**实验一数学表达式求值**

一、实验目的

熟练掌握栈、队列等基本操作及其在实际问题中的应用。

二、实验要求

基本要求：实现栈的基本操作，及表达式求值的实现。

1、实现栈的pop, push基本操作；

2、检测表达式是否是正确的数学表达式；

3、对于正确的数学表达式求取其值。

注：

1）数学表达式的判读与求值需支持加减乘除、小括号、中括号、大括号、幂运算。

2）不正确的表达式可能包括：字母、非运算符号、括号不匹配、运算符的排列不符合表达式形式的等多种情况。

3）表达式通过命令行参数获取。

输入：通过命令行参数输入一个表达式。

输出：（1）表达式正确的情况下输出结果；

（2）命令行参数不正确输出ERROR\_01；

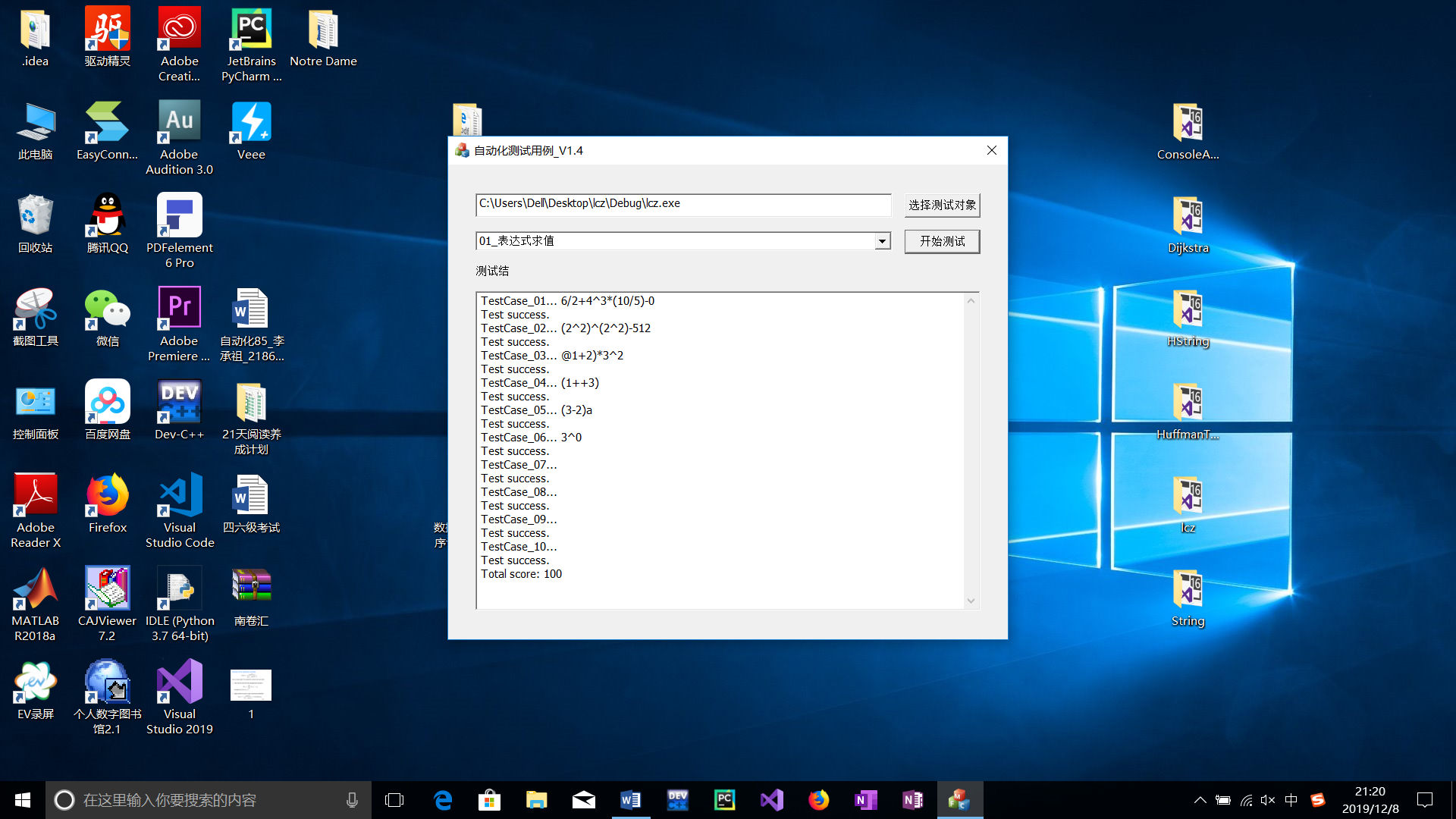
（3）表达式存在格式错误输出ERROR\_02；

（4）表达式在运算中出现逻辑错误输出ERROR\_03。

三、设计思路

主要设计思路为首先确定输入表达式是否合法，再进行计算。具体实现方法可利用栈的后进先出特性，建立两个栈，一个存储运算符，一个存储数字。根据运算符优先级的表，当当前运算符优先级低于栈顶运算符时入栈，等于（遇到括号）或高于时出栈，同时对于数字栈进行对应操作，即可完成运算操作。

四、运行结果



五、问题与思考

对于输入中的运算符，需要将所有可能提前考虑进来，并对其优先级进行排列。对于表达式中可能出现的表达式参数使用错误，如除数为0等，需要进行判别。同时，对于运算符使用错误，尤其是括号是否匹配、两个运算符是否可以连用等需要进行判断。

