数据结构化存储和查询

分布式计算作业一

1752124　朴 雪

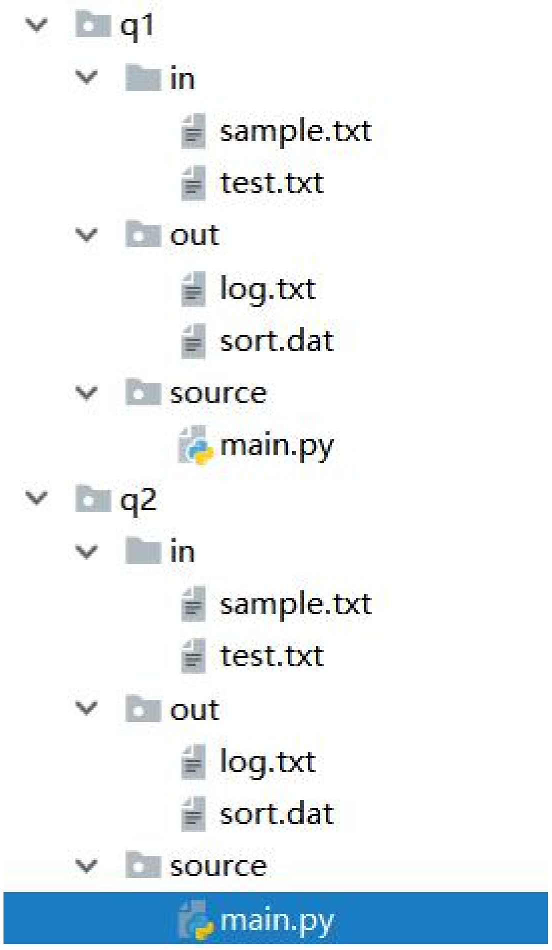
目录

1.开发环境平台

环境：win10
语言：python
开发 IDE：pycharm
作图工具：matplotlib

2.测试说明

项目目录如下：

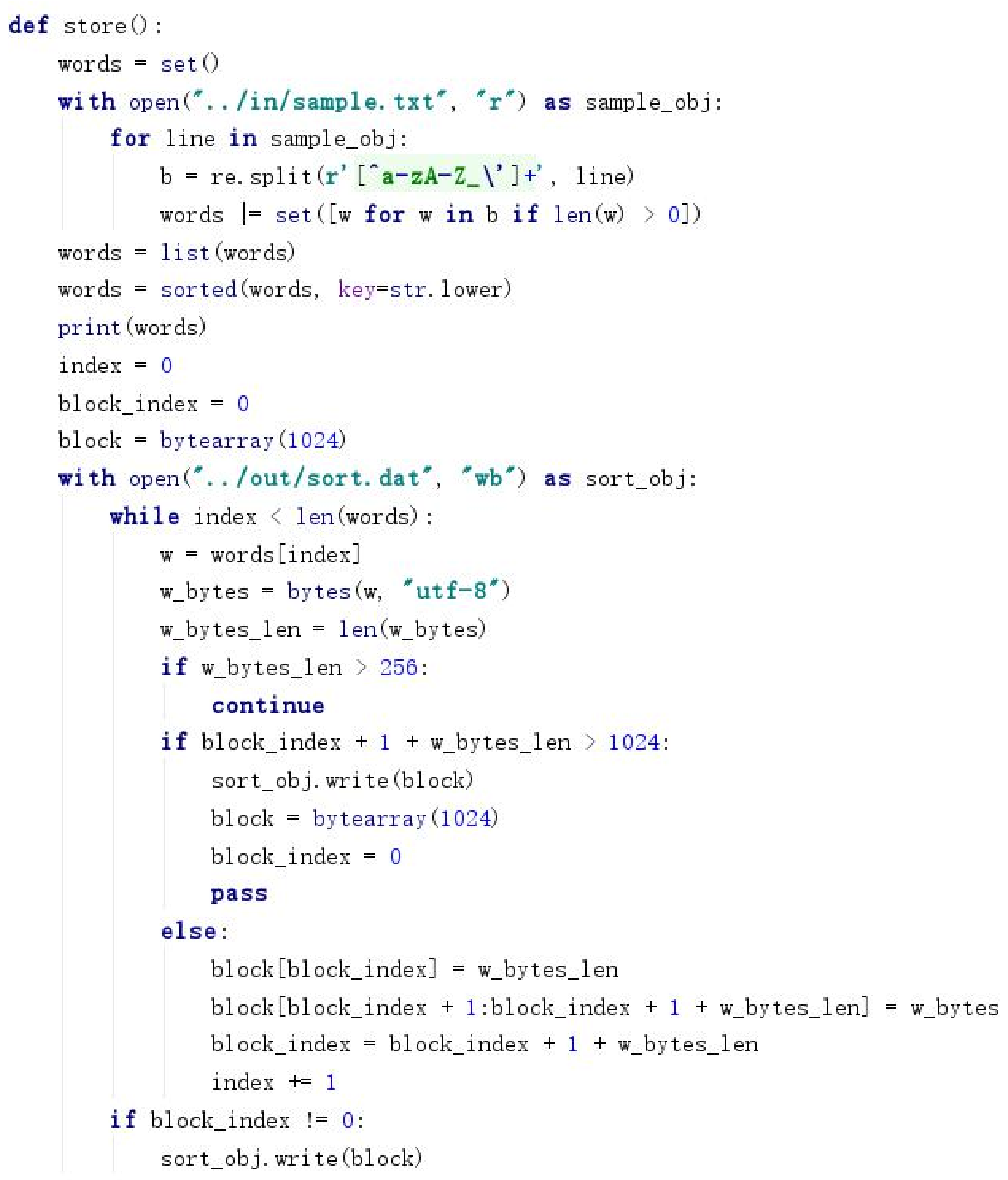


```
q1
    in
        sample.txt
        test.txt
    out
        log.txt
        sort.dat
    source
        main.py
q2
    in
        sample.txt
        test.txt
    out
        log.txt
        sort.dat
    source
        main.py
```

