上 海 大 学

2019-2020 冬季学期

《数据结构（2）》课程考核报告

学 号：18120189

姓 名：林艺珺

教 师：沈俊

课程考核评分表

序号	内容	分值	成绩
1	项目的设计	25	
2	项目的实现	25	
3	项目的测试	25	
4	报告的规范	25	
考核成绩			

计算机工程与科学学院

2020 年 5 月

《数据结构(2)》课程考核报告

一、题目 1：远离新冠病毒

1．主要数据结构

a. 数据文件格式

根据题目给出的输入样例，将其按格式写入.txt 文件中。

main.cpp 1.txt

```
7 9
1 2 50
1 3 60
2 4 120
2 5 90
3 6 50
4 6 80
4 7 70
5 7 40
6 7 140
2 7
-1 -1
```

图 1-1 输入信息

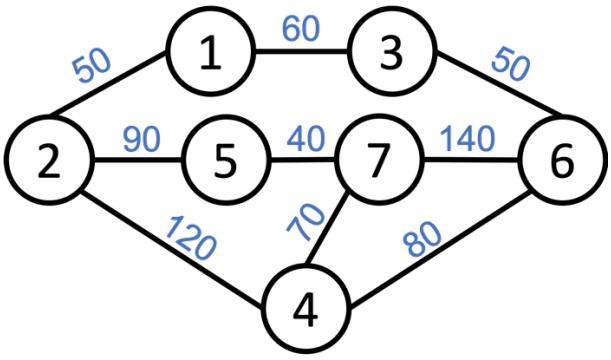

图 1-2 以图 1-1 为例的无向网

b. 数据结构设计

在 main 函数中共声明了以下几种变量。

代码 1. 1

```
#define ques 1000000
int vexNum, info;          //结点数、边信息条数
int arcs[100][100];        //邻接矩阵
int dist[100][100];        //经过变形 Floyd 算法后的信息矩阵
string path[100][100];     //选择路径的具体路径
```

2．主要算法设计

a. 问题分析

本题是传统 Floyd 算法的变形应用。传统 Floyd-Warshall 算法又称为插点法，是一种利用动态规划的思想寻找给定的加权图中多源点之间最短路径的算法，与 Dijkstra 算法类似。根据题目要求，将 Floyd 算法中的关键步骤：（其中初始 dist[i][j]的值为 i 与 j 之间的路径，最终 dist[i][j]的值为 i 与 j 之间的最短路径）

代码 1.2.1

```
for (int k = 0; k < vexNum; k++)
    for (int i = 0; i < vexNum; i++)
        for (int j = 0; j < vexNum; j++)
            if (dist[i][k] + dist[k][j] < dist[i][j] && dist[i][k] < infi
                && dist[i][k] > 0 && dist[k][j] < infi && dist[j][k] > 0)
            {
                dist[i][j] = dist[i][k] + dist[k][j];
                path[i][j] = path[k][j];
            }
```

更改为：（其中初始 dist[i][j]的值为 i 与 j 之间的路径，最终 dist[i][j]的值为 i 与 j 之间所有路段最小安全

值最大的路径的安全值）其中 min()函数及 max()函数为 algorithm 库的库函数。

代码 1.2.2

```
    for (int k = 0; k < vexNum; k++)
        for (int i = 0; i < vexNum; i++)
            for (int j = 0; j < vexNum; j++) {
                int ori = dist[i][j];
                dist[i][j] = max(dist[i][j], min(dist[i][k], dist[k][j]));
                dist[j][i] = dist[i][j];
```

为实现加分功能，在第三级 **for** 循环内添加代码 1.2.3，使得如果两个点之间有多条路径的最小安全值的最大值相等，则输出其中路径上道路条数最少的一条。如果道路条数也一样，则输出终点的前驱结点序号相对小的路径。

代码 1.2.3

```
if (i != j && i != k && k != j) {
    if (dist[i][j] != -1 && ori < min(dist[i][k], dist[k][j])) {
        path[i][j] = path[i][k] + " " + path[k][j];
        path[j][i] = path[j][k] + " " + path[k][i];
    }
    else if (ori == min(dist[i][k], dist[k][j])) {
        string tempathi = path[i][k] + " " + path[k][j];
        string tempathj = path[j][k] + " " + path[k][i];
        if (tempathi.length() < path[i][j].length() ||
                    tempathj.length() < path[j][i].length()) {
            path[i][j] = tempathi;
            path[j][i] = tempathj;
        }
        else if (tempathi.length() == path[i][j].length() ||
                    tempathj.length() == path[j][i].length()) {
            if (path[i][j][path[i][j].length()-1]>tempathi[tempathi.length()-1] ||
            path[j][i][path[j][i].length() - 1] > tempathj[tempathj.length()-1]) {
                path[i][j] = tempathi;
                path[j][i] = tempathj;
            }
        }
    }
}
```

b. 函数功能设计

main 函数中主要功能模块	
初始化二维数组 arcs[][]	声明一个二维数组 arcs[100][100]并使用-1 初始化其值
对称化二维数组 arcs[][]	由于题目为一无向网，则使邻接矩阵对称并令对角线元素为零
初始化二维数组 dist[][]	将 arcs[][]赋值给 dist[][]
对 dist[][]使用变形 Floyd 算法	计算 i 与 j 之间所有路段最小安全值最大的路径安全值并覆盖原 dist[i][j]
对路径选择提出要求	合理解决多条路径的最小安全值的最大值相等、道路条数一样等情况
判断询问结果并输出	根据题目要求输出所选择路径的最小安全值和具体路径或”no path”

文件输入及程序终止	
读入控制台输入文件编号	带备注版程序功能，选择测试文件
判断文件是否在列	带备注版程序功能，输出测试文件描述
判断是否成功打开测试文件	若无法打开指定文件，输出“无法打开文件！”
读入首行信息	得到无向网的结点数、边信息条数
逐行读入结点信息	写入二维数组 arcs[][]
逐行读入询问信息	搜索二维数组 dist[][]
读入末行信息	判断“-1 -1”结束程序

3．测试过程

[输入数据]

输入文件中第一行有二个整数：n、m。n 表示图中的顶点数（在无向图中，我们将所有结点从 1 到 n 进行编号），m 表示图中的边数；接下来 m 行，每行用三个整数 a,b,c 描述一条连接结点 a 和 b 的边，以及 ab 路段上的安全值 c；再接下来是若干行的询问，每个询问输入两个整数 a 和 b，表示要询问的两个顶点，输入二个-1 表示询问结束。

[输出数据]

输出结果有若干行，每行表示一次询问的结果，如果 a 和 b 是连通的则输出所选择路径的最小安全值和具体路径（最小安全值与路径之间用冒号分隔，顶点编号之间用空格分隔），否则输出“no path”。

