哈夫曼树、编码设计说明书

https://github.com/lingbai-kong
同济大学

2020 年 7 月

第一部分 算法实现设计说明

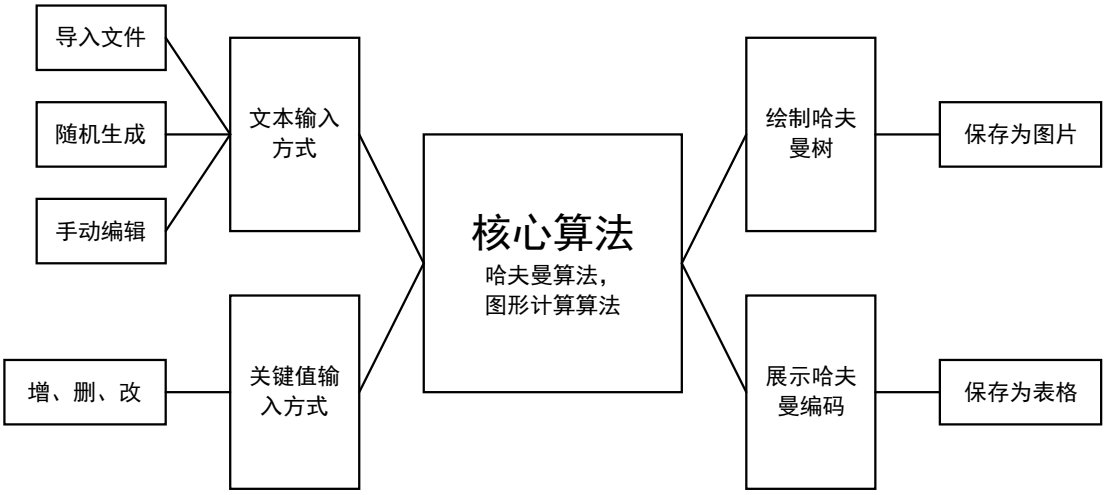
1.1 题目

6.哈夫曼树、编码

给出一组关键值，建立哈夫曼树，显示该哈夫曼树，并给出每个关键值的哈夫曼编码。

说明：关键值的获得可以选择通过以下途径：（1）给定的一组关键值；（2）给定的一个文本；（3）随机输入的一段文本。

1.2 软件功能



（功能结构图）

1.2.1 程序能够导入 TXT 格式文本文件内容到文本编辑框中。

实现方式：先弹出打开文件对话框获取文件路径及文件名，再打开指定文件并读取文件内容，最后将文件内容显示到文本编辑框中。

1.2.2 程序能够随机生成文本到文本编辑框中，随机文本可以指定为数字文本，字母文本，字符串文本，中文文本。

实现方式：先根据用户在下拉选择器中选中的文本类型和数字输入框中的文本长度，再生成随机文本，最后将随机显示到文本编辑框中。

1.2.3 程序能够让使用者在文本编辑框中输入文本，支持复制粘贴等操作。

实现方式：利用 PyQt5 提供的文本编辑框控件实现上述功能。

1.2.4 程序能够让使用者在关键值编辑框中逐个输入关键值，支撑关键值的删除、更改、重置操作。

实现方式：关键值的获取通过 PyQt5 提供的整数输入对话框采集，关键值编辑框利用 PyQt5 提供的表格控件实现相关功能。

1.2.5 程序能够在给定的文本或关键值上运行哈夫曼算法，绘制出哈夫曼树，罗列出哈夫曼编码表。

实现方式：如果在文本上运行，则先统计文本字符频度，生成关键值，如果直接在关键值上运行，则直接从关键值编辑框中获取关键值。得到关键值后运行哈夫曼算法生成哈夫曼树和哈夫曼编码，哈夫曼树要先进行坐标计算，然后才能在绘图区内绘制出哈夫曼树，哈夫曼编码表利用 PyQt5 提供的表格控件直接显示到界面。

1.2.6 程序的运行结果可以保存。哈夫曼树可以保存为图片，图片支持四种格式：jpg、png、tiff、bmp；哈夫曼编码表可以保存为 xls 表格

实现方式：首先通过文件保存对话框选中保存文件的路径和文件名称，然后获取绘图区/表格区的相关数据，再把数据保存为指定格式。

1.3 设计思想

1.3.1 底层算法

1.3.1.1 哈夫曼算法

原理简介：[1]

