



FDA分享_0-1的一次德州扑克_20220527

德州扑克介绍

德州扑克是目前世界上最流行的扑克游戏，全世界有众多相关的比赛，例如是 WSOP, WPT, EPT 等，也让这款游戏的玩法变得层出不穷，丰富多变。

德州扑克英文全称是 *Texas Hold' em poker*，一般是2-10个人在用52张牌（去掉大小王）进行游戏，游戏方式是通过「比各自的牌型大小」or「使对手弃牌」，目的是赢取其他玩家的筹码。

德州扑克的最基础流程是这样的：

1. 玩家们围着一张桌子坐下，每个位置有特地的顺序和优势。
2. 每个玩家会收到2张牌，这两张牌是玩家的「底牌」，且只有自己知道。
3. 发完牌后，大、小盲位必须要下注后，其他玩家选择
 - a. 跟注：放下相同的筹码
 - b. 加注：加注，并且提高“跟注门槛”
 - c. 弃牌：放弃手牌，不玩了
4. 翻牌圈：展示3张公共牌
 - a. 此时场上3张牌 + 2张手牌 = 5张牌，玩家分析目前5张手牌的牌力大小
 - b. 跟注、加注
 - c. 弃牌：放弃手牌，失去已经下注的筹码
5. 转牌圈：展示1张公共牌

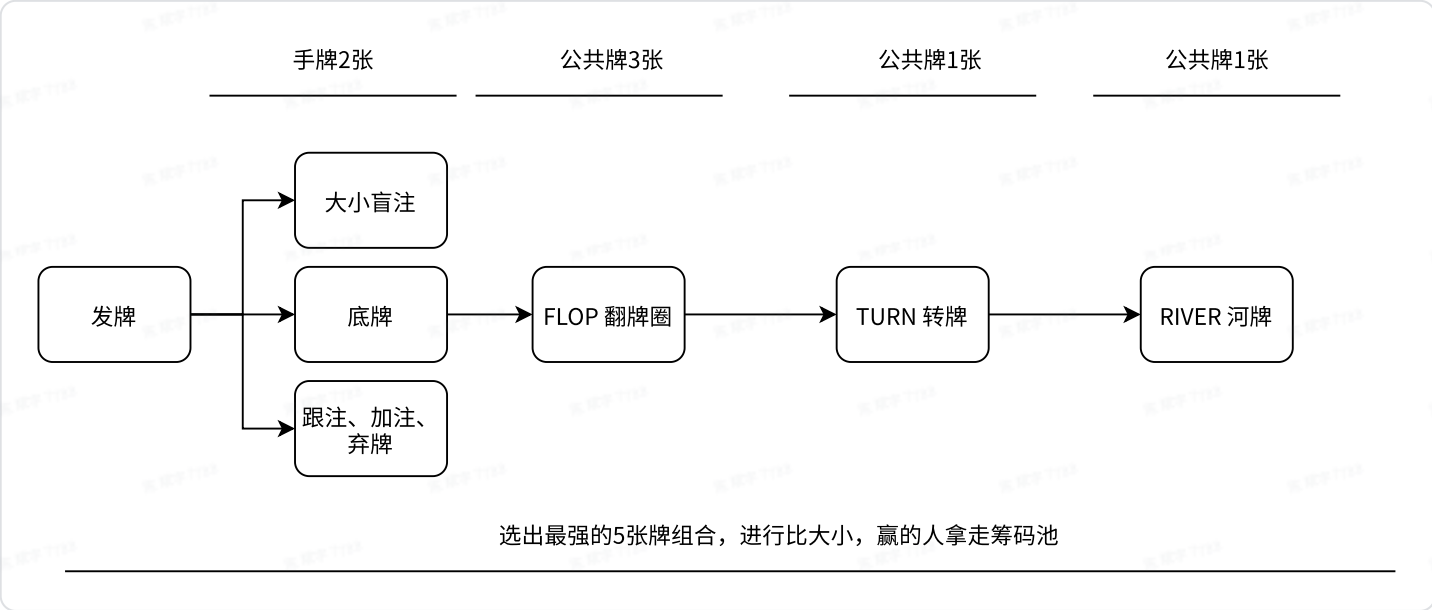
- a. 此时场上4张牌 + 2张手牌 = 6张牌，**玩家选出最强的5张牌**，分析牌力
- b. 跟注、加注
- c. 弃牌：放弃手牌，失去已经下注的筹码

