

基本架构

服务器套接字对象 s 负责监听

c1 c2用来建立TCP连接，建立后对三个客户端轮询。

三个客户端保持循环接受json格式语句，读取相关操作要求。

json的发送：dict格式（字典）转str格式（字符串）转byte格式（比特）然后发送至套接字链接的缓存中。

json的读取：定期读取缓存，若不为空字节则接受缓存内容。比特转字符串转字典后直接读取。

防止读取粘连：服务器发送连续的两个json之间必须等待一定时间使得远方客户端有机会清空缓存，若无，则连续两个json会出现读取错误。这种情况在本地测试中，由于各自设置了0.5-1s的等待，故不会发生。若网络中前一个json的延迟过大（大于1s），则可能出错。

同时丢包的存在可能导致json残缺，理论上会报错，处理方法待开发。

默认的设置是都获取本地IP地址，欲建立远程连接，请修改套接字的地址。如果需要请注意主机是否对局域网开启了DMZ映射。

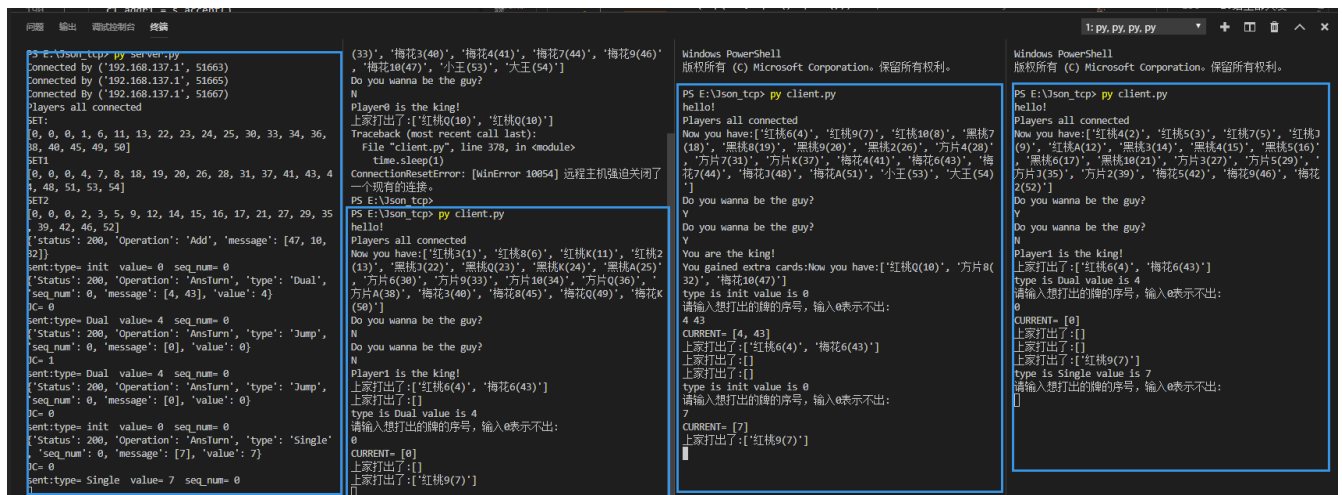
当前版本样例：(发送0号牌表示不出，后续应在服务端单独识别并广播【上家选择不出】)

示范了抢了两轮地主，后地主首先发出对6，两家不要，地主发单张红桃9。

为调试，各类参数会被打印出来，请注意发牌前打印出的牌型和值要求的变化。

服务器端打印了客户端返回的json串，同时打印要发给下一家的牌型要求和jump计数器。

有关jump计数器请看下面的解释。



```
PS E:\json_tcp> py server.py
Connected by ('192.168.137.1', 51663)
Connected by ('192.168.137.1', 51665)
Connected by ('192.168.137.1', 51667)
Players all connected
SET:
[0, 0, 0, 1, 6, 11, 13, 22, 23, 24, 25, 30, 33, 34, 36, 38, 40, 45, 49, 50]
SET1
[0, 0, 0, 4, 7, 8, 10, 19, 20, 26, 28, 31, 37, 41, 43, 4, 48, 51, 53, 54]
SET2
[0, 0, 0, 2, 3, 5, 9, 12, 14, 15, 16, 17, 21, 27, 29, 35, 39, 42, 46, 52]
{'Status': 200, 'Operation': 'Add', 'message': [47, 10, 32]}
sent:type= init value= 0 seq_num= 0
{'Status': 200, 'Operation': 'Ansturn', 'type': 'Dual', 'seq_num': 0, 'message': [4, 43], 'value': 4}
IC= 0
sent:type= Dual value= 4 seq_num= 0
{'Status': 200, 'Operation': 'Ansturn', 'type': 'Jump', 'seq_num': 0, 'message': [0], 'value': 0}
IC= 1
sent:type= Dual value= 4 seq_num= 0
{'Status': 200, 'Operation': 'Ansturn', 'type': 'Jump', 'seq_num': 0, 'message': [0], 'value': 0}
IC= 0
sent:type= init value= 0 seq_num= 0
{'Status': 200, 'Operation': 'Ansturn', 'type': 'Single', 'seq_num': 0, 'message': [7], 'value': 7}
IC= 0
sent:type= Single value= 7 seq_num= 0
[

(33)', '梅花3(40)', '梅花4(41)', '梅花7(44)', '梅花9(46)', '梅花10(47)', '小王(53)', '大王(54)']
Do you wanna be the guy?
N
Player0 is the king!
上家打出了:['红桃Q(10)', '红桃Q(10)']
Traceback (most recent call last):
  File "client.py", line 378, in <module>
    time.sleep(1)
ConnectionResetError: [WinError 10054] 远程主机强制关闭了一个现有的连接。
PS E:\json_tcp>
PS E:\json_tcp> py client.py
hello!
Players all connected
Now you have:['红桃3(1)', '红桃8(6)', '红桃K(11)', '红桃2(13)', '黑桃3(22)', '黑桃Q(23)', '黑桃K(24)', '黑桃A(25)', '方片6(30)', '方片9(33)', '方片10(34)', '方片Q(36)', '方片A(38)', '梅花3(40)', '梅花8(45)', '梅花Q(49)', '梅花K(50)']
Do you wanna be the guy?
N
Do you wanna be the guy?
Y
You are the king!
You gained extra cards:Now you have:['红桃Q(10)', '方片8(32)', '梅花10(47)']
type is init value is 0
请输入想打出的牌的序号，输入0表示不出：
4 43
CURRENT= [4, 43]
上家打出了:['红桃6(4)', '梅花6(43)']
上家打出了:[]
上家打出了:[]
type is init value is 0
请输入想打出的牌的序号，输入0表示不出：
7
CURRENT= [7]
上家打出了:['红桃9(7)']
hello!
Players all connected
Now you have:['红桃4(2)', '红桃5(3)', '红桃7(5)', '红桃3(9)', '红桃A(12)', '黑桃3(14)', '黑桃4(15)', '黑桃5(16)', '黑桃6(17)', '黑桃18(21)', '方片3(27)', '方片5(29)', '方片Q(35)', '方片2(39)', '梅花5(42)', '梅花9(46)', '梅花2(52)']
Do you wanna be the guy?
Y
Do you wanna be the guy?
Y
Do you wanna be the guy?
N
Player1 is the king!
上家打出了:['红桃6(4)', '梅花6(43)']
type is Dual value is 4
请输入想打出的牌的序号，输入0表示不出：
0
CURRENT= [0]
上家打出了:[]
上家打出了:['红桃9(7)']
type is Single value is 7
请输入想打出的牌的序号，输入0表示不出：
]
```

