第57回 プログラミング･シンポジウム 2016.1.8-10

3人の賢者の問題 —— 愚者は賢者をシミュレートできるか？

竹内郁雄 [1,a]

概要 以前，私が考案した単純そうなパズル「3 人の賢者」を深く掘って調べてみると，ある賢者が，持っている情報を完全には知り得ない他の賢者の思考を推論することをプログラムで表現することの難しさを実感することができた．実社会でも，他者の思考をシミュレーションすることは重要である．「3 人の賢者の問題」はそういう現実的な問題の非常に小さなモデル課題となっていると思われる．今後，さらによい完璧な解法が出ること祈念したい．

キーワード 3 人の賢者の問題，不完全情報，他者の思考の推論，不可能問題

1. はじめに

私が 2014 年 7 月号の「数学セミナー」誌に出題した「3 人の賢者の問題」[1] は，Douglas Hofstadter の凡庸（mediocre）[2] をベースにした不完全情報ゲームである．3 人の賢者が巡回しながら，ゲームについて分かったことを発言していくという仕立てになっている．全員がずっと「分からない」と発言し続けると，そのうち誰かが「誰それの勝ちだ」と宣言できるようになるという不思議なゲームである．これは各賢者が他の賢者の「分からない」発言から情報が得られるという原理に基づく．一見情報がないのに，情報が次第に明らかになってくるという意味で，有名な Hans Freudenthal の “Impossible Problem[3]” に通ずるものがある．

賢者が他の賢者の発言から何をどこまで理解できたかをシミュレートするプログラムを作ることは面白い問題である．ここでは問題の面白さとそれのプログラミングついて述べる．

実は，これは私が 2015 年の夏のプログラミング・シンポジウムで「下呂でしか作れないプログラム」で挑戦した 2 つのプログラムのうちの 2 つ目の問題である．残念ながら，下呂では完成しなかった．予想以上に手強い問題であった．その後，自宅で一応完成させ，本稿執筆時点では一応動作しているが，エレガントなプログラムとは言えない．

[1,] 東京大学名誉教授

[a] nue@nue.org

2. 出題した「3 人の賢者の問題」と正解

「数学セミナー」誌での，出題の原文通りの全文と解答を紹介する．

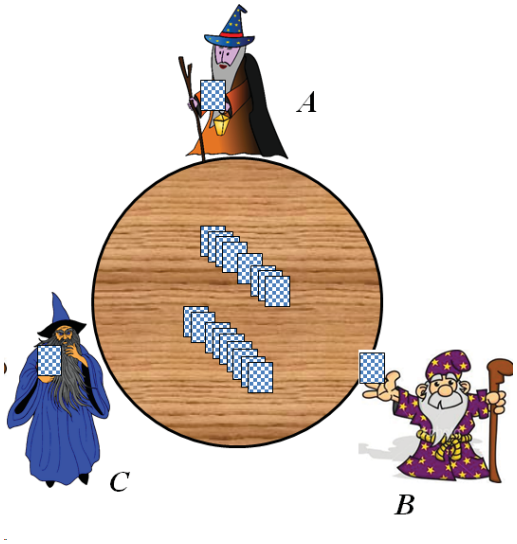

図 1. 3 人の賢者の問題

3 人の賢者 A，B，C がカード遊びをしています．1 から 20 までの数が書かれた 20 枚のカードが円卓上に裏向きに置いてあり，賢者たちは 1 枚ずつ手元に取ります（図 1）．大小比較で中間の数を取った賢者が勝利します．でも，一斉にカード

をオープンするだけでは脳がへたれるというので，賢者たちは次のようにゲームを変形しました．

取ったカードを自分の右隣り（B は A，C は B，A は C）にだけ見せます．そして，A から順に右回りに勝敗について確定したことを発言させるのです．ただし，勝敗に関する新しい確定情報がないときはニッコリ会釈するだけにします．

最初からニッコリが連続したあと，ある賢者が「じゃあ，わしの勝ちじゃな．最初からニッコリがこれより長く連続することはなかろうて」と言いました．誰がどのように勝ったのでしょうか？

3 人が出した数のうち中間の数が勝利するというのは，まさに Hofstadter が Scientific American に連載した Metamagical Themas で紹介した凡庸（その後，月並と改称）である．これはその後私がトランプで遊べる最中限というゲームに発展させ，愛好者によって分析されている [4]．

