## 一个扑克牌随机生成器：shuffle函数的实现和使用

你不实现它就别想使用它。

算法：(依次随机为每个元素找一个新的位置，当然，这个位置可以是它原来的位置）

def shuffle(seq):

import time,random

random.seed(time.time())

max\_size= len(seq)

for i in range(max\_size):

new\_index = random.randint(0,max\_size-1)

if new\_index != i and seq[i]!=seq[new\_index]: #when it becomes complicated to compare two elements,this shall not go on

temp = seq[i]

seq[i] = seq[new\_index]

seq[new\_index] = temp

if \_\_name\_\_=='\_\_main\_\_':

import random

seq = [ 3, 4 , 5,64,12]

seq2 = seq.copy()

print(seq)

shuffle(seq)

random.shuffle(seq2)

print(seq)

print(seq2)

## 扑克牌的类

Card

color

number/name

---

PokerCards

cards

---

注：python中的classmethod和staticmethod的区别在于前者将第一个参数视为类

## 发牌函数引入的问题：从一个序列中随机抽取指定的元素

算法1：对牌进行乱序之后，选择其中的前num张

输入：cards, num

输出：selected,left

if num<len(cards):

shuffle(cards)

x = cards[:num]

cards = cards[num:]

else:

x = cards

cards = []

return x,cards

问题：如果是连续的多次抽牌，实际上只需要一次shuffle就可以了。因此连续多次抽牌，算法性能有所下降。

解决方法：引入参数 usePreviousShuffle:boolean类型。保证在连续的n次抽牌中只使用了一次shuffle函数

算法2：引入状态参数

输入：cards, num,usePreviousShuffle

输出：selected,left

if num<len(cards):

if not usePreviousShuffle or not shuffled:

shuffle(cards)

x = cards[:num]

cards = cards[num:]

else:

x = cards

cards = []

return x,cards

附：代码见源文件

## 基于原子的牌型的判别

牌的分类：单牌，顺子，对子，三带一，连对，炸弹，杂牌

问题：给定一组牌，将其视为原子，判断其类型

先将牌排序，然后使用有限状态转换机判定。

（0张牌）-->单牌-->判别是否相同，判别是否相连，-->否：-->...

## 基于整牌的牌型分类：牌成分析

所谓的牌成分析，就是分析一副牌里面 对子有哪些，只能构成单牌的有哪些等，并且随着牌的打出，牌成也相应的变，如何动态识别出这些变化？

牌成分析算法：对一组牌，先将其排序，然后依次对每张牌进行分析，依次询问：对子，连牌，三个，炸弹，连对。如果以上都不符合，就是单牌。单牌永远都是单牌，不会变化。如果至少有一处符合，就将其记录进字典结构中。所有同样的牌作同样的分析。

有一处改进：如果是三个同样的牌，就不必记录进对子中，只需要更新的时候记录就行了了。

输入：一组牌

输出：一个记录牌的成分映射的字典

首先，统计牌的数量多余1个牌类。然后分析连牌，连对，飞机（两个及以上的三连牌）

1. 如果是先手，出什么牌
2. 如果是后手，出什么牌应付

简单AI算法：

先手出牌算法：出牌顺序：没人打得过的牌但也打不过人的牌 > 估价最低的牌 > 万能牌

后手出牌算法：如果下家是地主，就采用顶牌策略； 如果己方是地主以及己方是地主下家，出最小的牌，如果没有牌，就过。

## 牌的大小比较：1.给定两组具有大小关系的牌，判断大小次序 2.给定一组合法的出牌，在待选牌中选出一副能大的牌

1.出牌的所有可能：

单牌 -- 1张牌

对子 -- 2张牌

连牌 -- 至少5张牌

连对 -- 至少6张牌

3个 -- 恰好三张牌

3带1 -- 恰好4张牌

飞机 -- 至少6张牌 （两个连续的3对）

炸弹 -- 恰好4张牌或2张牌

除了同类型的牌以及炸弹具有可比性之外，其他的不具有可比性。

因此顺序十分容易确定。代码见PokerCards.compareCards。

X\_TYPE1\_TELL\_TYPES=['Zero','Unknown','Single','Double','Triple','TripleWithOne','Boomb','ConnectedSingles','ConnectedDoubles','Nothing','MultiTriples']

