麻将胡牌番数的数法

摘要:实现了计算机模拟麻将胡牌算番数，通过输入麻将牌的编码及手牌信息，将其信息进行整合处理后，可以自动算出其合适 的番种和对应的番数。

关键词:麻将 番数 番种

1 引言

国标麻将是麻将的一种玩法，由中国国家体育总局于1998年制定规则。玩家通过游戏规则拼凑牌型，当牌型满足m*AAA+n*ABC+DD(m,n可为0)获胜，称为和牌或胡牌。

胡牌的牌型有对应的番种。国标麻将有81种番种，根据组成难度的不同对番种进行量化评价，可记88番、64番、48番、32番、24番、16番、12番、8番、6番、4番、2番和1番的番数。(本文采用国标麻将规则)

2 问题描述

麻将胡牌番数的计算较复杂，不仅需要胡牌的牌型信息，还需要游戏过程的事件信息（如吃、碰、杠等）、场面信息（如门风）等。除此之外，还有番种重复（一服胡牌牌型对应多个番种）等问题。要解决这个问题，需先将麻将牌型进行编码，将必要的信息作为参数输入，再将其整合处理成程序所需类型，再从高番数番种到低番数番种进行逻辑判断，最终算出对应的番种，并得出对应的番数，将所有的番数求和并补花，即可得到最终番数。

3 方案

3.1 牌型编码

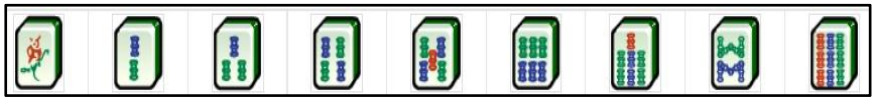
为方便计算机输入与计算，首先要对麻将的牌型进行编码。 国标麻将有144张牌，包括序数牌、字牌、花牌。

序数牌：

万牌 一万到九万，每种4张，总计36张

饼牌：一饼到九饼，每种4张，总计36张

条牌：一条到九条，每种4张，总计36张



字牌：

风牌：东、南、西、北，每种4张，总计16张

箭牌：中、发、白，每种4张，总计16张

花牌：

春、夏、秋、冬，梅、兰、竹、菊，每种1张，总计8张。

以上牌型共计 44 种，将其进行简单编码，如下：

1饼 1	1条 10	1万 19	东 28	春 35
2饼 2	2条 11	2万 20	西 29	夏 36
3饼 3	3条 12	3万 21	南 30	秋 37
4饼 4	4条 13	4万 22	北 31	冬 38
5饼 5	5条 14	5万 23	中 32	梅 39
6饼 6	6条 15	6万 24	发 33	兰 40
7饼 7	7条 16	7万 25	白 34	竹 41
8饼 8	8条 17	8万 26		菊 42
9饼 9	9条 18	9万 27		

3.2 输入参数

程序采用 matlab 编程，所需参数有两个：OrgBrand 和 message 。

OrgBrand 为牌型对应编码的向量,包括名牌和暗牌，可以无序，不记花牌。

message 为结构体，具备必要的牌信息，如吃，碰，杠所获的牌型的编码向量、花牌向量、将牌值、是否通天 、是否自摸、是否门风、是否报听等等。(用 1 或 0 表示布尔值)

OrgBrand 和 message 的结构如下所示

```
OrgBrand  %[] byte 未处理的所有牌的值(包括明牌) 不计花牌 可无序

%牌信息,如吃，碰，杠等
message=struct(...
    'chi',[],...   %吃得的顺子
    'peng',[],...  %碰得的刻子
    'gang',[],...  %杠
        ...
    'HuaPai',[],...
    'Jiang' ,0, ...
    'TongTian' ,0, ...
    'Zimo' ,0, ...
    'MenFeng' ,0, ...
    'TingBool' ,0 ...
);
```

注意:程序可通过遍历 81 个番种后判断所输入的牌型是否胡牌。但输入的牌型和信息应保持基本正确，否则会在处理时产生错误。可用已提供的用例进行测试。

3.3 加工整合

我们需要对 OrgBrand 和 message 的信息进行整合，加工成程序便于处理的结构体 MahjongCount

MahjongCount 的结构如下

```
%麻将组
   MahjongCount=struct(...
    'Array_all' ,[], ...[]byte 所有牌的值
    'Arrat_num',0,  ...byte 总牌的数量
    ...刻与杠不重复计算，能记杠则不记刻。
    'Array_ke' ,[], ...  []byte 刻子数组 (只列第一个值) (包含暗刻和碰)（如顺子111万 只记19）
    'Array_m_ke'[], ...     []byte 明刻  （碰）
    'Array_a_ke' ,[], ...  []byte 暗刻
    ...
    'Array_shun' ,[], ...  []byte 顺子数组 (只列第一个值)
    'Array_c_shun',[], ...  []byte 吃牌获得的顺子 (吃)
    'Array_h_shun',[], ...  []byte 手牌中的顺子
    ...
    'Array_gang' ,[], ...  []byte 杠数组 (只列第一个值) (包含明杠 暗杠)
    'Array_m_gang',[], ...  []byte 明杠  (杠)
    'Array_a_gang' ,[], ...  []byte 暗杠
    ...
    'Array_dui',[],... []byte  对子 (只列第一个值) (由杠可得两个对子 由刻子可得一个对子)
    ...
    'HuaPai',[],...[]byte 花牌
    'Jiang' ,0, ...   byte  将牌值
    'QiDui' ,0, ...  boolean (1 or 0)  是否七对
    'TongTian' ,0, ...  boolean (1 or 0)  是否通天
    'Zimo' ,0, ... boolean (1 or 0) 是否自摸
    'MenFeng' ,0, ...  boolean (1 or 0) 当前胡牌玩家门风
    'TingBool' ,0    ... boolean (1 or 0) 是否报听
      );
```

