

信州大学工学部

学士論文

マスターマインドにおける最悪手数を最小化する戦略
とそれに対する堅牢な出題

指導教員 西新 幹彦 准教授

学科 電気電子工学科
学籍番号 10T2002K
氏名 荒井 航

2014 年 3 月 31 日

目次

1	はじめに	1
2	マスターマインドのルール	1
3	戦略の構造	2
3.1	戦略の形式的定義	2
3.2	戦略の完全性	2
4	Knuth の戦略	3
4.1	戦略の補完	4
4.2	正答するまでの手数の分布	4
5	Knuth の戦略への対策	5
5.1	ゲーム理論的考察	6
5.2	情報理論的考察	7
6	まとめ	8
7	今後の課題	8
	謝辞	8
	参考文献	8
	付録 A 補完した戦略	10
	付録 B 補完に用いたプログラム	17

1 はじめに

マスターマインドは、二人のプレイヤーに分かれて互いの隠されたピンの色をヒントを元に推理するボードゲームである。1970 年代に前半にイギリスのインヴィクタ社から発売され、その後アメリカではハスプロ社から発売され、世界中で発売された [1]。また、1946 年からはアメリカで Twenty Questions[2]、日本ではそれをモデルにした二十の扉という番組が放送されていた。近年では、フジテレビのゲーム番組である NumerOn などさまざまなバリエーションがあり、長く多くの人に楽しまれてきた推理ゲームの一つである。本研究では様々な研究をされてきたマスターマインドの戦略の内から Donald E. Knuth の論文 [1] に掲載された戦略を使用する。Knuth の研究では、マスターマインドにおいて最悪 5 手で出題者のコードを決定する戦略を発見している。しかし、この論文に掲載された戦略は、一部省略されている部分があり、実際のゲームでは使用するには不完全である。そこで、最初に実際のゲームで使用できるよう足りない部分を補完し、それを用いてゲーム理論的、情報理論的な観点から、Knuth の戦略に対して堅牢な出題コードについて解析していく。

2 マスターマインドのルール

本研究で対象とするのは、以下のようなオリジナルのルールに基づくマスターマインドである。

プレイヤーは出題者と解答者に分かれ以下の手順に従って勝敗を決める。ボードゲームでは 6 種類の色のピンを使うが、本論文では色の種類を 1 から 6 までの数字で表す。

1. 出題者は解答者に分からないように 6 色のピンの中から重複を許して 4 本を選び並べる。この並べた 4 本のピン組み合わせをコードと言うこととし、 $x_1x_2x_3x_4$ と表すこととする。また、出題者が 4 本のピンを決めることを、コードを設定すると言うこととする。
2. 解答者は配置を予想して 4 本ピンを並べる。予想した 4 本のピンの並びをテストパターンと言うこととし、 $y_1y_2y_3y_4$ と表すこととする。また、テストパターンを出題者に提示することをコードを解答すると言うこととする。
3. 出題者はテストパターンの予想を見てヒントを解答者に示す。ヒントは赤ピンと白ピンの数によって示される。赤ピンの数は色も位置も合っているピンの数で、白ピンの数は色だけが合っているピンの数であるが、厳密には以下のように決められる。
 - 左から j 番目のシンボルが $x_j = y_j$ である時、これを満たすシンボルの数だけ赤ピンを立てる。

- 左から j 番目と k 番目のシンボルが $x_j \neq y_j$ であるが $x_j = y_k$ である時、これを満たすシンボルの数だけ白ピンを立てる。ただし、 k 番目のシンボルは上記の条件で使用していないシンボルである。

また、コードとテストパターンを比較してヒントを決めることを判定と言うこととする。

4. ヒントとして赤ピンが 4 本立って、テストパターンがコードと等しいことが分かったら終了する。このことをコードを正答と言うこととする。赤ピンが 4 本でなければ手順 2 に戻って解答者は次のテストパターンを作る。また、正答するまでに繰り返した 2-3 の手順の回数を手数と言うこととする。

勝敗の決め方としては、それぞれが出題者になり、答えるまでの回数が少ないほうの勝ちとする。本研究ではコードが正答されるまでの手数を考察し、勝敗については直接扱わない。

3 戦略の構造

3.1 戦略の形式的定義

戦略の形式的定義を説明するにあたりヒントの種類について説明する。ヒントは得られた赤ピンと白ピンの組み合わせにより以下のように分けられる。

赤ピンと白ピンの数をそれぞれ r, w と表し、ヒントを (r, w) と表す。可能なヒントは

$(r, w) = (0, 4), (0, 3), (0, 2), (0, 1), (0, 0), (1, 3), (1, 2), (1, 1), (1, 0), (2, 2), (2, 1), (2, 0), (3, 0), (4, 0)$ の 14 通りである。