上記の出題に対して，37 名から解答が寄せられたが，正解は 8 名のみであった．以下に私が用意した正解を紹介する．

以下，賢者 A，B，C のカードの数値も A，B，C と書く．図 2 はニッコリが連続したときにどのような情報が明らかになっていくかを表している．ニッコリが最初の A から始まり，この三角螺旋が内側から外側に回る向きに続いていく．図 2 の楕円の中の m-n はそこでニッコリした賢者のカードの値 x が $m \leq x \leq n$ であること，その左隣りの賢者のカードの値 y も $m \leq y \leq n$ であることが公知になることを意味する．

一番最初の A のニッコリでは，$2 \leq A \leq 19$ ということが公知になる．$A \in \{1, 20\}$ なら，A は自分が勝てない，つまり負けであることが宣言できるからである．なお，A は B を知っているので，$B \in \{1, 20\}$ であれば，A は B が自分で言う前に B が勝てないと宣言する．つまり，ここでの A のニッコリは $2 \leq B \leq 19$ も公知にする．また例えば，$A = 1$，$B = 2$ なら，自分が勝てないではなく，B の勝ち，$A = 2$，$B = 1$ なら自分，つまり A の勝ちなどと，情報量のより多い宣言をすることに注意する．なにしろ，みんな賢者なので，情報量を最大にする発言をする．

次に B がニッコリしたということで，$2 \leq B \leq 19$ に加えて，$B \notin \{2, 19\}$ であることが公知になる．もし $B = 2$ であれば，$A > 2$ であることが分かり，B は C を知っているので，その時点で B の勝ち負け（場合によっては別の勝者）が分かるからである．$B = 19$ も同様．つまり，ここで $3 \leq B \leq 18$ が公知となり，$3 \leq C \leq 18$ も公知となる．

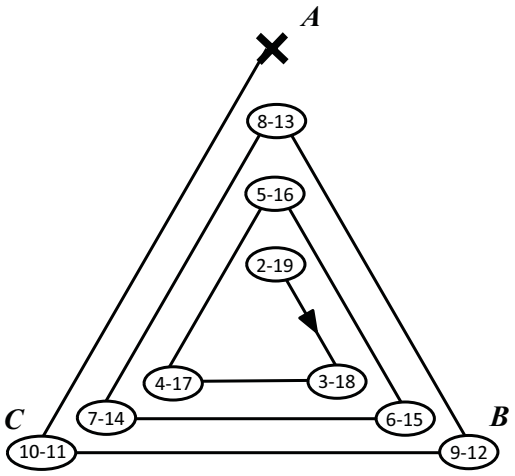

図 2. 3 人の賢者の推論三角

次に C がニッコリしたときも同様で，$4 \leq C \leq 17$，$4 \leq A \leq 17$ が公知となる．このようにして，9 つめのニッコリで $10 \leq C \leq 11$，$10 \leq A \leq 11$ が公知になったあと，A の番で以下の推論が成立する．

図 2 から，この時点で $9 \leq B \leq 12$ も公知である．ここで可能なカードの組合せは 4 通りである．

$A = 10$，$B = 9$ （このとき，$C = 11$）

$A = 11$，$B = 12$ （このとき，$C = 10$）

のいずれかの場合は A が「自分の勝ち」，つまり「A の勝ち」と宣言できる．

$A = 10$，$B = 12$ （このとき，$C = 11$）

$A = 11$，$B = 9$ （このとき，$C = 10$）

のいずれかの場合は A が「C の勝ち」と宣言できる．これは最初からニッコリのみの最長連続回数のあとの勝敗判明になる．A が自分の勝利を宣言したのだから，可能なのは前 2 つの場合に限定される．

つまり，正解は以下の通り．最初からニッコリが 9 回続いたあと，A が，$A = 10$，$B = 9$，$C = 11$，

あるいは，$A = 11$，$B = 12$，$C = 10$ というカードの組合せで勝った．

3. 一般のケース

実は，誰もがずっとニッコリを連続するというのは，逆説的だが問題をかなりやさしくしている．途中で誰かが「うむ，何某（自分も含む）が負けることが分かった」と発言したときのほうがずっとややこしいことになる．このことは「数学セミナー」誌の解答欄でも言及した．