哈夫曼二叉树是使得带权路径长度最小的二叉树。二叉树的带权路径长度的计算公式是 $WPL=\sum_{k=1}^{n} w_k l_k$，其中 wk 是第 k 个叶子结点的权值，lk 是第 k 个叶子结点的深度。对哈夫曼二叉树的一种直观感受是：如果将带权路径长度看作是访问这个叶子结点的代价，那么在哈夫曼二叉树上遍历所有叶子结点花费的代价最少。

构造哈夫曼二叉树的方法如下：

1.根据给定的 n 个权值{w1, w2, …, wn}构成 n 颗二叉树的集合 F={T1, T2, …, Tn}，其中每颗二叉树 Ti 中只有一个带权值 wi 的根结点，其左右子树均为空。

2.在 F 中选取两棵根结点的权值最小的树作为左右子树构造一颗新的二叉树，且置新的二叉树的根结点的权值为左右子树上根结点的权值之和。

3.在 F 中删除这两棵树，同时将新得到的二叉树上的根结点加入到 F 中。

4.重复 2，3，直到 F 中只含一个结点为止。这棵树便是哈夫曼二叉树。

哈夫曼编码是一种前缀编码，可以通过哈夫曼二叉树设计前缀编码。约定二叉树的

[1] 参考自 严蔚敏 吴伟民，《数据结构（C 语言版）》，清华大学出版社，P154-P156

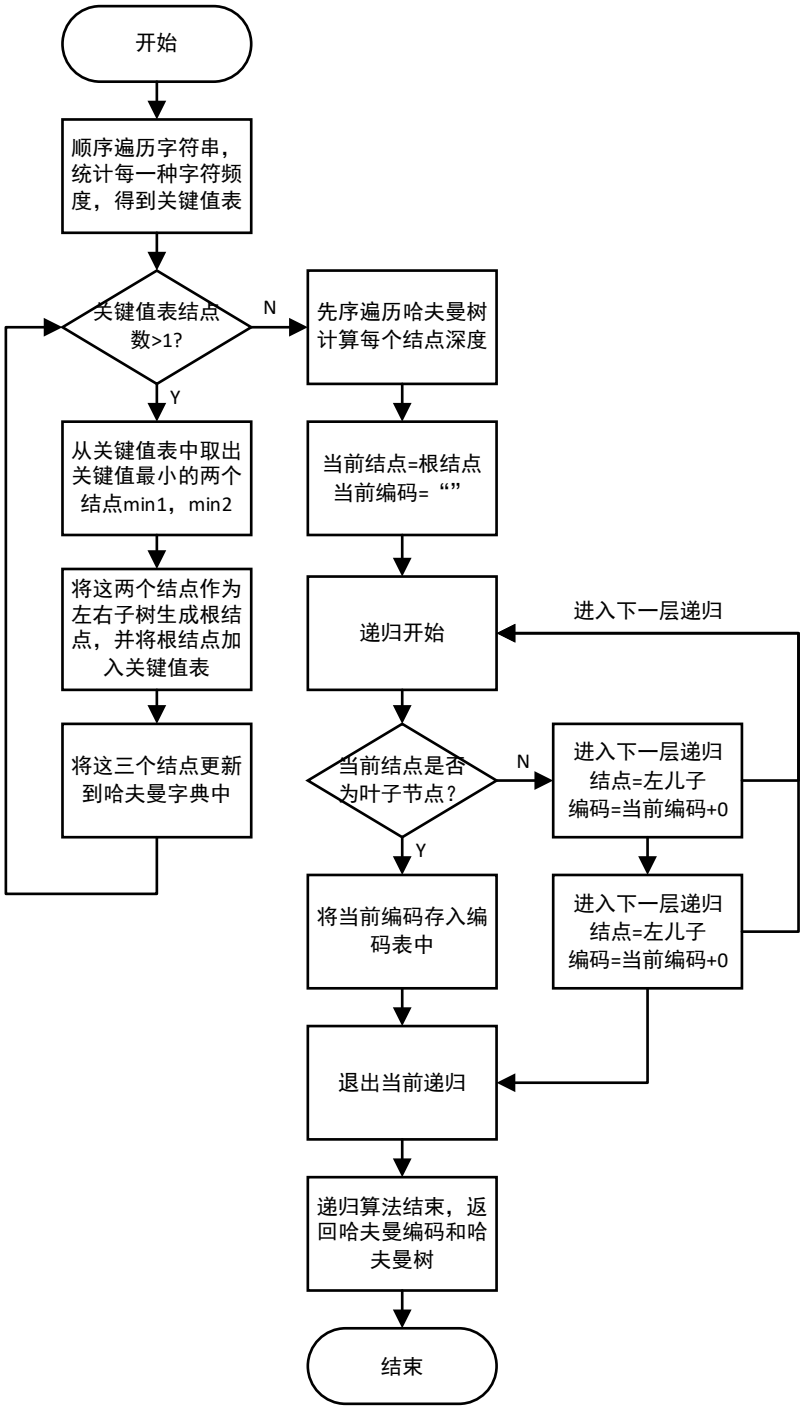
左分支表示字符‘0’，右分支表示字符‘1’，把从根结点到叶子结点的路径上分指字符组成的字符串作为该叶子结点字符的编码。可以运用深度优先遍历哈夫曼树的方法生成哈夫曼编码表。

