- 玻璃桥游戏相关概率的计算与验证
  - 摘要
  - 关键词
  - 引言
  - 正文
  - 代码
  - 参考文献

玻璃桥游戏相关概率的计算与验证

摘要

本文将计算韩剧《鱿鱼游戏》中第七关"玻璃桥"游戏在理想情况下的各种相关概率并用计算机模拟加以验证。

关键词

统计模拟，二项分布，归纳推广

引言

游戏简介：

- 有16位选手要过一条玻璃桥。
- 玻璃桥由18组玻璃构成。每组玻璃由两块玻璃平行组成，1块是钢化玻璃，1块是普通玻璃。
- 钢化玻璃能承重，如果选手踩到钢化玻璃可以继续前行；但如果踩到普通玻璃，玻璃会碎掉，玩家淘汰。
- 玩家按号码依次前行，通过18组玻璃后算作存活。
- 后面的玩家可以记忆并跟随前一个人的路径

简化：

- 选手的判断完全随机，互相没有交流
- 在模拟游戏中并不考虑时间效应

正文

分析可得游戏的一个重要特征是某个玩家淘汰前的所有玩家都已被淘汰，通关后的所有玩家都会通关

先考虑只有四个桥结点的情况:

显然号码超过4的玩家必然通过，因为4块板最多淘汰4个玩家，故只考虑编号1-5的玩家

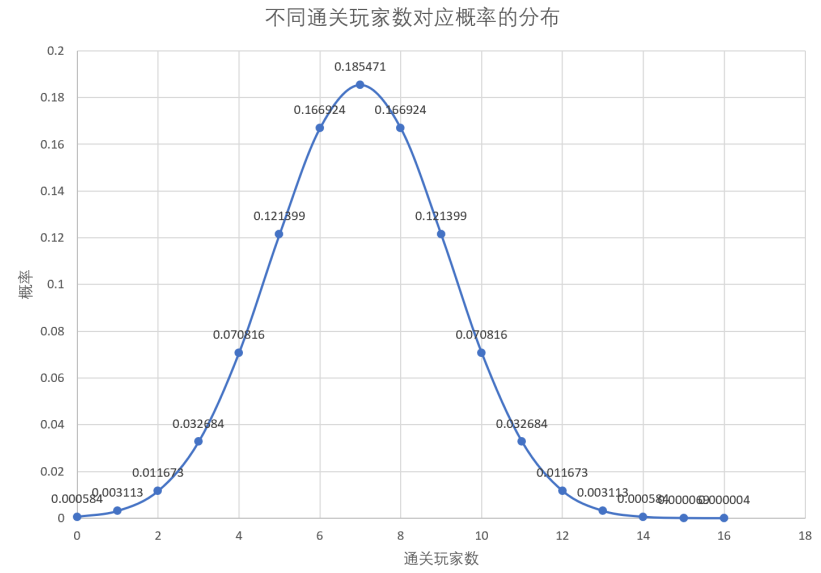
记p(i)为游戏过程中有i个玩家被淘汰(i块玻璃破碎),易知i个玩家一定是前i个玩家

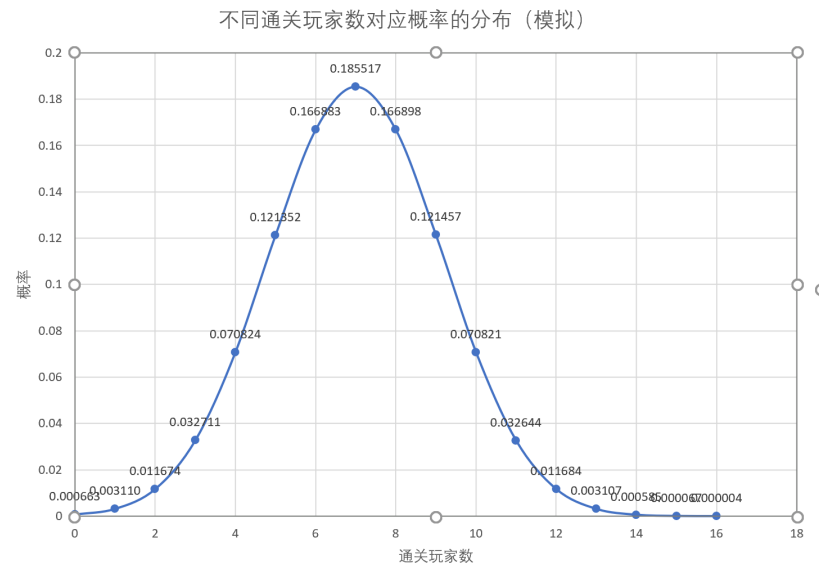
- $$p(0)=C_{4}^{0}\frac{1}{2}^0\frac{1}{2}^4=C_{0}^{4}/2^4$$