私はいくつかのケースを手作業で分析してみたが，とてもスンナリとは行かない．途中でこんがらかってきてしまうのである．例えば，一番最初に A が「自分は負けた」と発言すると，B には $A = 1$ なのか，$A = 20$ なのかが分からない．ただし，$B = 2$ だったら，$A = 20$ であることが B に分かる．なぜなら，$A = 1$ だったら，A は「自分の負け」と言わずに,「B の勝ち」と言うはずだからである．C は A を知っているので，$A = 1$ だったら，$B \neq 2$ であることが分かる．また，A，B，C はそれぞれ異なる数なので，C は B の札を知らないとはいえ，少なくとも C とも A とも異なる数であることは最初から分かっている．いろいろ考えると，ともかく，賢者の思考を推論することはそうやさしくない．

このことがずっと気になっていて，いつか時間のあるときに賢者の思考をきっちりとプログラムで表現したいと考えていた．2015 年の夏のプログラミング・シンポジウム「プログラム詠み会」という機会があったので,「詠み会」の 3 つの課題のうちの 1 つ「下呂でしかできないプログラミング」で，この問題に挑戦することになったのである．

実際にプログラムを書いてみると，やはり一筋縄ではいかない．賢者の思考を完全にシミュレートすることが結構難しい．どの賢者も自分の札と，自分の左隣りの賢者の札，そして自分の番の前のすべての賢者の発言（ここではニッコリも発言とする）を知っている．知らないのは，自分の右隣りの賢者の札の正確な値だけである．自分の左隣りの賢者の発言と右隣りの賢者の発言から，右隣りの賢者の札の値の範囲を絞りこんでいく必要がある．ここで重要なのは，誰かが発言すると，その左隣りの賢者の「知識」も右隣りの賢者の「知識」も変化することである．

ところで，カードを 1 から 20 に限定する必要はない．1 から n ($n \geq 3$) と一般化すべきである．$n \bmod 3$ によってなにか違う現象が起こるのかも気になるところであるが，実際にはなんの差もないようである．

4. 賢者のシミュレーションの当初の準備

各賢者の内部知識（持っている情報）の表現は，当初，簡単なものだと考えていた．自分の札の数と左隣りの賢者の札の数，そして右隣りの賢者の札の数の可能セット（つまり，札の数として可能なものの集合）が基本で，あとは「誰それが負けた」という発言がむし返されないようにするためのフラグとか，もっと基本的なところでは，自分は誰（A，B，C のうちの誰？），自分の右隣り，左隣りは誰？という情報などである．

下呂では，使っていた言語（TAO/ELIS の TAO，いまは DAO と呼んでいる）のオブジェクト指向をなかなか思い出せなくて，単純なリスト構造で作ったのだが，賢者（wizard）をオブジェクトで実現すると，そのインスタンス変数は以下のようになる．

- 不変で public

  `name`: 自分の名前（シンボル）
  `left`: 自分の左隣りの賢者（オブジェクト）
  `right`: 自分の右隣りの賢者（オブジェクト）

- 可変で public

  `range`: 自分の札の全員に公知の可能範囲
  （最小値から最大値までの昇順リスト）
  `state`: 自分の負けが宣言されたときに真

- 不変で private

  `mycard`: 自分の札
  `lcard`: 自分の左隣りの賢者の札

- 可変で private

  `rcards`: 右隣りの賢者の札の可能セット
  （昇順にソートしたリスト，ただし
  `mycard` や `lcard` は除いておく）

public とか private と書いてあるが，プログラム上はすべて public 扱いにしてある．デバッグの便を優先した．賢者が他の賢者の private 変数にアクセスしないようにするのは自主規制である．

ゲームは札の大小関係だけで決まるので，難しいことはなにもない．単独の数とリストになった数の集合との大小比較の関数を作るだけである．もちろん，この大小比較では大小関係が成立しないときも偽となる．たとえば，(Lisp の記法で書くが)

```
(<* 5 (list 2 3 4)
(<* 5 (list 4 6 7 8))
```

はどちらも偽である．

5. プログラムを書いてみると

上述したように，全員がずっと「ニッコリ」だと，比較的安直にプログラムを書いても大体正しく動く．ニッコリした途端，各賢者の「知識」は割りと単純なロジックでアップデートされるからである．

ところが，単純に書いたプログラムではいろいろ試してみると，一般のケースでは不都合が一杯出てくる．この不都合を判定するためには，プログラマ，つまり私自身がいろいろ考えないといけなかったので非常に効率が悪い．愚者が賢者になるのは大変なのである．以下のリスト記法の意味は，例えば

```
(A 10) (B 7) (C (1 2 3 4 5 6 8 9))
```

は，A の知識，$A = 10$，A が知っている自分の左隣りの $B = 7$，A が推論できる右隣りの C の札の範囲を昇順に並べた 1, 2, 3, 4, 5, 6, 8, 9 を表わしている．昇順のリストから 10 と 7 が抜かされていることに注意．