样例 1 题目样例 1

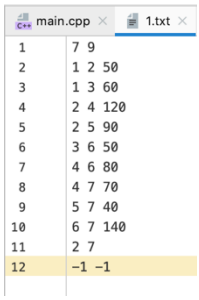

图 1-2-1(1) 远离新冠病毒样例 1 输入文件

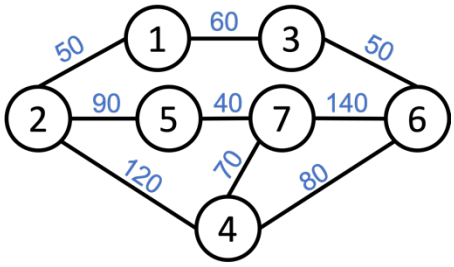

图 1-2-1(2) 远离新冠病毒样例 1 无向网

样例 1 测试结果

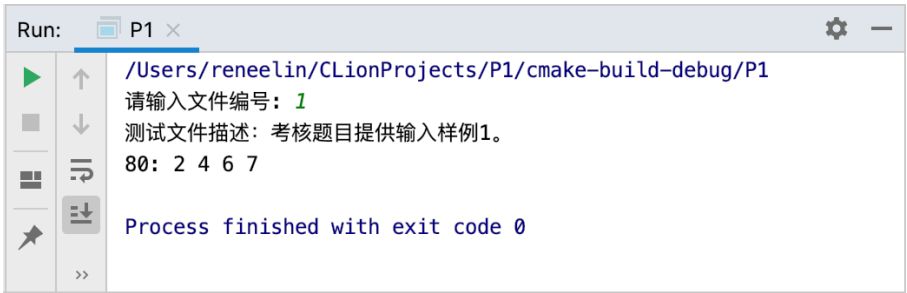

图 1-2-1(3) 远离新冠病毒样例 1 测试结果

样例 2 题目样例 2

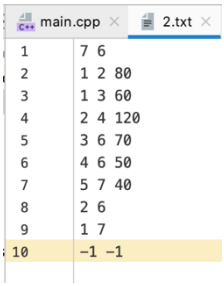

图 1-2-2(1) 远离新冠病毒样例 2 输入文件

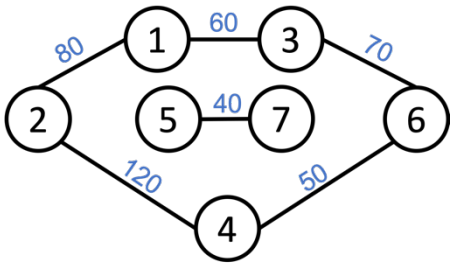

图 1-2-2(2) 远离新冠病毒样例 2 无向网

样例 2 测试结果

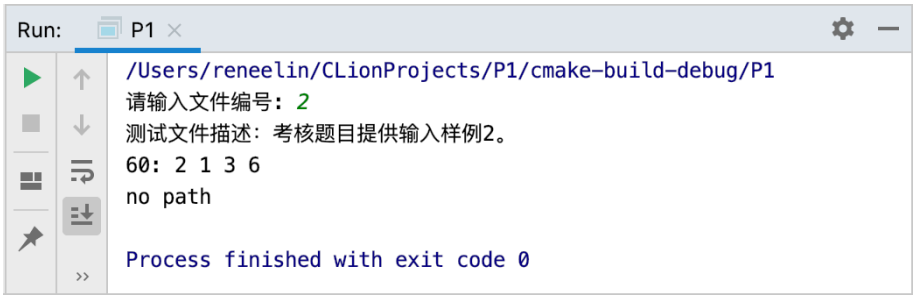

图 1-2-2(3) 远离新冠病毒样例 2 测试结果

样例 3 两点之间多条路径的最小安全值的最大值相等，输出其中路径上道路条数最少的一条。

根据图 1-2-3(2)无向网，本例中，2 号与 6 号点之间有两条路径的最小安全值的最大值相等，分别为：“40: 2 1 3 6”及“40: 2 5 6”。据题目中加分要求，输出其中路径上道路条数最少的一条“40: 2 5 6”。

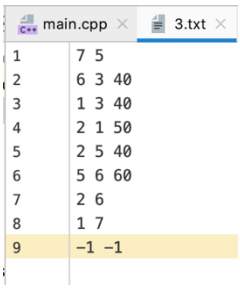

图 1-2-3(1) 远离新冠病毒样例 3 输入文件

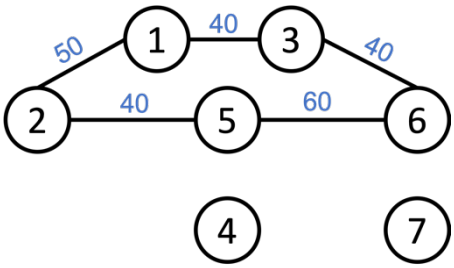

图 1-2-3(2) 远离新冠病毒样例 3 无向网

样例 3 测试结果

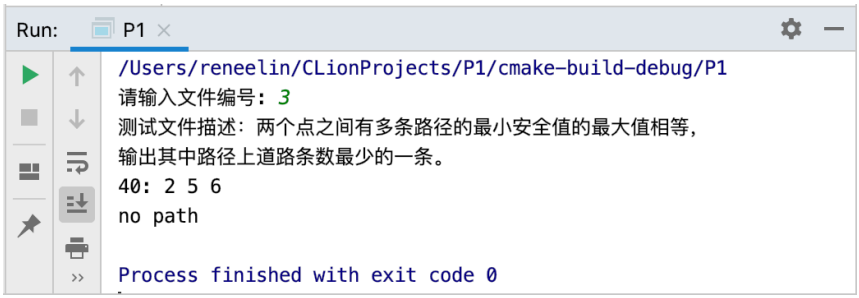

图 1-2-3(3) 远离新冠病毒样例 3 测试结果

样例 4 两点之间多条路径的最小安全值的最大值相等，道路条数一样，输出终点前驱结点序号相对小的路径。

根据图 1-2-4(2)无向网，本例中，2 号与 6 号点之间有两条路径的最小安全值的最大值相等，且道路条数一样，分别为：“40: 2 4 6”及“40: 2 5 6”。据题目中加分要求，输出终点前驱结点相对小的路径“40: 2 4 6”。

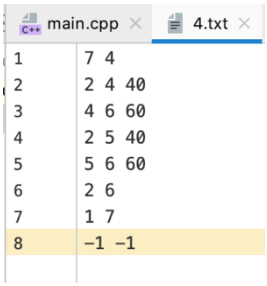

图 1-2-4(1) 远离新冠病毒样例 4 输入文件

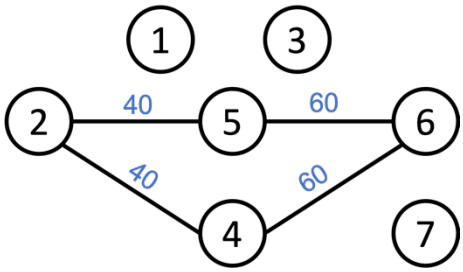

图 1-2-4(2) 远离新冠病毒样例 4 无向网

样例 4 测试结果

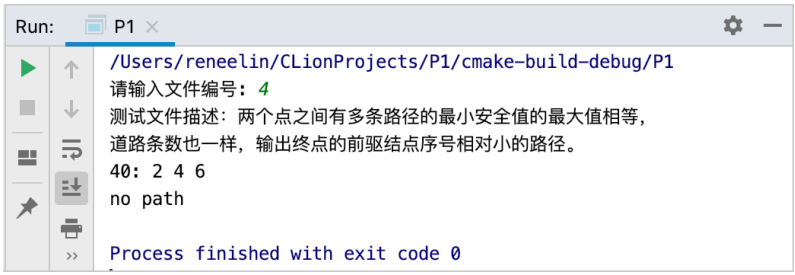

图 1-2-4(3) 远离新冠病毒样例 4 测试结果

样例 5 两个点之间有多条路径的最小安全值的最大值相等，共三条，其中有两条道路条数也一样。

根据图 1-2-5(2)无向网，本例中，2 号与 6 号点之间有三条路径的最小安全值的最大值相等，分别为：“40: 2 1 3 6”、“40: 2 4 6”及“40: 2 5 6”，且其中“40: 2 4 6”与“40: 2 5 6”道路条数一样。据题目中加分要求，输出其中路径上道路条数最少的一条且终点前驱结点相对小的路径“40: 2 4 6”。