六、源码



**实验二基于串的模式匹配**

一、实验目的

熟练掌握抽象数据类型串的实现，学会使用串及其模式匹配算法解决具体应用问题，从而体会串的特点。

二、实验要求

基本要求：是现在HTML文件内容中的特定字段查找。

1、用KMP算法实现模式匹配；

输入：通过命令行参数输入字符串和关键字。

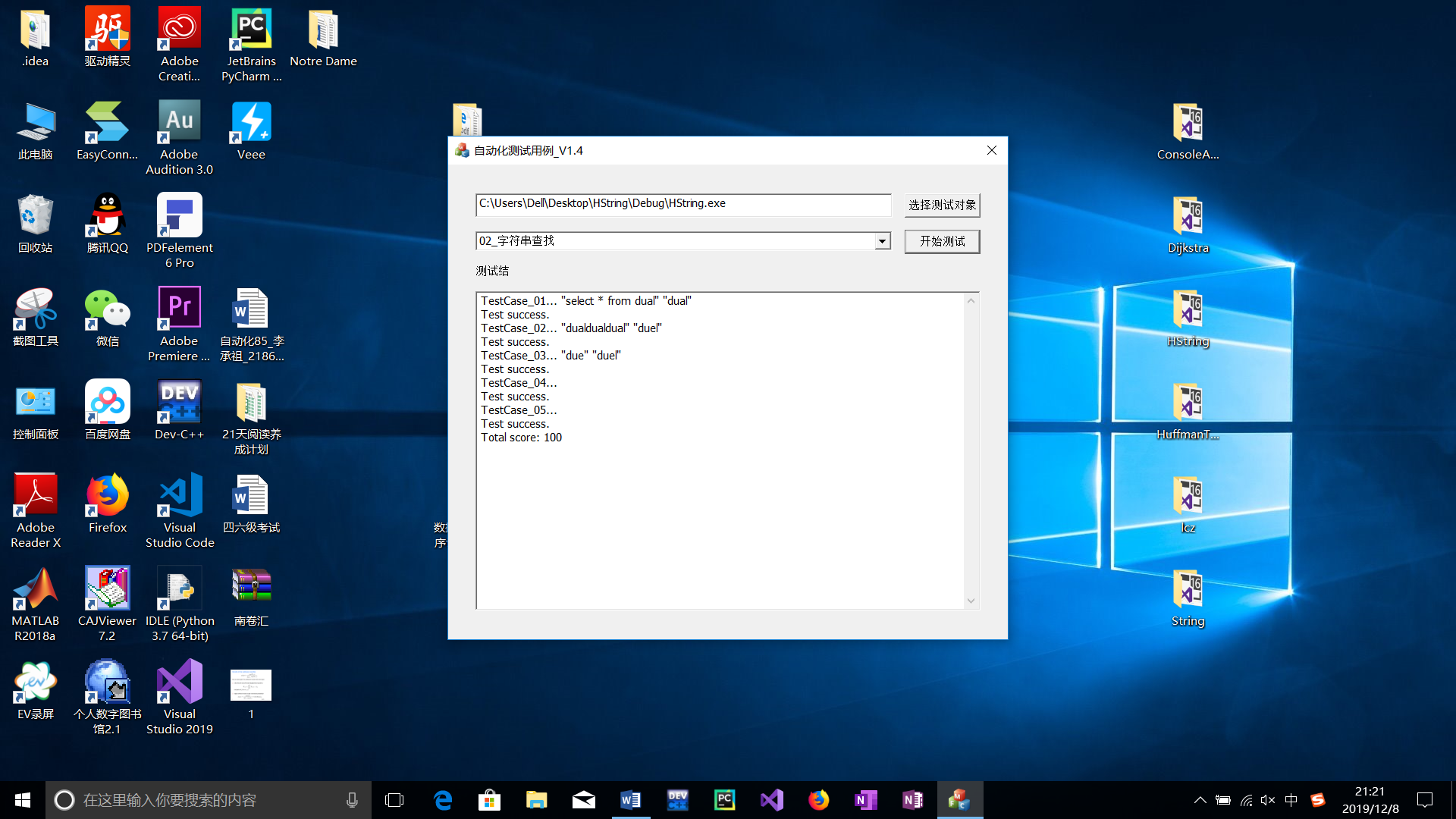
输出：（1）关键字在字符串中的位置，如果不存在则输出-1；

（2）命令行参数不正确则输出ERROR\_01；

三、设计思路

为了减小时间复杂度，考虑到子串中可能出现重复序列，因此使用串的模式匹配算法，即引入next数组标记如果某元素失配，该位置应与子串中重新匹配的对应位置，最终基于next数组将模式串与子串进行匹配。

四、运行结果



五、问题与思考

对于模式匹配算法，要求子串与模式串长度满足子串长度小于模式串长度，否则返回ERROR\_01。Next数组同样还有一种改进方法，即引入nextval数组，它在next数组基础上，考虑到某点失配，若next值对应字符相同则同样不可能匹配，因此它将next数组进行改动，保证了nextval值对应字符与该处字符不相同，提高了匹配效率。

六、源码



**实验三基于霍夫曼树的编码/译码**

一、实验目的

掌握二叉树的生成、遍历等操作，及霍夫曼编码/译码的原理。

二、实验要求

基本要求：掌握树的操作。

1、基于该霍夫曼树，实现非递归的先序遍历算法，输出该树的所有节点、节点的权值、节点的度和节点所在的层数；

2、在实现时要求霍夫曼树的左右孩子的大小关系满足，左孩子节点权值小于右孩子的节点权值。左右孩子权值相等的情况下，按ASCII码表中的编号，编号小的在左孩子，编号大的在右孩子。