初期値 $n = 10$，$A = 10$, $B = 7$, $C = 8$ に対して，A が最初から「A（自分）が負けた」と言った．

```
(A 10) (B 7) (C (1 2 3 4 5 6 8 9))
(B 7) (C 8) (A (1 10))
(C 8) (A 10) (B (2 3 4 5 6 7 9))
```

簡単にプログラムした段階では，上のようになったが，C の B に関する知識

```
(2 3 4 5 6 7 9)
```

は間違っている．9 が余計なのである．なぜなら，$B = 9$ なら，A が B の勝者を宣言をしないので $A = 1$ のはず，これは C の $A = 10$ の知識と反する．

もう 1 つの例を示しておく．初期値 $n = 10$, $A = 1$, $B = 9$, $C = 8$ に対して，A が最初から「A（自分）が負けた」と言った．

```
(A 1) (B 9) (C (2 3 4 5 6 7 8 10))
(B 9) (C 8) (A (1 10))
(C 8) (A 1) (B (3 4 5 6 7 9 10))
```

ここで，B の A に関する知識

```
(1 10)
```

は間違っている．10 が余計である．なぜなら，$A = 10$ なら，A は自分の負けでなく，B の勝利を宣言するはずということが B に推論できるからである．また，C の B に関する知識

```
(3 4 5 6 7 9 10)
```

の 10 も余計である．$B = 10$ なら，A が C の勝利宣言をするはずということが C に推論できるからである．

これらの例を見て分かるように，発言した賢者の左隣りの賢者も右隣りの賢者も，発言した賢者の思考を推論しないといけない．どちらも一番最初の A の発言で「負け」の発言があった例だが，途中で「自分以外の賢者の負け」を宣言することもあるので，もう少しいろんなことを考えないといけない．

当初，私はある程度のヒューリスティックスを駆使すればそこまで面倒なプログラムは書かなくてもいいと考えていたが，もぐら叩きのような「パッチ・デバッグ」ではどうにもならないことに気がついた．

6. 設計のやり直し

結局，他の賢者 (つまり発言した賢者) の思考を，残りの 2 人の賢者はそれぞれちゃんと推論しないといけないのである．そのために，賢者のインタンス変数を 3 個追加する．

可変で private

`l-rcards`: 自分の左隣りの `rcards` の推測
`r-rcards`: 自分の右隣りの `rcards` の推測

可変で public

`decl-winner`: 勝者を宣言すると真

文字どおり，両隣りの賢者の知識のうち漠としたもの，つまり `rcards` を（さらに漠と）推論する変数を用意したわけである．

全体で 200 行以内なので，決して大きなプログラムではない．

`init` ほか: 宣言や比較関数など
`game`: 1 回のゲームの進行
`turn`: 賢者が自分で番で行う思考と発言
`simulate`: `turn` とほぼ同じで副作用を除いた思考
　他の賢者の思考を推論するために使用
`update-by-right`:
　右隣りの賢者の発言による知識の更新
`update-by-left`:
　左隣りの賢者の発言による知識の更新

この中で一番デバッグの一番難しかったのが，当然ではあるが，他の賢者の発言で自分の知識 `rcards` を更新する 2 種類の `update` である．つまり，他の賢者の思考を推論する部分である．行数は短いが最後まで手間取った．特に面倒だったのが，誰かが誰かの勝利を宣言したとき，つまりゲームが終了したときに，自分の `rcards` を最終的に適正なものにすることであった．やさしいと思っていたのに，それだけのために，比較的多くの行数を費すことになった [2]．

プログラムは本稿を執筆している時点では，一応動作しているが，モグラ叩きパッチの当たった美しくないコードになっている．そのため，最新版を

`https://www.dropbox.com/s/qfnddaybtwqv5l5/three-wizards.pdf?dl=0`

に置いておく．プログラムの構造や，プログラムの読解に必要なコメントをたっぷりつけておくので，この問題に挑戦されるときには参考にしていただきたい．あるいは，参考にしないほうがいいかもしれない．現状ではまだある種のヒューリスティックスが紛れ込んでおり，これを排除した完全の意味での，あるいは別の言い方をすると「杓子定規」な形での推論（思考）の推論を行うべきではないかと考えている．

