深 圳 大 学 实 验 报 告

课程名称 高级算法设计与分析

项目名称 基于树的搜索算法

学　　院 计算机与软件学院

专　　业 计算机科学与技术

指导教师 _____

报 告 人 _____ 学号 _____

实验时间 2022 年 4 月

提交时间 2022 年 4 月

教务处制

背景

库恩扑克（Kuhn Poker）是一个形式简单的扑克游戏，它由哈罗德·库恩（Harold W. Kuhn）开发，模拟两人在不完全信息下的零和博弈。

在库恩扑克游戏中，一副牌只包括三张扑克牌，如 1，2 和 3（或者是 J，Q 和 K）。游戏开始时，向每位玩家发一张牌，可以像标准扑克一样下注。如果两个玩家都下注或两个玩家都过牌，则拥有较高牌的玩家获胜，否则，下注玩家获胜。

具体游戏流程如下，

- 每名玩家先押注一枚赌筹（Ante）。
- 每个玩家分发三张牌中的一张，第三张牌被放在一边，不能被看到。
- 玩家一可以选择让牌（CHECK）或是加注（BET）。
  - 如果玩家一选择**让牌**，那么玩家二可以选择让牌或加注。
    - 如果玩家二也选择**让牌**，那么两人翻牌比较大小，牌大者拿走 2 枚筹码（牌大者从对方赢取 1 枚赌筹）。
    - 如果玩家二选择**加注**，玩家一可以选择不跟（FOLD）或跟注（CALL）。
      - 如果玩家一**不跟**，那么玩家二拿走 3 枚筹码（从玩家一赢走 1 枚赌筹）。
      - 如果玩家一**跟注**，那么两人翻牌比较大小，牌大者拿走 4 枚筹码（牌大者从对方赢取 2 枚赌筹）。
- 如果玩家一选择**加注**，那么玩家二可以选择不跟或跟注。
  - 如果玩家二**不跟**，那么玩家一拿走 3 枚筹码（从玩家二赢走 1 枚赌筹）
  - 如果玩家二**跟注**，那么两人翻牌比较大小，牌大者拿走 4 枚筹码（牌大者从对方赢取 2 枚赌筹）。

