**《算法设计与分析》实验报告**

我保证没有抄袭别人作业!

1. **实验名称**

贪心选择

1. **问题描述**

1. 输入：n个活动的集合S={1, 2, …, n}，每个活动占用资源的时间为 F={ [si，fi) }，1≤i≤n

输出：S 的最大相容活动集合

2. 分数背包问题：给定3个物品，背包容量为20斤，各个物品的价值分别为

v=(25,24,15), 各物品的重量分别为：w=(18,15,10)。物品允许选择部分,

请给出一种物品选择方案使得背包总价值最大，并输出该最大价值

请按以下贪心策略设计算法, 比较算法结果的异同

1）贪价值 2）贪重量 3)贪单位重量价值

要求对结果进行分析，分析讨论正确的贪心策略是什么，为什么其它的策

略有问题

3.哈夫曼编码

1. **算法设计与分析**
2. **算法选择**

贪心算法有以下两个性质:

1最优子结构：问题的最优解由子问题的最优解构成

2 贪心选择性质: 可以按照某个标准选择（贪什么），通过不断构造局部的最优解（怎么贪），最终得到整个问题的最优解

相比动态规划，二者都有最优子结构性质，但贪心选择自顶向下，每一步依靠某个标准选择，动态规划自底向上，每一步依靠前面前面的最优解。

1. **算法设计**

**算法名称**：最多活动安排

**输 入：**活动的开始时间s,结束时间j

**输 出：**能相容的最多活动集合A

**算法步骤：**

(1) 先判断结束时间j是否有序,无序则先排序

(2) 选择第一个活动，寻找开始时间在第一个活动结束后的第二个活动，依次继续寻找第三个…

**算法名称**：分数背包

**输 入：** 物品的重量数组w,价值数组p，背包容量C

**输 出：**最大价值

**算法步骤：**

(1) 先按照某个标准将原数组排序（单位重量价值，重量或价值）

(2) 从排好序的数组中选择，如果背包容量足够，就全装

否则取一部分装入后，退出

返回最大价值

1. **算法复杂度分析**

1.活动时间安排

**（1）时间复杂度**

如果结束时间有序，整个算法只需要遍历一遍f数组，时间复杂度为O(n).如果无序需要对f，s进行排序，采用的排序算法时间复杂度更高O(nlogn),总体为O(nlogn)

**（2）空间复杂度**

算法过程需要布尔数组A记录选择的活动，所以空间复杂度为O(n)

2.分数背包

**（1）时间复杂度**

和上述的活动时间安排相似，分数背包需要根据需求先进行排序，然后再进行选择，排序的时间复杂度为O(nlogn),而排序后选择的过程只需要遍历一次数组为O(logn),所以时间复杂度为O(nlogn)

**（2）空间复杂度**

需要额外存储n个物品对象，因此空间复杂度为O(n)

3.哈夫曼编码

**（1）时间复杂度**

1.从文件读取字符，统计字符数量，这一部分时间复杂度为O(n)

2.从统计字符数量来编码哈夫曼编码，由于字符集总是一个常数C,所以生成哈夫曼树，深度最大为logC, 而C种字符需要进行C-1次合并，所以生成树的时间复杂度为O(ClogC),为其编码需要遍历整个树，为O(C),

3.对源文件进行编码，需要遍历n个字符，然后查表，这一过程为O(n)