6. 河牌圈：展示1张公共牌

- a. 此时场上5张牌 + 2张手牌 = 7张牌，**玩家选出最强的5张牌**，分析牌力
- b. 跟注、加注
- c. 弃牌：放弃手牌，失去已经下注的筹码

7. 开牌：所有玩家比大小

- a. 如果是同牌型比副牌
- b. 如果完全一样，平局对半分筹码





















































不要被简单的游戏规则而误导，复杂多变的比赛状况，让这款游戏在高水平的竞技中会变得非常复杂，这也让人们为德州扑克给出了这样一句评价“用一刻就能学会，但要用一生才能掌握”。

牌型大小是什么样的

所以简单来说，德州扑克的规则底层就是比大小，有些牌型很“大”，拿到几乎是必赢，有些牌型很容易抽到，但是赢面很小，那么具体牌型是哪些呢？

如下图可见，牌面从大到小排序：

1. ROYAL FLUSH	皇家同花顺					
2. STRAIGHT FLUSH	同花顺					
3. FOUR OF A KIND	四条					
4. FULL HOUSE	葫芦					
5. FLUSH	同花					
6. STRAIGHT	顺子					
7. THREE OF A KIND	三条					
8. TWO PAIR	两对					
9. ONE PAIR	一对					
10. HIGH CARD	高牌					

暂停一下：

其实德州扑克是什么，基础的逻辑是什么就足够了。

接下来其实不是想介绍德州扑克的古往今来或者具体怎么才能赢，因为世界上关于如何玩好德州扑克的玩法太多了，有许许多多的专业赌徒，他们会有各种心得策略，比如：

1. 有人说要自我控制，切忌上头，冷酷冷静地处理每一次牌。
2. 有很多人认为要学好bluffing(虚张声势)，还有read对方，把这个游戏看得像无间道一样。观察每个玩家玩牌的方式，解读对方的人设。同时建立自己的人设。如果您打牌一直扎扎实实，从未被人发现有虚张声势的表现，那么您就拥有了bluffing的本钱。
3. 不争胜负，管理资金。你可以赢很多次，但是亏本钱也许只需要一次，管理资金，量化每一次EQ才是真的赢家。
4.

但其实这篇分享...

真正要分享的是，当我接触到德州扑克时，作为一个数据分析师，对于其做的一些浅层的小研究，以及在做这个项目之中对于数据结构、算法模型，甚至产品化过程中的一些感悟和理解。最终：

1. 与大家分享自己的感悟
2. 期待有趣的思维逻辑碰撞

于是...

基于我在这个项目上的探索研究的过程，我按时间线分为了4个章节，向大家阐述我的心路历程。也希望大家能代入进来，一起思考，enjoy。

Part1. 接触后的好奇

在51假期机缘巧合接触过德扑之后我有了以下一些疑问：

如何通过数据表现德州扑克？为什么3条的赢面比顺子小？德州扑克是否能有一个算法，告诉我什么时候加注，什么时候弃牌？我的胜率是不是能量化？

带着这些问题，我通过jupyter去探索，模拟，设计一些底层数据结构与算法。

后续思路

在有了对德州扑克的基础概率以及数据结构一个初步的探索之后，我拥有了一些可以复用的函数，可是我依然不知道我的胜率是多少；如果在一局游戏每轮胜率是动态的，何时应该弃牌；我的具体胜率是什么？那么该如何解决这个问题呢？