玩家一和玩家二每一步可能进行的操作流程如图 1 所示，其中蓝色是玩家一的操作，橙色是玩家二的操作，加粗是游戏评判的结果。

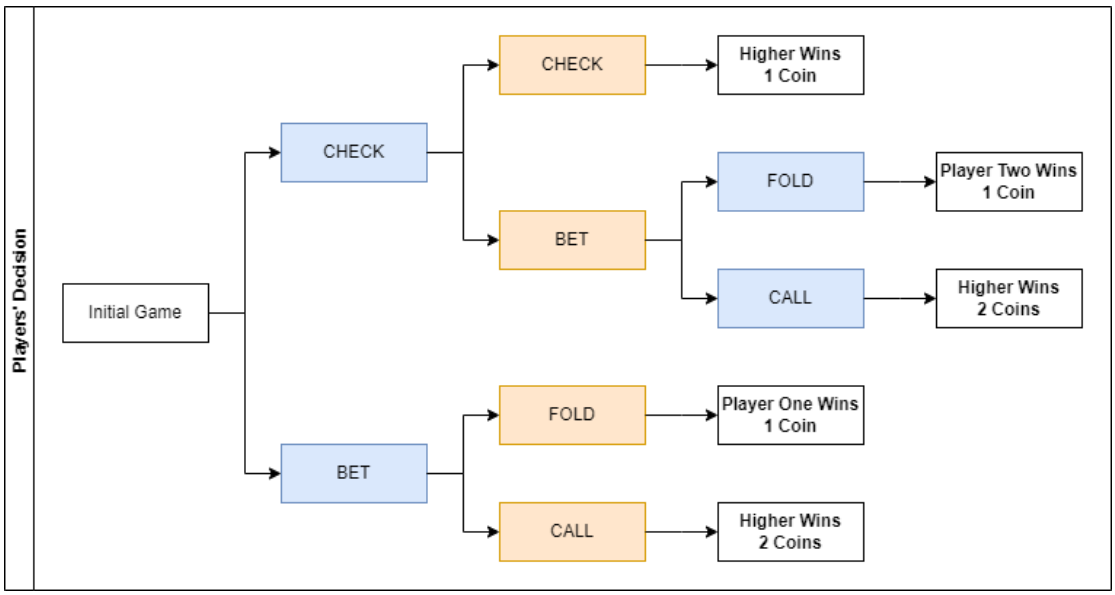

图 1 库恩扑克流程图

相关工作

关于库恩扑克的求解已经有许多相关工作取得进展，Lazaric 等人使用强化学习在无知识、完全知识和不完全知识的情况下给对手建模提高算法性能；Szafron 等人研究了三人库恩扑克问题，并且设计出一个参数化均衡策略族；Metzner 等人研究了作弊对库恩扑克游戏的影响，并且设计了对作弊的检测方法；Billingham 等人先是在三人库恩扑克问题中建立常微分模型，发现对策的平衡解作为常微分方程的解是渐近稳定的，在之后的工作中又使用基于规则的多项式方程和不等式研究了 N>3 情况下库恩扑克的平衡策略；Billings 等人建立了 Loki 扑克模型，该模型能观察对手，再通过建立对手模型和动态调整来发挥最佳利用对手的信息的优势。

算法与模型设计

博弈树的设计

博弈树的数据结构如图 2 所示（以 Python 语言设计为例），它主要包含 3 个类，其中 GameTreeNode 类是基础类，提供游戏涉及的两个关键信息扑克信息和操作记录信息；InitNode 类是表示游戏的类，在一场游戏开始时，初始化分配玩家的扑克、玩家的操作记录和赌注变化表；PlayerNode 类是表示玩家的类，它需要区分本轮的玩家还有目前位置的操作记录，并且根据策略进行出牌，判断游戏结果。

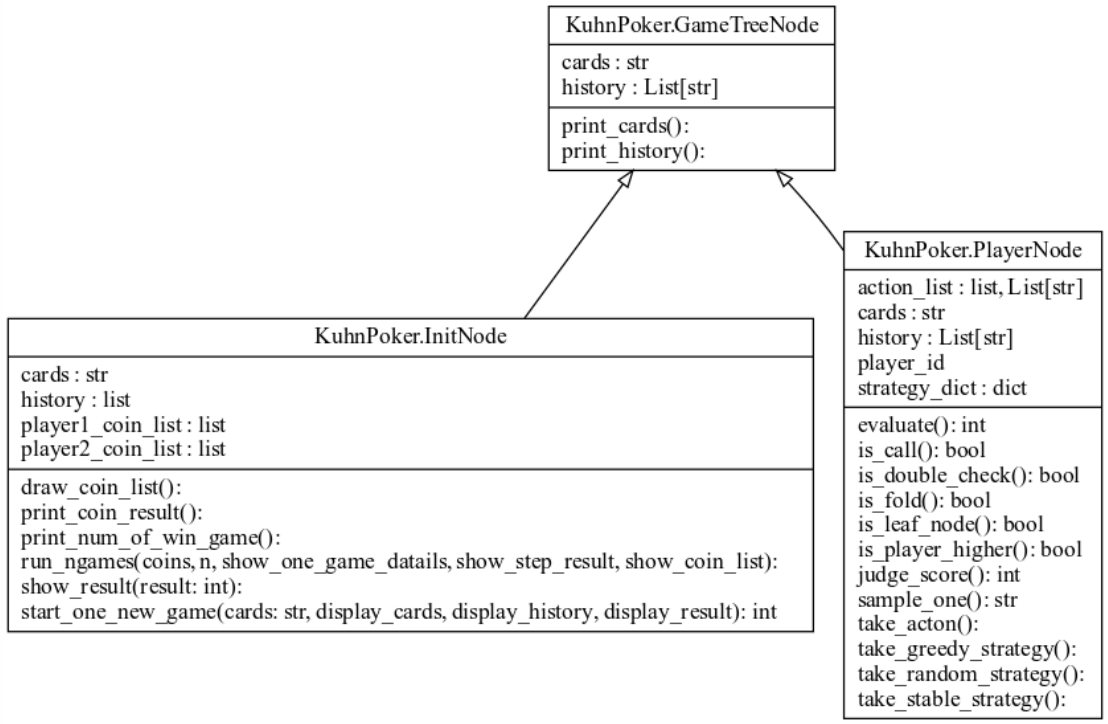

图 2 博弈树的数据结构

玩家策略设计

玩家在游戏中的出牌策略可以结合筹码的数量、历史操作记录、手中牌的点数和游戏的轮次等信息进行决策。本次实验，主要设计了三个简单的玩家策略，随机策略、固定策略、贪婪策略（我的算法）。这三种策略都相对简单，只使用了本轮的信息，并没有结合历史信息。下面具体说明三种策略，并提供策略的伪代码。