所以时间复杂度为O(n+ClogC),n为输入需要编码字符的数量,C为该字符集的大小

**（2）空间复杂度**

读文件过程需要存储字符频率表，额外需要O(C)个空间，生成哈夫曼树需要存储，这需要O(C)，存储哈夫曼编码需要O(C),整体需要O(C)的额外空间

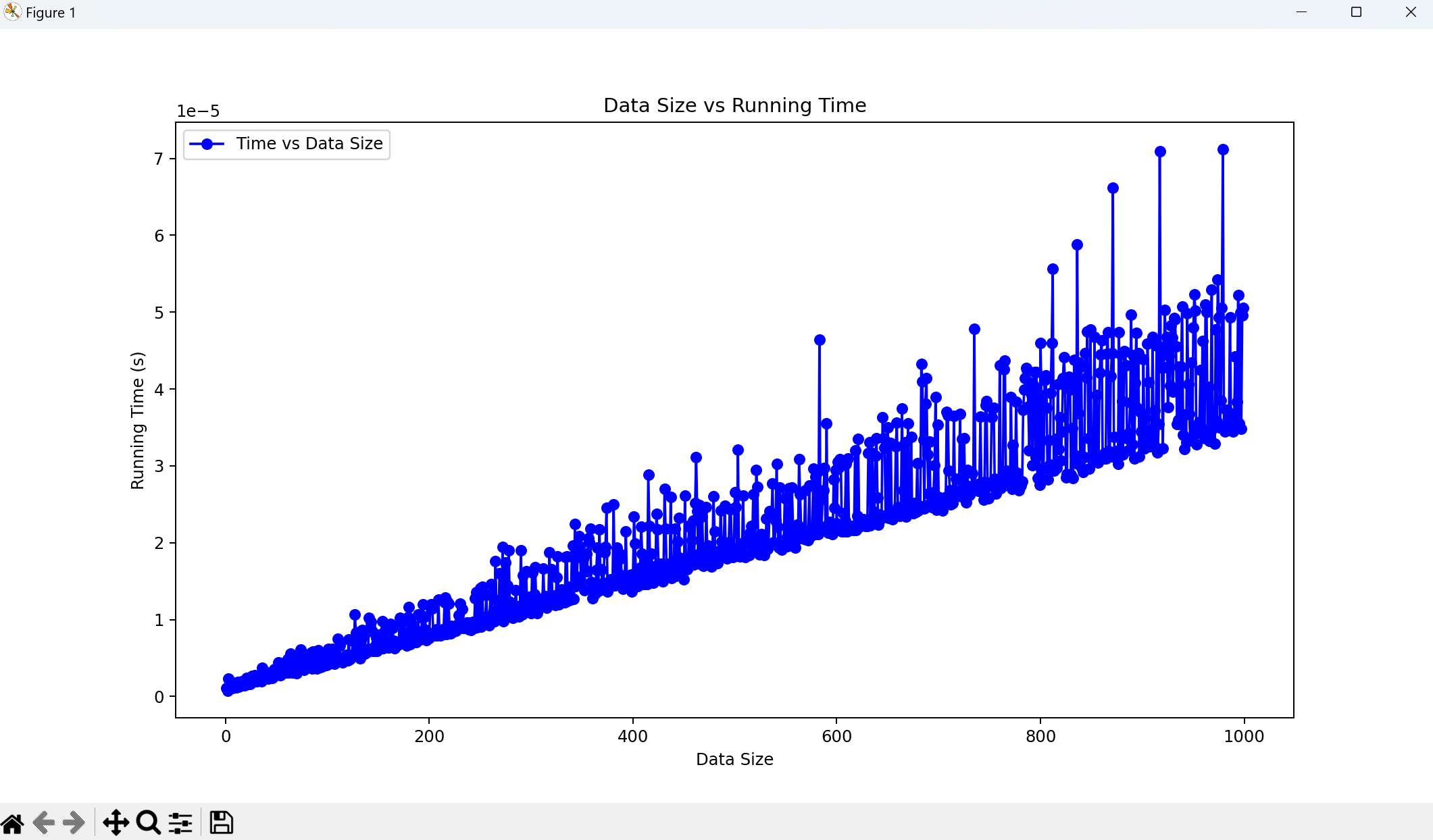
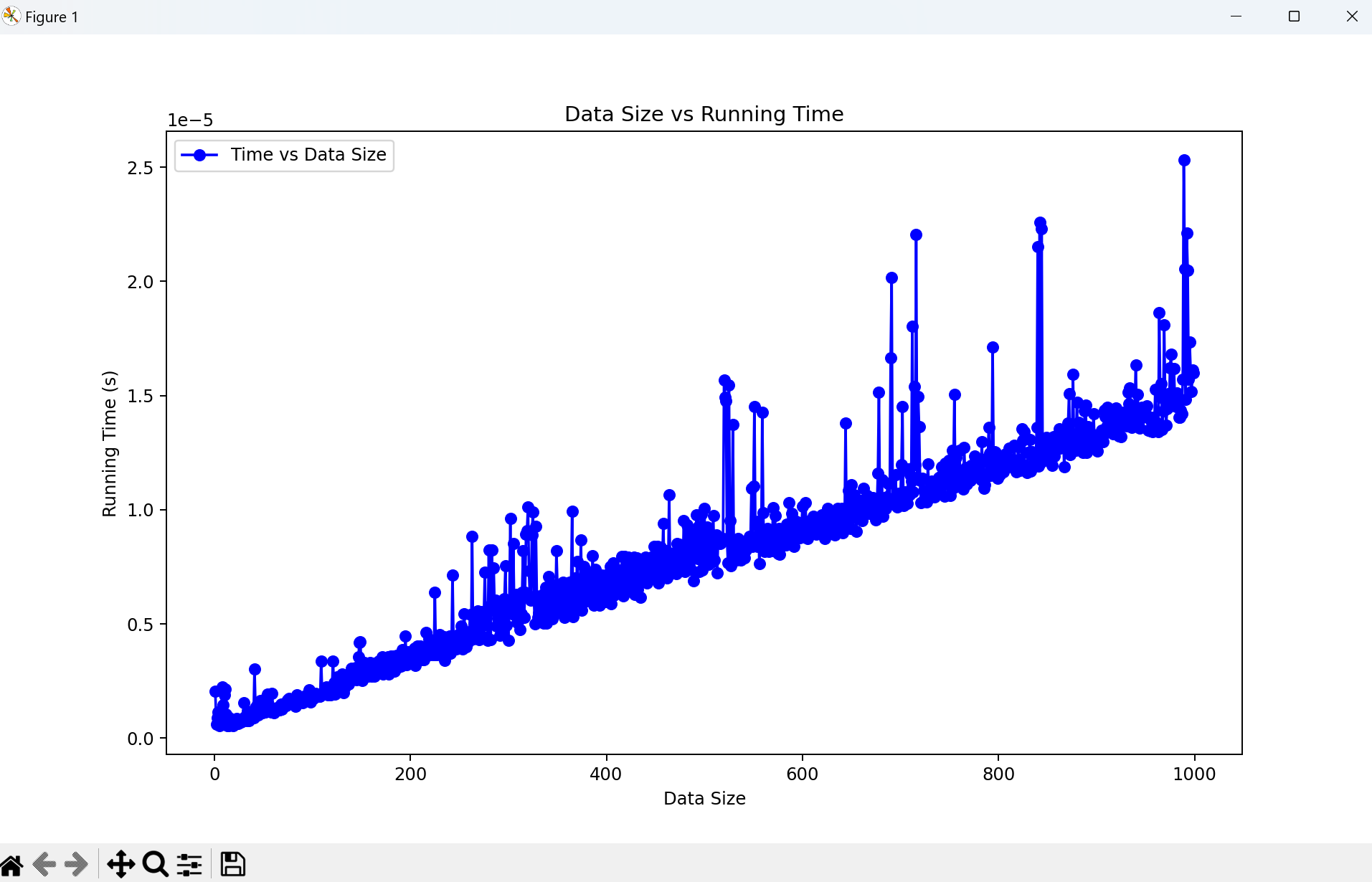
1. **实验测试**
2. **测试用例及结果分析**