1. 胜率和人数无关？

首先在德州扑克中是无法知道对手的手牌，所以纯概率层面来说，我们关注的点不是“我们能不能赢具体的某个他”，而是我们的手牌“会被多少种手牌打败”

2. 我们拿到各种牌型的概率是什么？

在每一轮，我们拿到各种牌型的概率随着发出的牌会有变化。这也意味着，我们最终的函数需要针对不同发牌阶段计算我们的胜率。

3. 如何计算胜率？

- a. 索引法：德州扑克游戏中一共有 $c_{52,5} = 2598960$ 种组合，查询每一种牌型的“排行”，就能知道输给多少牌，赢了多少牌。
 - i. 查询效率：用什么方式存储数据？数据库的查询效率如何？矩阵还是DataFrame？
 - ii. 排序逻辑：这张结果维表的排序逻辑是什么？需要哪些结果字段？
 - iii. 索引id：如何保证牌型的索引id一致？
 - iv. 这算法最大的缺陷是计算的胜率是「实际胜率」，而不是「估计胜率」
- b. 穷举法：穷举自身的每一种牌型的情况，并穷举每种牌型的所有“敌人”，最终用0,1 记下结果
 - i. 算法流程：该如何设计每个函数的主、子关系
 - ii. 复杂度：如果不利用任何缓存的话那么