随机策略

随机策略是不管当前的牌型，随机地选出当前可能执行的操作。它的主要设计思路如算法 1 所示，

Algorithm 1: take random strategy

Input: *card*, player' poker;
id, player's id;
history, players' actions history
Output: *list*, list with all possible actions

```
1.  if history is empty:
2.      list = [CHECK, BET]
3.  else if BET is history's last element:
4.      list = [FOLD, CALL]
5.  else:
6.      list = [CHECK, BET]
7.  return list
```

固定策略

固定策略对于玩家一是大牌（如 3）一直加注（BET 或 CALL），小牌（如 1）一直过牌(CHECK 或 FOLD)，中牌（如 2）则先过牌（CHECK），再加注（CALL）。

对于玩家二则稍有不同，对于大牌和小牌，设计与玩家一相同，但玩家二只有一个回合，所以对于中牌，玩家二总是选择过牌。具体来说，当玩家一选择加注后，玩家二直接不跟（FOLD），选择认输；玩家一让牌，玩家二也让牌（CHECK）进行比较大小。

固定策略对玩家一和玩家二是不对称的，玩家一的灵活度更高一些。固定策略的主要设计思想如算法 2 所示，

Algorithm 2: take stable strategy

Input: *card,* player's pokers;
id, player's id;
history, players' actions history
Output: *list*, list with all possible actions

```
1.  if id is 1:
```

```
    if history is empty:
        if card is 3:
            list = [BET]
        else:
            list = [CHECK]
    else:
        if card is 1:
            list = [FOLD]
        else:
            list = [CALL]
if id is 2:
    if BET is history's last element:
        if card is 3:
            list = [CALL]
        else:
            list = [FOLD]
    else:
        if card is 3:
            list = [BET]
        else:
            list = [CHECK]
return list
```

贪婪策略

这个策略是本次实验个人设计的一种策略，它的设计思路是尽可能赢取赌筹。

贪婪策略对于小牌有 30%的概率行进加注，对于中牌 60%的概率进行加注，对于大牌则一定加注。

并且贪婪策略相对固定策略来说，玩家一和玩家二的操作是对称的。贪婪策略的主要设计思想如算法 3 所示，

Algorithm 3: take greedy strategy

Input: *card,* player's pokers;
id, player's id;
history, players' actions history
Output: *list*, list with all possible actions

```
if card is 1:
    if history is empty:
        list = [BET]*3 + [CHECK]*7
    else if BET is history's last element:
        list = [CHECK]
    else:
```

```
            list = [BET]*3 + [CHECK]*7
    if card is 2:
        if history is empty:
            list = [BET]*6 + [CHECK]*4
        else if BET is history's last element:
            list = [CALL]*6 + [FOLD]*4
        else:
            list = [BET]*6 + [CHECK]*4
    else:
        if history is empty:
            list = [BET]
        else if BET is history's last element:
            list = [CALL]
        else:
            list = [BET]
    return list
```

关键代码实现如下，以 Python 实现为例。

```python
def take_greedy_strategy(self) -> []:
        """
        Returns actions player maybe takes in greedy(my)
        strategy.
        Players will try best to BET in the game,
        if they get poker 1, they would BET(or CALL) with
        a probability of 0.3, for poker 2 is 0.6,
        and for poker 3 is 1.
        """
        if self.cards[self.player_id] == CARD1:
            if len(self.history) == 0:
                return [CHECK]*7 + [BET]*3
            else:
                return [FOLD] if self.history[-1] == BET \
                    else [CHECK]*9 + [BET]
        elif self.cards[self.player_id] == CARD2:
            if len(self.history) == 0:
                return [CHECK]*4 + [BET]*6
            else:
                return [FOLD]*4 + [CALL]*6 if \
                    self.history[-1] == BET \
                    else [CHECK]*4 + [BET]*6
        else:
            if len(self.history) == 0:
```

```
            return [BET]
        else:
            return [CALL] if self.history[-1] == BET \
                else [BET]
```

实验环境

我们实验使用的软件和硬件信息如下：

- Windows 10(64 bit)
- RAM 8.00 GB
- CPU 2.70 GHz
- Anaconda(Python 3.9)

