卒業研究報告書

題目

限定ジャンケンの必勝法について

指導教員

石水 隆　講師

報告者

12–1–037–0083

佐藤 立康

近畿大学理工学部情報学科

平成 29 年 1 月 23 日提出

概要

ジャンケンは偶然性に多くを支配されるゲームであるが、相手の癖や心理を読むことにより勝率を挙げられる場合もある。例えばサザエさんジャンケン研究所 [1] では、アニメ「サザエさん」のジャンケンのコーナーの出し手パターンの統計を取ることにより、7 割弱もの勝率を挙げているという。

また、レクリエーション等では様々な趣向を凝らしたジャンケンのバリエーションも多く用いられる。その一例として、相手は敢えて手を先に出し自分はその手に対して勝つ手を出すもの、誰よりも早くジャンケンで一定の回数だけ勝利を挙げることが目的となるものが挙げられる。[2]

こうしたジャンケンのバリエーションの一つに限定ジャンケン [3] がある。限定ジャンケンは、漫画「賭博黙示録カイジ」に登場するギャンブルの一つである。グー・チョキ・パーを出せる回数を制限することにより、どの手を残しておくのか、相手にどの手を出させるか、等の心理戦の要素が加わったものとっている。本研究では、この限定ジャンケンにおける戦略を検証する。

目次

1 序論

1.1 本研究の背景

ジャンケンの 3 つの手は、同じ手はあいことなるが、その他の 2 つの手において、一方の手には勝ち、一方の手には負ける。このように、ジャンケンには絶対的優位な立場を持つ手は全く存在せず、それぞれの手は三すくみの関係になっている。

出す手に優劣が無い以上、ランダム手を出しても特に有利でも不利でもなく、そこには戦略を練る余地は無さそうに思える。しかし、人間の出す手は何かしらの癖があり、必ずしも全ての手が 1/3 の確率で出るとは限らない。その癖を読み取ることができれば、有利な手を出しやすくする可能性はある。

また、限定ジャンケンでは予め一つのゲーム内で出す手が限られてくる。そのため、自分及び相手の、各々の手の残り回数によって、それぞれの手の出しやすさが異なってくる。したがって、限定ジャンケンではどう手の出しやすさを変えるかに、戦略が生まれてくるのである。今研究ではこのことについて重きを置いて研究を進めて行くこととする。

1.2 ジャンケンに関する既知の結果

ジャンケンについては様々な研究が行われてきた。複数人数で従来のグー、チョキ、パーを用いる 3 手ジャンケンでは、例えば 2 人なら、ジャンケンを行う回数が 1.5 回、3 人なら 2.3 回となるが、10 人の場合は 24.4 回となるなど、人数に比例しそれの平均回数が増えていく。[6]

また、ジャンケンに遺伝アルゴリズムを適用するという興味深い実験もある。対戦相手の出す手のパターンに癖があることを利用し、勝ちやすい手を遺伝アルゴリズムで選択しようとしたものである。実際に人間と対戦したところ、全体的な勝利数は 4 割弱に対し敗北数は 3 割程度といった結果が出ており、ある程度コンピュータは人の出す手を読めていると言える。[5]

さらに、石、鋏、紙、壷の 4 種類を用いたフランス式ジャンケンと呼ばれる 4 手ジャンケンや、水、鳥、石、板、ピストルの 5 種類を用いた 5 手ジャンケンがある。それを元に、1 つの手が他の手よりも、その手が勝てる手数が多い無駄な手を省いた。その「面白さ」最大のジャンケンである 6-トーナメントなどが編み出されてきた。実際にプレイングはなされていないものの、より多人数かつ面白いジャンケンを編み出すことに重きを置いている。[4]

一方で、自由度 B と時間 D の統計量の指標$\sqrt{\mathrm{B/D}}$よりゲームの面白さを把握するため、ランドマッチジャンケンを実際に試行した例もある。更に特別ルールとして連続で勝利した場合はその時点でゲーム終了とする。10 回のラウンドマッチジャンケンにおいて、2 連続勝利が 3 回で即勝利が、チェスの$\sqrt{\mathrm{B/D}}$の値が最も近いという結果が出ている。実験結果には長い歴史を経て淘汰されたチェス ($\sqrt{\mathrm{B/D}}$=0.074) と比較しており、10 回のラウンドマッチジャンケンもその値に近い値が出ている。[7]

最後に、繰り返し対戦型ジャンケンにおける戦略の特徴についても研究されている。相手の手に関係なく完全ランダムに手を出すでたらめ戦略、一つ以上前の相手の手と同じ手を出すものまね戦略、自分の一つ前の勝負の結果が、勝ち、引き分け、負けのそれぞれの場合の時、その次の手を 3 種類のいずれかの手にあらかじめ決めておく場合分け戦略、過去の相手の手の系列を、ある長さの履歴パターンの出現回数として記憶しておき、その履歴から、次の相手の手を予測し、それに勝つような手を出す履歴学習型戦略など、様々な戦略を決めておく。また、各々の戦略ごと内に幾つかパターンが考えられるため、それらも場合分けしておく。その結

果、計 39 パターンの総当たりを行ったところ、1 試合 1000 回、5000 回、10000 回いずれの場合においても履歴学習型戦略の全てのパターンが 8 割以上の勝率を上げている。次いで勝率が高いのはでたらめ戦略で、ほぼ五分五分の勝率となっている。[8]

1.3 限定ジャンケン [3]