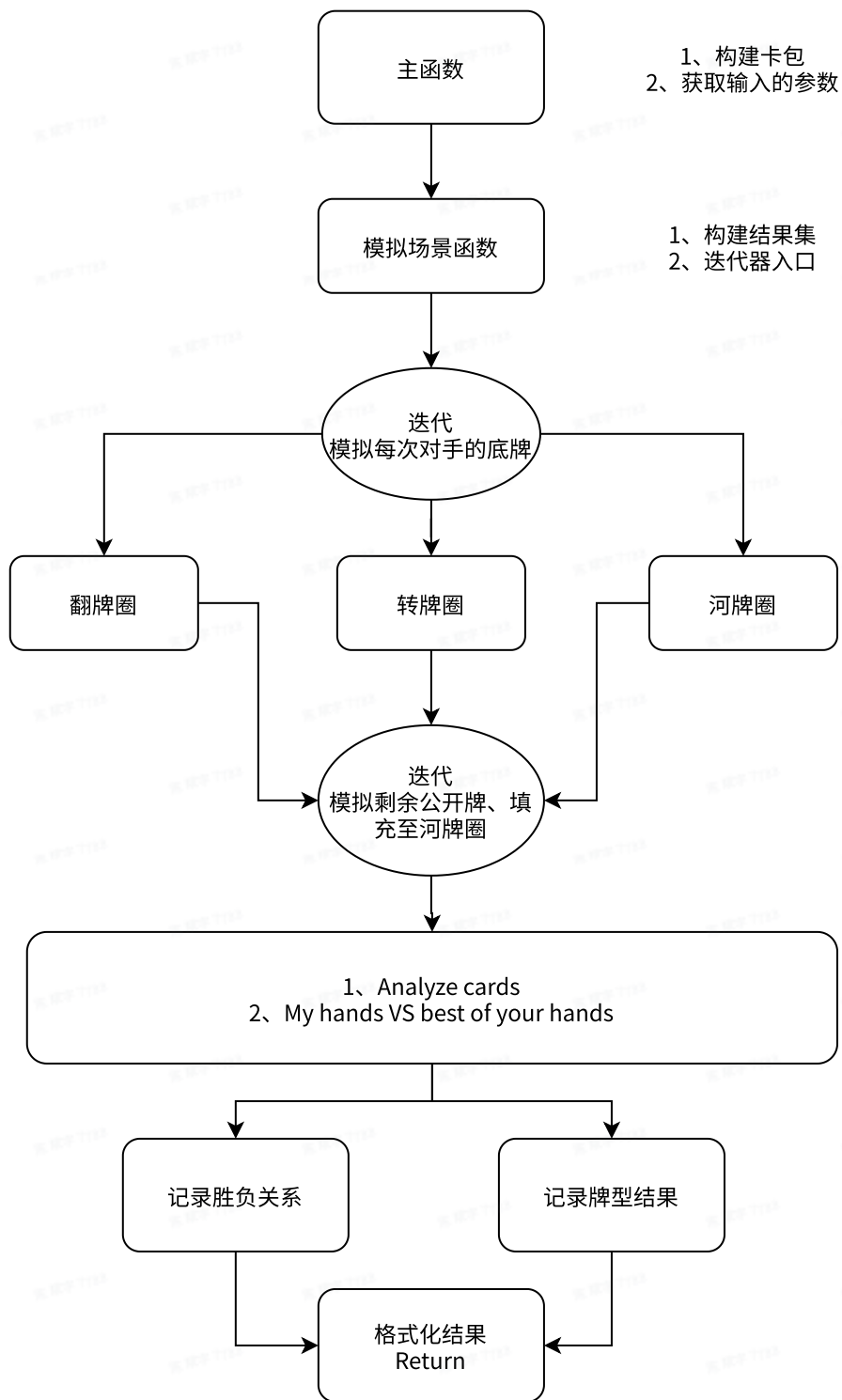
Part2. 我可能需要一个工具

在头脑风暴之后，我意识到需要开发一个算法，去帮助我实现上述所有的构思和需求。

所以最后的主函数的伪代码如下：

```
1 def poker_run(我的卡牌: list, 对手的开牌: list, 桌上的卡牌: list):
2     return 胜率, 每种牌型的可能性
```

算法流程：



以上就是整个算法程序的流程图，整体流程非常简单，但是中间层的子函数传参问题确实有些让人看起来眼花缭乱，不过不管怎么样，还是完成了，测试了多个case，也能跑通。但随之而来的是另一个巨大的问题。

算法运行时间过长

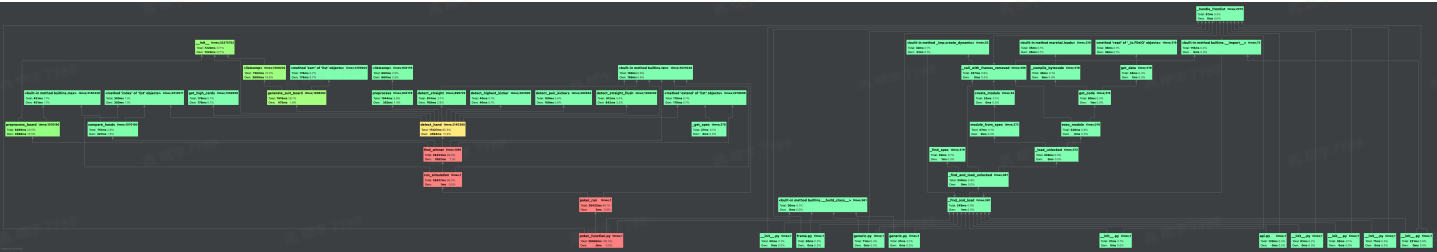
在跑通这个模型后，我测试了几个场景下的运行效率。

	A	B
1	平均运行时长(s)	
2	翻牌圈	26
3	转牌圈	1.3
4	河牌圈	0.5

优化前

	A	B	C	D	E
1	执行模块	call count	time (ms)	own time(ms)	
2	poker_function.py	1	26682	0	
3	poker_run	1	26432	3	
4	run_simulation	1	26427	1	
5	find_winner	1081	26423	1882	7.10%
6	detect_hand	2140380	17427	4594	17.20%
7	generate_suit_board	1208202	7916	475	1.80%
8	<listcomp>	1208202	7262	3889	14.60%
9	__init__	32275752	7222	7222	27.10%
10	preprocess_board	1070190	6269	3598	13.50%
11	preprocess	932178	1044	383	
12	detect_straight_flush	1208202	912	842	
13	detect_straight	898722	814	769	
14	compare_hands	1070190	751	427	
24	get_high_cards	1352928	179	179	
28	detect_pair_kickers	390852	169	169	
31	get_code	319	90	1	