Testdata.hpp文件中的Testdata类函数生成，可以生成一维数组，乱序或有序，二维数组，以及随机字符串，可以根据数据所需求的规模设定生成测试用例。而测试时间由Timetracker.hpp中的measuretime函数实现，可以根据具体算法可以调整运行次数，以得到函数较为准确的运行时间，并且避免运行时间过久。

后测试用例的数据规模会记录到datascale.txt文件中，而对应的运行时间记录在file.txt文件中，file为算法名称（与cpp文件同名），再经过Draw.py(该文件调用了python的matpilot库)将其绘制为坐标图，以更加直观的方式展现运行时间随着数据规模的变化，更好进行性能分析。

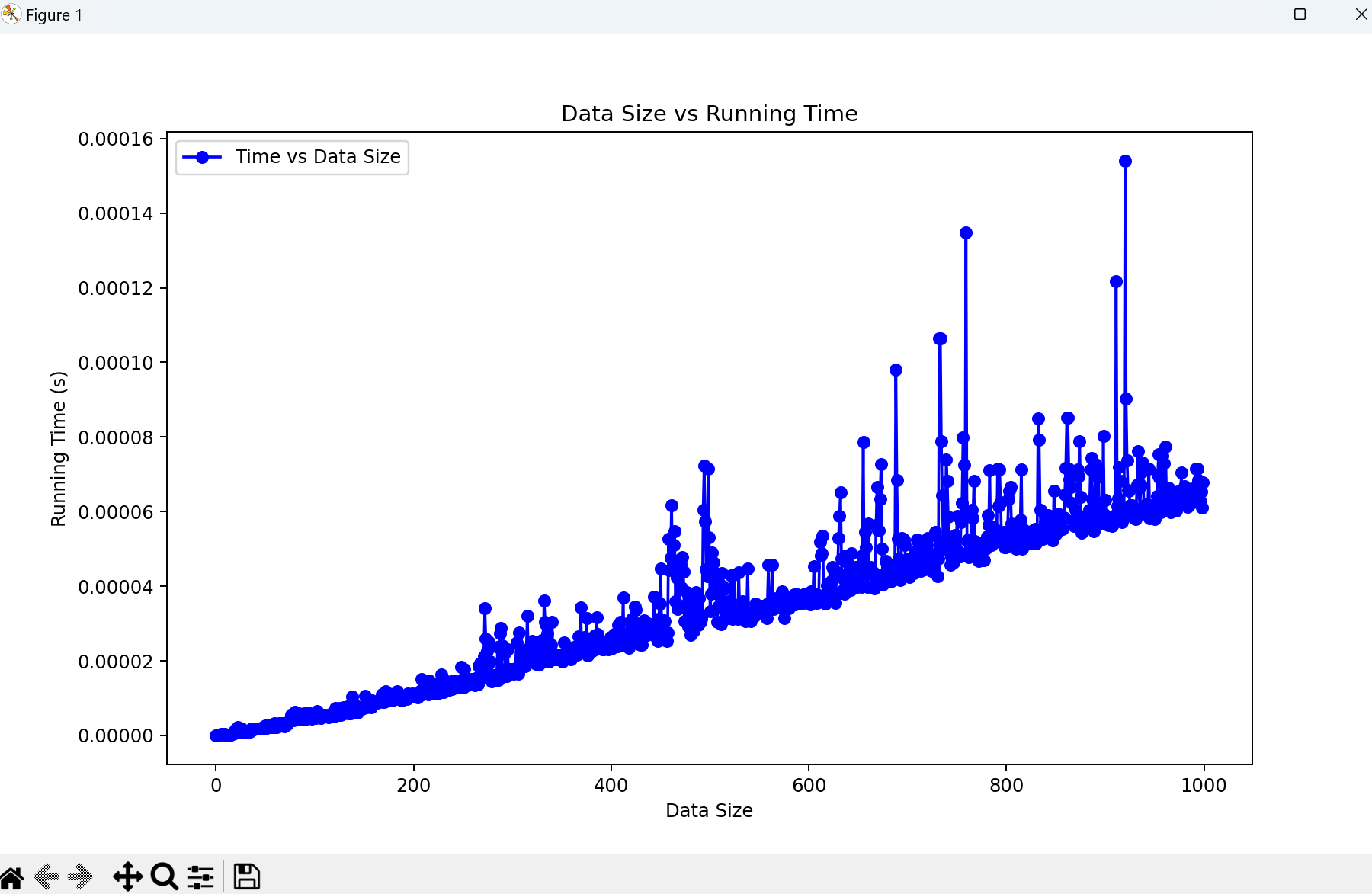
操作步骤，先运行算法.cpp文件，生成两个txt数据文件，再运行python文件根据数据文件进行绘制，即可复现以下测试用例的结果（由于测试数据每次随机生成，可能略微不同，为确保数据来源真实以及结果准确，会保留展示的测试用例数据，放在工作目录下的testdata.txt，同时便于进行算法的正确性验证）

1.活动时间安排

上图为生成的活动结束时间没有排序，而下图是生成活动排好序的，理论上上图的时间复杂度为O(nlogn),不应该为线性，与预期的曲线相差较大，但对比下图，可以看到下图的线性更为明显，这样看来上图的散点过大的原因确实是一部分测试用例不符合线性时间增长的规律。

猜想原因是因为这里的排序函数sort直接采用的标准库函数，标准库的sort函数可能性能优化极佳，在1000数据不太会显示O(nlogn)的特性，下面又选取1000个数据对sort进行测试，结果如下