实验结果

胜率比较

统计 1000 局游戏，玩家一和玩家二在分别使用随机策略、固定策略和贪婪策略（我的算法）的胜率（只考虑每局是否赢得赌筹，不考虑每次赢得赌筹的数目），结果取 10 次模拟的平均值，统计结果列于表 1，具体实验数据见于附录表 3。

表 1 胜率对比统计表

玩家一 \ 玩家二	随机策略	固定策略	我的算法
随机策略	56.0%:44.0%	58.4%:41.6%	49.9%:50.1%
固定策略	49.0%:51.0%	50.1%:49.9%	49.6%:50.4%
我的算法	58.7%:41.4%	54.3%:45.7%	52.4%:47.7%

观察表 1 可以发现，对于玩家一，采取贪婪算法可以有较高的胜率。特别是使用贪婪策略对抗使用随机策略的玩家二，这可能是因为玩家有时候手牌为 1 时加注，而玩家二此时选择了过牌，玩家一以较小的牌获胜，增大了总体的赢面。

玩家一采取固定策略应对玩家二（不考虑玩家二采取的策略），相对采取随机策略和规定策略，方差较小，胜率相对稳定。

对于玩家二，虽然贪婪策略并没有像玩家一使用时那么高的胜率，但相比于玩家二采取的其它两种策略，胜率有明显提升。并且使用贪婪策略对抗玩家一时，方差较小，胜率相对稳定。

当玩家一和玩家二采取相同策略对抗时，三种策略都是玩家一占优，而固定策略胜率差距最小，相对公平。

赌筹数量比较

统计 1000 局玩家一和玩家二在分别使用随机策略、固定策略和贪婪策略（我的算法）的最终赌筹（不考虑每局是否赢得赌筹，只考虑每次赢得赌筹的数目），结果取 10 次模拟的平均值，统计结果列于

表 2，详细实验数据见于附录表 4。

表 2 赌筹数量对比统计表

玩家一 \ 玩家二	随机策略	固定策略	我的算法
随机策略	3106.1:2893.9	2919.2:3080.9	2753.7:3246.3
固定策略	3141.0:2859.0	2841.0:3159.0	2941.2:3058.8
我的算法	3286.1:2731.9	2889.4:3110.6	2946.2:3053.8

观察

表 2 可以发现，对于玩家一，并没有策略能在 1000 局后保持正收益，但贪婪策略在三种策略中，赌筹的收益结果最大（损失最小）。

对于玩家二，随机策略的效果很差，对抗玩家一时均为负收益。而固定策略和贪婪策略的结果不相上下，固定策略相对稳定，贪婪策略更有可能创造最大收益。

当玩家一和玩家二采取相同策略时，随机策略使玩家一占优，固定策略和贪婪策略使玩家二占优。

赌筹数量变化

假定每名玩家有 3000 个赌筹，用图形表示出连续玩 1000 局之后玩家的筹码数量变化。为了避免出现某个特殊情况，计算 10 次的平均值。

注意玩家有先后，分别对 A 和 B 进行统计。如果筹码数量不能支持玩家玩 1000 局，则增加筹码。

如图 3 所示，使用堆叠条形图表示玩家一和玩家二在使用不同策略出牌 1000 局游戏后筹码的数量，R 表示 RANDOM，S 表示 STABLE，G 代表 GREEDY