运行详情



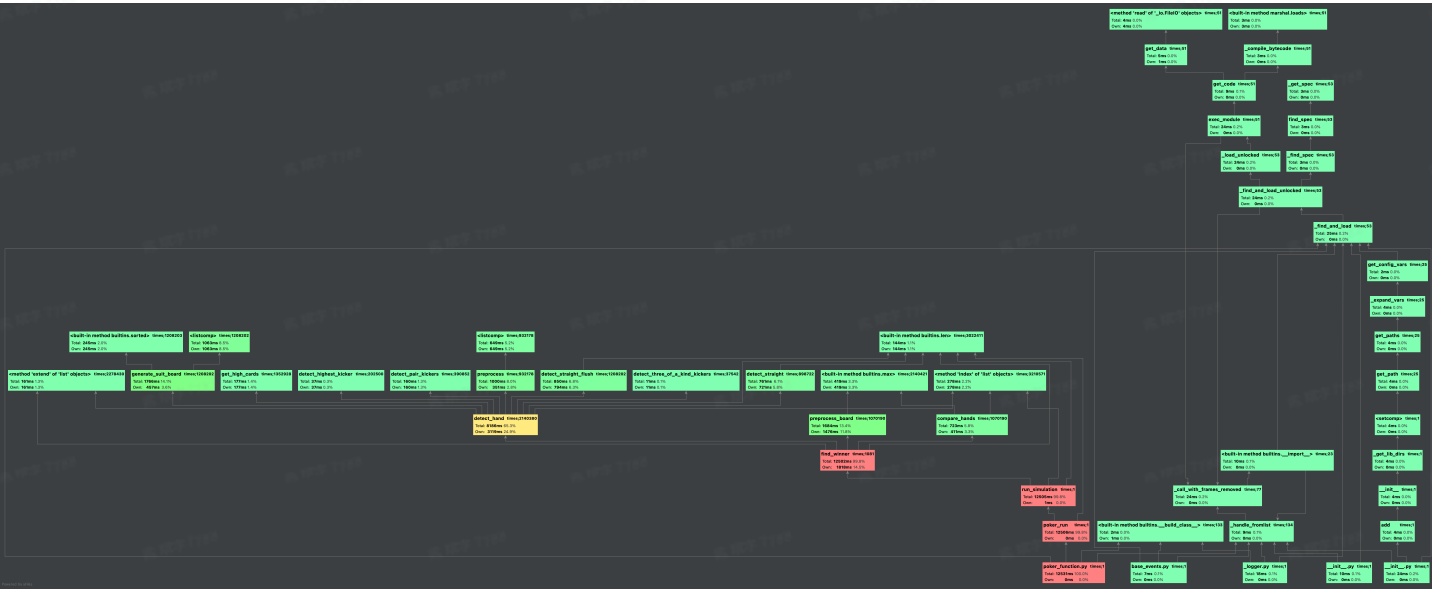
优化方式：

- 1. 将扑克对于花色和牌面的类去除，使用列表索引和字典代替
- 2. 精简了判断手牌函数
- 3. 优化了前置格式化函数中，降低部分复杂度

优化后：

	A	B	C	D	E
1	执行模块	call count	time (ms)	own time(ms)	变化
2	poker_function.py	1	12531	0	
3	poker_run	1	12506	0	
4	run_simulation	1	12505	1	
5	find_winner	1081	12502	1818	
6	detect_hand	2140380	8186	3119	
7	generate_suit_board	1208202	1766	457	
8	preprocess_board	1070190	1684	1476	
10	preprocess	932178	1000	351	
11	detect_straight_flush	1208202	850	794	
12	detect_straight	898722	761	721	
13	compare_hands	1070190	723	411	
18	get_high_cards	1352928	177	177	
20	detect_pair_kickers	390852	160	160	
22	detect_highest_kicker	202500	37	37	
30	detect_three_of_a_kind_kickers	37542	11	11	
34	get_code	51	9	0	

