通过编程对遍历查找算法和二分查找算法性能测试的研究

赵 水

（浙江工业大学 计算机学院，浙江 杭州 310014）

遍历查询一般是指对于指定的某个需要查找的值，在数组中找到之前将从头搜索整个数组，若找到对应值，则中断查找；二分查找则需在有序数组中，对于所需查找的值，不断地将数组二分并逐渐缩小值存在的范围，当范围不能再继续缩小时，若还没能找到指定的值，则数组中不存在，若找到，则中断查找；以下将探究两种算法的时间复杂度在数据量递增时的规律。

1 实 验

下面使用 python 语言用算法来实现上述算法和实验：

以下为代码：

```
# -*- coding: utf-8 -*-
import random
import time
import numpy as np
import matplotlib.pyplot as plt
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False


def createRandomArray(numOfArray,numRange):#定义函数用来生成大小为 numOfArray，在 0 到 numRange 范围内的整数的数组
    arr = []
    for i in range(0,numOfArray):
        arr.append(i+1)
    return arr

def findNumInArrayWithErgodic(array,key):#遍历查找法
    result = False;
    for i in range(0,len(array)):
        if array[i] is key:
```

```
            result = True
            break
    return result;

def findNumInArrayWithHalfSep(array,key):#二分查找法
    left = 0
    right = len(array) - 1
    middle = (left + right)//2
    while left <= right:
        middle = middle = (left + right)//2
        if array[middle] is key:
            return True
        elif key < array[middle]:
            right = middle - 1
        else:
            left = middle + 1
    else:
        return False

PowerArray = []#数据量，用于充当图表的 x 轴坐标
TimeArrayForBinary = []#在当前数据量下二分查找所用的时间，用于充当图表的 y 轴坐标
TimeArrayForErgodic  = []#在当前数据量下遍历查找所用的时间，用于充当图表的 y 轴坐
标

power = 1 #数据量从 1 开始累加
while power <= 10000000:
    PowerArray.append(power)#画点
    key = random.randint(0, power) #生成所需查找的数值
    TimeBinaryTemp = [] #
    TimeErgodicTemp = [] #保存一百组数据的测试结果，用于进行去平均运算
    print(power)
    for j in range(0,100):#每个 power 容量下测试一百次，重复实验
        array = createRandomArray(power, power)

        #遍历查找
        oldtimeForErgodic = time.time()#当前时间
        result = findNumInArrayWithErgodic(array, key)
        newtimeForErgodic = time.time()#运算结束后时间
        TimeErgodicTemp.append(newtimeForErgodic-oldtimeForErgodic)

        #二分法
```

```
        oldtimeForBinary = time.time()
        result = findNumInArrayWithHalfSep(array, key)
        newtimeForBinary = time.time()
        TimeBinaryTemp.append(newtimeForBinary-oldtimeForBinary)

    TimeArrayForErgodic.append(np.average(TimeErgodicTemp))#取平均运算后的值
    TimeArrayForBinary.append(np.average(TimeBinaryTemp))
    power *= 10 #增大容量继续试验


plt.xlabel("样本数")
plt.ylabel("所用时间")

plt.plot(PowerArray,TimeArrayForBinary,'r')
plt.plot(PowerArray,TimeArrayForErgodic,'b')
plt.legend(['二分',"遍历"])
plt.show()#绘制图表
```

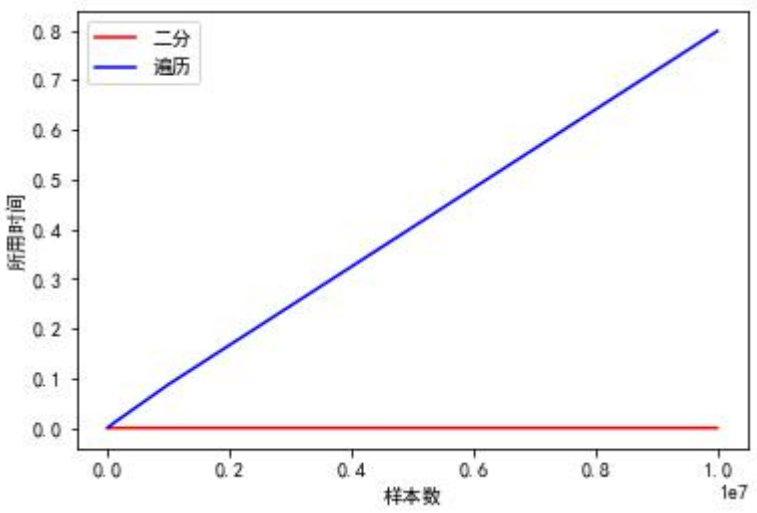

图 1 某次实验通过上述代码所运行出来的结果

2 结 论

多次实验下来，都得到了与图 1 相似的概率曲线，而从图中可以看出，遍历算法所用的时间呈 O（n）线性增长，而二分算法所用时间基本为零，实际上为 logn。