猜想得到验证，标准库sort函数的优化极佳，在1000组数据内的运行时间几乎是线性，所以导致上述两个图，有无排序的对图像走势影响不大，但是从运行时间上还是可以看出差异，已排序的活动时间组要明显更快，且更符合线性时间O(n)的特性。

2.分数背包问题

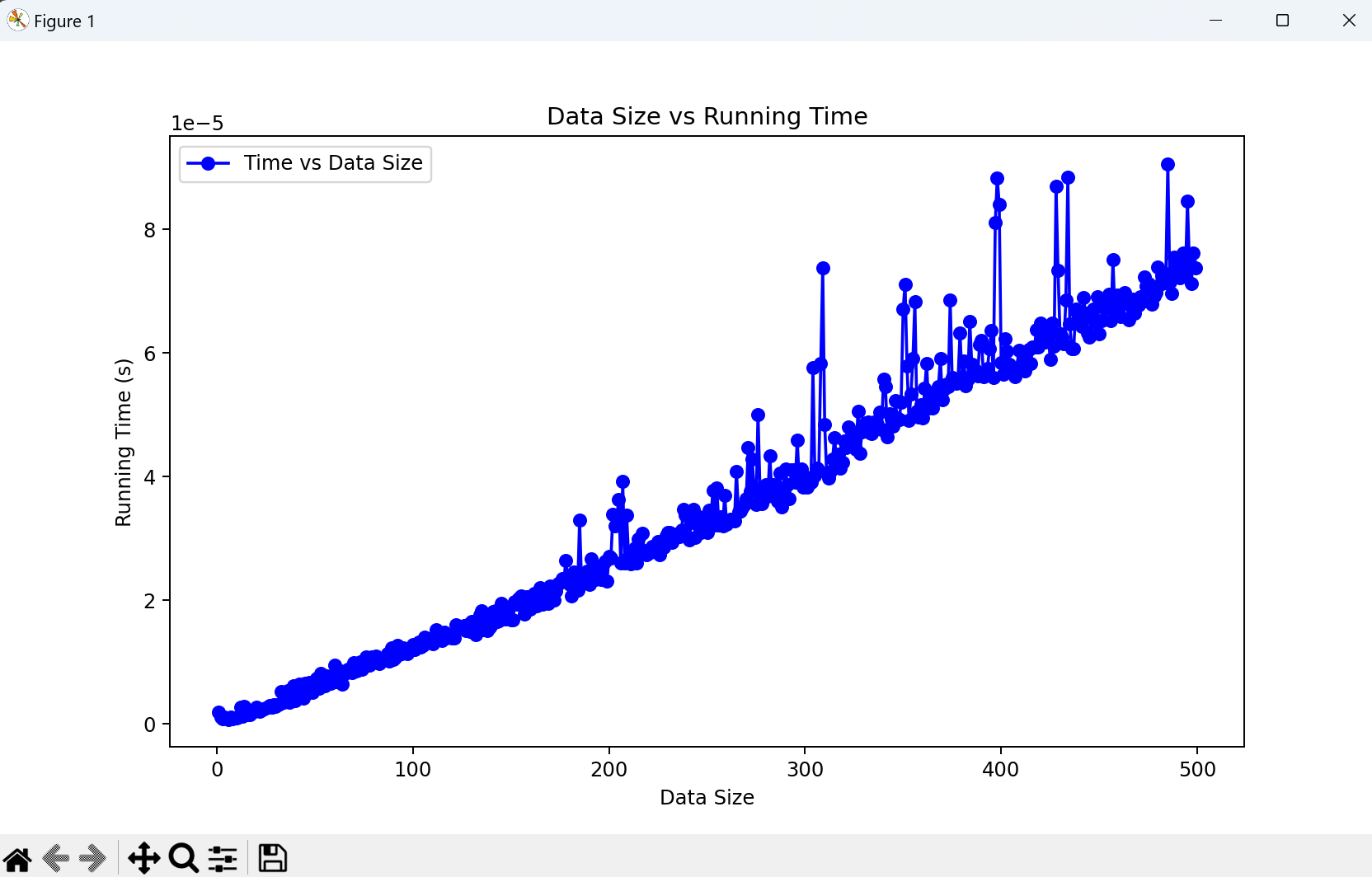
正确性分析:

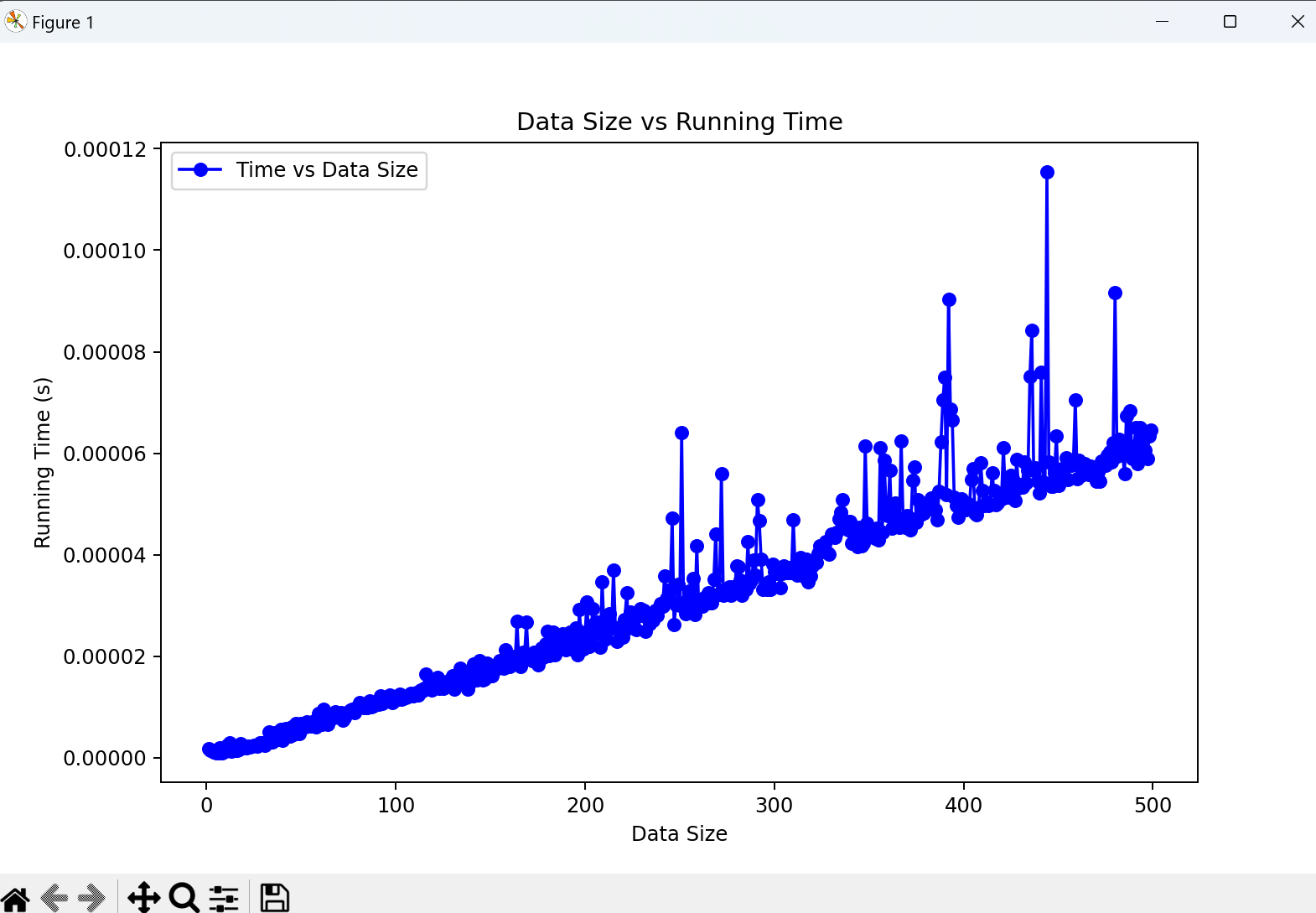
将排序的标准分别修改为单位重量的价值，价值，重量。

结果分别为31.5,28.2,31.最后能得到最大价值的为按单位重量价值排序的可以获得最大价值。

为什么其他的策略有问题，因为虽然最终的目的是要求价值最大，但是因为有背包容量这个限制，该问题实际的目的是在该容量下，能达到的最大价值，即单位重量价值最大，而另外两个策略，一个是能确保将价值最大的物品拿走最多，而另一个是能将尽可能多的物品拿走。都没有达到我们的目的，所以有问题。正确的贪心策略应该要充分考虑限制的条件下，