解答者はそれまでに得たヒントから次のテストパターンを決定する。この手順が記述されたものを戦略と呼ぶ。戦略は基本的に 14 分木の構造をしている (図 1)。各内部ノードにはコードの集合が対応しており、特にルートノードにはコード全体が対応している。さらに内部ノードにはテストパターンが記述されている場合があり、そのときそのノードは高々 14 個の子ノードを持ち、子ノードへの枝には 14 通りのヒントのうちのいずれかがラベルとして重複なく振られている。子ノードに対応しているコードの集合は、親ノードがもつコードのうち、テストパターンに対するヒントが枝のラベルと一致するもの全体である。

3.2 戦略の完全性

1296 通りのコードの候補に対して判定を行うと、コードの候補がヒントによって分けられ、ノードに絞り込まれた候補が集まる。この候補に対して、ノードに記述されているテストパターンと再び判定を行う。例えば、Knuth の戦略を持つ解答者は、ルートノードに書かれている '1122' というテストパターンを解答する。これを判定した結果、ヒントが $(2, 0)$ となったとする。すると解答者は $(2, 0)$ というラベルが降られた枝の先の子ノードを参照し、次のテスト

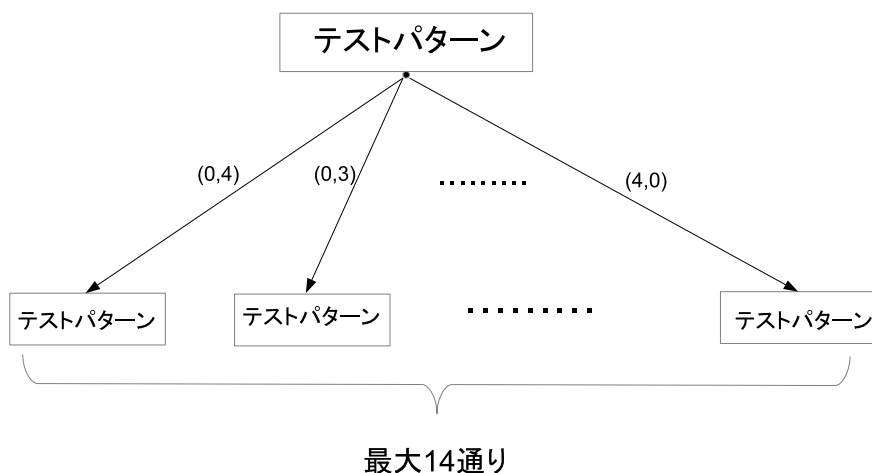


図 1 14 分木の構造図

パターンを得る．これを繰り返し，コードを正答する．どの葉に対しても対応しているコードの数が 1 つであるような戦略を完全な戦略と呼び，それ以外を不完全な戦略と呼ぶ．完全な戦略を用いると，その戦略に従うことによってどんなコードでも正答することができる．

4 Knuth の戦略

マスターマインドにおいて最悪 5 手で出題者のコードを正答する戦略が Knuth の論文 [1] で示された．この最悪 5 手は最適な手数である．仮に最悪 4 手以下で正答できる戦略があるとする．この戦略の平均手数は 4 手より短くなる．これに対して Koyama らの論文 [3] に掲載されている最適な戦略の平均手数は 4.341 手である．最悪 4 手以下の戦略はこのことと矛盾する．従って，Knuth の戦略の最悪手数は最適である．

Knuth の論文に掲載されている戦略は，自明な部分のテストパターンが省略されているので，形式的には不完全な戦略である．さらに，ノードに対応しているコードの集合も省略されている．

この章では，まず省略されているテストパターンを補完して完全な戦略を作成し，次に得られた戦略の基本的事項を検証する．

4.1 戦略の補完

Knuth の論文に掲載されている戦略でテストパターンが省略されているのは，対応しているコードの集合のサイズが 1 または 2 のノードである．したがって省略されたテストパターンを決定するためには，ノードに対応しているコードの集合を明らかにすればよい．そのために，不完全な戦略を記述したファイルを読み込んで，複数のコードが対応している葉に対して，それらのコードを表示するプログラムを作成した．この表示されたコードのうち適切なものをテストパターンとして手動で戦略に書き込むことによって完全な戦略が得られる．