- $$p(1)=C_{4}^{1}\frac{1}{2}^1\frac{1}{2}^3=C_{1}^{4}/2^4$$
- $$p(2)=C_{4}^{2}\frac{1}{2}^2\frac{1}{2}^2=C_{2}^{4}/2^4$$
- $$p(3)=C_{4}^{3}\frac{1}{2}^3\frac{1}{2}^1=C_{3}^{4}/2^4$$
- $$p(4)=C_{4}^{4}\frac{1}{2}^4\frac{1}{2}^0=C_{4}^{4}/2^4$$

显然此事件满足二项分布b(4,1/2),淘汰i个玩家等价于通关5-i个玩家，推广可得通关k个玩家的概率$P_{succNum}(k)=p(M-k)=C_N^{M-k}/2^N$,k为通关玩家数，N为桥结点总数,M为玩家总数

16位玩家18个桥节点的情况下的概率分布如下图(程序模拟次数为一亿轮)





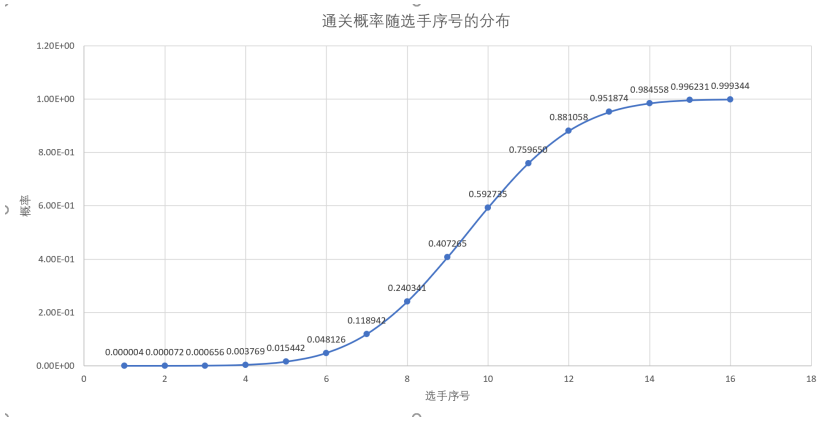
四个桥结点的5种情况对应5号玩家通关的各种情况，显然有5号玩家通关的概率$P_{success}(5)=\sum_{i=0}^4p(i)=\sum_{i=0}^4C_4^i/2^4=1$，如果4号未被淘汰，即p(4)对应的事件未发生，得到的将是$P_{success}(4)=\sum_{i=0}^3 p(i)=(\sum_{i=0}^3C_4^i)/2^4$.

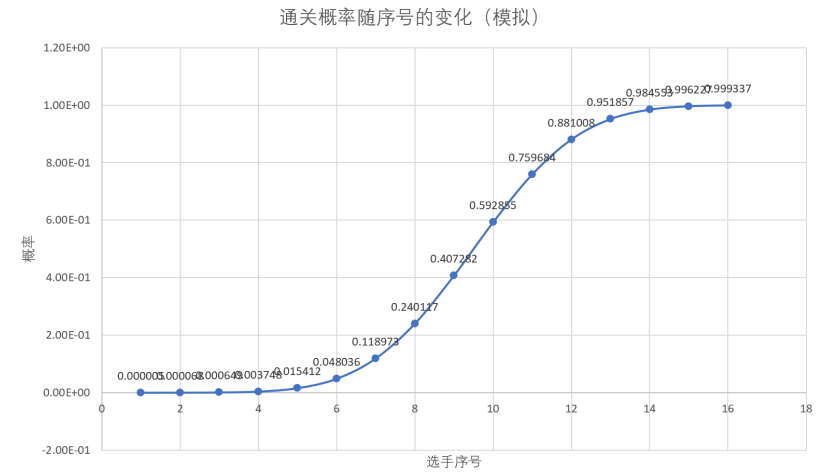
推广即可得$P_{success}(i)=(\sum_{i=0}^{i-1}C_N^i)/2^N,i<=N$，其中i为玩家编号，显然$P_{success}(i)=1,i>N$