注意:

1. 以上的向量都需经过排序处理。

2. 因程序目的是计算番种和番数，故顺子、刻子、杠、对子的计算是相互独立的。（如一个杠可同时记两个对子）

3. 若以上 MahjongCount 的信息不足以处理所有番种的计算，还可用自行添加 。

3.3.1 杂项

Array_all、Arrat_num、HuaPai、Jiang、TongTian、Zimo、MenFeng、TingBool 这些信息是可以经过直接获取 message 信息或简单处理后得到。

3.3.2 刻子和杠

全刻、明刻、暗刻　Array_ke、Array_m_ke、Array_a_ke

全杠、明杠、暗杠　Array_gang、Array_m_gang、Array_a_gang

首先预处理，通过判断牌型向量里有 3 个相同数或 4 个相同数，即可得到 Array_ke 和 Array_gang。

```
%获得去重牌组
brand_wr = unique(OrgBrand);
%刻子和杠子的预处理
arr_ke=zeros(1,length(brand_wr));
arr_gang=zeros(1,length(brand_wr));
%有三个相同的数，为刻子；有四个相同的数，为杠
for i=1:length(brand_wr)
    if(length(find(OrgBrand==brand_wr(i)))==3)
        arr_ke(i)=brand_wr(i);
    end
    if(length(find(OrgBrand==brand_wr(i)))==4)
        arr_gang(i)=brand_wr(i);
    end
end
arr_ke=arr_ke(arr_ke>0);
arr_gang=arr_gang(arr_gang>0);
```

根据规则 Array_m_ke=message.peng；Array_m_gang=arr_m_gang;

而对于暗刻和暗杠，可分别用全刻和明刻、全杠和暗杠求补集（可利用刻或杠不会重复的性质求，详见代码）

注意:刻与杠不重复计算，能记杠则不记刻。

3.3.3 顺子

Array_shun、Array_c_shun、Array_h_shun

对 Array_shun:

对牌向量去重得到去重后向量，从中求出存在的顺子（设为 arr）。遍历 arr，计算顺子 n 三个数(n,n+1,n+2)在原向量里的数量，以三种的最小值作为这个顺子的 n 的数目，得到 Array_shun。

对 Array__shun:

Array_c_shun=message.chi

对 Array_h_shun:

由 Array_shun、Array_c_shun 求出。（顺子不同于刻和杠，存在重复的情况，故在求补集时要去重）

```
%用arr_shun和arr_c_shun求 arr_h_shun
index=zeros(1,length(arr_c_shun));
for i = 1:length(arr_c_shun)
    ind=find(arr_shun==arr_c_shun(i));
    index(i)=ind(1);
end
index=index(index>0);
arr_h_shun=arr_shun;
arr_h_shun(index)=[];

if(isempty(arr_h_shun))
     arr_h_shun=[];
end
arr_h_shun=sort(arr_h_shun);
MahjongCount.Array_h_shun=arr_h_shun; %手牌中的顺子
```

3.3.4 对子和七对

对子 Array_dui 七对 QiDui（1 或 0）

遍历已经处理好的刻和杠，一个刻可以得到一个对子，一个杠可以得到两个对子，同时对原牌向量去重。之后再遍历去重的牌向量，得到剩余的对子。

通过对子的数量，判断是否符合七对。

3.4 判断番种

通过上述步骤，得到结构体 MahjongCount

将 81 番种排序，本文将低番数番种到高番数番种排序，如：

```
HUAPAI=1; ...花牌
...
DASIXI=81; ...大四喜
```

建立 1*81 默认为 0 的向量 confirm、ban，其下标分别对应 81 种的序号。利用 MahjongCount 编写 81 番种的业务逻辑函数，判断其是否符合此番种。当符合此番种时，将对应下标的 confirm 设为 1，与此番种冲突(不计)的番种对应下标的 ban 设为 1。

花牌另外计算，返回补花数。

按番种序号的倒序进行遍历判断，即大番种到小番种，如：

```
[confirm,ban]=DASIXI(MahjongCount,confirm,ban); %大四喜
...
[confirm,ban]=ZIMO(MahjongCount,confirm,ban); %自摸
%花牌补记
[add,confirm,ban]=HUAPAI(MahjongCount,confirm,ban); %花牌
```