连续遇到两次jump后下一家可自由发牌的相关逻辑代码：

```
#...
if recjs['type']!='Jump':
    if jumpCounter==1:
        type='init'
        value=0
```

```

        seq_num=0
    else:
        jumpCounter=1
    else:
        jumpCounter=0
        type=recjs['type']
        value=recjs['value']
        seq_num=recjs['seq_num']
#...
```

待开发:

- 1.用户选牌发牌后需要检查是否存在在手牌库，若不存在则退回。若存在则扣除这些手牌。（因为测试需要，关闭此功能）
- 2.在套接字相关操作时应用catch try结构，并且返回不同的status以向程序员反馈发生了何种处理。

牌型表

映射数组定义:

```

A=['红桃','黑桃','方片','梅花']
B=['3','4','5','6','7','8','9','10','J','Q','K','A','2']
POKERS=[]
n=1
for i in A:
    for j in B:
        POKERS.append(((i+j+'('+str(n)+')')))) #初始化映射表
        n+=1
POKERS.append('小王(53)')
POKERS.append('大王(54)')
```

实际运算使用一个1-54的数组，每个牌都对应其特定一个序号，点数为其的模13，0代表无牌。

映射的代码如下:

```

def map_card(Cno):
    if Cno==0:
        return None
    else:
        return POKERS[Cno-1]

test=[1,0]
def show_card(Card):
    print('Now you have:',end="")
    print(list(filter(None,(list(map(map_card,sorted(Card)))))))
```

牌的特征码（来自符老师），特征码简化了牌型的分类逻辑，旧的分类函数（未完成）在1.0-1.1版本依然存在，它的名字是card_select

```
def prase_key(CARD): #返回牌的特征码
    for i in range(len(CARD)):
        if CARD[i]==53 or CARD[i]==54:
            continue
        else:
            CARD[i]=CARD[i]%13
    CARD=sorted(CARD)
    KEY=[]
    for j in range(0,len(CARD)-1):
        KEY.append(CARD[j]-CARD[j+1])
    return KEY
```

json命令表:

服务端:

- message 表示发送纯文本，要求客户端打印message内的内容
- AskS 要求客户端询问用户是否抢地主，并且发送回复。
- init 要求客户端重置手牌，并替换为message内的内容。
- Add 要求客户端在手牌库中增加message内的手牌
- SetTurn 表示现在轮到此客户端发牌，要求该客户端返回信息
 - Type:init 表示无限制发牌
 - Type:reply 表示有限制发牌
 - value 限定发牌要大过的值
 - type 限定发牌的类型
- Announce 一定为群发，告诉全部人打出了什么牌 牌在message里，要求客户端映射并显示

客户端:

- AnsS 表示回复抢地主结果，要求服务端读取message内的
- AnsTurn 表示发牌.
- Clear 表示该客户端牌库已清空

牌型表

'type':

- Jump : 不出
- Single: 单张牌
- Dual: 对子
- Tri: 三连
- Quad: 炸弹
- DualKing:对王
- sequ:顺子(必须大于5) -seq_num:顺子的牌数
- 3+1: 三带一

- 3+2: 三带二
- doubleSequ:双顺子 (必须大于6, 即三个对子)
- triSequ: 三顺子(必须大于6 即2个对子)
- triSequPlus:飞机带翅膀(被删除)
- 4+2:四带二(被删除)

所有顺子都要带seq_num 所有类型牌都要带: type: 牌型 message:牌 value:该牌相对的值

服务器收到AnsTurn之后的反应: 1.给全部人发message内信息。 2.把json里的三个变量刷新到服务器全局变量的:
type value seq_num,然后放进set turn函数, 发送。

注意: card_select函数被废弃。 改用card check