运行详情



思考题：
如何判断n张牌里有5张牌的顺子？

踩坑 and 心得：

- 1. pycharm可以说是最好用的python工程代码编译器了，yyds

2. 谨慎用类
3. 多用内置函数提高效率
4. 不要在算法里用pandas! 不要在算法里用pandas! 不要在算法里用pandas! !
5. 冗余的package不要import
6. 如果没有跑满算力，那么多进程才有用

本节代码github地址: <https://github.com/binyuc/Texas-poker-analyze>

Part3. 可用性与产品开发

一个好的idea需要有一个成功的载体，开发一个算法很容易，但开发一个产品比想象中的还要复杂

1. Idea与产品:

我最近在玩一款塔防手游。手游的内容不是重点，重点是我发现了2、3这款手游的玩家自制软件。基本上是，我点进去看过一个，附了他的github地址。然后这个手机游戏的玩家们就制作了一个产品，很多优秀的产品内核，其实就源自一些朴实无华的idea，比如说健身镜，比如说。很多他们往往都是源自于一个想法“要是有一个XX就好了”。当我做了这么多，我也随之迸发出了这个想法，要是有一个界面、一个工具，拥有一个载体之后我就可以稳定持续更新它了。

2. 开发的故事:

作为一个数据分析师，其实我的视角更多聚焦在很后期，我知道开发一个产品会有哪些流程，但是从没有自己参与过。开发一个“产品”，首先你得有个服务器。其次是得有一个整体的服务架构，我后端框架用的是Flask，前端框架是jinja2+Bootstrap。再然后我脑子里想象了一个简单的页面，我大概需要把这些东西放在哪，页面ui是什么样子，用户上来应该点什么。

平时我们容易忽视的每一个按钮，每一个浮窗，任何一个小到不能小的功能，只有自己在开发的时候，才能意识到这些东西组合起来的不易，因为每个点背后都是大大小小的逻辑和代码。

德州扑克胜率计算器

2022-05-15 by [Binyu](#)

Flop 翻牌 花色 红桃 牌面 A	Flop 翻牌 花色 梅花 牌面 Q	Flop 翻牌 花色 黑桃 牌面 K	Turn 转牌 花色 方片 牌面 5	River 河牌 花色 None... 牌面 Choose...
我的手牌1 花色 方片 牌面 J	我的手牌2 花色 方片 牌面 Q			

胜率计算

德州扑克胜率算法结果

100.00%
计算耗时 0.54855秒
整体胜率情况 我的胜率是 71.81%，打平的概率是 3.14%，输掉的概率是 25.05%
我的牌型概率 牌型是【高牌】的概率是 0.0%，牌型是【一对】的概率是 60.87%，牌型是【两对】的概率是 26.09% 牌型是【3条】的概率是 4.35%，牌型是【顺子】的概率是 8.7%，牌型是【同花】的概率是 0.0% 牌型是【葫芦】的概率是 0.0%，牌型是【炸弹】的概率是 0.0%，牌型是【同花顺】的概率是 0.0% 牌型是【皇家同花顺】的概率是 0.0%
对手的牌型概率 牌型是【高牌】的概率是 29.83%，牌型是【一对】的概率是 48.79%，牌型是【两对】的概率是 14.6% 牌型是【3条】的概率是 2.52%，牌型是【顺子】的概率是 3.7%，牌型是【同花】的概率是 0.0% 牌型是【葫芦】的概率是 0.53%，牌型是【炸弹】的概率是 0.02%，牌型是【同花顺】的概率是 0.0% 牌型是【皇家同花顺】的概率是 0.0%