限定ジャンケンとは、漫画「賭博黙示録カイジ」に登場するギャンブルの一つであり、第 2.1 章のルールに則り行われた。原作ではグー、チョキ、パーが描かれたカードを 4 枚、計 12 枚を持ち、相手とジャンケンをする際にカード (手) を出しジャンケンを行い、其のとき使ったカードはボックスに回収され再度使用は不可となる。そのため、自分が出せるそれぞれの手の回数はカードの枚数によって決まる。

ジャンケン以外の用途でカードを破棄するなどのルール違反を行った場合、即失格となる。本研究でも (星が 0 個になり途中でゲームが終了する場合を除いて) 必ず 12 枚のカードを使い切らなければならないものとする。

1.4 本研究の目的

本研究では限定ジャンケンにおける勝率を上げるため、どのような AI が最も勝ちやすいかを調べることにある。そのために、実際にプログラムを動かし、それぞれの AI 同士を戦わせることが考えられる。本研究の詳細は章で詳しく解説する。

1.5 本報告の構成

本報告の構成は以下の通りである。まず第 2.2 章で限定ジャンケンのルールと作成する AI について述べる。第 3 章で本研究で作成したジャンケンプログラムについて説明し、第 4 章で実験結果を示し、考察を行う。最後に第 5 章でまとめと今後の課題について述べる。

2 研究内容

2.1 限定ジャンケンのルール

本来の限定ジャンケンは、不特定多数の人数が参加する、２回目以降の参加者と初参加の者とは勝利に必要な得点が違う、その他様々な心理戦を要する追加ルールがあるが、今回はそれら追加ルールは使用せず、基本ルールのみを用いる。

限定ジャンケンの基本ルールを以下に記載する。

- 2 人で対戦する。
- 各プレイヤーはそれぞれグー、チョキ、パーを 4 回まで出せる。
- 各プレイヤーは、最初“星”と呼ばれる持ち点 3 点を持っており、勝てば +1 点、負ければ-1 点される。
- 12 回ジャンケンを行うか、どちらかの星が 0 点になった時点で終了。星が多い方の勝利となる。(同点の場合は引き分けとなる)

また、追加ルールを以下に記載する。

- 参加者が多数おり、任意の相手と同意があれば対戦できる。同じ相手と繰り返し対戦してもいいし、対戦ごとに相手を変更してもよい。
- 12 回ジャンケンを行うか、星が 0 になった時点で終了。終了時の星の数が、初参加の参加者は 3 個以上、2 度目以降の参加者は 4 個以上あれば勝利となる。

追加ルールでは、各参加者は任意のタイミングで対戦を行うため、参加者ごとに対戦回数が異なる場合がある。このため、残りカード枚数が対戦相手によって異なり、残りカードの少ない相手を選んで対戦する、といった戦略が生まれる。

本研究では、限定ジャンケンに対する最適な戦略を検証するために、様々な戦略対するジャンケン AI を作成し、対戦させる。

2.2 作成する AI とそれらの狙い

本研究では以下の 6 つの戦略を従うジャンケン AI を作成し、各 AI 間で対戦させた。

- 戦略 A…カードの枚数に関係なくランダムで手を出す。
- 戦略 B…相手の出せる手が二つに限定されている場合、自分はその二つの手に勝てるか引き分ける手を優先的に出す。
- 戦略 C…戦略 B の内容に加え、自分の各々の残り回数が 1 となった手は絶対に使わず、2 の場合は通常の半分の確率で出す。
- 戦略 D…自分の現在のカードの枚数に比例して手を出す。
- 戦略 E…戦略 D の内容に加え、相手の手の残り回数が多ければ、その手に対し勝てる手を出し易くする。
- 戦略 F…戦略 E の内容に加え、負ける手を出しにくくする。

以下各戦略の狙いを説明する。

全ての手を無作為に出す戦略 A は対照実験としてのサンプルを取る為の AI である。

戦略 B の理論は至って単純である。例えば相手がチョキ或いはパーしか残っていない場合について考える。

- 自分がグーを出す時…相手がチョキならば勝ち、パーならば負ける。
- 自分がチョキを出す時…相手がチョキならば引き分け、パーならば勝つ。
- 自分がパーを出す時…相手がチョキならば負け、パーならば引き分ける。

以上より、仮に相手の出す手がランダムだったとすると、引き分けるか勝つかのパターンしか考えられず、決して負けることのないチョキを最優先で出すこととする。また、チョキが 1 回も出せない状況の場合、勝つか負けるかのパターンしか考えられないため、次に優先度が高いのはグーとなる。チョキとグーが両方とも 1 回も出せない状況の場合はパーしか出せないため、パーを出す。また、章より三すくみが成り立つため、相手がパー或いはグーしか残っていない場合、相手がグー或いはチョキしか残っていない場合においても同様に確からしく証明できる。

続いて戦略 C だが、元々の戦略 B に、以下の内容を付け加えたものである。そもそも戦略 B は相手の手が 2 種類に限定された時に有利に発揮する戦略であるため、その逆で自分の手がなるべく 2 種類にならないように手を出すようにする。そのため、現在の残り回数が 1 である手を出せば必ずその手はそれ以降出せず、2 種

類以下しか出せないため絶対出さないようにする。また、その残り回数が 1 の状態をなるべく起こさないため、残り回数が 2 の時は通常の半分の確率で出すようにする。

戦略 D は戦略 C の亜種であり、単純に各々の手の残り回数が多い手ほど出しやすくする。こちらは戦略 C よりランダム性が高く、例え残り回数が 1 の手も可能性は低いが出し得る。後述の戦略 E,F との対照実験として用いるための AI である。