#0 1 2 3 4 5 6 7 8 9 10

1. 出牌

将所有的出牌策略分为两类：1.使用动态估价函数（动态性） 2.使用静态策略（静态策略）。出牌后的，所有的牌成分析都失效。所以不建议将牌成分析作为牌的状态。

（静态策略）

牌成分析后，获取要打的牌的类型，然后在自己相应的类型中选择能打的牌，返回能打的牌。

定义牌型顺序关系：单牌<对子<连牌<连对<飞机

对所有的牌按小到大的顺序进行遍历

依次查询每张牌在其以上的牌型中情况，如果存在，就选择下一张，否则就选择之。

这种策略没有伸缩性，不够良好。

单牌，对子，三个，炸弹：

选择一张数量够的牌，然后检查这张牌可能破坏的牌型。 计算出其对连续牌型的破坏程度。比如现在有牌 33445678999,牌成分析为{ cs:{3:7}}，对5。可以选择出 6 7 8 9

,出6，7两部分都被破坏， 出8只有右边被破坏，出9则不破坏连牌，但是破坏了3个9。 理论上来说出8好一点。

破坏程度的衡量：对连牌/连对/飞机 牌将被分为n个部分，设为 P1,P2,...,Pn。

初始的时候，牌有一个总价值V0。

3带1：

在3个的基础上，选择除这个牌之外的单牌

定义牌型顺序 ct > cd > cs

连牌：

在cs中选择，

连对：

在cd

## 组合概率问题，牌的估价策略

估价策略：如果一组牌出现的概率越小，则其价值越大。即 （1 - P） \* 100为其价值。

从54张牌中抽出m张牌，其中某张牌（不是猴子）有k张的概率为:

C(4,k)\*C(50,17-k)/C(54,17)

猴子有k张的概率为：

C(2,k)\*C(52,17-k)/C(54,17)

由此计算有：

2,3~A: 0张--0.209 1张--0.418 2张--0.286 3张--0.080 4张--0.008

G: 0张--0.465 1张--0.440 2张--0.095

记P0(i)为i出现0次的概率（在54分发的17张牌中），P01(i)为出现0~1次，P02(i)出现0~2次

连牌概率：(起始-i，张数--j)

至少有1个i且1个i+1,....1个i+j-1,

每张牌至少有一个的概率为1-P0,所以概率为

 （增函数，在直观上显然是错误的）

从另一个方面考虑，对于34这样的连牌先抽各一张3、4，再到剩下的52张中抽15张。概率即为 ，但是这显然包含重复。如果牌中只有34就是正确的，但是当有3，4在闲牌中可以替换时，就要考虑这种重复。分两种情况考虑：仅仅34，有多余的3，4。即按3，4的个数分别考虑，(1~4,1~4)共16种情况考虑

设3,4的数目为依次为k0,k1,则概率为：



让k0,k1从1~4变化即取得其概率和。复杂度为n^j,为j的指数级变化

连对概率：（起始-i，对数--j）

类似的，概率为上述概率的至少是两个的部分概率和

飞机不带单牌的概率:（起始--i,连续数--j)

概率为上述概率的至少3个的部分概率和

表：

连牌：

5 0.2758735593779525

6 0.20442031724314189

7 0.14869144572026982

8 0.10586858787728441

9 0.07352593810529773

10 0.04963054065938481

11 0.03270913316234403

12 0.022830007518722224

连对

3 0.03739357009998862

4 0.009430086570461776

5 0.001911869312407955

6 0.000284487433750871

7 0.000026152917

飞机

2 0.005241727361358032

3 0.00018901499101807673

4 0.00000314325294

玩家拿到某副特定牌的概率：(因为花色不同但是牌相同的算一副）

根据猴子的数目来分： 0个猴子 1个猴子 2个猴子 。这三类之间自己的各副牌的概率是相同的。因此，概率分别为0.465 0.440 0.095。所以拿到一副牌的概率只需要看猴子的个数就知道了。

注：公式

A(n,m) = n!/(n-m)!

C(n,m) = A(n,m)/m!

## 量化：牌的价值估算（这似乎才是最重要的部分）

建议参考： 已经出的过牌 是否任意出 牌的数量 牌的类型是否单一 牌的拆分法（牌不共用）以及每一种拆分法的牌共分为几个部分 每种牌型在自己的牌型之内能打的牌（即到底是累赘还是王牌）

每种牌在自己牌型中的价值计算法：

大部分牌可以用 三元组 (start,repeat\_time,connected\_time)来描述。一般来说，这些三元组之间是具有大小关系的。

比如 444555666 描述为 (4,3,3) 可以打 334455 333444555 33 44 55。

但是不妨扩展一下，将所有牌用四元组来描述 (start,repeat,connected,others)来描述。先确认所有可能的四元组，然后再这些四元组中计算 1个四元组所能大的未知的（就是要参照已经知道的牌）四元组的数量，将其除以所有的还可能存在的四元组就是这个四元组的价值。

17张牌

## 基于深度学习理论的AI算法