为便于程序计算，可写为递推式$P_{success}(i)=\begin{cases} 0,i=0\P_{success}(i-1)+C_{N}^{i-1},0\leq i\leq N\1,i\geq N\ \end{cases}$

借助递推式利用程序计算有16个玩家，18个桥结点时各玩家通关的概率

结果为





可见16号主角是稳赢的

期望的计算

- $\sum (P_{success}(i)\times1)=7$
- $\sum(P_{succNum}(i)\times i)=7$
- 模拟结果:6.99988

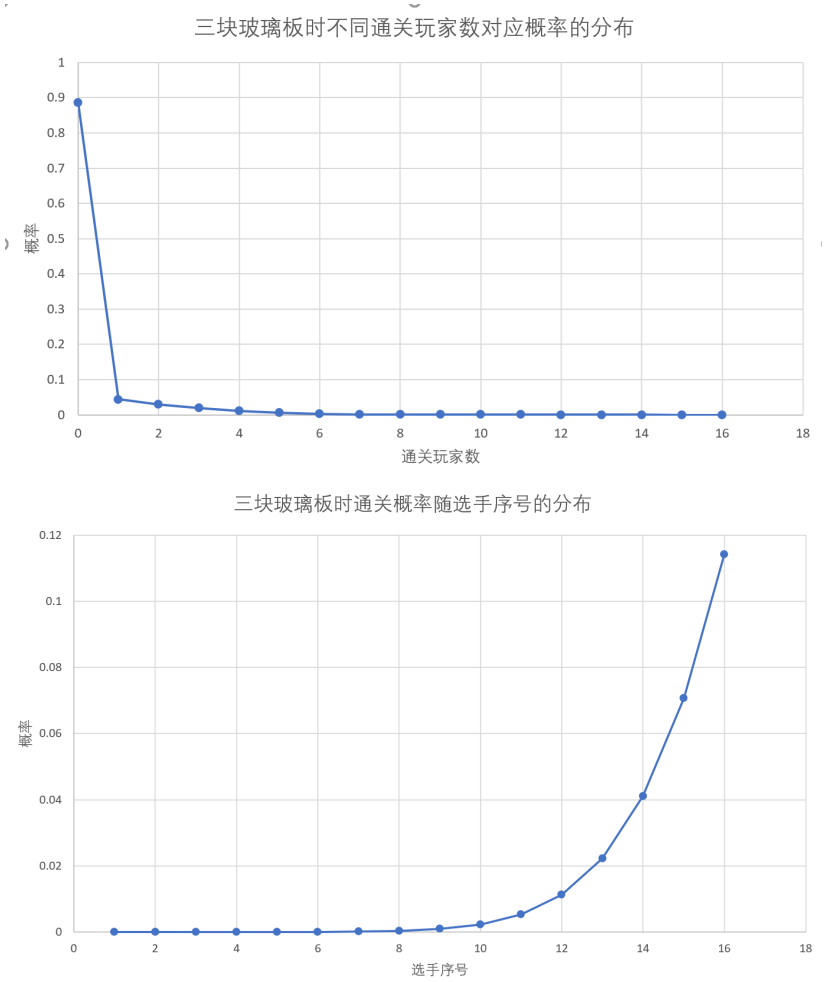
还可以从更直观的角度得到期望，每个玩家被一个玻璃板淘汰的概率为1/2,所以淘汰一个玩家平均需要2个玻璃板，即两个桥结点，本例中18个桥结点可平均淘汰9位玩家，剩余7位玩家。即$E(i)=N-M/2$