将 sample.txt 和 test.txt 输入到 in 文件夹之后，直接进入相应的 source 文件夹中执行 main.py 文件，即可看到控制台输出的情况以及 out 文件夹中输出的 log.txt 日志文件。（执行时间均为执行 1000 次所用时间，q2 执行 5*5 次任务）

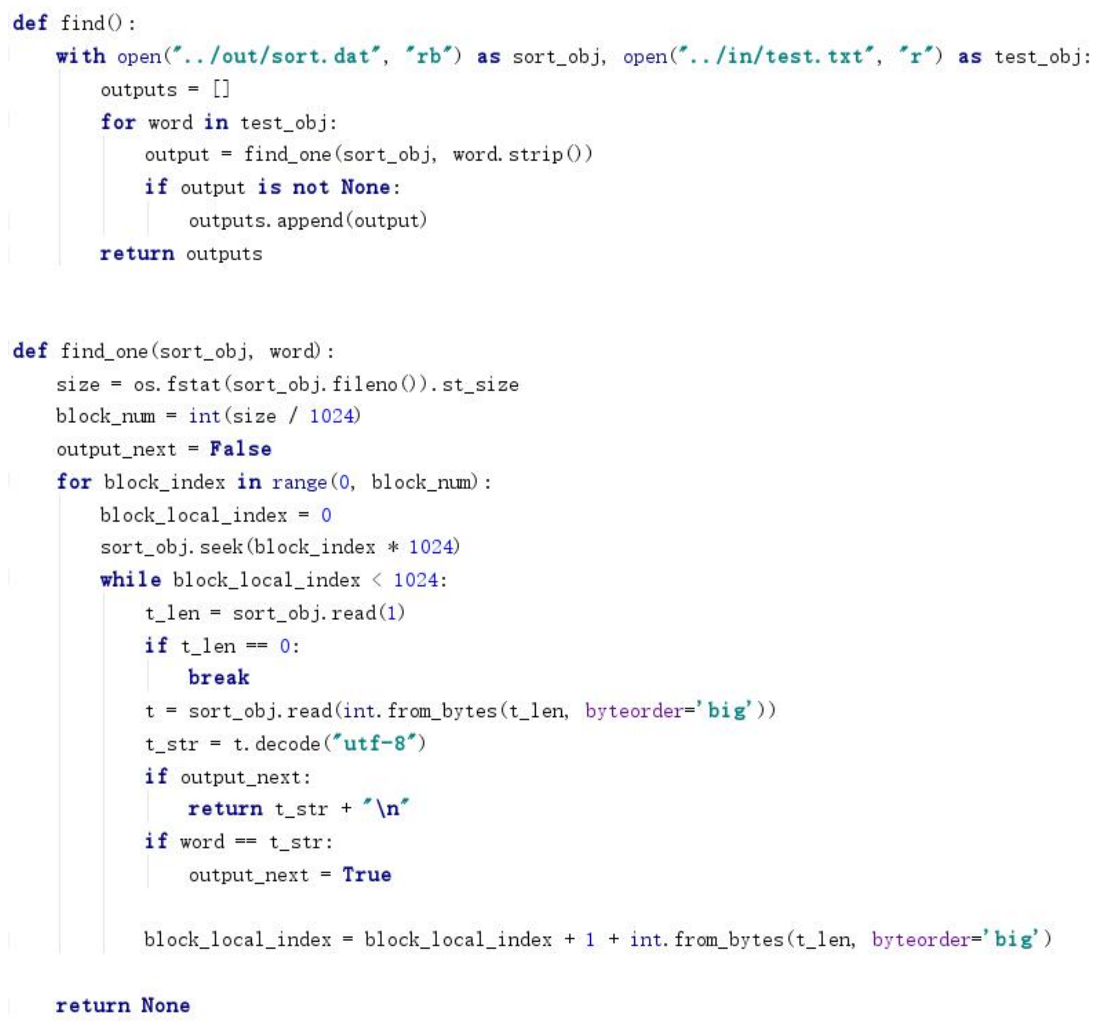
3.核心代码

3.1q1 存储



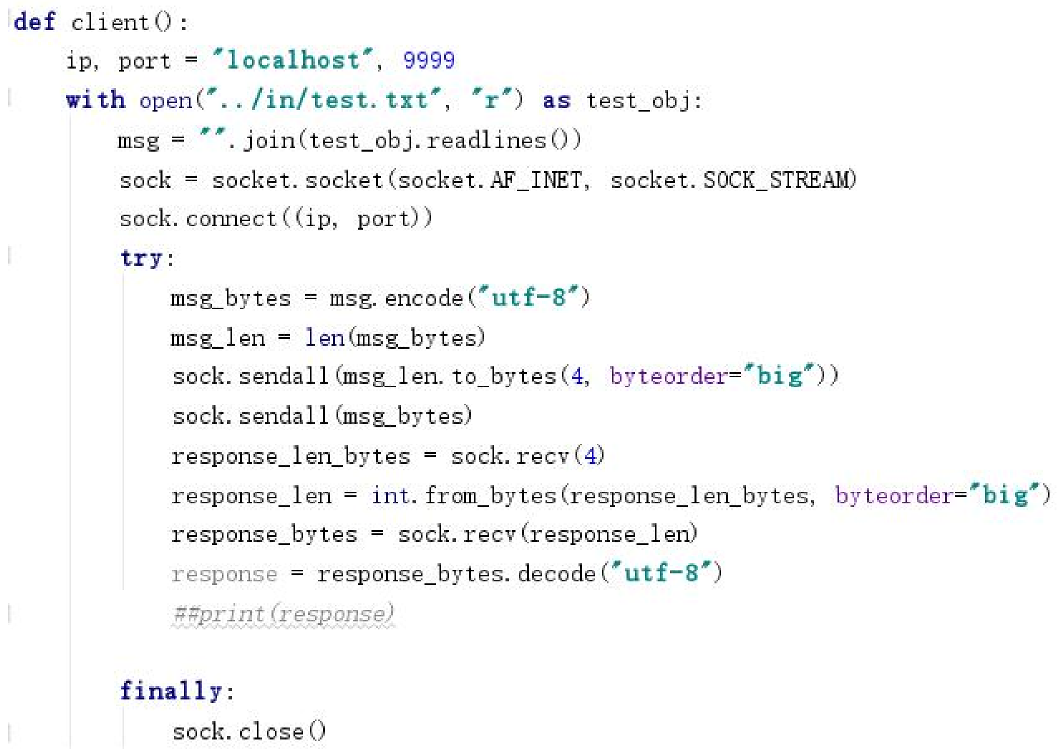
```python
def store():
    words = set()
    with open("../in/sample.txt", "r") as sample_obj:
        for line in sample_obj:
            b = re.split(r'[^a-zA-Z_\']+', line)
            words |= set([w for w in b if len(w) > 0])
    words = list(words)
    words = sorted(words, key=str.lower)
    print(words)
    index = 0
    block_index = 0
    block = bytearray(1024)
    with open("../out/sort.dat", "wb") as sort_obj:
        while index < len(words):
            w = words[index]
            w_bytes = bytes(w, "utf-8")
            w_bytes_len = len(w_bytes)
            if w_bytes_len > 256:
                continue
            if block_index + 1 + w_bytes_len > 1024:
                sort_obj.write(block)
                block = bytearray(1024)
                block_index = 0
                pass
            else:
                block[block_index] = w_bytes_len
                block[block_index + 1:block_index + 1 + w_bytes_len] = w_bytes
                block_index = block_index + 1 + w_bytes_len
                index += 1
        if block_index != 0:
            sort_obj.write(block)
```