戦略 E,F は勝つ可能性が高い手ほど出しやすくする戦略である。

戦略 E の内容を説明する。例えば自分の出す手の残り回数において、グー、チョキ、パーが 1、3、4 とする。相手も同様に 3、1、4 とする。自分がグーを出すとき相手はチョキであれば勝つので、このパターンが起こる可能性はそれぞれの残り回数を乗算した 1 × 1=1 とする。同様に自分がチョキの場合は 3 × 4=12、パーの場合は 4 × 3=12 とする。即ち、この場合で自分が最も勝ちやすい手はチョキとパーとなる。また、この戦略は戦略 B の内容と、戦略 C の内容の一部を踏襲しており、例えば相手のグーの残り回数が 0 の時、自分がパーを出す場合に勝つ可能性は、それの残り回数に関係なく 0 となるため、全く出さないことが可能となる。また、戦略 D 同様に自分が手を出す時に、自分の手の残り回数に応じて出す確率を変えるため、戦略 C の内容の一部も含まれている。

戦略 F はそれを更に高度化させたもので、負ける手を出しにくくするものである。それを表現するために、相手に勝つ手の残り回数の二乗と、引き分ける手の残り回数をかけたものを可能性として算出している。先述の通りの例で挙げると、自分がチョキの場合は 3 × 4 × 4 × 1=48 となり、一方でパーの場合は 4 × 3 × 3 × 4=144 となる。つまり、戦略 E の場合は同等の確率だったチョキとパーに対し、戦略 F では明らかにパーが出しやすくなっている。

3 ジャンケンプログラム

本章では、本研究で作成したジャンケンプログラムについて説明する。付録に本研究で作成したジャンケンプログラムのソースを示す。

3.1 GenteiJanken.java

限定ジャンケンのルール根幹、並びに各種戦略を搭載したプログラムである。

- フィールド、コンストラクタ…各種設定。肝心の自分と相手の AI(戦略) 設定はこのプログラムのフィールド部分で書き換える。それぞれの戦略と数値の対応について、戦略 A…0、戦略 B…1、戦略 C…2、戦略 D…3、戦略 E…4、戦略 F…5 となる。
- view,judge,result メソッド…それぞれ各回のジャンケン結果表示、ジャンケンの勝ち負け判定、12 回ジャンケン終了後の勝敗表示を行うメソッド。
- main,game メソッド…main メソッドで game メソッドの処理を行わせ、game メソッドで実際に上記の処理を記述している。なお、game メソッド内には後述の ai メソッドをコメントアウトし、戦略比較用として全てグーを出すプログラムも記載している。game メソッドは後述の GJRepeat.java で実行結果を反映させるため、実行結果を int 型で返している。
- ai メソッド…各々の戦略について記述している。詳細は後述。

メインとなる ai メソッド部分に関して記載する。各々の戦略に対して分岐する。分岐先の各々のプログラムは原則独立しているが、ランダムで手を出す部分等、ある程度同じ内容のプログラムは random メソッドでまとめた。ai メソッドで確定させた手はメソッドの終了に該当の手の残り回数を減らす上に、各々のプレイヤーに対し、各々の戦略で実行される。そのため、例えば相手プレイヤーを実行させる場合、自分プレイヤーの残り手の回数が減った状況で行われてしまうため、帳尻合わせのために、一時的に自分の出す手を足しておく。

このプログラムを実行すると限定ジャンケンが 1 度だけ試行される。

3.2 GJRepeat.java

GenteiJanken.java に記載されたプログラムを、複数回反復して行わせるプログラムである。その回数もやはり GJRepeat.java のフィールドで設定する。

- フィールド…各種設定。試行回数はこのフィールド内で設定する。
- main メソッド…試行回数分だけ限定ジャンケンを行う。28 行目、int x = gj.game(); では、この試合で勝利すれば 0、敗北すれば 1、引き分ければ 2 を、x の値に入力する。0,1,2 の数がそれぞれ勝利数、敗北数、引き分け数の回数となるため、最終的にそれらの回数がどうなったか表示される。

このプログラムを実行すると限定ジャンケンが指定された回数分試行される。今回は 10000 回固定で行わせた。

4 実験結果

各戦略間で対戦させた実験結果を表 1 に示す。表 1 の数値は、各戦略で AI1、AI2 が対戦したときの AI1 の勝率:敗率である。それぞれの試行回数は 10000 回である。

表 1　各戦略間の対戦における勝率 (試行回数 10000 回)

AI1 \ AI2	A	B	C	D	E	F
A	34:34	34:36	33:36	34:35	37:32	41:31
B	37:34	40:40	49:30	34:35	42:30	39:33
C	37:33	30:48	43:42	35:35	34:38	27:43
D	34:34	33:35	34:35	34:35	35:35	33:37
E	32:37	30:41	37:33	36:34	33:33	33:35
F	31:40	32:40	44:26	37:32	35:33	34:33

4.1 考察

AI1 が戦略 B の時、戦略 D に対して敗率が僅かに上回った事以外は、勝率が敗率を大きく上回っている。従って、勝率を総合的に評価し、最も強い戦略は B であることが判明した。これは中盤以降で起こりうる、相手のそれぞれの手の回数が制限されたときに真価を発揮する戦略の内容が非常に有効的であったことを証明