模拟结果

N\M	15	16	17
17	6.49903	7.49907	8.50006
18	6.00097	7.00046	8.00159
19	5.5041	6.50197	7.49751

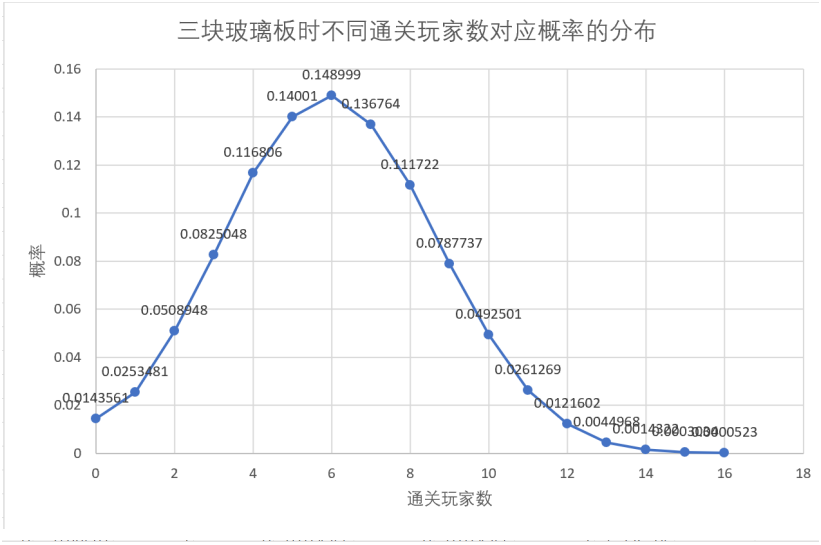
再将情况进一步推广，假设一个桥结点有3块玻璃，其中只有一块钢化玻璃，即普通玻璃的数量为2N,由于每次选择时被淘汰的概率不定(有三块玻璃时被淘汰概率是2/3，一块普通玻璃已破碎的情况下被淘汰概率是1/2)，故此时概率随通关人数的分布不再duiy服从二项分布，小规模的枚举也变得困难，限于笔者水平，在此只列举计算机模拟的结果