3.2q1 查询



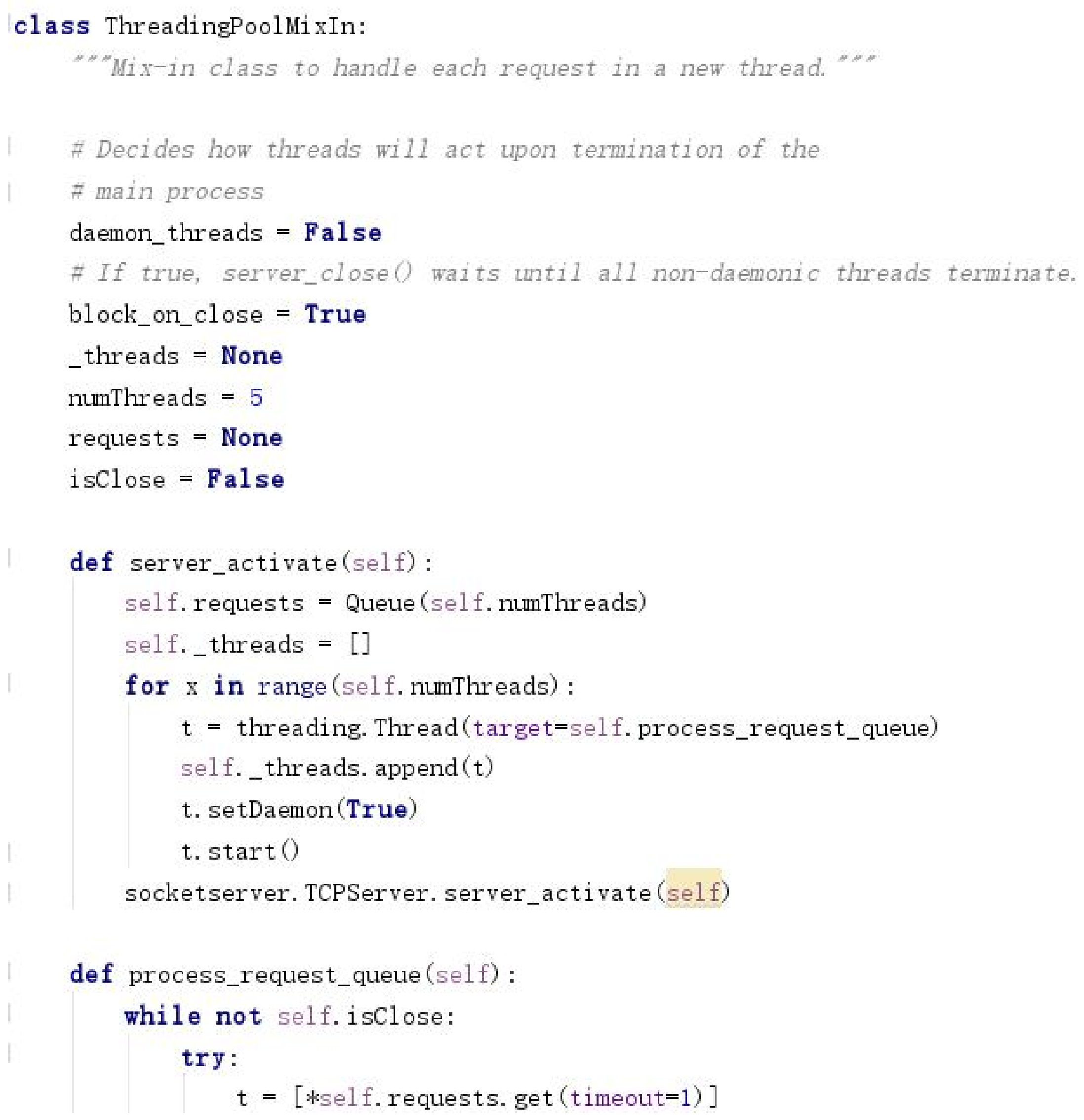
```
def find():
    with open("../out/sort.dat", "rb") as sort_obj, open("../in/test.txt", "r") as test_obj:
        outputs = []
        for word in test_obj:
            output = find_one(sort_obj, word.strip())
            if output is not None:
                outputs.append(output)
        return outputs


def find_one(sort_obj, word):
    size = os.fstat(sort_obj.fileno()).st_size
    block_num = int(size / 1024)
    output_next = False
    for block_index in range(0, block_num):
        block_local_index = 0
        sort_obj.seek(block_index * 1024)
        while block_local_index < 1024:
            t_len = sort_obj.read(1)
            if t_len == 0:
                break
            t = sort_obj.read(int.from_bytes(t_len, byteorder='big'))
            t_str = t.decode("utf-8")
            if output_next:
                return t_str + "\n"
            if word == t_str:
                output_next = True

            block_local_index = block_local_index + 1 + int.from_bytes(t_len, byteorder='big')

    return None
```