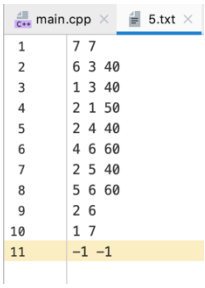

图 1-2-5(1) 远离新冠病毒样例 5 输入文件

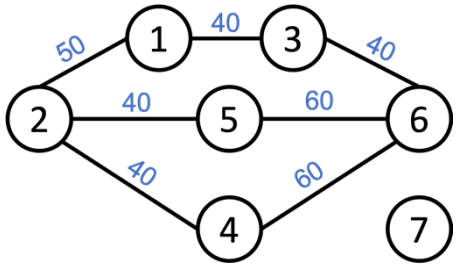

图 1-2-5(2) 远离新冠病毒样例 5 无向网

样例 5 测试结果

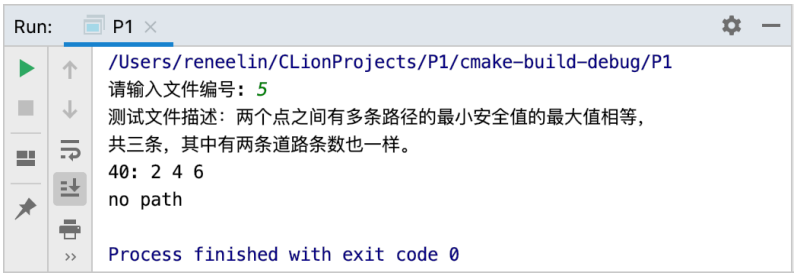

图 1-2-5(3) 远离新冠病毒样例 5 测试结果

样例 6 五结点无向网完全图

根据图 1-2-5(2)无向网，本例中，2 号与 6 号点之间有三条路径的最小安全值的最大值相等，分别为：“40: 2 1 3 6”、“40: 2 4 6”及“40: 2 5 6”，且其中“40: 2 4 6”与“40: 2 5 6”道路条数一样。据题目中加分要求，输出其中路径上道路条数最少的一条且终点前驱结点相对小的路径“40: 2 4 6”。

main.cpp | 6.txt

```
1   5 10
2   1 2 10
3   1 3 20
4   1 4 30
5   1 5 40
6   2 3 50
7   2 4 60
8   2 5 70
9   3 4 80
10  3 5 90
11  4 5 100
12  1 1
13  1 2
14  1 3
15  1 4
16  1 5
17  2 2
18  2 3
19  2 4
20  2 5
21  3 3
22  3 4
23  3 5
24  4 4
25  4 5
26  5 5
27  -1 -1
```

图 1-2-6(1) 远离新冠病毒样例 6 输入文件

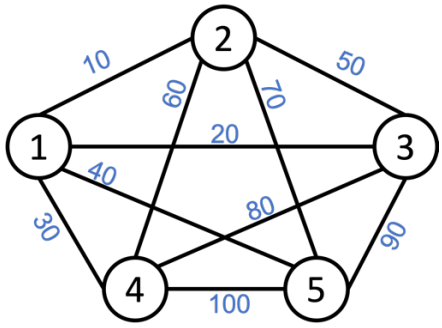

图 1-2-6(2) 远离新冠病毒样例 6 无向网

样例 6 测试结果

图 1-2-6(3) 远离新冠病毒样例 6 测试结果

违规样例

样例 7 不存在的文件

图 1-2-7 远离新冠病毒样例 7 测试结果

样例 8 空无向网或无边图(空文件结果相同)

main.cpp | 8.txt

```
1   0 0
```

图 1-2-8(1) 远离新冠病毒样例 8 输入文件

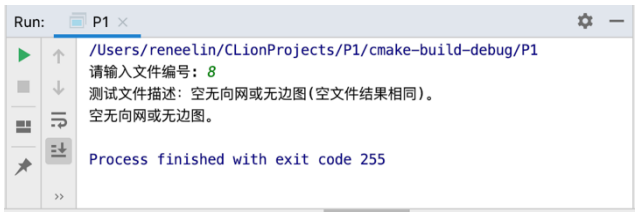

图 1-2-8(2) 远离新冠病毒样例 8 测试结果

样例 9 结点数大于 100

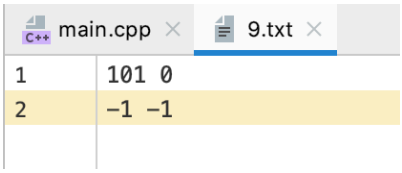

图 1-2-9(1) 远离新冠病毒样例 9 输入文件

图 1-2-9(2) 远离新冠病毒样例 9 测试结果

样例 10 边信息条数大于 1000

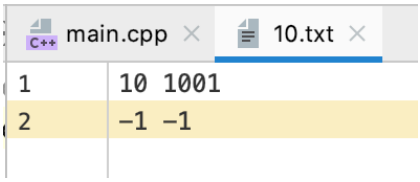

图 1-2-10(1) 远离新冠病毒样例 10 输入文件

图 1-2-10(2) 远离新冠病毒样例 10 测试结果

样例 11 输入不在列的文件编号

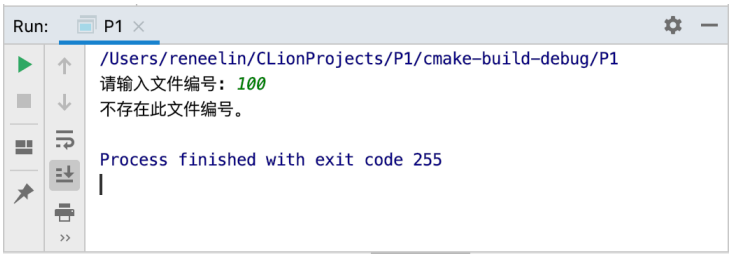

图 1-2-11 远离新冠病毒样例 11 测试结果

4．其它说明

为方便对测试样例进行查看，在测试版程序中设置了输入式菜单，在报告中展示的为典型实现验证样例 1-6 及违规样例 7-10 及输入式菜单独有的样例 11，程序中另外设置了一结点数为 100 的样例，文件编号为 11，实现情况良好。

所提供的测试文件编号为 1-6 及 8-11，缺失的 7.txt 为样例中不存在的文件测试。

随报告的代码文件包中另含有 OnlineJudge 纯享版程序，即可以在 OnlineJudge 平台中直接验证，不会输出无关的测试文件描述等，以题目样例一为例，输入文件后输出数据如题目中提供的结果。

加分功能的实现在样例 2、3、4 中体现，进行两个加分功能的单独测试以及融合测试。

二、题目 2：例句搜索

本题使用 Python 语言实现，开发环境为 Mac OS 下的 JetBrains Pycharm IDE。最终成品为 Python 命令行版本以及 Web 版（Django + Vue.js）。

1．主要数据结构

(1) 语料库

a. 数据文件格式

所有的语料均以 txt 格式存储，文件首行为语料出处（如资讯标题等），对文件标题并无要求。本次课程考核所收集的语料均为 Info 网站上近日的计算机前沿进展与新闻。