结果明显表示出贪婪策略对随机策略有很大的优势。不管是玩家一还是玩家二使用该策

略对抗随机策略，1000 局游戏后大约能赢近 300 枚赌筹。

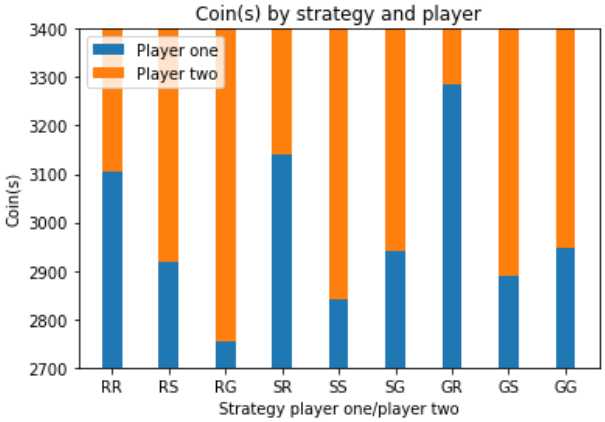

图 3 赌注变化图

讨论

结合表 1 和

表 2 可以发现，胜率和赌筹数量具有一定的 trade-off 效应，不可能兼顾这两个条件，并且没有某种策略同时适合玩家一和玩家二。

但总体来说，贪婪策略能使玩家一有较高的胜率收益，赌筹损失相对较少；并且使玩家二稳定的赌筹收益，胜率也较高。

总之，相对于随机策略和固定策略，贪婪策略显示出具有一定的合理性。

结论

我们设计的算法，玩家一胜率对随机算法是 58.7%，赌筹多了 286.1；玩家二胜率对随机算法使 50.1%，赌筹多了 246.3。玩家一对固定规则算法胜率是 54.3%，筹码多了-110.6；玩家二对固定规则算法胜率是 50.4%，筹码多了 58.8。

我们设计的算法，还可以在下面的一些地方进行改进：

- 本算法只考虑了一局游戏中的信息，还可以考虑多局游戏之间的联系改进；
- 本算法只考虑玩家自己的信息，还可以从考虑对手可能的信息、决策改进；
- 本算法在相似条件开局下选择了重复计算，还能从保存历史信息加速决策改进；
- 本算法无法兼顾胜率和赌筹两方面的优势，应对 trade-off 效应能力较差，还能从动态调整加注概率等方向改进。

参考文献

[1] Szafron D, Gibson R G, Sturtevant N R. A parameterized family of equilibrium profiles for three-player kuhn poker[C]AAMAS. 2013, 13: 247-254.

[2] Ernest J, Foglio P, Selinker M. Dealer's choice: The complete handbook of Saturday night poker[M]. Overlook Press, 2005.

[3] Bard N, Bowling M. Particle filtering for dynamic agent modelling in simplified poker[C]AAAI. 2007: 515-521.

[4] Ganzfried S, Sandholm T W. Computing equilibria by incorporating qualitative models[J]. 2010.

[5] Lazaric A, Quaresimale M, Restelli M. On the usefulness of opponent modeling: the kuhn poker case study[C]Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3. 2008: 1345-1348.

[6] Metzner A, Zwillinger D. Kuhn Poker with Cheating and Its Detection[J]. arXiv preprint arXiv:2011.04450, 2020.

[7] Billingham J. Equilibrium solutions of three player Kuhn poker with N> 3 cards: A new numerical method using regularization and arc-length continuation[J]. arXiv preprint arXiv:1802.04670, 2018.

[8] Gilpin A, Sandholm T. Lossless abstraction of imperfect information games[J]. Journal of the ACM (JACM), 2007, 54(5): 25-es.

[9] Billingham J. Full street simplified three player Kuhn poker[J]. arXiv preprint arXiv:1707.01392, 2017.

[10] Heinrich J, Silver D. Self-play Monte-Carlo tree search in computer poker[C]Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence. 2014.

[11] Abou Risk N, Szafron D. Using counterfactual regret minimization to create competitive multiplayer poker agents[C]AAMAS. 2010: 159-166.

[12] Begovich K, Cao X, Dohm P, et al. The Mathematics of Poker-like Games Project Report, Fall 2018[J]. 2018.

[13] Nash Jr J F. Equilibrium points in n-person games[J]. Proceedings of the national academy of sciences, 1950, 36(1): 48-49.

[14] Southey F, Bowling M P, Larson B, et al. Bayes' bluff: Opponent modelling in poker[J]. arXiv preprint arXiv:1207.1411, 2012.

[15] Zinkevich M, Johanson M, Bowling M, et al. Regret minimization in games with incomplete information[J]. Advances in neural information processing systems, 2007.

附录

实验数据

表 3 胜率统计表（玩家一）

玩家一策略	R	R	R	S	S	S	G	G	G
玩家二策略	R	S	G	R	S	G	R	S	G
1	56.9	59.0	49.6	51.7	50.3	47.6	57.1	53.8	50.4
2	56.3	57.5	50.5	49.8	47.5	52.2	58.6	55.7	53.8
3	54.2	59.0	49.8	50.6	50.8	51.8	57.6	53.3	53.6
4	57.0	57.2	47.6	47.4	48.3	49.8	60.7	52.6	50.7
5	54.7	59.6	51.8	49.4	52.4	49.9	60.6	54.7	50.5
6	54.3	57.3	51.1	47.6	50.8	49.6	59.3	56.0	52.2
7	57.7	60.4	50.0	47.0	51.0	48.3	58.7	54.4	52.5
8	57.0	57.5	48.8	48.6	52.3	48.1	60.0	54.2	53.5
9	55.2	58.0	50.0	47.2	48.4	47.1	57.6	52.2	53.8
10	56.3	58.6	49.6	50.8	49.1	51.2	56.3	56.0	52.5
平均值	56.0	58.4	49.9	49.0	50.1	49.6	58.7	54.3	52.4