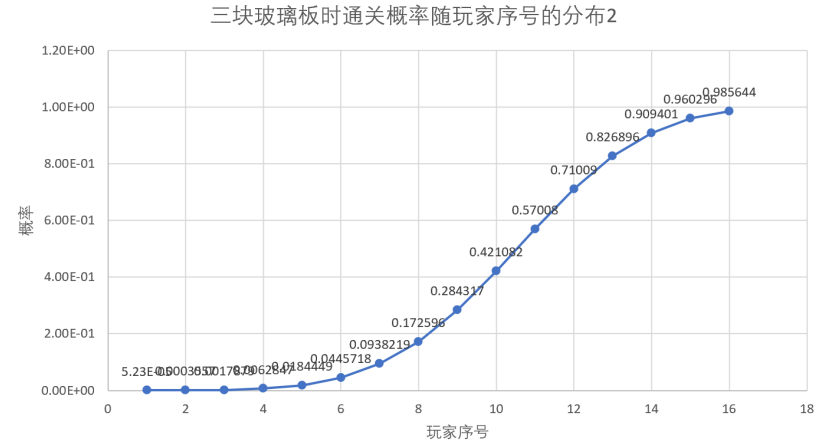
计算的结果如下



可以看出此时大概率没有人能过桥，"稳操胜券"的16号也只有大约百分之10的概率通关,游戏并没有什么意义

通过减小桥与玩家数的比例可以使游戏更合理，可以调整桥结点数量为9,人数依然为16





代码

```
#include <iostream>
#include <cstring>
#include <ctime>
#include <cmath>
#include <random>
#include "/home/fengsc/Desktop/cpp/CppTest/class/src/timeCount.h"
using namespace std;
//选择函数，选到0成功
#define select(x)                                                   \
    (                                                               \
        {                                                           \
            static default_random_engine e(time(NULL));             \
            static uniform_int_distribution<unsigned> u(0, x);      \
            u(e);                                                   \
        })
struct PlayerNode
{
    double passCount; //总的通关次数
    int outLocation;  //每场的淘汰位置
    PlayerNode() : passCount(0), outLocation(0) {}
};
struct BridgeNode
{
```

```
    int condition; //玻璃板状态，等于结点的普通玻璃数，破碎一块玻璃减一
};
struct Group
{
    PlayerNode *players;  //玩家数组
    BridgeNode *bridges;  //桥结点数组
    int *survivalNumber;  //各场存活玩家总数统计数组
    int *eachPassCount;   //各种通关人数统计数组
    int playerNumber;     //玩家总数
    int bridgeNumber;     //桥结点总数
    int glassNumber;      //一个结点上玻璃板总数(有且仅有1块钢化板)
    double totalNumber;   //试验次数总数
    double survivalTotal; //所有场次存活玩家总数
    Timer timer;          //计时成员
    Group(int _playerNumber, int _bridgeNumber, int _glassNumber, int
_totalNumber) :
     playerNumber(_playerNumber),bridgeNumber(_bridgeNumber),
survivalTotal(0), glassNumber(_glassNumber),
totalNumber(_totalNumber), players(new PlayerNode[_playerNumber + 1]),
eachPassCount(new int[_playerNumber + 1]()),
 survivalNumber(new int[_totalNumber + 1]), bridges(new
BridgeNode[_bridgeNumber + 1])
    {
        for (int i = 1; i <= _totalNumber; i++)
            survivalNumber[i] = _playerNumber;
        for (int i = 1; i <= _bridgeNumber; i++) //桥结点初始状态
            bridges[i].condition = _glassNumber - 1;
        players[0].outLocation = 1; //使第一位选手从第一个结点开始
    }
    ~Group()
    {
        delete[] players;
        delete[] bridges;
        delete[] survivalNumber;
        delete[] eachPassCount;
    }
    void reset() //重置准备下一轮
    {
        for (int i = 1; i <= bridgeNumber; i++)
            bridges[i].condition = glassNumber - 1;
        for (int j = 1; j <= playerNumber; j++)
            players[j].outLocation = 0;
    }
};
int main()
{
    Group G(16, 18, 2, 100000000); //G.playerNumber-G.totalNumber/2;平均两块
板淘汰一个玩家
    for (int k = 1; k <= G.totalNumber; k++)
    {
        for (int i = 1; i <= G.playerNumber; i++)
        {
            for (int j = G.players[i - 1].outLocation; j <= G.bridgeNumber;
j++)
```

```
            {//后一位可以保证到达前一位淘汰的位置的前一个位置,即在前一位淘汰的位置继续
判断
                if (G.bridges[j].condition &&
select(G.bridges[j].condition)) //需要选择，选了普通玻璃板，淘汰
                {
                    G.bridges[j].condition--; //破一块
                    G.players[i].outLocation = j;
                    G.survivalNumber[k]--;
                    //cout<<i<<' ';
                    break;
                }
            }
            if (G.players[i].outLocation == 0) //i号玩家顺利通关
            {
                G.players[i].outLocation = G.bridgeNumber + 1; //更新未淘汰玩
家的淘汰位置使后面的玩家顺利通关
                G.players[i].passCount++;
            }
        }
        G.eachPassCount[G.survivalNumber[k]]++;
        G.survivalTotal += G.survivalNumber[k];
        G.reset();
    }
    for (int i = 1; i <= G.playerNumber; i++)
        cout << "survivalRate of player " << i << " is " <<
G.players[i].passCount / G.totalNumber << endl;
    for (int i = 0; i <= G.playerNumber; i++)
    {
        cout << "Rate of " << i << " survivals is" << G.eachPassCount[i] /
G.totalNumber << endl;
    }
    cout << endl;
    cout << "Mockexpectation is " << G.survivalTotal / G.totalNumber <<
endl;
     //利用公式计算，仅适用于两块玻璃的情况
    if (G.glassNumber == 2)
    {
        int C[G.bridgeNumber + 1][G.bridgeNumber + 1] = {0};
        C[0][0] = 1;
        for (int i = 1; i <= G.bridgeNumber; i++) //组合数计算模块
        {
            C[i][1] = 0;
            C[i][0] = 1;
            C[0][i] = 0;
            C[1][i] = 1;
        }
        for (int i = 1; i <= G.bridgeNumber; i++)
            for (int j = 1; j <= G.bridgeNumber; j++)
                C[j][i] = C[j - 1][i - 1] + C[j - 1][i];
        double p[G.playerNumber + 1];
        p[0] = 0;
        double Count[G.playerNumber + 1] = {0}; //各个通关人数对应的概率
        double expection = 0;                   //期望
        cout << "ability of different succNumber: ";
```

```
        for (int i = 0; i <= G.playerNumber; i++)
        {
            Count[i] = C[G.bridgeNumber][G.playerNumber - i] / pow(2,
G.bridgeNumber);
            expection += Count[i] * i;
            cout << Count[i] << ' ';
        }
        cout << endl;
        cout << "exception is " << expection << endl;
        cout << endl;
    }
}
```

参考文献

[《《鱿鱼游戏》中【过玻璃桥】游戏的概率问题》RoseCanoe]
(https://www.cnblogs.com/rosecanoe/p/15440539.html)

[《使用统计学分析《鱿鱼游戏》中“玻璃垫脚石”的生存概率》deephub]
(https://zhuanlan.zhihu.com/p/424324137)