する。

一方で、AI2 が戦略 C の時は戦略 B の時よりも全体的に勝率が劣り、特に戦略 B と対戦させた成績は、戦略 C の勝率が敗率を大きく下回った。これは自分が均等よく手を出すことによって終盤で勝ちにくくなってしまい、裏目となってしまったと考えられる。

戦略 D の時は戦略 A とほぼ同様に、相手のそれぞれの手の残り回数を参照しない、ランダム要素の強いものであるため戦略 A とは大差変わらない結果となった。

さて、戦略 E の勝率であるが、戦略 A と戦略 B の敗率を下回り、戦略 C の敗率は上回った一方で、戦略 D,E,F とは敗率と大差ない結果となった。先述の通り、戦略 E の内容は戦略 C のそれを踏襲した形となっているため、同様のことが言えるではないかと考えられる。

一方で戦略 F の勝率は戦略 E のそれと大差ないが、戦略 C の敗率を大きく上回っている。また、戦略 E での勝率では敗率よりわずかに上回っている。戦略 F の内容は戦略 E のそれを踏襲し、先述の通り戦略 E と C においても同様のことを示す。以上により、前文の内容が実証されたと思われる。

また、全体的に勝率を 5 割を超えた対戦は一つも存在しない一方で、勝率を 3 割を切った対戦は一つ存在するのみに留まった。一方で、勝率と敗率がほぼ同じ対戦が多かった。章に示す通り、ジャンケンは偶然性に多くを支配されるゲームであるため、ランダム性の高いゲームではある。しかし、今回の実験にあたり、戦略 A に対する勝率が 5 割を超える、決定的な必勝法となる戦略を期待していたが、1 つも見つからなかった。

5　結論・今後の課題

本研究では限定ジャンケンの戦略を検証した。

今後の課題としては、より勝率の高い戦略を作ること、また、計算機による試行だけではなく、その戦略が確率的に最適となることを証明することが挙げられる。

具体的には、戦略 B のような相手のそれぞれの手の残り回数が制限された終盤で発揮した内容とは対照的に、比較的自分や相手の手の残り回数が大差ないような序盤や中盤で勝率を上げる方法を模索することにある。或いは、戦略 E や F とは別の方法で勝率を上げる内容を考案することも考えられる

また、追加ルールについても検証し、より原作の限定ジャンケンに近づけたルールで勝率を上げる方法で実践することも今後の課題である。先述の追加ルールを一つずつ追加していき、前述の通り計算機のみならず実践によって証明していき、より実践に近い戦略を確立することが目標となるだろう。

謝辞

卒研レジュメやこの卒業論文の推敲のみならず、卒業研究のためのプログラム作成の際、上手いようにデータが取れないために私の作成したプログラムのチェックを受けて下さった、石水隆講師には多大な感謝と敬意を表します。

参考文献

[1] サザエさんじゃんけん研究所, http://park11.wakwak.com/~hkn/

[2] 弥延浩史：バリエーションいろいろ！「じゃんけんレク」 - 絆を深める×笑顔があふれる, (2016) http://www.meijitosho.co.jp/eduzine/icebreak/?id=20160053

[3] 福本信行：賭博黙示禄カイジ, 講談社 (1996)

[4] 伊藤大雄, 一般化ジャンケン, オペレーションズ・リサーチ, Vol.18, No.3, pp.156-160, 2013, http://www.orsj.or.jp/archive2/or58-03/or58_3_156.pdf

[5] 山下将臣, じゃんけんゲームに対する遺伝的アプローチ, 高知工科大学 情報システム工学科 平成 20 年度学士学位論文, 2009, http://www.kochi-tech.ac.jp/library/ron/2008/2008info/1090396.pdf

[6] 服部太輔, 細江政範, 石黒友一, ジャンケンの文化的側面と数理解析, 南山大学 数理情報学部 数理科学科 2006 年度卒業研究, 2007, http://www.seto.nanzan-u.ac.jp/st/gr-thesis/ms/2006/osaki/03mm015.pdf

[7] 柏木理志, 飯田弘之, ゲームの洗練法 ジャンケンを題材として, 情報処理学会研究報告ゲーム情報学, Vol.2003-GI-010, pp.9-13 (2003), http://id.nii.ac.jp/1001/00058569/

[8] 牧野泰裕, 西野順二, 小高知宏, 小倉久和, 繰り返し対戦型じゃんけんにおける戦略の特徴, 福井大学 工学部 研究報告 第 45 巻 第 1 号, pp.71-79, 1997, http://repo.flib.u-fukui.ac.jp/dspace/bitstream/10098/3425/1/AN00215401-045-01-007.pdf