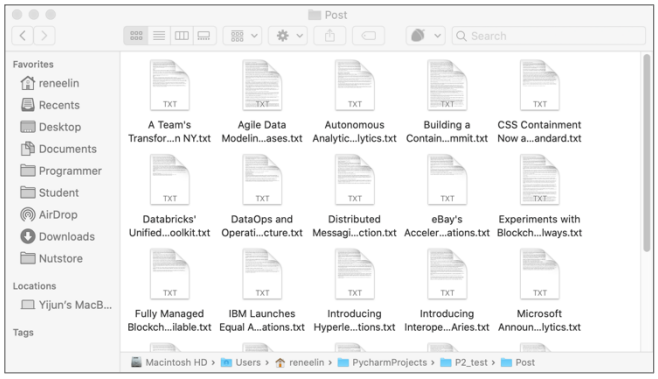

图 2-1-1(1) 语料库数据文件格式-1

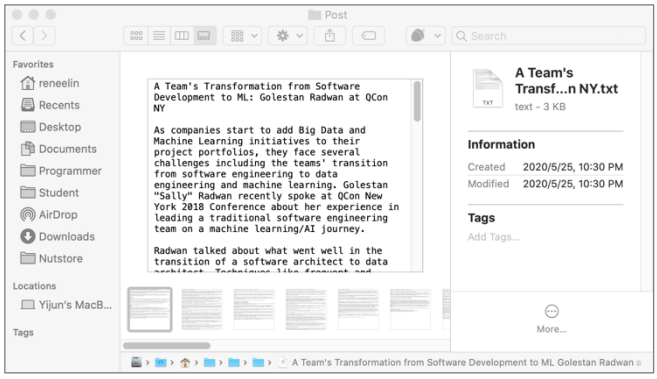

图 2-1-1(2) 语料库数据文件格式-2

b. 数据结构设计

txt 文件使用 Python 中的 open()函数以及 read()函数打开并读入为 Python 字符串类型。

(2) 例句库

a. 数据文件格式

经过语料处理后，所有被筛选的例句均被存入一 Python 列表（List）类型中，后存入 txt 文件备份。

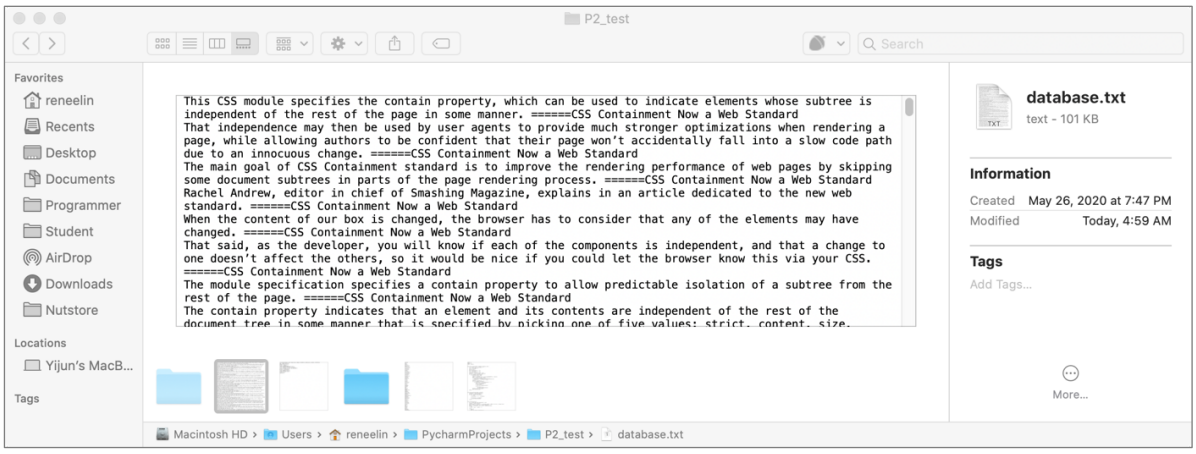

图 2-1-2(1) 例句库数据文件格式

b. 数据结构设计

在命令行版本中，列表中的每一项为一个 Python 字符串类型，字符串为例句本身及其出处的结合；而 Web 版本中，列表中的每一项为一个 Python 列表类型，包含两个字符串，分别为例句本身及其出处，在 Web 请求过程中，列表会转化为 Json 数据格式中的 Array，如图 2-1-2(2)所示，此为“blockchain”一词在语料库中的例句库。

代码 2.1.1　　Web 版本中例句库列表举例

```
example: [['这是一句语料库里的例句', '这是例句的出处(Info 的标题)'], ['这是一句语料库里的例句', '这是例句的出处(Info 的标题)']]
```

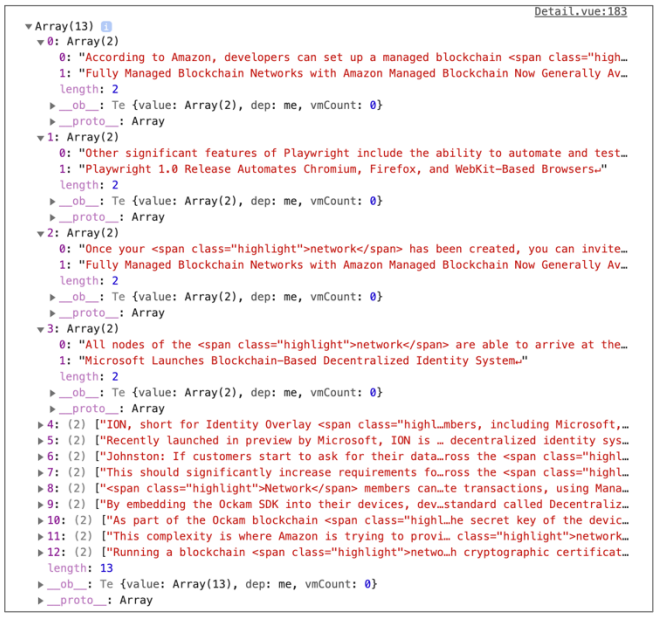

图 2-1-2(2)　例句库数据文件格式 Web 版 Json

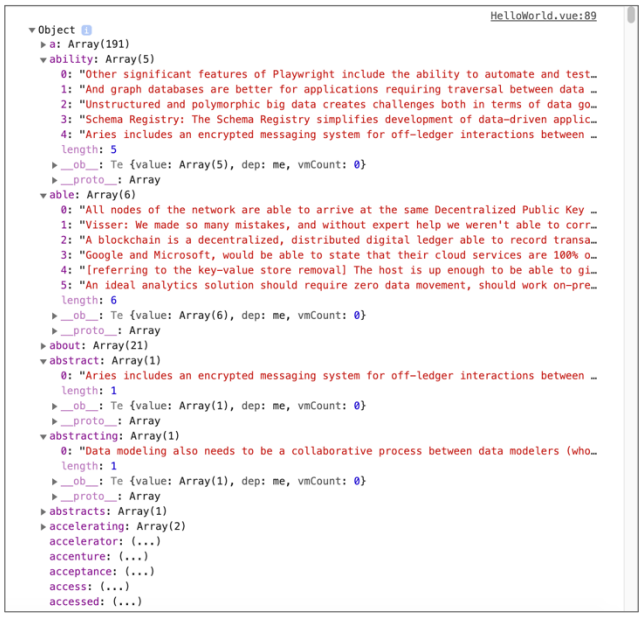

图 2-1-3　索引字典库数据文件格式 Web 版 Json

(3) 索引字典库

a. 数据文件格式

经过语料处理后，建立索引后的词句被存入一 Python 字典（Dictionary）类型中，字典类型更类似于 Json 数据格式，并未存入本地文件备份。

b. 数据结构设计

在命令行版本中，字典中的每一项，索引为单词（Python 字符串类型）；Web 版本中，采用与命令行版本一样的组织结构，在 Web 请求过程中，字典会转化为 Json 数据格式中的 Object，列表会转化为 Array，如图 2-1-3 所示，此为索引字典库的 Json 格式。