注:每个番种判断函数都先对其对应位的 ban 是否为 1，提高效率。

3.5 计算总番数

在遍历 81 番种的判断函数后,根据 confirm、ban，以查表法得到对应番种的名称和其对应番数值。总番数等于各番种番数之和加上补花数。若总番数为 0，则输入的牌并未胡牌。

注意：本文程序并未全部实现 81 种番种判断函数，故合适番种可能并不全面。请用已提供的用例进行测试。

4.代码结构

本文附代码说明如下：

- main.m:为外部接口，负责调用 function 文件夹函数进行番数计算，并打印结果。 function main(OrgBrand,message)
- Test.m:为提供的测试用例，可直接运行
- function 文件夹
  - calFan.m:番数计算 function [fanName,fanNum]=calFan(OrgBrand,message)
  - tool 文件夹
    - process.m:将参数 OrgBrand,message 进行整合处理，得到 MahjongCount function MahjongCount = process(OrgBrand,message)
    - FanNum.m:根据番种 id 获取番种对应的番数 function num=FanNum(index)
    - IndexForName.m:根据番种 id 获取番种中文名称 function name = IndexForName(index)
  - FanTypeCalculation 文件夹

- 此文件夹存放 81 种番种判断函数(本文代码仅实现部分)
- IndexForName 类:属性为各番种的英语名称,其值为番种对应的 id 号。用于使代码更为简便。

注意：代码文件为多层结构，在使用 Test.m 进行测试时，应把当前文件夹调整至 Test.m 文件所在的文件夹！

5 测试

test1

输入：

```
%% 大四喜
clc;
clear;
OrgBrand=[29 29 29 29 30 30 30 30 31 31 31 31 28 28 28 28];
message=struct(...
    'chi',[],...   %吃得的顺子
    'peng',[],...  %碰得的刻子
    'gang',[29 31],...  %杠
         ...
    'HuaPai',[40 42],...
    'Jiang' ,30, ...
    'TongTian' ,0, ...
    'Zimo' ,1, ...
    'MenFeng' ,0, ...
    'TingBool' ,0 ...
);
%注意，点左边框运行时，要先移动到main函数对应的文件夹
%否则会报错：未定义与 'struct' 类型的输入参数相对应的函数 'main'。
main(OrgBrand,message);
```

输出：

```
MahjongCount:
       Array_all: [28 28 28 28 29 29 29 29 30 30 30 30 31 31 31 31]
       Arrat_num: 16
          HuaPai: [40 42]
           Jiang: 30
        TongTian: 0
            Zimo: 1
         MenFeng: 0
        TingBool: 0
        Array_ke: []
      Array_m_ke: []
      Array_a_ke: []
      Array_gang: [28 29 30 31]
    Array_m_gang: [29 31]
    Array_a_gang: [28 30]
      Array_shun: []
    Array_c_shun: []
    Array_h_shun: []
       Array_dui: [28 28 29 29 30 30 31 31]
           QiDui: 0
```

```
符合番种:
    {'大四喜'}
    {'字一色'}
    {'双明杠'}
    {'自摸' }
    {'花牌' }

番数:
159番
```

test2

输入：

```
%% 三暗刻
clc;
clear;
OrgBrand=[1 1 10 10 10 20 20 20 3 3 3 11 11 11];
message=struct(...
    'chi',[],...   %吃得的顺子
    'peng',[10],...  %碰得的刻子
    'gang',[],...  %杠
        ...
    'HuaPai',[],...
    'Jiang' ,1, ...
    'TongTian' ,0, ...
    'Zimo' ,0, ...
    'MenFeng' ,0, ...
    'TingBool' ,0 ...
);
%注意，点左边框运行时，要先移动到main函数对应的文件夹
%否则会报错：未定义与 'struct' 类型的输入参数相对应的函数 'main'。
main(OrgBrand,message);
```

输出：

```
MahjongCount:
       Array_all: [1 1 3 3 3 10 10 10 11 11 11 20 20 20]
       Arrat_num: 14
          HuaPai: []
           Jiang: 1
        TongTian: 0
            Zimo: 0
         MenFeng: 0
        TingBool: 0
        Array_ke: [3 10 11 20]
      Array_m_ke: 10
      Array_a_ke: [3 11 20]
      Array_gang: []
    Array_m_gang: []
    Array_a_gang: []
      Array_shun: []
    Array_c_shun: []
    Array_h_shun: []
       Array_dui: [1 3 10 11 20]
           QiDui: 0

符合番种:
    {'全小'}
    {'三暗刻'}

番数:
40番
```

6 结果及讨论

1. 如上所示,运行测试用例后,即可打印输出对应的MahjongCount结构、番种和总番数。

2.本文设置的message及得到的MahjongCount以可以处理大部分番种判断的情况。若出现特殊情况，如改用其他的麻将规则等，可以直接在相应的番种判断函数里做处理，或对message添加相关信息并补充MahjongCount。

7.小结

本文根据输入的牌型和事件等必要信息，处理并计算出番种及番数。本文主要为数据处理和提供结构体MahjongCount，结构体MahjongCount可用于番种判断函数的编程。番种判断函数为业务逻辑代码，本文仅完成了部分函数。

8.参考

思路参考博客:http://t.csdn.cn/ceV5t