付録 A　ソースプログラム

本研究で作成した限定ジャンケンプログラムのソースファイルを下に示す。

```
package janken;

import java.util.Random;

public class GenteiJanken {

	/*
	 * ジャンケンの定義、並びにＡＩの内容を記すクラス。
	 * ＧＪＲｅｐｅａｔを実行する前に、ここでａｉの値（戦略の内容）を変えておく。
	 * このクラスを実行するとジャンケンが１回だけ行われる。
	 */

	//　残りの星の数。０になるとその時点で負けが決まる
	public int star = 3;
	//　自分の残りグー、チョキ、パーの回数
	public int ig;
	public int ic;
	public int ip;
	//　相手の残りグー、チョキ、パーの回数
	public int yg;
	public int yc;
	public int yp;
	// 自分と相手のＡＩ設定
	/* 検証の際はここの値を変更する。ｉａｉはレジュメでいえばＡＩ１、ｙａｉはＡＩ２の、それぞれの戦略を指す
	 * それぞれの戦略と数値の対応について、戦略Ａ…０、戦略Ｂ…１、戦略Ｃ…２、戦略Ｄ…３、戦略Ｅ…４、戦略Ｆ…５
	 */
	public int iai = 2;
	public int yai = 1;
	//　ジャンケンを行う回数
	public int rounds;
	// 手の種類（０：グー　１：チョキ　２：パー）
	public int hand[];
	Random rnd;

	GenteiJanken() {
		ig = ic = ip = yg = yc = yp = 4;
		rounds = 12;
		hand = new int[2];
		rnd = new Random();
	}

	void view() {
		System.out.print("you:" + hand[0] + "/rival:" + hand[1] + "/");
	}

	void judge() {
		if (hand[0] == hand[1]) {
			System.out.print("Draw:Star" + star);
```

```
        } else if ((hand[0] == 0 && hand[1] == 1)
                        || (hand[0] == 1 && hand[1] == 2)
                        || (hand[0] == 2 && hand[1] == 0)) {
                star++;
                System.out.print("Win!:Star" + star);
        } else {
                star--;
                System.out.print("Lose:Star" + star);
        }
}

public static void main(String[] args) {
        GenteiJanken gj = new GenteiJanken();
        int x = gj.game();
        // System.out.println(x);
}

int game() {
        System.out.println("GAME START!!");
        int round = 0;
        while (star < 6 && star > 0 && round < rounds) {
                round++;
                System.out.print("ROUND" + round + "//");
                // 手の選択
                ai(0, iai);
                ai(1, yai);
                // 戦略比較用
                /*
                if(round <= 4) hand[1] = 0;
                else if(round <= 8) hand[1] = 1;
                else if(round <= 12) hand[1] = 2;
                */
                // それを表示
                view();
                // 勝ち負けの判定
                judge();
                System.out.println();
        }
        System.out.println();
        return result();
}

// レジュメ第３稿よりプログラム内容を一部変更
// 引き分けの時星（持ち点）が３個の場合は引き分けを搭載
int result() {
        int winner;
        System.out.println("RESULT!!");
        if (star > 3) {
                System.out.println("WINNER!!");
                winner = 0;
        } else if(star < 3){
                System.out.println("LOSER...");
                winner = 1;
```

```
        } else {
                System.out.println("DRAW...!");
                winner = 2;
        }
        return winner;
}

// ここまでルール根幹に関わるプログラム

/*
 * ジャンケンＡＩの考察
 *   ・例えば相手がグーしか無ければ自分は何投げようが同じ
 *   ・例えば相手がグーとチョキなら自分はグー、パー、チョキの順で優先して投げる
 *   ・例えば相手がパー１枚しかなければ自分は上と同じ
 *   ・例えば自分がパー１枚しかなければ相手はグーが出し易くなるのでグー、パー、チョキの順で優先して投げる
 *   ・そもそもサンプルプログラムのをまんま参考にするAI
 */
/*
 * 以下はに関するプログラムAI
 */

// とにかくランダムで手を出す。元々はａｉ　＝　０のプログラムだったがまとめた。
int random(int player, int num) {
        int a = -1;
        while (true) {
                a = rnd.nextInt(num);
                if (player == 0) {
                        if (!((a == 0 && ig == 0) || (a == 1 && ic == 0)
                                        || (a == 2 && ip == 0))) {
                                break;
                        }
                } else if (player == 1) {
                        if (!((a == 0 && yg == 0) || (a == 1 && yc == 0)
                                        || (a == 2 && yp == 0))) {
                                break;
                        }
                }
        }
        return a;
}

// 戦略C用テスト（没案)。一部動作をメソッドとして独立。
/*
int random6(int a,int b,int c,int d,int e,int f){
        int g = 0;
        int h = 0;
        while (true) {
                g = rnd.nextInt(6);
                if (g == 0 && a != -1){
                        h = a;
                        break;
                }else if (g == 1 && b != -1){
                        h = b;
```

```
                        break;
                }else if (g == 2 && c != -1){
                        h = c;
                        break;
                }else if (g == 3 && d != -1){
                        h = d;
                        break;
                }else if (g == 4 && e != -1){
                        h = e;
                        break;
                }else if (g == 5 && f != -1){
                        h = f;
                        break;
                }
        }
        return h;
}
*/

/*
 * 各々の戦略を卒研レジュメに倣い、以下に記す
 * 戦略Ａ…カードの枚数に関係なくランダムで手を出す。
 * 戦略Ｂ…相手の出せる手が二つに限定されている場合、自分はその二つの手に勝てるか引き分ける手を優先的に出す。
 * 戦略Ｃ…戦略Ｂの内容に加え、自分の各々の残り回数が１となった手は絶対に使わず、２の場合は通常の半分の確率で出す。
 * 戦略Ｄ…自分の現在のカードの枚数に比例して手を出す。
 * 戦略Ｅ…戦略Ｄの内容に加え、相手の手の残り回数が多ければ、その手に対し勝てる手を出し易くする。
 * 戦略Ｆ…戦略Ｅの内容に加え、負ける手を出しにくくする。
 * プログラム上はａｉ＝０を戦略Ａ、ａｉ＝１を戦略Ｂとし、以下ａｉ＝５まで同様に続く。
 */
void ai(int player, int ai) {
        // 正しい統計を取るため、自分の手を参照し、一旦足しておく
        if (player == 1) {
                if (hand[0] == 0)
                        ig++;
                if (hand[0] == 1)
                        ic++;
                if (hand[0] == 2)
                        ip++;
        }

        int a = -1;
        // 戦略Ａ…取り敢えず完全ランダムＡＩ。ここから発展させる。
        if (ai == 0) {
                // ｒａｎｄｏｍメソッドでランダムに数字を生成。そこで生成した数字をａへ代入
                a = random(player, 3);
        } else if (ai == 1) {
                // 戦略Ｂ…例えば相手がグーとチョキなら自分はグー、パー、チョキの順で優先して投げる。以下同様
                if (player == 0) {
                        if (yg != 0 && yc != 0 && yp == 0) {
                                if (ig != 0)
                                        a = 0;
```

```
                        else if (ip != 0)
                                a = 2;
                        else
                                a = 1;
                } else if (yc != 0 && yp != 0 && yg == 0) {
                        if (ic != 0)