输入：通过命令行参数输入字符串（长度>=20）和码字。

输出：（1）命令行参数不正确输出ERROR\_01；

（2）编码失败输出ERROR\_02；

（3）译码失败输出ERROR\_03；

（4）在同一行中输出编码结果和译码结果，中间用空格隔开。

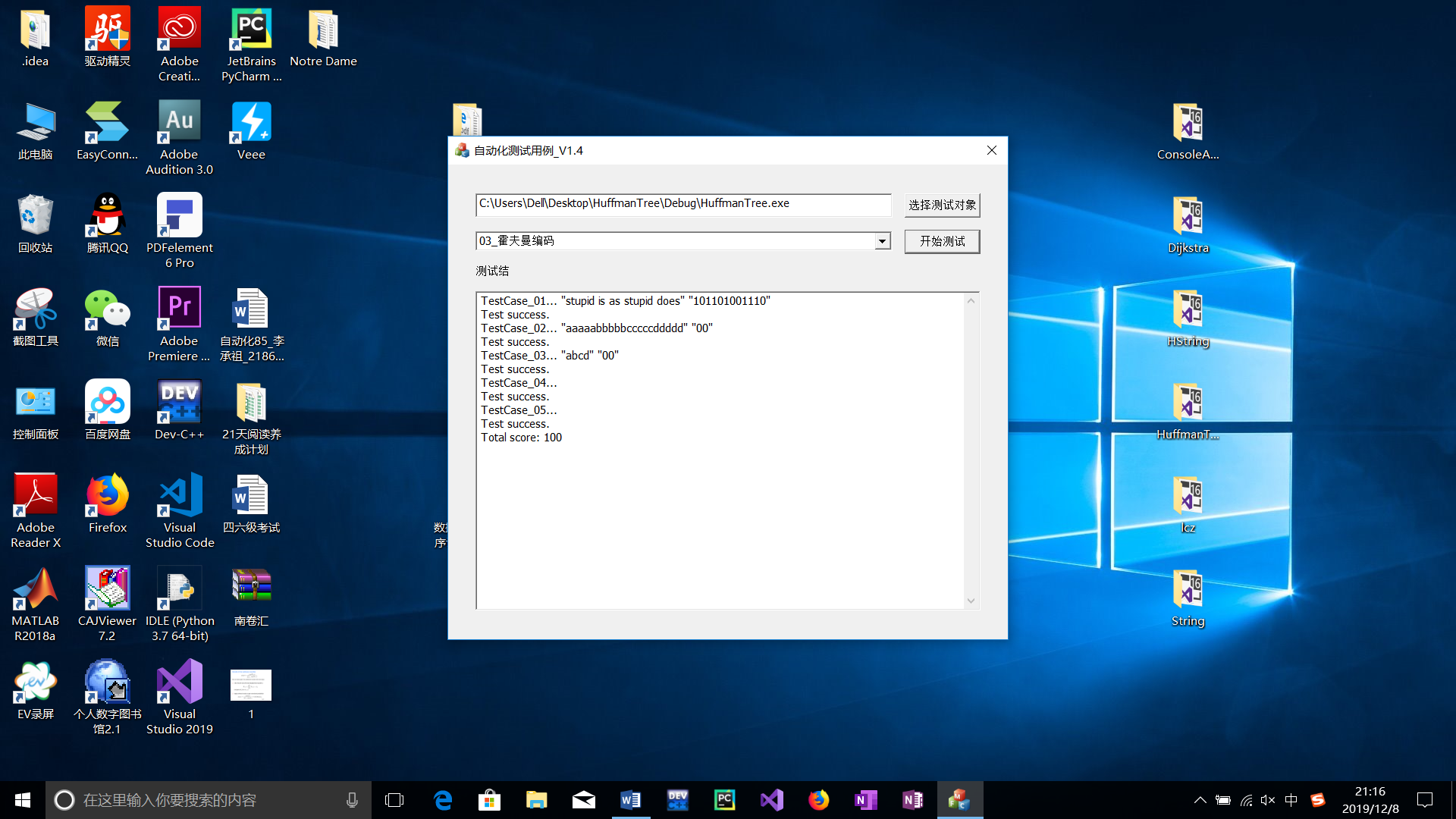
三、设计思路

首先统计输入中每个元素的对应权重，基于权重即可构建出Huffman树。具体构建方法，则为每次选出其中两个权值最小的Huffman树，组合构建成为一个新的Huffman树，其权值等于先前二者权值之和，并删去先前两个Huffman树。