3.3q2 客户端



```
def client():
    ip, port = "localhost", 9999
    with open("../in/test.txt", "r") as test_obj:
        msg = "".join(test_obj.readlines())
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((ip, port))
        try:
            msg_bytes = msg.encode("utf-8")
            msg_len = len(msg_bytes)
            sock.sendall(msg_len.to_bytes(4, byteorder="big"))
            sock.sendall(msg_bytes)
            response_len_bytes = sock.recv(4)
            response_len = int.from_bytes(response_len_bytes, byteorder="big")
            response_bytes = sock.recv(response_len)
            response = response_bytes.decode("utf-8")
            ##print(response)

        finally:
            sock.close()
```

3.4q2 多线程



```python
class ThreadingPoolMixIn:
    """Mix-in class to handle each request in a new thread."""

    # Decides how threads will act upon termination of the
    # main process
    daemon_threads = False
    # If true, server_close() waits until all non-daemonic threads terminate.
    block_on_close = True
    _threads = None
    numThreads = 5
    requests = None
    isClose = False

    def server_activate(self):
        self.requests = Queue(self.numThreads)
        self._threads = []
        for x in range(self.numThreads):
            t = threading.Thread(target=self.process_request_queue)
            self._threads.append(t)
            t.setDaemon(True)
            t.start()
        socketserver.TCPServer.server_activate(self)

    def process_request_queue(self):
        while not self.isClose:
            try:
                t = [*self.requests.get(timeout=1)]
                self.process_request_thread(*t)
            except Empty:
                pass

    def process_request_thread(self, request, client_address):
        """Same as in BaseServer but as a thread.

        In addition, exception handling is done here.

        """
        try:
            self.finish_request(request, client_address)
        except Exception:
            self.handle_error(request, client_address)
        finally:
            self.shutdown_request(request)

    def process_request(self, request, client_address):
        self.requests.put((request, client_address))

    def server_close(self):
        super().server_close()
        if self.block_on_close:
            threads = self._threads
            self._threads = None
            self.isClose = True
            if threads:
                for thread in threads:
                    thread.join()


lass ThreadPoolTCPServer(ThreadingPoolMixIn, socketserver.TCPServer):
    pass
```

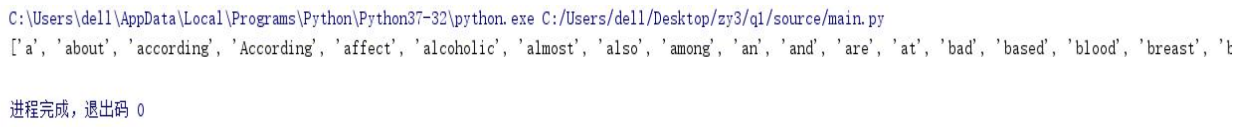
3.5q2 画图

```
def main():
    x = [0 for i in range(25)]
    y = [0 for i in range(25)]
    z = [0 for i in range(25)]
    num=0
    host, port = "localhost", 9999
    for ts in range(1, 6):
        for tc in range(1, 6):

            server = ThreadPoolTCPServer((host, port), MyTCPHandler)
            server.numThreads = ts
            server_thread = threading.Thread(target=server.serve_forever)
            server_thread.daemon = True
            server_thread.start()
            start = timeit.default_timer()
            for i in range(tc):
                client_thread = threading.Thread(target=client())
                client_thread.daemon = True
                client_thread.start()
            end = timeit.default_timer()
            find_time = end - start
            print("{0}  {1}  {2}".format(ts, tc, find_time*1000))
            x[num]=ts
            y[num]=tc
            z[num]=find_time*1000
            num=num+1
            server.shutdown()
            server_thread.join()
            server.server_close()
    # new a figure and set it into 3d
    fig = plt.figure()
    ax = fig.gca(projection='3d')


    # set figure information
    ax.set_title("3D_Curve")
    ax.set_xlabel("server")
    ax.set_ylabel("client")
    ax.set_zlabel("time")

    # draw the figure, the color is r = read
    figure = ax.plot(x,y,z, c='r')

    plt.show()
```