设计实现：

数据结构设计：

程序需要统计文本中每一种字符的频度也就是哈夫曼树每一个叶子结点的关键值，容易知道每一个关键值和每一个字符一一对应，因此选用 python 中的字典类型构建，键值对为字符和其频度。同理，哈夫曼树、哈夫曼编码表也采取字典类型构建，哈夫曼树的键值对为结点名称和该结点信息，其中结点信息以列表构建具体包含该结点的关键值，该结点的后继，该结点的前驱和该结点的深度；哈夫曼编码的键值对为字符和其编码。python 字典的本质是一张哈希表，在本程序中，字典帮助代码完成了将字符串散列到哈希表中查找所需数据的功能。通过“以空间换时间”的策略将查找的时间复杂度由 O(n)降低至 O(1)。

算法设计：



（哈夫曼算法流程图）

1.顺序遍历给定字符串，统计每一种字符的频度，得到初始关键值表。

2.从关键值中取出关键值最小的两个结点，构建起一棵二叉树，新的根结点的关键值为其两个儿子结点关键值之和，将新的根结点及其关键值加入关键值表中。

3.计算两个儿子结点和根结点的关键值、后继、前驱到其数据列表中，将这三个结点及其数据记录至哈夫曼树字典中。

4.执行 2-3 直到关键值表中仅剩余一个结点，这个结点就是整个哈夫曼树的树根。

5.从根结点开始使用递归法先序遍历哈夫曼树，计算每个结点的深度并将深度加入至数据列表中

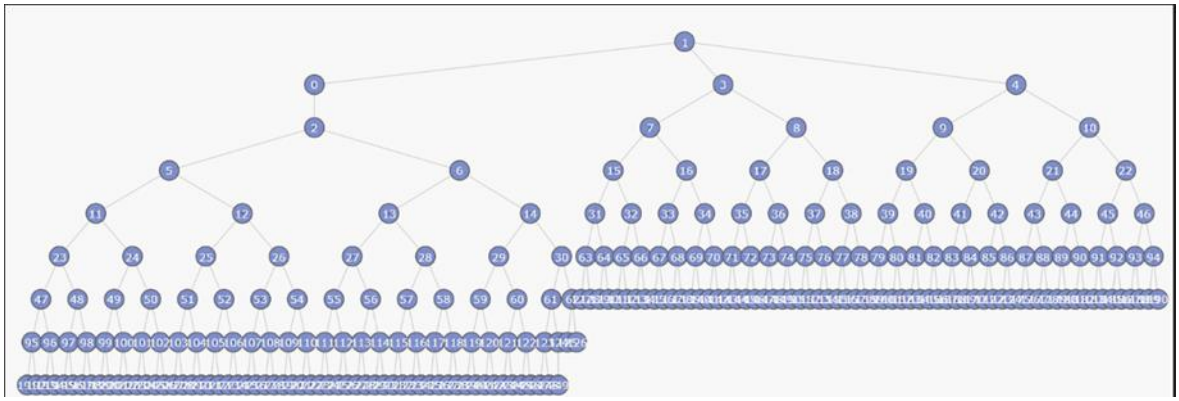
6.开始深度优先遍历哈夫曼树，初始时，结点设定为根结点，编码设定为空字符串。

7.如果当前结点是叶子结点，将编码存入编码字典后退出这一层递归。否则将下一层结点设定为当前结点的左儿子，在当前编码后加‘0’再次调用 7 进入下一层递归。然后再将下一层结点设定为当前结点的右儿子，在当前编码后加‘1’ 再次调用 7 进入下一层递归。