Huffman树构建好后，从根结点开始确定对应字符的Huffman编码，规定向左为0向右为1。最终即可得到对应字符串的赫夫曼编码。

对于译码，则可以按照0向左子树1向右子树的规律寻找，知道找到赫夫曼树的叶子结点，即可确定对应字符，完成译码任务。

四、运行结果



五、问题与思考

对于译码失败，可能原因为当给出的待译串已经结束但是此时并未处于Huffman树叶子节点处，无法翻译出对应字符。

对于先序遍历，可以利用栈的后进先出性质，在节点的左右子树遍历过或不存在时pop，在存在且未遍历时push，按照根-左-右顺序即可。

六、源码



**实验四无向图最短路径搜索**

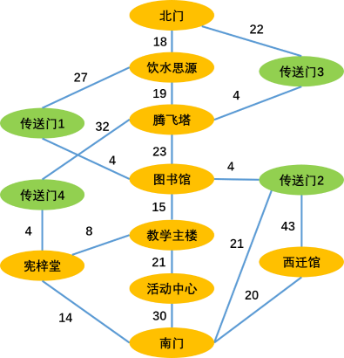
一、实验目的

掌握Dijsktra算法的原理。

二、实验要求

基本要求：熟练掌握图的操作。

1、针对所给图进行构建数据结构和算法，通过命令行参数输入任意两建筑物名称，可查询建筑物间的最短路径长度，并输出最短路径。

注：

1）右图为校内知名建筑物示意平面图（其中传送门用于增加网络复杂度），以边表示建筑物间的路径，各条路径上方的数字表示路径长度。