3. UX:

UX是用户在使用产品过程中建立起来的一种纯主观感受，虽然是一种主观的态度，但最终体现出来的是各个漏斗上的指标数据，甚至关乎到一个用户最终会不会留存。日常我们做看板也好，体验产品也好，总是下意识的去考虑一些使用效果，“这样设计业务or用户用起来是否方便？”

本节成品网站地址：<https://www.chereby.com/poker>

本节代码github地址：https://github.com/binyuc/chereby_blog

Part4. 策略、博弈与AI

最后回到我们的主题德扑，我们聊一下德扑里的博弈，以及目前有哪些工具。

什么是博弈论

简单讲就是一群聪明人，在规则之下，如何选取**最有利自己的策略**进行竞争。

比如大家一起用筹码玩扑克，无规则的情况，一旦有人为了不输钱开始偷看，随着时间累加，大家都会采取对自己的最优策略，于是大家都开始偷看牌，这样偷看牌的策略就都**抵消**了，这也是为什么规则诞生。

什么是纳什均衡

在说GTO之前我们再聊一下什么是纳什均衡：

大家都知道的「囚徒困境」，两名罪犯被警察抓住，谁先坦白将从宽量刑，仅判 3 个月；而被供出的人将被判 10 年；如果双方都坚持，最后都将无罪释放；但如果都交代，那么就各判 5 年。

在这种情形下，由于两人被隔离，他们无法串供，博弈就开始了，每个人都是从利己的目的出发。显然选择坦白交代是**最有利自己的策略**。

而博弈论展现在我们日常生活中无处不在。比如说公司中，每次MRD，研发和产品对需求，比如说投放和运营收割这种。

为解释纳什均衡，再举个栗子：A，B玩家博弈，规则是两个人各自选择从1到9的任意一个整数，如果两个人选的数字之和不大于10，则A、B玩家各自获得所选数目的奖金，反之双方一分钱也拿不到。

如果A选择“4”，B为了是**利益最大化**会选“6”，记为(4,6)；相反，如果B选择的是“6”，A为了利益最大化就只能选“4”，因此(4,6)就是一个纳什均衡点。相应地，(1,9)、(2,8)、(3,7)、(5,5)、(6,4)、(7,3)、(8,2)、(9,1)也都是纳什均衡点。但以下几点是例外：

1. 如果信息透明，也就是像下象棋一样，A先选，B知道A选择之后再选择，这种情况下，一定是(9,1)
2. 如果信息不对称、不透明，就涉及到有没有bluffing，有没有欺骗等等，所以结果很有可能达不到纳什均衡点。