4.运行截图

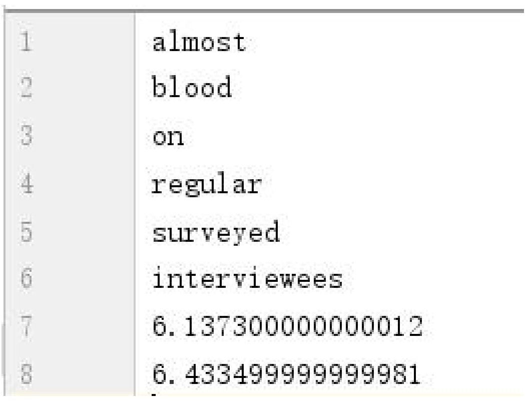
4.1q1 运行结果

控制台：



```
C:\Users\dell\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/dell/Desktop/zy3/q1/source/main.py
['a', 'about', 'according', 'According', 'affect', 'alcoholic', 'almost', 'also', 'among', 'an', 'and', 'are', 'at', 'bad', 'based', 'blood', 'breast', 'b

进程完成，退出码 0
```

日志文件：

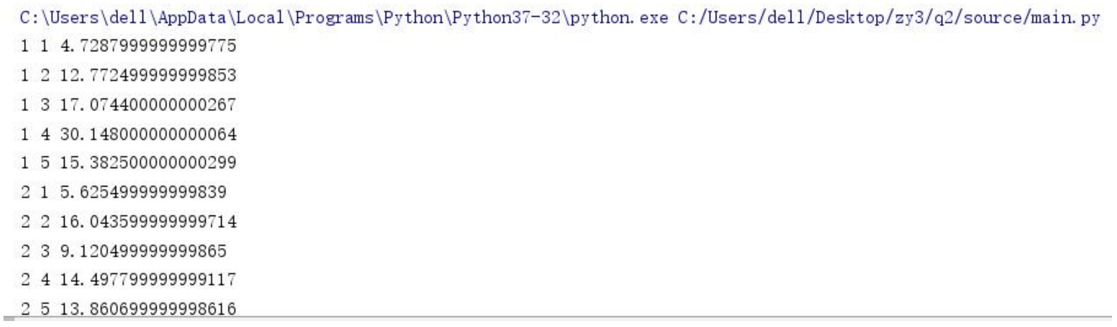


```
almost
blood
on
regular
surveyed
interviewees
6.137300000000012
6.433499999999981
```