输入：通过命令行参数输入起点和终点的位置名称。

输出：（1）命令行参数不正确输出ERROR\_01；

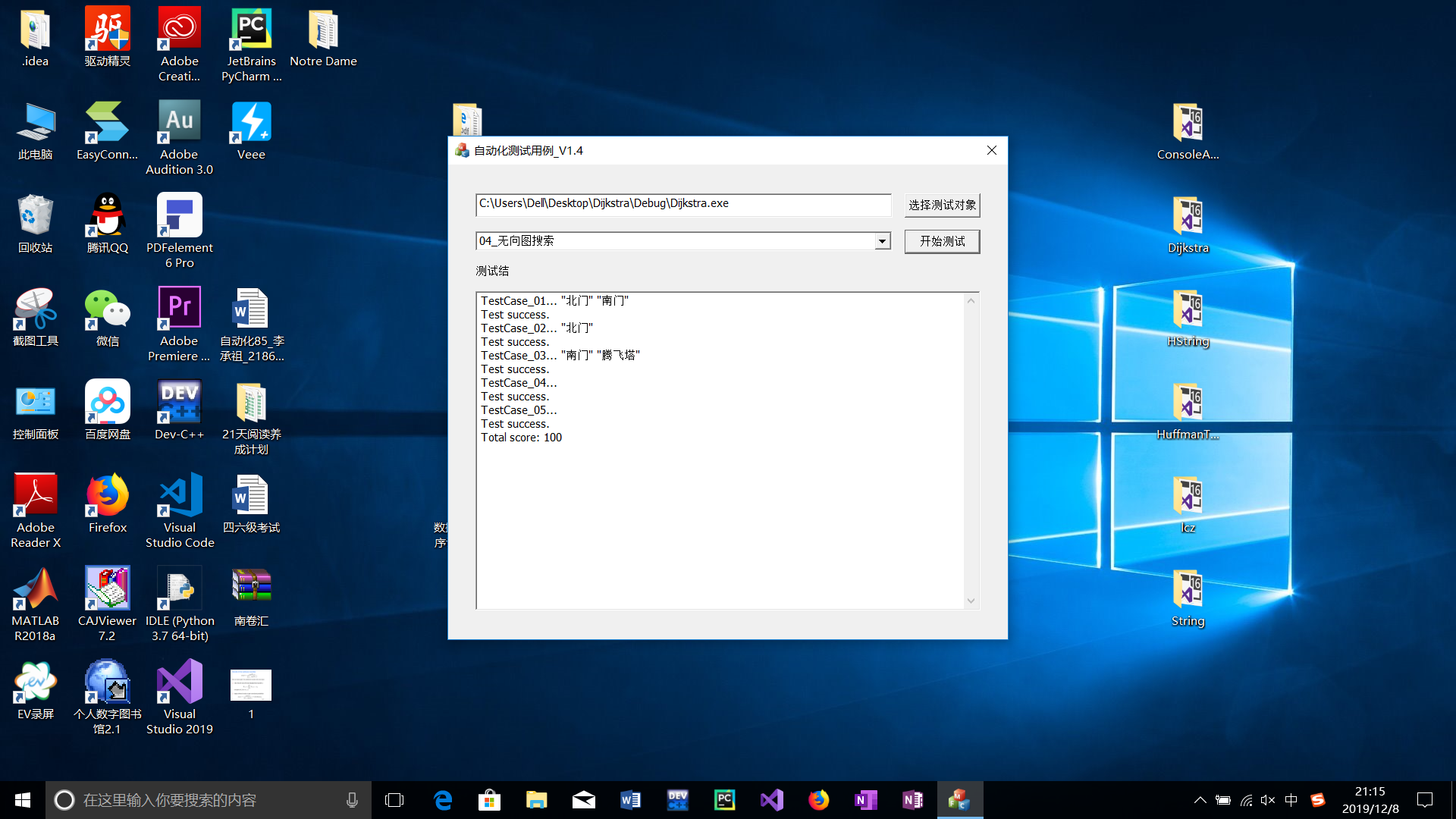
（2）获取最短路径失败时输出ERROR\_02；

（3）获取最短路径成功时输出路径长度。

三、设计思路

在输入图中所有节点以及路径权重后，利用Dijsktra算法，对其进行改进，构建一个D的二维数组记录每个节点路径上的前一节点，构建一个final的向量记录最短路径是否被找到，以及一个path二维向量记录每一个节点的路径上的节点顺序。当对于path二维向量进行处理时，可以借鉴原Dijsktra算法中的想法，每次更新路径时将上一节点的原路径全部覆盖到该节点路径上再加上该节点本身。

四、运行结果



五、问题与思考

对于原Dijsktra算法，其P矩阵并不能表示最短路径上节点的先后顺序，只能表示最短路径上是否有该节点。因此对此在本实验中进行了改进，引入path矩阵，相较于P矩阵修改对应位置处的值，path矩阵直接在每行向量后添加节点，以达到记录先后次序的目的。

六、源码