(4) 备份词库

a. 数据文件格式

经过语料处理后，所有被筛选的单词均被存入一 Python 列表（List）类型中，后存入 txt 文件备份。

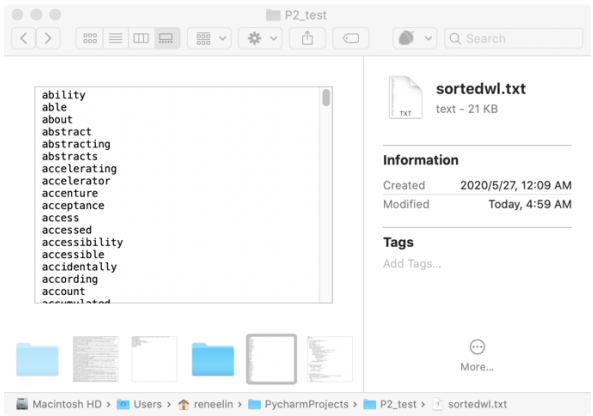

图 2-1-4　备份词库数据文件格式

b. 数据结构设计

在程序中，此备用词库 wordlist（List 数据类型）是用来检测在语料库中是否存在例句所用。

(5) 词库

a. 数据文件格式

在程序中，词库（即包含中文释义、词性、词形转换、同反义词等）并未存在本地，而是使用 Request 方法从金山词霸在线词典中获取（爬虫），词库并未储存在本地。

b. 数据结构设计

在命令行版本中，并未对爬虫获得的数据进行整合和组织；而 Web 版本中，涉及到对于 API 的 GET 与 POST 请求，因此封装为 Json 数据格式进行参数传递。如图 2-1-5 所示，此为“network”在程序中传递的相关参数。代码 2.1.2 为其中的索引字段及其默认值。Phonetic 为音标（List 数据类型，包含两个字符串类型），Meaning 为释义（List 数据类型，包含若干个字符串类型），Exchange 为词形转换（List 数据类型，包含若干个字符串类型），Synonym 为同义词（List 数据类型，包含若干个 List 类型，每个 List 又包含两个字符串类型），Antonym 为反义词（List 数据类型，包含若干个 List 类型，每个 List 又包含两个字符串类型）。

代码 2.1.2　　Web 版本中单词数据结构举例

```
word: '感谢使用',
phonetic: ['英[没有内容呢]', '美[也没有内容呢]'],
meaning: ['啊噢~ 没有你要找的单词呢~是不是输错啦~Σ(ο゜д゜ο/)'],
exchange: ['啊噢~', '没有你要找的单词呢~', '是不是输错啦~', 'Σ(ο゜д゜ο/)'],
synonym: [['啊噢~没有你要找的单词呢~', '是不是输错啦~Σ(ο゜д゜ο/)']],
antonym: [['啊噢~没有你要找的单词呢~', '是不是输错啦~Σ(ο゜д゜ο/)']],
// 对应参数
phonetic: ['英[英音音标]', '美[美音音标]'],
meaning: ['词性 该词性释义'],
exchange: ['转换形式 1', '转换形式 2', '转换形式 3', '转换形式 4'],
synonym: [['单词词性/类型', '同义词']],
antonym: [['单词词性/类型', '反义词']],
```

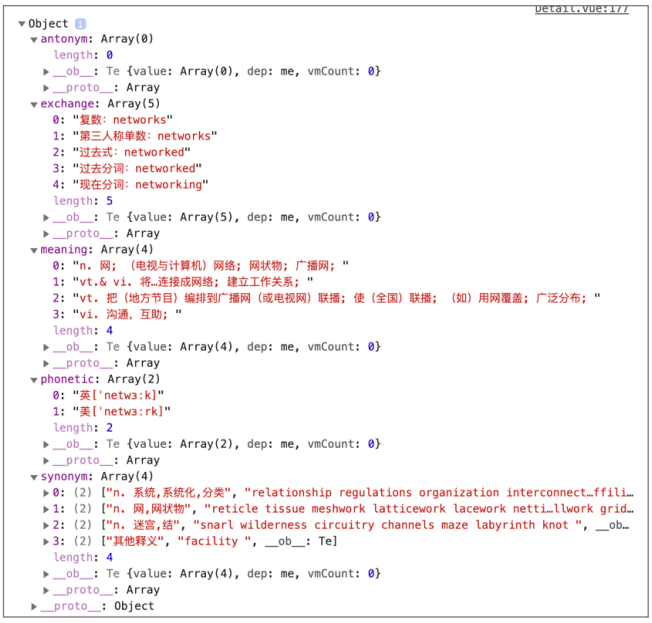

图 2-1-5 单词 network 数据文件格式 Web 版 Json

2．主要算法设计

main 函数中主要功能模块（命令行版本及 Web 版本基本通用）	
process_post(filename, database, wordlist, dict)	处理语料：对语料进行清理、分词、建立索引、生成词典等。
process_word(unsortedwl, wordlist)	转存备份词库：用于检测在语料库中是否存在例句所用。
word_meaning(word)	支持解释单词中文意思和查询音标、词形变化。
word_synonym(word)	支持同义词、反义词信息查询。
Web 版本专享接口（配置在 backend/views.py 中）	
api_init(request)	初始化例句库接口：处理语料，转存备份词库。
api_input(request)	输入传值接口。
api_dict(request)	调用查询接口：支持解释单词中文意思，支持相关信息查询。
api_resplit(request)	再处理例句接口：高亮查询单词，分离例句与出处。

(1) 通用功能函数

此处通用功能函数指命令行版本与 Web 版本皆有的功能实现性函数，根据课程考核基本要求与加分功能进行阐述。若命令行版本与 Web 版本有出入，将会备注此处引用代码版本。

代码 2.2.1　准备语料：寻找一些英文文章，例如托福、GRE 文章、英文小说、英语新闻等。

处理语料：对准备好的语料进行清理、分词、建立索引、生成词典等。

re 模块：Python 自 1.5 版本起增加了 re 模块，它提供 Perl 风格的正则表达式模式。

re.split()：split 方法按照能够匹配的子串将字符串分割后返回列表。

filter()函数：用于过滤序列，过滤掉不符合条件的元素，返回由符合条件元素组成的新列表。

```
def process_post(filename, database, wordlist, dict):
    file = open("./backend/Post/" + filename, "r") # 读入语料库文件夹
    title = file.readline()                         # 读取首行标题
    raw = file.read()                               # 读取正文文本
    raw = raw.replace(". ", ".")                    # 预处理语句结束符
    raw = raw.replace("? ", "?")
    raw = raw.replace("! ", "!")
    sentences = re.split('[.?!\n]', raw)           # 使用正则表达式分词进行语料分句
    sentences = list(filter(None, sentences))      # 过滤空白分句结果，并转存为 list 类型
    for s in range(0, len(sentences)):              # 进入初步分词及建立索引
        if len(sentences[s]) > 100:                 # 过滤过短短句及不当分句
            sentence = sentences[s]
            # 使用正则表达式进行例句分词
            words = re.split('[ ,;:\'’`()\[\]{}/1234567890"“@&]', sentence)
            words = list(filter(None, words))       # 过滤空白分句结果，并转存为 list 类型
            if sentences[s][len(sentences[s])-1] == ':':
                sentences[s] = sentences[s][0:len(sentences[s])-1]
            sentences[s] += '. ======' + title
            sentence = sentences[s]
            for w in range(0, len(words)):
                if words[w][0] != '-' and words[w][0] != '+' and \
                  words[w][0] != '$' and words[w][0] != '&' and \
```

```
                    words[w][0] != '%':             # 从首字符过滤非法词汇
                        word = str.lower(words[w])      # 转换为小写
                        wordlist.append(word)           # 添加进入 Wordlist
                        # 建立词句索引
                        if word in dict:                 # 如果该词已存在索引
                            dict[word].append(sentence) # 值列表增加例句
                        else:                            # 如果该词不存在索引
                            dict[word] = []              # 新建该词索引
                            dict[word].append(sentence)  # 值列表增加例句
                database.write(sentences[s])            # 所有例句写入 database.txt 例句库
        return
```