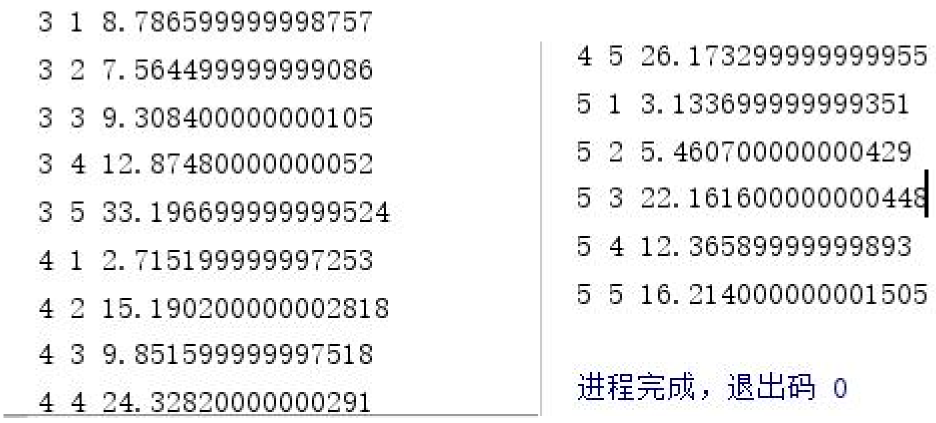
4.2q2 运行结果

控制台：

（执行 5*5 次共 25 次任务）



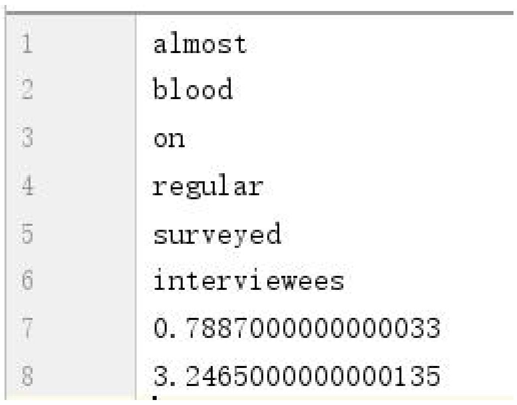
```
C:\Users\dell\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/dell/Desktop/zy3/q2/source/main.py
1 1 4.7287999999999775
1 2 12.772499999999853
1 3 17.074400000000267
1 4 30.148000000000064
1 5 15.382500000000299
2 1 5.625499999999839
2 2 16.043599999999714
2 3 9.120499999999865
2 4 14.497799999999117
2 5 13.860699999998616
```



```
3  1  8.786599999998757
3  2  7.564499999999086
3  3  9.308400000000105
3  4  12.87480000000052
3  5  33.196699999999524
4  1  2.715199999997253
4  2  15.190200000002818
4  3  9.851599999997518
4  4  24.32820000000291
```

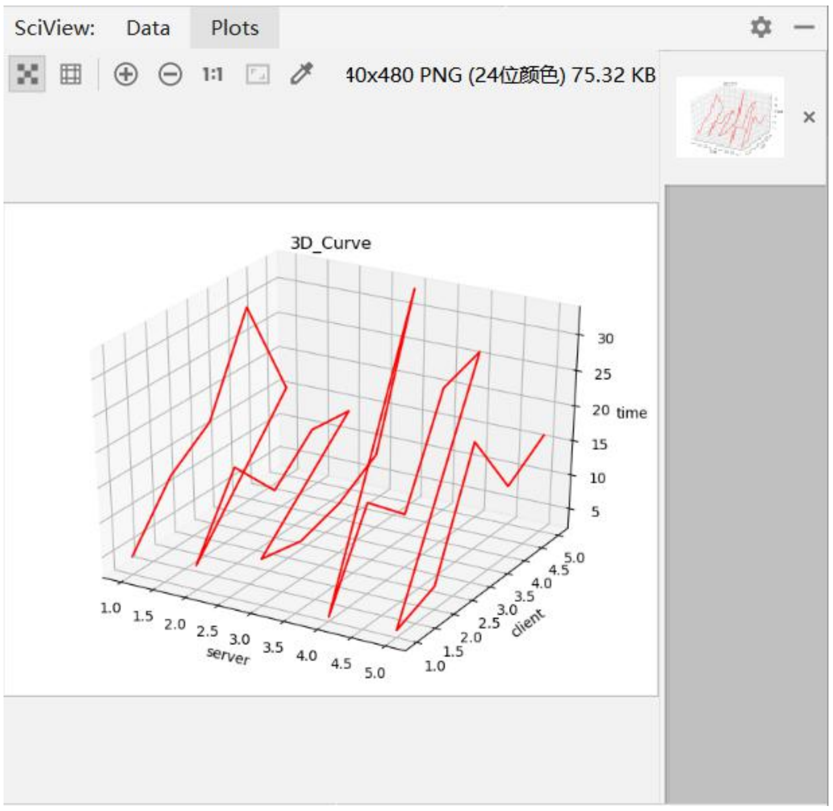
```
4  5  26.173299999999955
5  1  3.133699999999351
5  2  5.460700000000429
5  3  22.161600000000448
5  4  12.36589999999893
5  5  16.214000000001505

进程完成，退出码 0
```

日志文件：



```
almost
blood
on
regular
surveyed
interviewees
0.7887000000000033
3.2465000000000135
```

实时三维曲线图：


SciView:
Data
Plots
40x480 PNG (24位颜色) 75.32 KB
3D_Curve
server
client
time