8.算法结束返回哈夫曼树字典和哈夫曼编码字典。

1.3.1.2 坐标计算算法

原理简介：



上图是一个标准的树的图形，观察这个图形可以发现以下特点：

（1）不管是否在同一层，所有叶子结点的水平距离相等

（2）任意一个根结点在其最左孩子和最右结点构成的线段的中垂线上

（3）每一层结点都在同一条水平线上，相邻层结点的竖直距离相等

根据以上特点我们发现，计算结点的 y 坐标是容易的，仅仅需要构造一个与结点深度相关的线性函数。难点主要在于计算结点的 x 坐标。下面将结点分为两类、一类是叶子结点、一类是非叶子结点。对于叶子结点而言其 x 坐标与其遍历的次序正相关、因此需要一个变量 xCount 记录当前叶子结点是第几个遍历到的结点。对于非叶子结点而言其 x 坐标仅与其孩子结点有关。可见 x 坐标的计算应该是自底向上计算的。根据这一特点选择深度优先遍历。

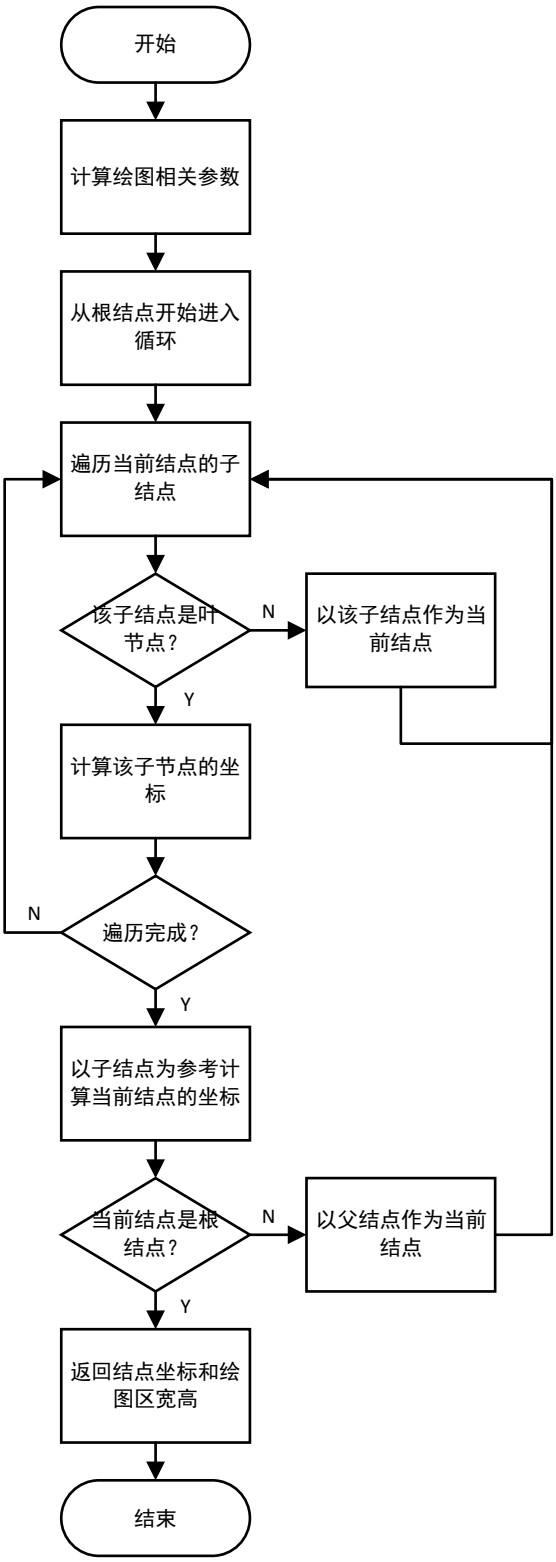
设计实现：

数据结构设计

因为每个结点和其坐标是一一对应的关系且每个结点的名字唯一，因此采用 python

中的字典类型，键值对为结点名字和其坐标。

算法设计



（坐标计算算法流程图）

首先指定 x 方向最小距离(xUnit)和 y 方向最小距离(yUnit)（x 方向最小距离就是叶

子结点的水平距离，y 方向最小距离就是每一层结点的竖直距离），左右边框宽度(xBondary)和上下边框宽度(yBondary)，然后根据哈夫曼树深度和叶子结点数和以上信息计算出整个哈夫曼树需要绘制的窗口大小具体计算公式为

$$\begin{cases} X = xUnit * 叶节点数目 + 2 * xBondary \\ Y = yUnit * 树最大深度 + 2 * yBondary \end{cases}$$