代码 2.2.2　再次处理语料：剔除重复例句。d 为代码 2.2.1 建立的 dict 索引，wl 为代码 2.2.1 建立的 wordlist。

set()函数：创建一个无序不重复元素集，可进行关系测试，删除重复数据，计算交集、差集等。

```
for w in wl:
    d[w] = list(set(d[w]))
```

代码 2.2.3　转存备份词库。

```
def process_word(unsortedwl, wordlist):
    unsortedwl = list(set(unsortedwl))              # 读取先前建立词库
    sortedwl = sorted(unsortedwl)                   # 按字母排序
    for w in range(0, len(sortedwl)):
        if w != 0:
            writeword = '\n' + sortedwl[w]
        else:
            writeword = sortedwl[w]
        wordlist.write(writeword)                   # 转存备份
    return
```

代码 2.2.4　根据索引进行查询：输入单词，返回相应的例句，要求支持重复查询。（命令行版）

```
var = 1
MIN = 5
while var == 1:                         # 支持重复查询
    print('欢迎来到计算机英语句句搜。\n作者：林艺珺 18120189\n输入单词进行查询。输入">>>"退出。\n')
    target = input()                    # 输入单词
    if target in d:
        word_meaning(target)            # 调用爬取释义等函数
        print('例句')
        if len(d[target]) < MIN:  # 最多输出 MIN 条例句
            for i in range(0, len(d[target])):
                print(d[target][i])
        else:
            for i in range(0, MIN):
                print(d[target][i])
```

代码 2.2.5　支持解释单词的意思。中文解释。（Web 版，带返回值）

支持一些相关信息查询。如同义词、反义词、词形变化等。

requests 模块：使用 Apache2 licensed 许可证的 HTTP 库。

re.findall()：findall 方法在字符串中找到正则表达式所匹配的所有子串，并返回一个列表。

replace()函数：把字符串中的旧字符串替换成新字符串，第三个参数 max 即替换不超过 max 次。

```
def word_meaning(word):
    url = "http://www.iciba.com/word?w=" + word     # 金山词霸请求地址
    # 建立空单词索引字典
    detail = {'phonetic':[],'meaning':[],'exchange':[],'synonym':[],'antonym':[]}
    response = requests.get(url)                    # GET 请求返回 HTML
    # 筛选音标有效信息
    phoneticwhole    =    re.findall('<ul    class="Mean_symbols__5dQX7">(.*?)</ul>',
response.text)
    img = re.findall('<img (.*?)>', phoneticwhole[0])
    phonetic = re.findall('<li>(.*?)</li>', phoneticwhole[0])
    if img:
        img = "<img " + img[0] + ">"
    else:
        img = ""
    if phonetic:
        for i in range(0, len(phonetic)):
            # 处理冗余字符...此处省略 3 行
            detail['phonetic'].append(phonetic[i])   # 存入字典'phonetic'索引
    # 筛选释义有效信息.....省略 23 行，与上述类似
    # 筛选词形变化有效信息.....省略 9 行，与上述类似
    return detail

def word_synonym(word):
    url = "http://www.iciba.com/word?w=" + word     # 金山词霸请求地址
    response = requests.get(url)                  # GET 请求返回 HTML
    detail = {'synonym': [], 'antonym': []}         # 建立空同反义词索引字典
    # 筛选同义词有效信息
    synonymdiv = re.findall(
        '<li><div class="FoldBox_fold__1GZ_2"><h3 class="FoldBox_title__2YVcn">同义词
</h3><div                                          class="FoldBox_content__2q-bU"><div
class="Synonym_synonym__1Pp8x">(.*?)</div></div></div></li>',
        response.text)
    if synonymdiv:
        print("同义词")
        for i in range(0, len(synonymdiv)):
            synonymli  =  re.findall('<div  class="Synonym_mean__1vKzy">(.*?)</div>',
synonymdiv[i])
            for j in range(0, len(synonymli)):
                synoposmean = ''
                synopos = re.findall('<i>(.*?)</i>', synonymli[j])
                if synopos:
                    synopos[0] = synopos[0].replace("&amp;", "&")
                    synoposmean += synopos[0] + ' '
                synomean = re.findall('<span>(.*?)</span>', synonymli[j])
                for m in range(0, len(synomean)):
                    synoposmean += synomean[m]
                synowl = ''
                synonymwl = re.findall('<p>(.*?)</p>', synonymli[j])
                for k in range(0, len(synonymwl)):
                    synonymw = re.findall('href="/word\?w=(.*?)"', synonymwl[k])
                    for w in range(0, len(synonymw)):
                        synowl += synonymw[w] + ' '
                detail['synonym'].append([synoposmean, synowl]) # 存入字典'synonym'索引
    # 筛选反义词有效信息.....省略 26 行，与上述同义词近似
    if not (synonymdiv or antonymdiv):
        print('无同义词反义词。')
    return detail
```

代码 2.2.6　支持不断增加语料库：采用 Python os 模块，读取指定目录下文件列表并一一打开，因此只需要在文件夹内添加符合“1. 主要数据结构”中“(1)语料库”文件即可。（命令行版）

open()函数：用于打开一个文件，并返回文件对象，如果该文件无法被打开，会抛出 OSError。

os 模块：提供了非常丰富的方法用来处理文件和目录。

```
filelist = os.listdir("Post")   # 读取 Post 目录下文件列表
d = dict()                      # 新建空字典
wl = []                         # 新建空列表
db = open("database.txt", "w")  # 打开例句库文件
for fn in filelist:
    process_post(fn, db, wl, d)  # 处理语料
db.close()
sortedwl = open("sortedwl.txt", "w")
process_word(wl, sortedwl)       # 存储备份词库
sortedwl.close()
```

(2) Vue.js 调用 Django 函数 API 搭建 Web 版本应用

由于 Vue.js 以及 Django 不是本课程的教授内容，加上自己也是初学者，因此这一板块略写。

Django　Python 下一重量级 Web 框架，采用了 MVT 的软件设计模式，即 Model，View 和 Template。

Axios　是一个基于 Promise 的 HTTP 库，可以用在浏览器和 node.js 中。

Vue.js　是一套构建用户界面的渐进式框架；只关注视图层；API 实现响应的数据绑定和组合视图组件。

Element-UI　一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的桌面端组件库。

代码 2.2.7　预处理：**import** django 相关函数；解决前端 post 请求 csrf 问题；Vue.js 中 main.js 配置。

```
// views.py
from django.http import JsonResponse, HttpResponse
from django.views.decorators.csrf import csrf_exempt

// urls.py
from django.urls import path
from . import views
urlpatterns = [
    path('input/', views.api_input, name='api_input'),
    path('dict/', views.api_dict, name='api_dict'),
    path('init/', views.api_init, name='api_init'),
    path('resplit/', views.api_resplit, name='api_resplit'),
]
// main.js
import Vue from 'vue'
import App from './App.vue'
import ElementUI from '../node_modules/element-ui';    # import ElementUI 组件
import '../node_modules/element-ui/lib/theme-chalk/index.css';
import VueRouter from "vue-router";
import axios from 'axios'           # import axios
import qs from 'qs'                 # import qs 解决 axios 传参问题
Vue.use(VueRouter);                 # Vue 需要 use 才可生效（.....省略 qs,axios,UI 的 use）
# 路由配置
const router = new VueRouter({
  mode: "history",
  routes: [
    { path: "/", component: () => import("@/components/HelloWorld.vue") },
    { path: "/Detail", component: () => import("@/components/Detail.vue") },
  ]
});
```

```
// HelloWorld.vue <script>
this.$axios.get("/api/init/")
        .then(response => {
          this.d = response.data
        })
```