                                a = 1;
                        else if (ig != 0)
                                a = 0;
                        else
                                a = 2;
                } else if (yp != 0 && yg != 0 && yc == 0) {
                        if (ip != 0)
                                a = 2;
                        else if (ic != 0)
                                a = 1;
                        else
                                a = 0;
                        // 該当するものがなければランダム。手が一つなら自分は何をどの順番
に出そうが結果は同じ
                } else {
                        a = random(player, 3);
                }
                // これを相手に対しても同様に行う
        } else if (player == 1) {
                // ・例えば相手がグーとチョキなら自分はグー、パー、チョキの順で優先して投げる。
以下同様
                if (ig != 0 && ic != 0 && ip == 0) {
                        if (yg != 0)
                                a = 0;
                        else if (yp != 0)
                                a = 2;
                        else
                                a = 1;
                } else if (ic != 0 && ip != 0 && ig == 0) {
                        if (yc != 0)
                                a = 1;
                        else if (yg != 0)
                                a = 0;
                        else
                                a = 2;
                } else if (ip != 0 && ig != 0 && ic == 0) {
                        if (yp != 0)
                                a = 2;
                        else if (yc != 0)
                                a = 1;
                        else
                                a = 0;
                        // 該当するものがなければランダム。手が一つなら自分は何をどの順番
に出そうが結果は同じ
                } else {
                        a = random(player, 3);
                }
```

```
                }
        } else if (ai == 2) {
                /*
                 * 戦略C…逆に自分の各々の手の回数が0にならないように出す相手の出せる手の回数のいずれが0にならない限り、
                 * 自分の手の回数が1となった手は絶対に使わない、2の場合は通常の半分の確率で出す。
                 * 全ての回数が1になってしまった場合はそこからランダムで出すしかない。
                 */
                //   まずは優先作業（相手の出せる手の回数のいずれが0にならない限り…）を先に書く。戦略Bと同じ
                if (player == 0) {
                        if (yg != 0 && yc != 0 && yp == 0) {
                                if (ig != 0)
                                        a = 0;
                                else if (ip != 0)
                                        a = 2;
                                else
                                        a = 1;
                        } else if (yc != 0 && yp != 0 && yg == 0) {
                                if (ic != 0)
                                        a = 1;
                                else if (ig != 0)
                                        a = 0;
                                else
                                        a = 2;
                        } else if (yp != 0 && yg != 0 && yc == 0) {
                                if (ip != 0)
                                        a = 2;
                                else if (ic != 0)
                                        a = 1;
                                else
                                        a = 0;
                                // 全ての手の回数が1以下ならランダム。手が一つなら自分は何を殿順番に出そうが結果は同じ
                        } else if (ig <= 1 && ic <= 1 && ip <= 1) {
                                a = random(player, 3);
                                // 少々乱雑な方法だが、以下のように行うため…
                                /*
                                 * 自分の手の回数が1となった手は絶対に使わない、2の場合は通常の半分の確率で出す。
                                 */
                                // まず配列bに1（初期値）を入れ、残り手の回数が3以上なら2つに、なら1つに該当する手の数字を入れる?
                                //
                        } else {
                                int[] b = { -1, -1, -1, -1, -1, -1 };
                                if (ig >= 3)
                                        b[0] = b[1] = 0;
                                else if (ig == 2)
                                        b[0] = 0;
                                if (ic >= 3)
                                        b[2] = b[3] = 1;
                                else if (ic == 2)
```

```
                        b[2] = 1;
                if (ip >= 3)
                        b[4] = b[5] = 2;
                else if (ip == 2)
                        b[4] = 2;
                // a = random6(b[0],b[1],b[2],b[3],b[4],b[5]);
                // ０から５の範囲内で数字をランダム生成。とにかく１が当たらないよ
うに祈る。?

                while (true) {
                        int c = rnd.nextInt(6);
                        if (b[c] != -1) {
                                a = b[c];
                                break;
                        }
                }

        }
        // これを相手に対しても同様に行う
} else if (player == 1) {
        // ・例えば相手がグーとチョキなら自分はグー、パー、チョキの順で優先して投げる。
以下同様
        if (ig != 0 && ic != 0 && ip == 0) {
                if (yg != 0)
                        a = 0;
                else if (yp != 0)
                        a = 2;
                else
                        a = 1;
        } else if (ic != 0 && ip != 0 && ig == 0) {
                if (yc != 0)
                        a = 1;
                else if (yg != 0)
                        a = 0;
                else
                        a = 2;
        } else if (ip != 0 && ig != 0 && ic == 0) {
                if (yp != 0)
                        a = 2;
                else if (yc != 0)
                        a = 1;
                else
                        a = 0;
                // 全ての手の回数が１以下ならランダム。手が一つなら自分は何を殿順
番に出そうが結果は同じ
        } else if (yg <= 1 && yc <= 1 && yp <= 1) {
                a = random(player, 3);
                // 少々乱雑な方法だが、以下のように行うため…
                /*
                 * 自分の手の回数が１となった手は絶対に使わない、２の場合は通常の
半分の確率で出す。
                 */
```

```
                                        // まず配列bに１（初期値）を入れ、残り手の回数が３以上なら２つに、
２なら１つに該当する手の数字を入れる？