在进行测试之前，经前面的复杂度分析，该问题和活动时间安排问题类似，猜想测试结果应该也是相近的。

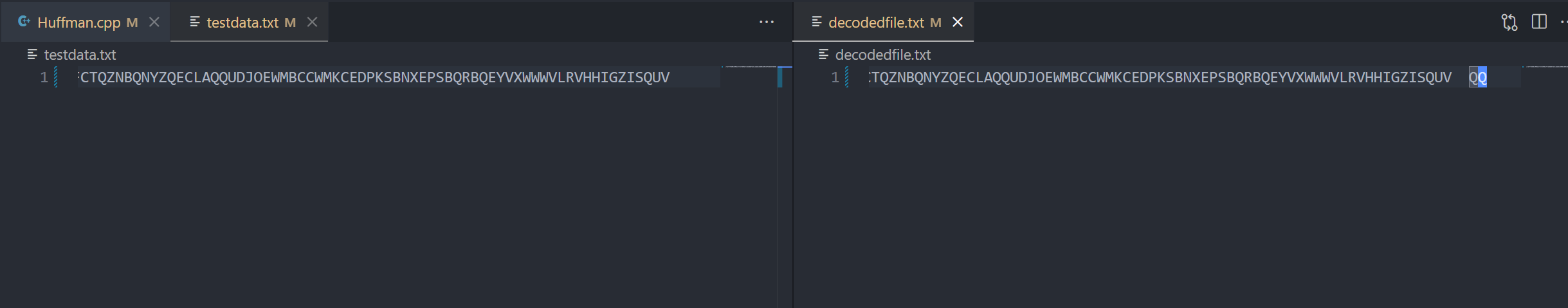


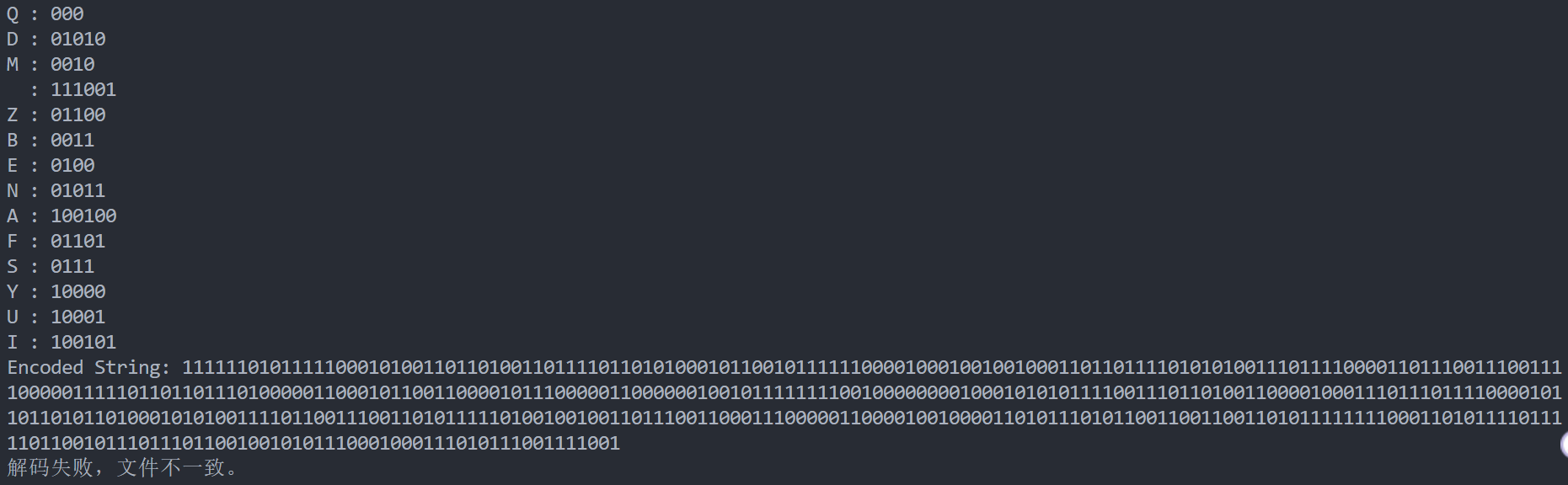


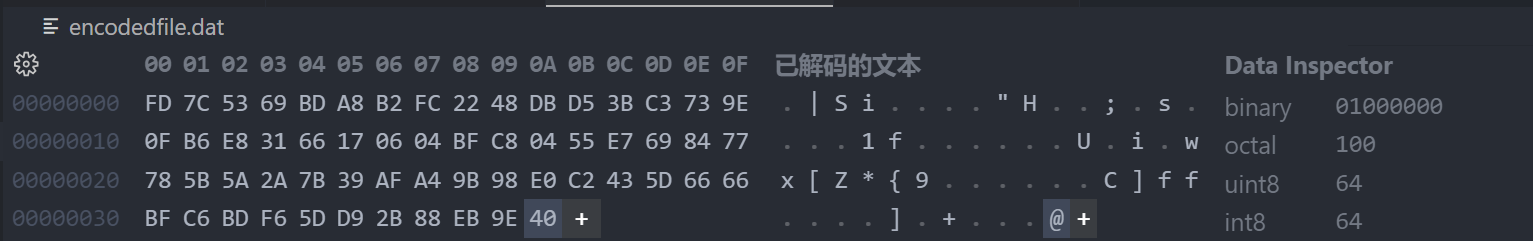
上图为未排序完全随机的状态，而下图为排好序的，发现两者的线性时间特性都较为明显，但后者的运行时间是明显低于前者。这里我没有事先判断物品是否有序，但sort函数在有序情况下可以极大提升排序速度（例如合并排序有无序提升不大），再次可见标准库中的sort函数的强大，相比自己手动编写的排序函数性能要强上很多

3.哈夫曼编码

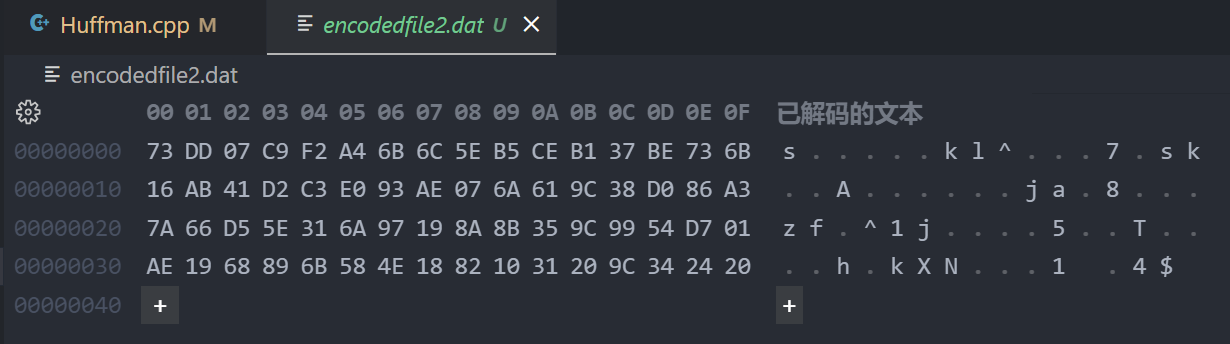
这一部分在第一次运行时，解码文件盒源文件可以对上，但偶然发现有一次结果不能对上，经过对比生成文件可以看到解码文件的最后会多出一些字符

例如这一次测试用例，发现最后会多出两个QQ，猜想是填充的0字符导致最后解码时将其识别成Q,于是在将Huffmancode和二进制结果打印在终端，并最后解析生成的二进制文件.dat,来确认猜想.