[2] 勝利の宣言があったとしても，それぞれの賢者の `rcards` がシングルトンになるわけではないことに注意．

7. むすび

私のプログラムがこの方針のままでいいのかどうかについては，現時点でまったく自信がない．ただ，これだけの短いプログラムに，賢者，すなわち自分が知り得ている範囲の情報をベースにして最大限の情報を引き出せる推論をできる人が，他の賢者に関する不完全情報をベースに，その賢者の思考，つまり推論を推論するということが「曲がりなりにも」表現できるという実例を示せたとは考えている．

その中で，賢者の思考と行動のプログラムと，それから行動の部分だけを抜いたプログラムの両方が必要だったというのは面白かった．いわば推論を推論するのだから当然とは言えるが，プログラミングを極めたいという立場からすると，どうしてちょっとだけ違う同じプログラムを 2 回書かないといけないのだろうかとも思った．

この問題の難しさのもう 1 つの理由だと思ったのは，3 角形に並んだ賢者の相対的な「右左」の循環関係である．ともかく，頭の中だけではなにかがぐるぐる回ってしまう．

それにしても，Freudenthal の「不可能問題」は不思議である．むすびで，長々とこれを紹介するのはちょっと気が引けるが，あまりにも面白いし，因縁もあるので紹介する．記述のバージョンはいろいろあるが，オリジナルは以下の通りである（オランダ語から英語への翻訳から日本語への翻訳）．

先生が Peter と Sam にこう言った．「私は $2 \leq x < y$，かつ $x + y \leq 100$ であるような 2 つの整数 x と y を秘密裡に選んだ．$s = x + y$ を Sam だけに教えた．また，$p = xy$ を Peter だけに教えた」

1. Peter 「僕には x と y が分からない」

2. Sam 「君にそれが分からないことは僕にはもう分かってたよ」
3. Peter 「そうか，じゃあ，僕には x と y が分かった」
4. Sam 「そうか，じゃあ，僕にも分かった」

これだけで，x と y がなんであったかを当てろというのが問題である．もちろん，これらの発言の間にはたっぷり時間がかかっていると考えるべきである．

これを Scientific American (邦訳: 「日経サイエンス」，日経サイエンス社) に Martin Gardner が 1979 年に「不可能問題」として紹介した．ただ，彼が少し問題を変えて出題したために，x と y が当てられないという，本当の不可能問題になってしまった．おかげで読者たちから大反響が起こったといういわくつきの問題である [3]．つまり，言い方を微妙に変えたり，等号・不等号をちょっと変えても，まったく違う問題になるという意味で，相当に不思議な問題だったわけである [5]．

この問題はプログラムを書いて，数の候補を篩にかけていく方法が有効であり，いろいろな人が試みている．

このほかにも,「分からない」ということが情報になるパズルがある．有名なのは，Muddy Children Puzzle[6] である．複数の子供たちが外から帰ってきたあと，親から指摘されて自分には見えない額についた泥が当てられるかどうかという問題である．1 回目の指摘で誰も分からなかったということから，2 回目以降（条件によって何回か親からの指摘が繰り返される），どこかで「あ，分かった」となる．3 人の賢者の問題に似ているといえば似ているが，ずっと簡単である．これのプログラムは（書いてはいないが）そう難しくはないはずである．

いずれにせよ，ある人が分からないということが他者に関する情報になるという問題は 1 つのカテゴリをなす．これのプログラムでそれをちゃんと書くとなると面倒なことが多くなるだろうというのが私の予想である．

[3] もちろん，Gardner はそれを詫びたのであるが，どういうわけか，その後出版された単行本や全集からこの問題は削除されたままとなっている．Gardner はよほど後悔したのだろうか．こんな面白い問題が Gardner の著作から消えるのは惜しい．

[参考文献]

[1] 竹内郁雄: エレガントな解答を求む，数学セミナー，2014 年 7 月号，10 月号，日本評論社.

[2] Douglas R. Hofstadter: Metamagical Themas, Basic Books, 1985. (邦訳) メタマジック・ゲーム，竹内郁雄，斉藤康己，片桐泰弘，白揚社，2005 (新版).

[3] Axel Born, Cor A.J. Hurkens, Gerhard J. Woeginger: The Freudenthal Problem and its Ramifications (Part 1), Bulletin of the EATCS, no. 90, pp. 175-191, 2006, European Association for Theoretical Computer Science.

[4] 副田俊介，田中哲朗: 最中限の終盤の分析，情報処理学会研究報告ゲーム情報学（GI）2003, pp.31-38, 2003-08-04 情報処理学会.