R 表示 RANDOM，S 表示 STABLE，G 代表 GREEDY，单位为百分比（%）。

表 4 赌筹统计表（玩家一）

玩家一策略	R	R	R	S	S	S	G	G	G
玩家二策略	R	S	G	R	S	G	R	S	G
1	10	2930	2732	3181	2852	2887	3224	2887	2887
2	3125	2889	2781	3156	2793	3009	3278	2921	3000
3	3055	2940	2759	3183	2845	2997	3258	2844	2999
4	3127	2893	2687	3108	2796	2958	3349	2857	2902
5	3066	2951	2776	3189	2905	2937	3331	2899	2905
6	3051	2896	2777	3099	2857	2933	3287	2919	2932
7	3148	2975	2768	3103	2859	2945	3302	2894	2955
8	3136	2911	2715	3131	2877	2885	3333	2904	2957
9	3105	2898	2773	3109	2803	2872	3272	2835	2976
10	3111	2909	2769	3151	2823	2989	3227	2934	2949
平均值	3106.1	2919.2	2753.7	3141.0	2841.0	2941.2	3286.1	2889.4	2946.2

R 表示 RANDOM，S 表示 STABLE，G 代表 GREEDY，无单位。

实验流程

在实验中，每一局的游戏详情如下，第一行给出正在进行第几局游戏，第二行向玩家一和玩家二发牌。之后的两行或三行给出游戏过程，也即玩家的操作。紧接着判断本局游戏，哪名玩家赢得多少赌注，另一名玩家输了多少赌注。最后给出本局游戏后，玩家一和

玩家二各有多少赌注。游戏流程可以参考图 4。

This is game 980.
Player one gets poker 1, player two gets poker 3.
Player one takes CHECK.
Player two takes BET.
Player one takes FOLD.
Player two wins 1 coin from player one.
Player one has 2950 coin(s), player two has 3050 coin(s).

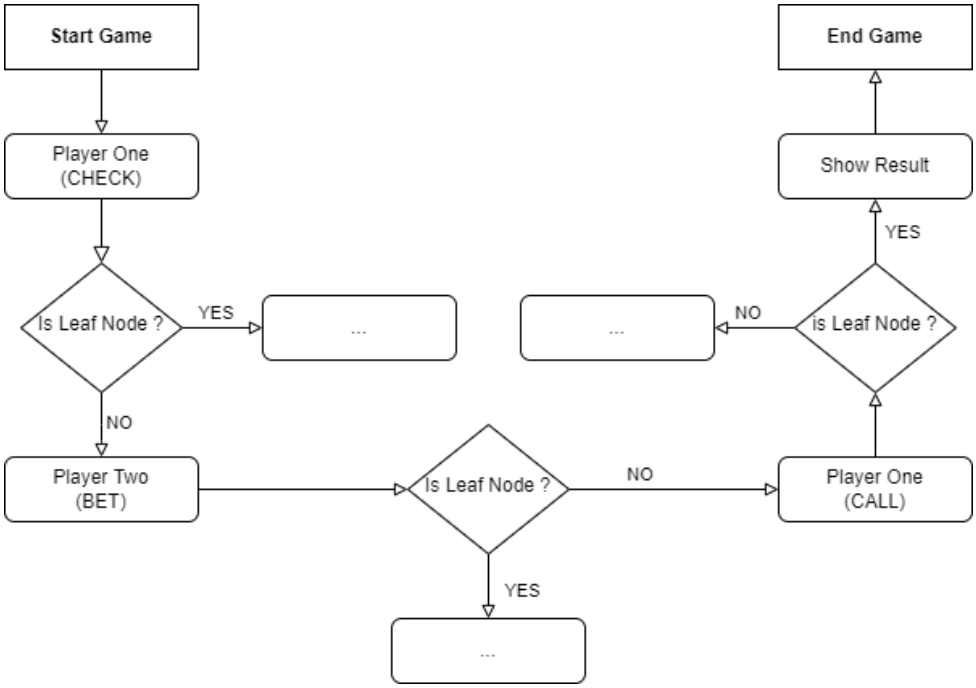

图 4 一个游戏例子

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。