什么是GTO

在德州扑克中，GTO和剥削玩法是两种极端反例。

GTO：观察对手的基准线后，通过计算EV，随机且平衡自己的玩法，如果对手不使用GTO策略，长期来看我们的EV就是好的。

剥削式打法：就是没有平衡，就是极度压榨你的价值，大程度改变自己的玩法去掠夺对方的价值。

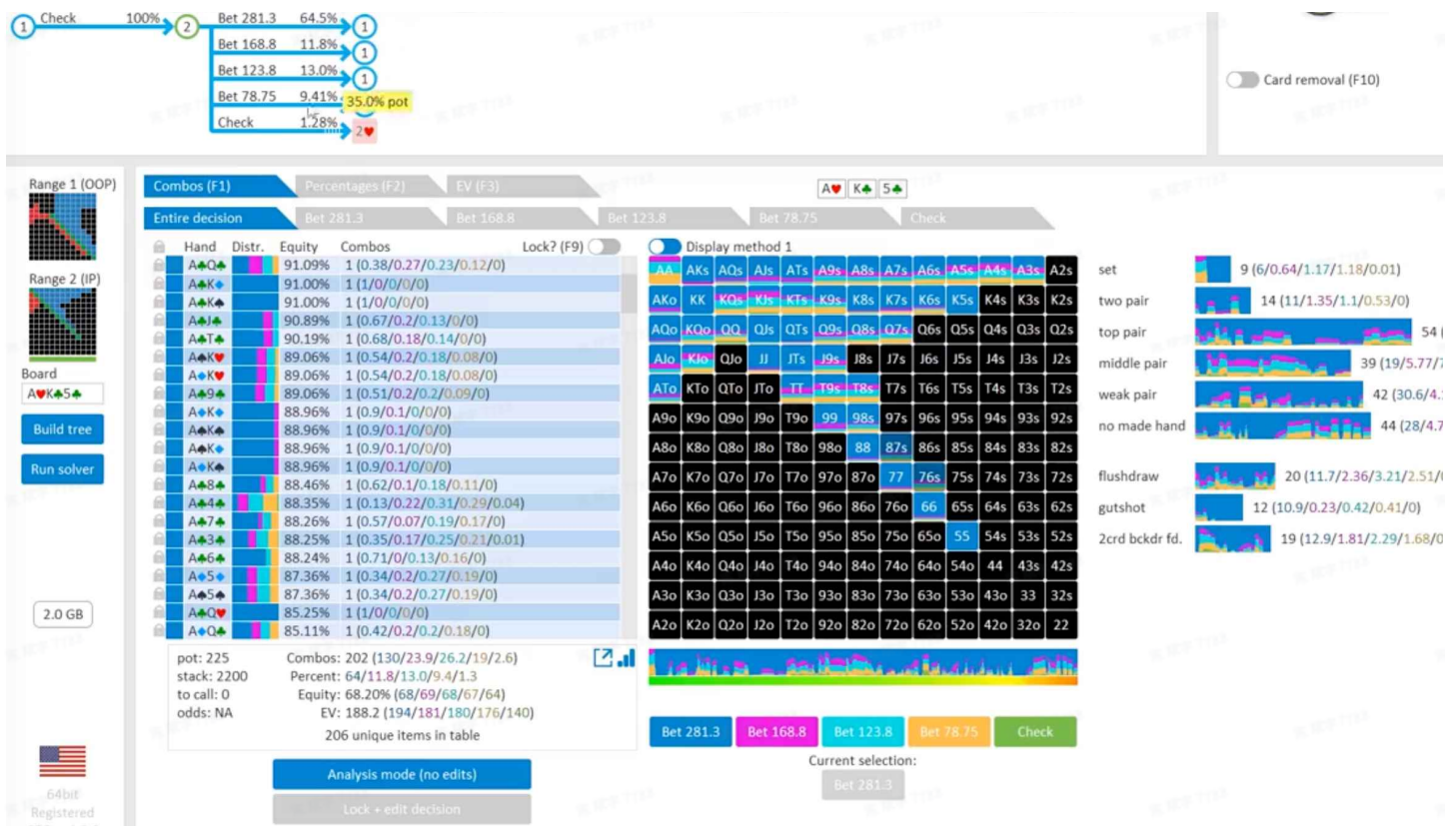
如果那足球比赛做类比：

GTO：攻守平衡，合理利用贝叶斯理论

剥削打法：猛攻对手右路弱点，但同时暴露自己后防的漏洞

人类只能尽可能模拟GTO，人类很难执行完美的GTO策略，因为对随机性要求很高。且人脑很难遍历复杂的GTO策略树。

现有的德扑软件、AI是什么样的



Reference:

Playing a toy poker game with Reinforcement Learning、All In! 我学会了用强化学习打德州扑克

Rich Zhu_GTO理论 - 德扑_德州扑克、2+2论坛 - 德扑_德州扑克

德州扑克中GTO理论的原理是什么? - 知乎

<https://www.zhihu.com/question/19804990/answer/720617947>

剑指 Offer 61. 扑克牌中的顺子 - 力扣 (LeetCode)