实际测试中发现 PyQt5 所支持的最大绘图窗口为 32767 像素×32767 像素的正方形，因此要求结点横纵间距参数需要根据不同哈夫曼树动态的改变，以保证窗口大小在要求范围内。具体实现方式就是上述公式的逆运算，已知量变为了窗口大小未知量变为了方向最小距离，具体计算公式为：

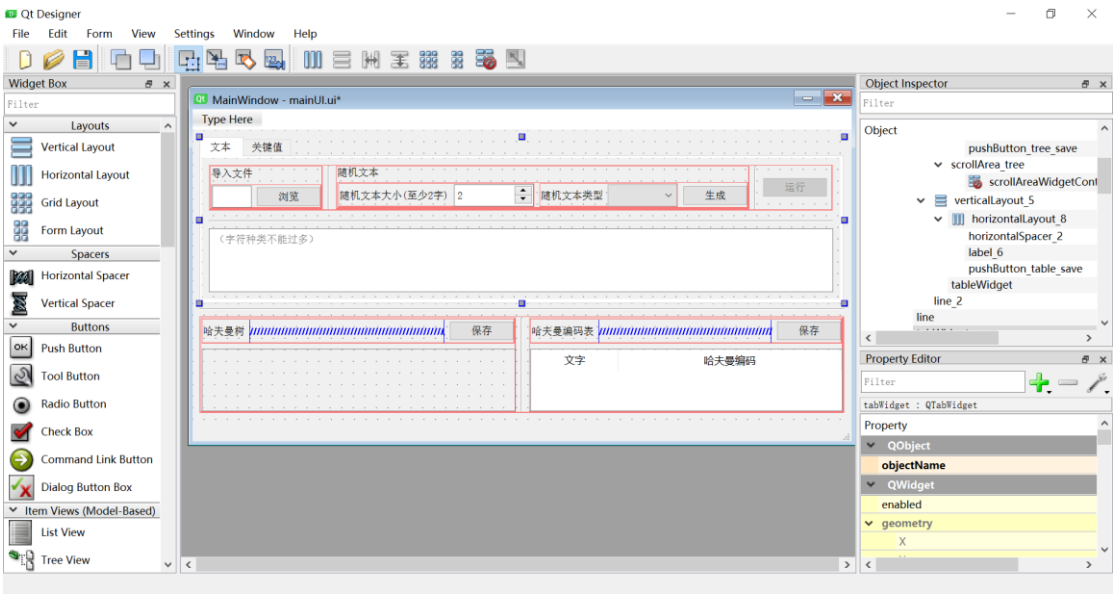
$$\begin{cases} xUnit = (xMax - 2 * xBondary) / 叶节点数目 \\ yUnit = (yMax - 2 * yBondary) / 树最大深度 \end{cases}$$

接着深度优先遍历哈夫曼树，当遍历到叶子结点时，根据 xCount 计算叶子结点的 x 坐标，并更新 xCount，继续遍历其他兄弟结点。当子结点的 x 坐标完全确定时，则平均这些 x 坐标得到父亲结点的 x 坐标。这样遍历直到根结点的 x 坐标被计算出，完成 x 坐标的计算。y 坐标的计算相对容易，它是关于结点深度的一次函数 $y[i] = deep[i] * yUnit + yBondary$。算法结束后将所有结点的坐标和窗口大小传出，供绘图程序使用。