初始化 api_init 代码类似代码 2.2.6，以 JsonResponse 返回 d（字典类型）。

代码 2.2.8　api_dict 代码：查询单词释义等。

word_meaning 与 word_synonym 见代码 2.2.5。

```
// views.py
def api_dict(request):
    print('dict')
    w = list(request.POST)
    print(w)
    word = w[0]
    detail = word_meaning(word)
    detail2 = word_synonym(word)
    detail.update(detail2)
    return JsonResponse(detail)
// detail.vue <script>
this.$axios.post("/api/dict/", this.word)
                .then(response => {
                    this.phonetic = response.data['phonetic']
                    this.meaning = response.data['meaning']
                    this.exchange = response.data['exchange']
                    if (this.exchange == '') this.exchange = ['暂无词性转换']
                    // ......省略 synonym、antonym 共 4 行，与上两行相似
                })
```

代码 2.2.9　api_resplit 代码：再处理例句字段，添加例句中单词高亮显示，以 JsonResponse 返回 e（字典）。

word_meaning 与 word_synonym 见代码 2.2.5。

```
// views.py
def api_resplit(request):
    print('resplit')
    dns = dict(request.POST)
    e = []
    for i in range(0, len(dns) - 1):
        i = str(i)
        ds = dns['d[' + i + ']'][0].split("======")
        if len(dns['word'][0]) == 1:
            word = " " + dns['word'][0] + " "
            highlight = "<span class=\"highlight\">" + word + "</span>"
            ds[0] = ds[0].replace(word, highlight)
            # ......省略大写首字母等处理 4 行
        else:
            # ......基本同 if，省略
        e.append(ds)
    js = {'example': e}
    return JsonResponse(js)
// detail.vue <script>
this.$axios.post("/api/resplit/",    qs.stringify({d:    this.d[this.word],    word:
this.word}))
                .then(response => {
                    this.example = response.data['example']
                    if (this.example == '') this.example = [['语料库暂无收录该词例句']]
                })
```

3．测试过程

根据课程考核基本要求与加分功能进行测试演示。

输入一个英文单词，返回相应的英语例句。具体步骤及功能实现：

(1) 准备语料：寻找一些英文文章，例如托福、GRE 文章、英文小说、英语新闻等。

(2) 处理语料：对准备好的语料进行清理、分词、建立索引、生成词典等。

(3) 根据索引进行查询：输入单词，返回相应的例句，要求支持重复查询。

(4) 支持解释单词的意思。中英文解释都可以，需要自己建词库。

(5) 支持不断增加语料库。

(6) 支持一些相关信息查询。如同义词、反义词、词形变化、固定搭配等。

a. 命令行版本

命令行版本完全由 Python 开发，使用常用模块 re、os 以及 requests。根据索引进行查询，输入一个英文单词，返回相应的英语例句，支持重复查询，支持中文解释单词的意思，支持不断增加语料库（直接向 Post 目录，即语料库目录添加符合要求的 txt 文件并重新运行程序即可），支持一些相关信息查询（如同义词、反义词、词形变化等）。

图 2-3-1 单词 ability 命令行运行截图 + 注释

单词 blockchain 在测试时报错是因为它的释义内容并不含有词性，无法正常提取并输出，修改代码使得程序更加健壮后输出如图 2-3-2。

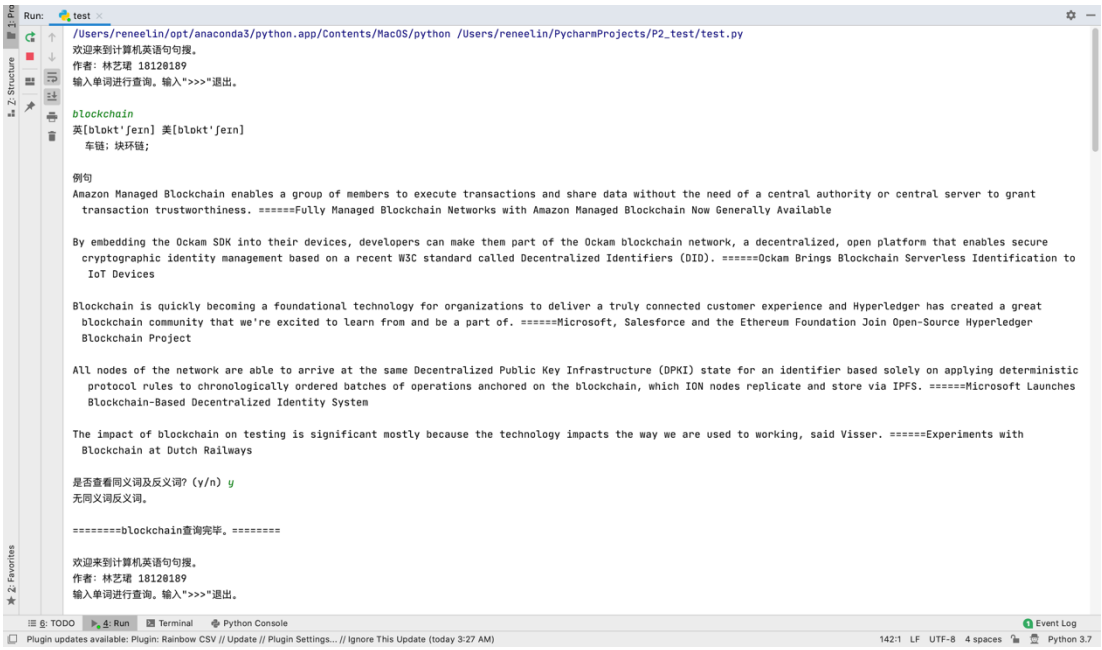

图 2-3-2 单词 blockchain 命令行运行截图

单词 contribute 是不存在反义词的例子。在命令行版本中，若同义词、反义词有一不存在，则该模块不输出，如图 2-3-3 所示，只输出了同义词。若同义词、反义词均不存在，则输出提示信息，详见图 2-3-2。

图 2-3-3 单词 contribute 命令行运行截图

图 2-3-4 及图 2-3-5 分别展示了命令行版本的程序在面对不在词库内的单词的输出以及手动输入“>>>”后退出程序的截图。相比较而言，命令行程序对非语料库整理出的词汇均输出“词库中不存在此单词例句”，而 Web 版本的程序除例句外，但凡单词存在并收录于金山词霸，都将显示基础内容，但不显示例句，详见图 2-3-6。

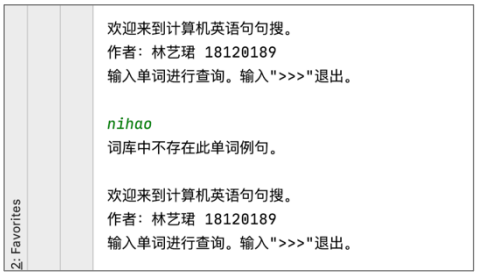

图 2-3-4 不在词库的单词 nihao 命令行运行截图

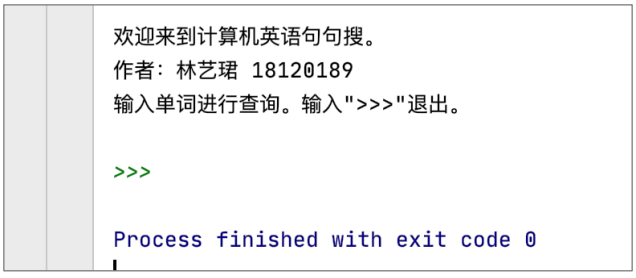

图 2-3-5 正常退出命令行运行截图

b. Web 版本

Web 版本主要功能与命令行版本一致，主要差异在更好的可视化、明晰的内容区分等。以下图 2-3-6 展示了不存在词汇的默认输出；图 2-3-7 展示了无反义词显示（无同义词、无词形变换同理）；图 2-3-8 为没有缺失项的显示；图 2-3-9 为多释义显示。