[5] Lee Sallows: The Impossible Problem, http://www.leesallows.com/files/THE_IMPOSSIBLE_PROBLEM2.pdf

[6] The Muddy Children Puzzles, http://sierra.nmsu.edu/morandi/coursematerials/MuddyChildren.html

[付録]

(1) オリジナルの問題に対するプログラムの適用

最初の [i j] はその賢者（左端に名前が出てくる賢者）の range である．続いてその賢者の札，左隣りの賢者の札，右隣りの賢者の札に対する推定値（rcards）である．$n = 20$, $A = 10$, $B = 12$, $C = 11$ の例.

```
A smiled calmly.
[2 19] (A 10) (B 12) (C (1 2 3 4 5 6 7 8 9 11 13 14 15 16 17 18 19 20))
[2 19] (B 12) (C 11) (A (2 3 4 5 6 7 8 9 10 13 14 15 16 17 18 19))
[1 20] (C 11) (A 10) (B (2 3 4 5 6 7 8 9 12 13 14 15 16 17 18 19))
B smiled calmly.
[3 18] (B 12) (C 11) (A (2 3 4 5 6 7 8 9 10 13 14 15 16 17 18 19))
[3 18] (C 11) (A 10) (B (3 4 5 6 7 8 9 12 13 14 15 16 17 18))
[2 19] (A 10) (B 12) (C (3 4 5 6 7 8 9 11 13 14 15 16 17 18))
C smiled calmly.
[4 17] (C 11) (A 10) (B (3 4 5 6 7 8 9 12 13 14 15 16 17 18))
[4 17] (A 10) (B 12) (C (4 5 6 7 8 9 11 13 14 15 16 17))
[3 18] (B 12) (C 11) (A (4 5 6 7 8 9 10 13 14 15 16 17))
A smiled calmly.
[5 16] (A 10) (B 12) (C (4 5 6 7 8 9 11 13 14 15 16 17))
[5 16] (B 12) (C 11) (A (5 6 7 8 9 10 13 14 15 16))
[4 17] (C 11) (A 10) (B (5 6 7 8 9 12 13 14 15 16))
B smiled calmly.
[6 15] (B 12) (C 11) (A (5 6 7 8 9 10 13 14 15 16))
[6 15] (C 11) (A 10) (B (6 7 8 9 12 13 14 15))
[5 16] (A 10) (B 12) (C (6 7 8 9 11 13 14 15))
C smiled calmly.
[7 14] (C 11) (A 10) (B (6 7 8 9 12 13 14 15))
[7 14] (A 10) (B 12) (C (7 8 9 11 13 14))
[6 15] (B 12) (C 11) (A (7 8 9 10 13 14))
A smiled calmly.
[8 13] (A 10) (B 12) (C (7 8 9 11 13 14))
[8 13] (B 12) (C 11) (A (8 9 10 13))
[7 14] (C 11) (A 10) (B (8 9 12 13))
B smiled calmly.
[9 12] (B 12) (C 11) (A (8 9 10 13))
[9 12] (C 11) (A 10) (B (9 12))
[8 13] (A 10) (B 12) (C (9 11))
C smiled calmly.
[10 11] (C 11) (A 10) (B (9 12))
[10 11] (A 10) (B 12) (C (11))
[9 12] (B 12) (C 11) (A (10))
A said C won.
[10 11] (A 10) (B 12) (C (11))
[9 12] (B 12) (C 11) (A (10))
[10 11] (C 11) (A 10) (B (12))
game-end
```

(2) 誰かが誰かの負けを宣言した例．こちらのほうがはるかに難しくなる．

A が一番最初に自分の負けを宣言した例．$n = 8,\ A = 6,\ B = 8,\ C = 3.$

```
A said B lost.
[2 7] (A 6) (B 8) (C (1 2 3 4 5 7))
[1 8] (B 8) (C 3) (A (2 4 5 6))
[1 8] (C 3) (A 6) (B (1 8))
B smiled calmly.
[1 8] (B 8) (C 3) (A (2 4 5 6))
[2 7] (C 3) (A 6) (B (1 8))
[2 7] (A 6) (B 8) (C (3 4 5))
C smiled calmly.
[2 7] (C 3) (A 6) (B (1 8))
[3 6] (A 6) (B 8) (C (3 4 5))
[1 8] (B 8) (C 3) (A (4 5 6))
A said A won.
[3 6] (A 6) (B 8) (C (3 4 5))
[1 8] (B 8) (C 3) (A (4 5 6))
[2 7] (C 3) (A 6) (B (8))
```