的一半；

(3) 按此规则进行处理，直到不能再添加自然数为止。

## 算法实现

### 算法一

#### 思想

使用set容器自动去重，但因为要计算出每个值所以内存开销大（不建议使用）

#### 代码（略）

#include <iostream>

#include <fstream>

#include <set>

#include <iomanip>

#include<math.h>

using namespace std;

int getn()

{

fstream fin;

fin.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-4\\Debug\\test.txt", ios::in);

if (!fin)

{

cout << "can't open it!" << endl;

return 0;

}

long n = 0;

while (!fin.eof())

{

int temp = fin.get() - '0';

if (temp < 0)

{

break;

}

n = temp + n \* 10;

}

fin.close();

return n;

}

set<long long> halfset(long long n, set<long long> a, int addhead)

{

for (int j = 1; j <= addhead; j++)

{

long long temp = n;

int length = 1;

while (temp >= 10)

{

temp = temp / 10;

length++;

}

a.insert(n + j \* pow(10, length));

//cout << fixed << setprecision(0) << n + j \* pow(10, length) << endl;

if (j / 2 > 0)

{

a = halfset(n + j \* pow(10, length), a, j / 2);

}

if (j == 50)

{

int temp = 0;

}

}

return a;

}

int main()

{

set<long long> a;

long long n = getn();

a.insert(n);

//auto i = a.begin();

//cout << \*i << endl;

a = halfset(n, a, n / 2);

fstream fout;

fout.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-4\\Debug\\output.txt", ios::out);

fout << a.size();

//for (auto i = a.begin(); i != a.end(); i++)

//cout << \*i << endl;

fout.close();

}

### 算法二：使用递归，只是计算个数

#### 思想

还是一样的递归，去重使用判断式，不再依赖容器

判断式来源：题目给的n<=201,所以n/2<=100,那么重复元素一定是一个两位数,且十位上的数字<=个位上的数字的一半,我们剔除组成十位上的数字的方案数即可

#### 代码

#include <iostream>

#include <fstream>

using namespace std;

int getn()

{

fstream fin;

fin.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-4\\Debug\\test.txt", ios::in);

if (!fin)

{

cout << "can't open it!" << endl;

return 0;

}

long n = 0;

while (!fin.eof())

{

int temp = fin.get() - '0';

if (temp < 0)

{

break;

}

n = temp + n \* 10;

}

fin.close();

return n;

}

int halfSet(int num)

{

int count = 1;

if (num == 1)

return 1;

for (int i = 1; i <= num / 2; i++)

{

count = count + halfSet(i);

if ((i / 10) \* 2 <= (i % 10) && (i > 10))

count -= halfSet(i / 10);

}

return count;

}

int main()

{

int n = getn();

n = halfSet(n);

fstream fout;

fout.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-4\\Debug\\output.txt", ios::out);

fout << n;

fout.close();

}

## 分析结果

测试文档1：test.txt 100

output.txt 9620

测试文档2：test.txt 6

output.txt 6

# 集合划分问题

## 问题描述

给定正整数n，计算出n个元素的集合{1，2，……，n}可以划分为多少个非空子集

Input：n

Output：非空子集个数

## 算法设计

#### 思想

使用递归式F(n-1,m-1)+F(n,m-1)\*m

#### 代码

#include <iostream>

#include <fstream>

using namespace std;

int getn()

{

fstream fin;

fin.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-7\\Debug\\test.txt", ios::in);

if (!fin)

{

cout << "can't open it!" << endl;

return 0;

}

long n = 0;

while (!fin.eof())

{

int temp = fin.get() - '0';

if (temp < 0)

break;

n = temp + n \* 10;

}

fin.close();

return n;

}

long long F(long long n,long long m)

{

if (m == 1)

return 1;

if (m == n)

return 1;

else

return F(n - 1, m - 1) + F(n - 1, m )\*m;

}

int main()

{

int n = getn();

long long sum = 0;

for (int i = 1; i <= n; i++)

sum = sum + F(n, i);

fstream fout;

fout.open("C:\\Users\\11732\\Desktop\\算法\\新建文件夹\\2-7\\Debug\\output.txt", ios::out);

fout << sum;

fout.close();

}

## 分析结果

测试样例1：input.txt: 4 测试样例2：input.txt: 30

output.txt: 15 output.txt: 6568040393494172515