图 2-3-6 不存在的单词“数据结构 2”网页截图

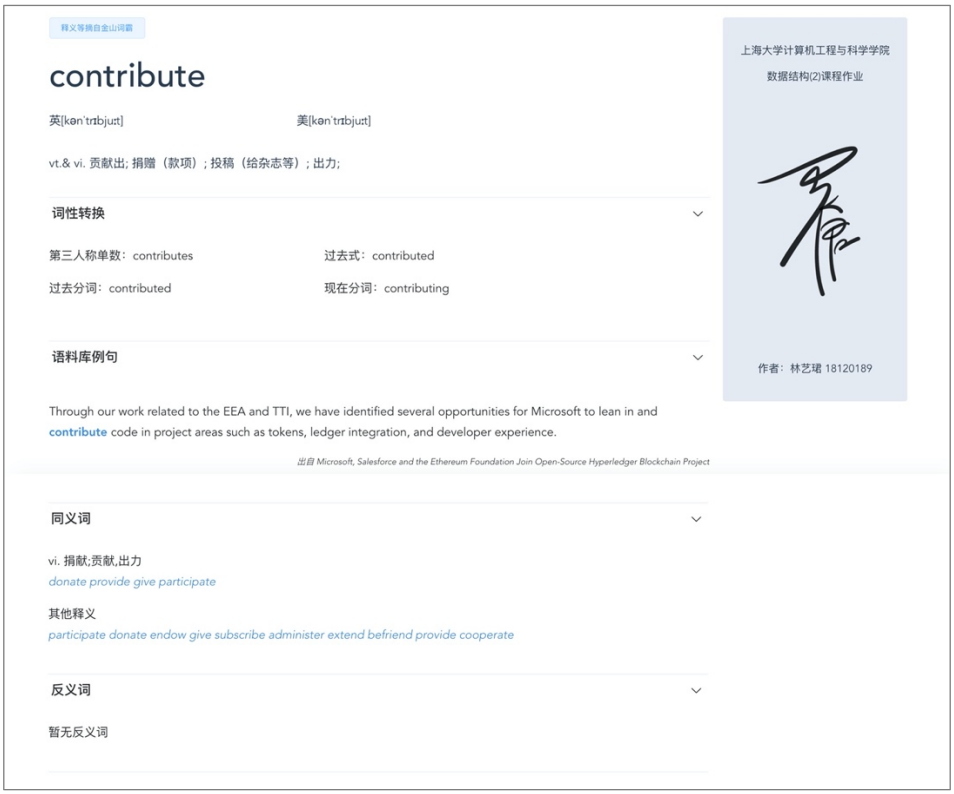

图 2-3-7 单词 contribute 网页截图

图 2-3-8 单词 able 网页截图

图 2-3-9 单词 network 网页截图

通过后端处理例句字段，返回带有高亮 HTML 的 Response，使得前端 Vue.js 处理时产生例句中词汇高亮的效果，更加明确清晰。

4．其它说明

关于 Web 版本程序改进。在多词条联动方面，理论上是可以实现的。以“love”一词为例，词条的 url 为：http://127.0.0.1:8000/Detail?input=love&d=%5Bobject%20Object%5D，即可以通过修改 url 来 redirect，实现多词条路由，可应用在词形变化和同义词反义词模块当中。但由于时间关系和先前组织数据结构时的遗留问题，导致开发此功能将逾期并影响到其他考试，因此暂且作罢，但之后会对此进行改进，因为个人对这个应用还是很感兴趣。在网页美化方面，第一次使用 Element-UI，先前其实也没有接触过 Vue.js，所以在这方面还有很大上升空间，值得提升并增强用户体验。另外，其实上周刚好购置了租赁的服务器，但由于先前从未接触过 CentOS 等技术，因此在尝试部署应用至服务器时，出现了一定的配置问题，碍于时间限制，没有十足的把握能够短时间解决这个问题，因此暂时没有外网链接可以供老师访问。但如果除去考核外，老师对此有兴趣，我会在我修整好后告知。

关于开发语言的选择。选择 Python，一是因为在数据处理方面，它的应用非常出色；二是因为相对来说开发时间会比 C++稍短一些，在考核的时间上更加有所把握。虽然我接触 Python 的时间比 C++足足晚了一年有余，但在 C++的基础上，对于 Python 的理解和上手还是相对迅速的。C++应当也可以完成此题的开发，在检索中也有了解到 Map，但是从 Python 的数据类型本身的优化来看，可能最终的效果是比完全构建的 C++会有一些数据优势，但也并不能完全确定，之后也会钻研一下 C++实现的可能性与可能用到的函数或数据类型等。

三、课程总结

1. 关于课程考核

选择 Vue.js+Django 进行开发其实是我先前没有尝试过的事情，这次课程考核时间挺紧张的，还有许多别的科目没有复习，但是确实给了我一个很大的动力去学习一直想学但是没有执行的事情。不过，虽然是勉勉强强成功了，但同样也是因为时间紧张，我认为自己的应用其实还存在不少潜在的问题。这次的课程考核我更加意识到计算机是一门工程学科，其实实践和应用是最后回馈于社会的最主要的途径，虽然课本的理论知识很重要，但是不实操其实是很难达成必要的效果的。

虽然说这次课程考核是特殊时期下的特殊产物，可能真的是前无古人后无来者了，但是我个人还是挺乐意做这样的课程项目来锻炼自己的能力的（当然最好是在没有其他大作业、大考试争抢时间的情况下）。对我来说，这次的课程考核是一次提升满足感的过程，虽然我算法方面包括熟练度，都远不如原本就已经掌握许多技术（比如 Java、数据库等）的同学，甚至我自己现在就可以想到有一些尚未解决和可以改进的问题。

这次的课程考核让我回想起大一秋季《高级语言程序设计》时的个人大作业，累并满足着。

2. 关于《数据结构(2)》

我想没有人会否认数据结构这门课在计算机专业中的重要程度。我还记得秋季学期为《数据结构(1)》考试成绩不佳而伤心许久。虽说成绩在短期内会成为衡量学习成果的唯一指标，但是数据结构应当是计算机专业学生一直记在心中并探索更优方法的一门学问。之前有听说过 LeetCode 偶尔会出现新题解，而这其实就是数据结构更优质的应用的实例，一道看似很简单的算法题，使用不同的数据结构可能导致完全不同的时间空间消耗，而在更大型的应用中，一点点时间空间消耗的巨量积累，最后会产生巨大的差异。而现在用户级别的产品也越来越注重用户体验，速度是必不可少的一环，而各类算法，比如排序、搜索等，优质的算法就能帮助应用或产品在市场中脱颖而出。其实数据结构远不是我先前所想的纯理论、不知应用，相反，它是一切应用的基石，也可以是跳板，它应用的模糊正是因为它所涉及的面太过广泛，以至于你难以用任何一个领域或者方向来概括。

数据结构是计算机人值得“终生学习”的课程，常学常新，当有了新的知识再回顾数据结构的时候，一定会有新的理解或见解。