1.3.2 UI 界面

1.3.2.1 前端设计

在可视化设计软件 QtDesigner 中设计整个界面的布局和主要控件的相关属性：



整个页面分为上半部分和下半部分。上半部分是输入区，文本输入方式和关键值输入方式各占一个标签页并且有其独立的运行按钮。文本输入方式中上面的工具栏包含文件导入、随机文本生成、程序运行三大功能块，下面的文本编辑框可以显示导入或生成的文本也可以让用户自行输入文本，为了保证哈夫曼编码的生成，只有文本编辑框内的字符多于两种时，运行按钮才会开放。关键值方式中上面的工具栏包含新增、删除、更改、重置、运行按钮，当关键值为两个及以上时才会开放运行按钮。页面的下半部分是结果输出区，左部的区域可以绘制哈夫曼树，使用 Scorll Area 控件以便在有限的空间内显示不同大小的哈夫曼树。右部的区域用于显示哈夫曼编码，绘图区和表格区的右上角各自带有保存按钮，用于保存算法的运行结果。

1.3.2.2 界面功能实现

绘图功能利用 PyQt5 提供的 QPainter 实现，通过先序遍历哈夫曼树，根据坐标字典绘制出哈夫曼树的结点和连线。

运行功能通过 run 函数和 run2 函数实现,两者的差别在于哈夫曼对象的建立方式，run 函数根据文本编辑框内容建立哈夫曼对象，run2 函数根据关键值表建立哈夫曼对象。建立起哈夫曼树后便调用哈夫曼算法生成哈夫曼树和哈夫曼编码，然后调用坐标计算算法，在绘图区绘制出哈夫曼树，并将哈夫曼编码显示到表格中。

随机生成文本功能通过 random 函数实现。获取下拉框选中内容和整数输入框内容调用随机生成字符串的函数将生成的文本显示到文本编辑框中。

打开文本文件功能通过 file 函数实现，先用文件对话框获取文件路径及文件名，将文件路径显示到文本框中。然后打开文件获取文件内容，将文件内容显示到文本编辑框中，对于空文件或文件操作错误的情况通过消息提示框实现。

保存哈夫曼树的功能通过 savetree 函数实现，通过文件对话框选定保存路径及文件名，然后导出绘图窗口数据至 QPixmap 对象，最后将 QPixmap 对象保存。

保存哈夫曼编码的功能通过 savetable 函数实现[2]，通过文件对话框选定保存路径及文件名，然后导出编码表数据到 Workbook 对象，最后将 Workbook 对象保存。

输入关键值时，对关键值的增删改操作分别有起独立的功能函数和按钮控件，其实现较为容易，对界面的表格控件进行操作即可[3]。

1.3.2.3 前后端对接

[2] 参考自网站资料 《python 使用 xlwt 模块操作 Excel》 https://www.jianshu.com/p/4e39444d5ebc
[3] 参考自完整资料 《QTableWidget 怎么得到鼠标位置所在的单元格》 https://bbs.csdn.net/topics/360266015

1.绘图窗口的对接使用 setWidget 方法将绘图窗口放入主窗口的滚动窗口（scrollArea_tree）中。

2.按钮和文本框的触发信号通过 connect 函数挂载到对应的功能函数上。

1.4 逻辑结构与物理结构

程序中操作的数据主要为哈夫曼树、哈夫曼编码表、关键值表和结点坐标表。

数据名	数据类型	逻辑结构	物理结构
哈夫曼树	字典	树形结构	结点：散列存储结构 结点数据域：顺序存储结构。
哈夫曼编码表	字典	集合结构	散列存储结构
关键值表	字典	集合结构	散列存储结构
结点坐标表	字典	集合结构	散列存储结构

1.5 开发平台

开发平台：Anaconda3 Spyder

开发语言：Python 3

具体版本：conda 4.7.12 python 3.7.1

核心使用库：PyQt5

运行环境：Windows 10（64 位）

1.6 开发结果

正确性:对于输入的文本或关键值组程序能够正确地构建出哈夫曼树和哈夫曼编码。可见程序能够输出正确结果。

稳定性：程序对于若干组“普通测试”、“边界测试”、“随机测试”均能在较短的时间内给出正确的结果。程序在运行期间无闪退、崩溃现象发生。可见程序稳定性良好。

容错性：首先“只有关键值多于两个才开放运行按钮”这一程序规则就避免了因为关键值不合法而导致程序出错的问题。其次程序在导入文本文件时如果程序是空文件或打开失败将会转入相应的错误处理代码，弹出错误信息。其三，文本编辑框的输入内容涵盖了 utf-8 的字符集，基本覆盖了所有日常使用的字符，避免了因为输入字符超范围而导致的错误。可见程序容错性良好。