                                        //
                                } else {
                                        int[] b = { -1, -1, -1, -1, -1, -1 };
                                        if (yg >= 3)
                                                b[0] = b[1] = 0;
                                        else if (yg == 2)
                                                b[0] = 0;
                                        if (yc >= 3)
                                                b[2] = b[3] = 1;
                                        else if (yc == 2)
                                                b[2] = 1;
                                        if (yp >= 3)
                                                b[4] = b[5] = 2;
                                        else if (yp == 2)
                                                b[4] = 2;
                                        // ０から５の範囲内で数字をランダム生成。とにかく１が当たらないよ
うに祈る。？
                                        while (true) {
                                                int c = rnd.nextInt(6);
                                                if (b[c] != -1) {
                                                        a = b[c];
                                                        break;
                                                }
                                        }
                                }
                        }
                } else if (ai == 3) {
                        // 戦略Ｄ…それぞれの手において、その手が出せる残り回数が多いほど出し易い
                        // よって、手の種類からではなく、カードの枚数から手をランダムに決める
                        if (player == 0) {
//                              Random rnd = new Random();
                                int d = rnd.nextInt(ig + ic + ip);
                                if (d < ig)
                                        a = 0;
                                else if (d < ig + ic)
                                        a = 1;
                                else if (d < ig + ic + ip)
                                        a = 2;
                        } else if (player == 1) {
//                              Random rnd = new Random();
                                int d = rnd.nextInt(yg + yc + yp);
                                if (d < yg)
                                        a = 0;
                                else if (d < yg + yc)
                                        a = 1;
                                else if (d < yg + yc + yp)
                                        a = 2;
                        }
                } else if (ai == 4) {
                        // 戦略Ｅ…戦略Ｄの内容に加え、相手の手の残り回数が多ければ、その手に対し勝てる手を出し
易くする。その際…
```

```
                        /*
                         * 例えば相手のグーの残り回数が０の場合、自分はグーに勝てるパーは出さない。更に、相手はチョキとパーのみであれば、
                         * 自分はそれに負けるか引き分けるパーは前述の通り出すことはない。
                         * 逆に自分のグーの残り回数が１の場合、極力出さないようにする（確率なので全くで無いわけでは無い）
                         * そのため、自ずとレベル２までの条件の大半を網羅することが出来る
                         * また、相手のグーの残り回数が３や４など多い場合、相手はグーが出易いと看做して自分はパーを出し易くする
                         */
                        // まずは自分の各々の手の重み（出し易さ）を定義する
                        int gw = 0;
                        int cw = 0;
                        int pw = 0;
                        if (player == 0) {
                                //   例えば相手のグーに勝てるのはパー。自分のパーの枚数が多いほど、相手のグーの枚数が少ないほど自分がパーを出し易くする
                                /*
                                 * エラー対策…例えば自分のグーが残り１回、相手グーが１回の場合だと
                                 * ｇｗ，ｃｗ，ｐｗ全て０になってしまう。ｎｅｘｔＩｎｔ（０）で
                                 * ランダム生成できないｉｇ，ｉｃ，ｉｐが１以上であれば１を足しておく
                                 */

                                gw = ig * yc;
                                cw = ic * yp;
                                pw = ip * yg;

                                if (ig != 0)
                                        gw++;
                                if (ic != 0)
                                        cw++;
                                if (ip != 0)
                                        pw++;

//                              Random rnd = new Random();
                                int d = rnd.nextInt(gw + cw + pw);
                                if (d < gw)
                                        a = 0;
                                else if (d < gw + cw)
                                        a = 1;
                                else if (d < gw + cw + pw)
                                        a = 2;これも書き方変えただけ

                                //
//                              Random rnd = new Random();
                                // 試しに書いただけ
                                /*
                                if(gw > cw && gw > pw && ig != 0) a=0;
                                else if(cw > gw && cw > pw&& ic != 0) a=1;
                                else if(pw > gw && pw > cw&& ip != 0) a=2;
                                else if(ig != 0) a=0;
                                else if(ic != 0) a=1;
                                else if(ip != 0) a=2;
```

```
                        */
                        /*
                        gw = ig * yc;
                        cw = ic * yp;
                        pw = ip * yg;

                        if (ig != 0)
                                gw++;
                        if (ic != 0)
                                cw++;
                        if (ip != 0)
                                pw++;

                        cw += gw;
                        pw += cw;

                        int d = rnd.nextInt(pw);
                        if (d < gw)
                                a = 0;
                        else if (d < cw)
                                a = 1;
                        else if (d < pw)
                                a = 2;
                        */



                } else if (player == 1) {