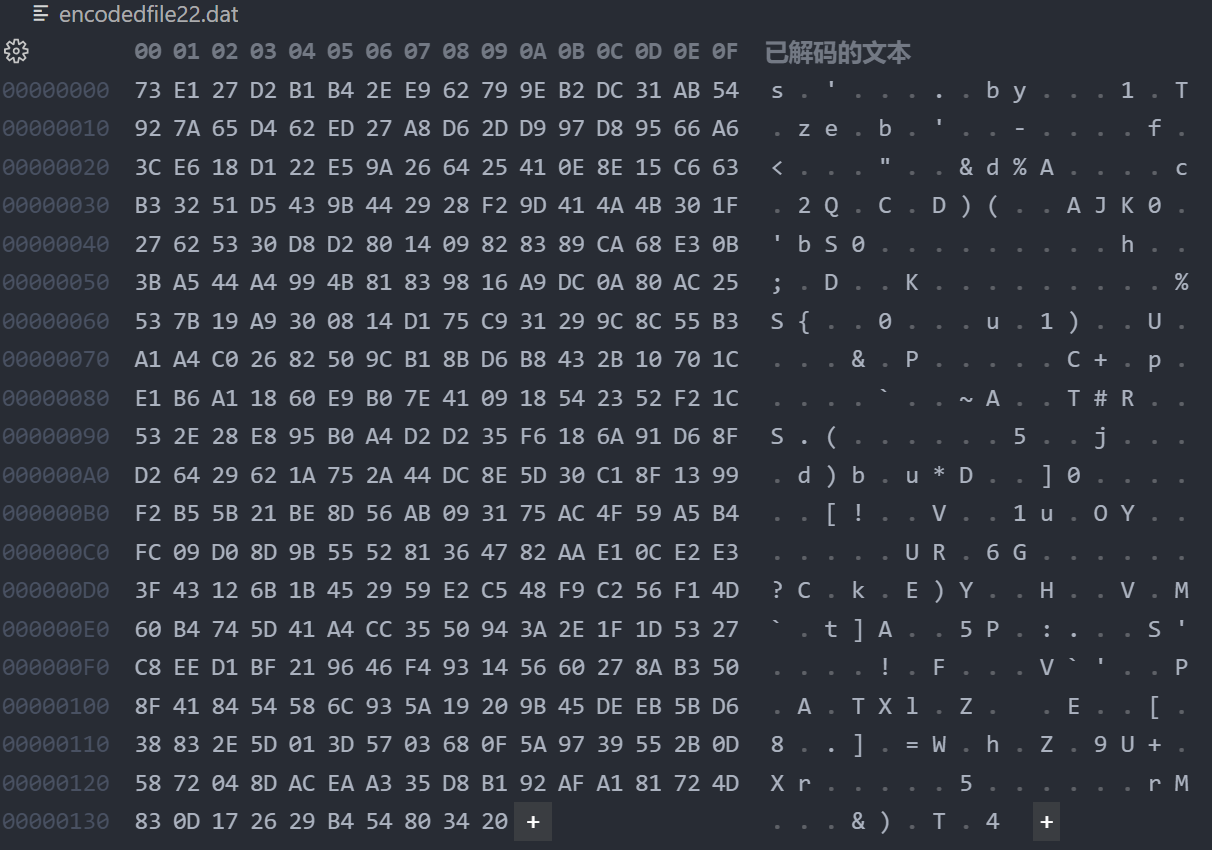
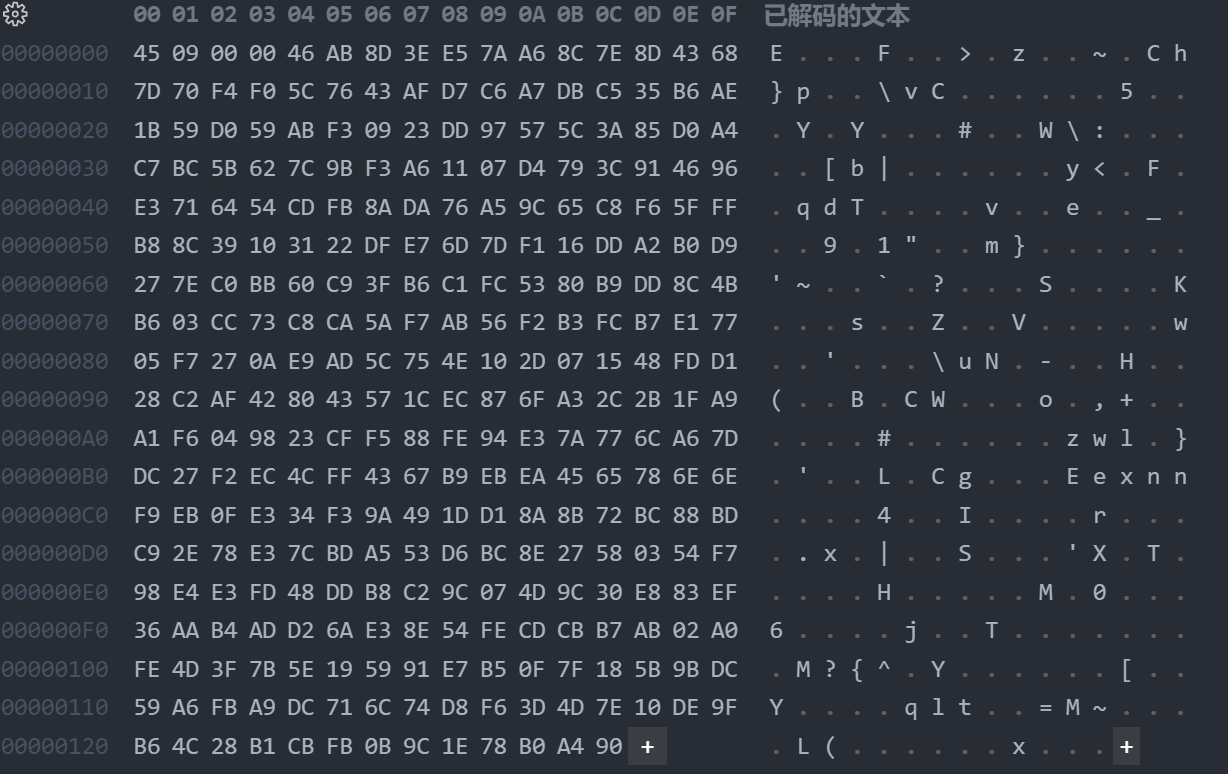