                        gw = yg * ic;
                        cw = yc * ip;
                        pw = yp * ig;

                        if (yg != 0)
                                gw++;
                        if (yc != 0)
                                cw++;
                        if (yp != 0)
                                pw++;

//                      Random rnd = new Random();
                        int d = rnd.nextInt(gw + cw + pw);
                        if (d < gw)
                                a = 0;
                        else if (d < gw + cw)
                                a = 1;
                        else if (d < gw + cw + pw)
                                a = 2;

                }
        } else if (ai == 5) {
                // 戦略Ｆ…戦略Ｅの内容に加え、負ける手を出しにくくする。
```

```
                        /*
                         * 負ける手を出しにくくする方法は苦労したが、
                         * 例えば自分の手の残り数（例えばグー）に勝てる手の二乗（相手のチョキの残り回数）と
                         * 引き分ける手（相手のグー）を掛けることで擬似的に表現した
                         */
                        int gw = 0;
                        int cw = 0;
                        int pw = 0;
                        if (player == 0) {
                                //   例えば相手のグーに勝てるのはパー。自分のパーの枚数が多いほど、相手のグーの枚数が少ないほど自分がパーを出し易くする
                                /*
                                 * エラー対策…例えば自分のグーが残り１回、相手グーが１回の場合だと
                                 * ｇｗ，ｃｗ，ｐｗ全て０になってしまう。ｎｅｘｔＩｎｔ（０）で
                                 * ランダム生成できないｉｇ，ｉｃ，ｉｐが１以上であれば１を足しておく
                                 */
                                gw = ig * yc * yc * yg;
                                cw = ic * yp * yp * yc;
                                pw = ip * yg * yg * yp;
                                if (ig != 0)
                                        gw++;
                                if (ic != 0)
                                        cw++;
                                if (ip != 0)
                                        pw++;
//                              Random rnd = new Random();
                                int d = rnd.nextInt(gw + cw + pw);
                                if (d < gw)
                                        a = 0;
                                else if (d < gw + cw)
                                        a = 1;
                                else if (d < gw + cw + pw)
                                        a = 2;
                        } else if (player == 1) {
                                gw = yg * ic * ic * ig;
                                cw = yc * ip * ip * ic;
                                pw = yp * ig * ig * ip;
                                if (yg != 0)
                                        gw++;
                                if (yc != 0)
                                        cw++;
                                if (yp != 0)
                                        pw++;
//                              Random rnd = new Random();
                                int d = rnd.nextInt(gw + cw + pw);
                                if (d < gw)
                                        a = 0;
                                else if (d < gw + cw)
                                        a = 1;
                                else if (d < gw + cw + pw)
                                        a = 2;
                        }
                }
```

```
                // 手を出すプレイヤーと実際に決めた手を参照し、それぞれの残り回数を減らす
                if (player == 0) {
                        if (a == 0)
                                ig--;
                        if (a == 1)
                                ic--;
                        if (a == 2)
                                ip--;
                        hand[0] = a;
                } else if (player == 1) {
                        if (a == 0)
                                yg--;
                        if (a == 1)
                                yc--;
                        if (a == 2)
                                yp--;
                        hand[1] = a;
                }

                if (player == 1) {
                        // ここで帳尻合わせ
                        if (hand[0] == 0)
                                ig--;
                        if (hand[0] == 1)
                                ic--;
                        if (hand[0] == 2)
                                ip--;
                }

                // System.out.println(hand[0]);
        }
}
```

```
package janken;

public class GJRepeat {

        /**
         * @param args
         * Ｇｅｎｔｅｉｊａｎｋｅｎ．ｊａｖａの実行を繰り返すために作られたクラス
         * レジュメで実際に検証する際はこのクラスを実行する。今回は１００００回試行。
         * ａｉの値（戦略の内容）については大元のＧｅｎｔｅｉｊａｎｋｅｎ．ｊａｖａの方に書かれてある。
         */

        //   ここに繰り返す回数を指定する。ａｉレベル等はＧｅｎｔｅｉｊａｎｋｅｎ．ｊａｖａの方をいじればいい
        static int games = 10000;
        //   勝利数と敗北数をメモる引数レジュメ第３稿のテストより引き分け数ｄｒａｗも記載
        static int win = 0;
```

```
	static int lose = 0;
	static int draw = 0;
	// Ｗｉｎｎｉｎｇ　ｐｅｒｃｅｎｔａｇｅ、つまり勝率
	static double wp = 0; 確認用
	//
	static int[] kakunin = new int[games];

	public static void main(String[] args) {
		// TODO 自動生成されたメソッド・スタブ
		for (int i = 1; i <= games; i++) {
			System.out.println("GAME:" + i);
			GenteiJanken gj = new GenteiJanken();
			int x = gj.game();
			// ｘが０なら勝ち、１なら負け
			if (x == 0){
				win++;
				kakunin[i-1] = 0;
			}else if (x == 1){
				lose++;
				kakunin[i-1] = 1;
			}else {
				draw++;
				kakunin[i-1] = 2;
			}
			// 見易くするよう取り敢えず３行開ける
			System.out.println();
			System.out.println();
			System.out.println();
		}
		// 見易くするよう取り敢えず３行開ける
		System.out.println();
		System.out.println();
		System.out.println();  全試合の勝敗を簡略化して見易くする（確認用）
		//
		for (int i = 1; i <= games; i++) {
			System.out.print(kakunin[i-1]);
			if(i%100==0) System.out.println();
		}
		// 最後の行に結果を記す
		wp = ((double)win/games)*100;
		System.out.println("WIN:" + win + "//LOSE:" + lose
				+ "//DRAW:" + draw + "//WP" + wp);
	}

}
```