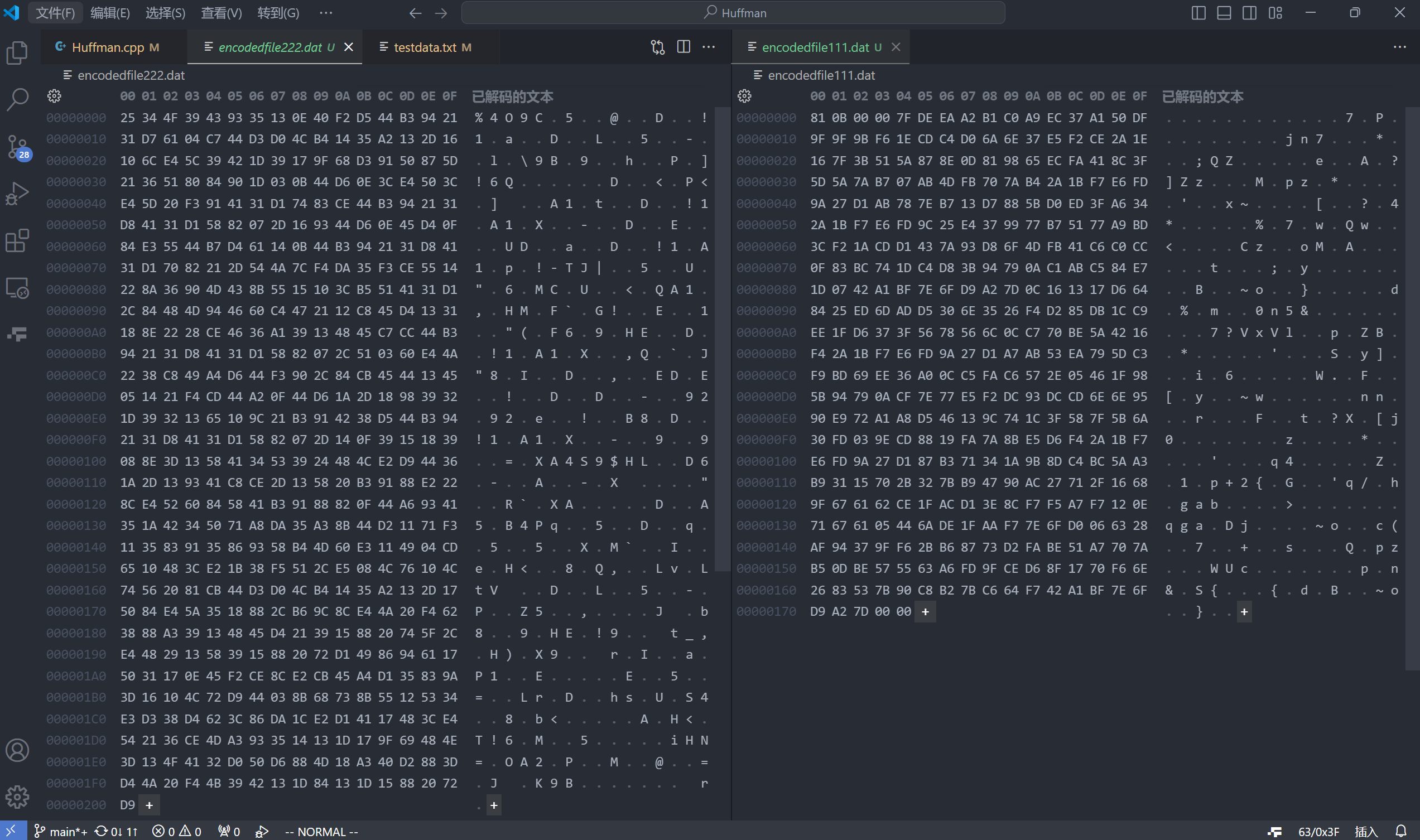
可以看到dat文件相比编码后的二进制字符串最后填充了6个零，而Q在该次生成的编码为000,所以最后会识别为两个Q，导致解码错误，这时又引出一个问题，如何识别最后哪些0时填充的，哪些不是。方法是记录编码后的有效位，在后续解码时每读取一位就减1，直到小于0即有效位读取完毕。



Codefile2为定长编码的结果，发现比哈夫曼编码多了一个字节。如果选取更长的字节应该差距会更加大。



结果发现500字符串发现也差不多，这时突然意识到哈夫曼编码可以优先筛选高频字符，所以该随机生成的测试数据各字符的频率相近，导致哈夫曼编码不能提现其优势。于是去网上找了一段英文段落作为测试用例。这时两者的区别很明显



1. **实验总结**

通过活动安排，分数背包和哈夫曼编码三个案例，体会了贪心算法的思想精髓，在于不需要前面的铺垫（例如动态规划），每一步都是基于当前状态给出的局部最优解，看似比较短视，但只要问题存在贪心性质和最优子结构，也可以做出最优解，即使一些情况不能得到最优解，贪心选择也可以得到一个较好的解。但在选择时一定要明确“贪什么”，这时该算法的关键。哈夫曼编码虽然也是贪心算法，但为了处理细节，花费的时间精力远远超过体会贪心算法的一小部分，以致写完都忘了这是在贪心算法章节,但不得不承认这是贪心算法一个很典型的应用，并且很有实用范围。