第 3.2 節に従って 1296 通りのコードを判定していく．葉まで判定を行い，候補のコードが複数存在する場合はコードをヒントごとに分けて表示する．これを基に手動でノードの先にヒント別に新たなノードを作成し表示されたコードをテストパターンとして書き加える．テストパターンを追記したファイルを再び読み込んで，候補のコードが表示されなくなるまで繰り返す．完成した戦略を付録 A に掲載した．

4.2 正答するまでの手数分布

正答にかかる手数の分布を調べた．この分布は 1296 通りのコード一つずつに対して戦略を使用し，正答するまでの手数を数えることによって求めた．その分布が図 2 と表 1 である．

表 1 正答するまでの手数の分布

手数	コードの数
1	1
2	6
3	62
4	530
5	697

図 2 の横軸は手数，縦軸はコードの数である．図 2 を見て分かるように 4 手，5 手にコードの分布が集中していることが分かる．

この分布から，正答するまでの平均手数を求めることが出来る．Knuth の戦略を用いると，1296 パターンのコードが全て等確率で選ばれる際に 1 つのコードを導き出すためにかかる平均手数は，手数の総数を 1296 で割ったものであるので，先の結果から

$$5804/1296 \approx 4.478 \quad (1)$$

手である．

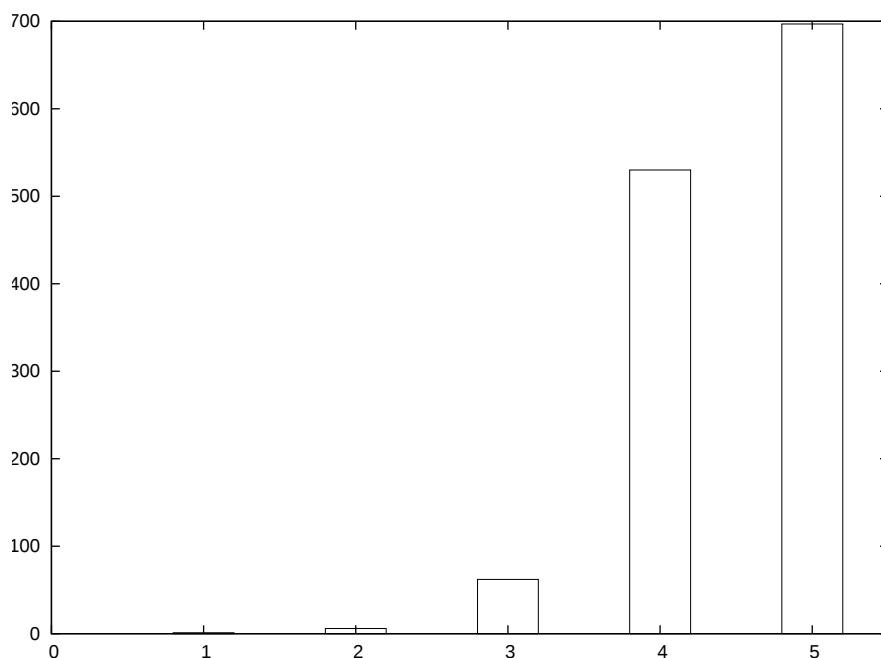


図2 手数の分布

また，この平均手数が最適な戦略が，1993 年に Kenji Koyama と Tony W. Lai により発表されている [3]．この論文の戦略を用いると最悪 5 手で出題者のコードを解答する戦略の平均手数が

$$5626/1296 \approx 4.341 \quad (2)$$

手となりより早くコードを導き出す戦略となっていることが分かる．

5 Knuth の戦略への対策

Knuth の戦略を持つ解答者に対して出題者はどのようにコードを設定すべきなのかを考える．解答者が使用する戦略には複数の種類が存在する．戦略は，シンボルに対する色の当てはめ方を換える事により新たな戦略が得られ，色は 6 種類あるので $6! = 720$ 通りの戦略が存在する．また，戦略のシンボルの位置を入れ換えることにより新たな戦略を作成でき $4! = 24$ 通りの戦略が存在する．以上より

$$6! \times 4! = 17280 \quad (3)$$

通りの戦略を解答者は使用できる．解答者はこの中から 1 つの戦略を選んで使用するが，どの戦略を選ぶかは出題者には分からない．

5.1 ゲーム理論的考察

ゲーム理論的にマスターマインドを考察する．ゲームは勝つことに意味があるので，何度もゲームを行い統計的に勝率を高めるよりも，一回のゲームで確実に何手までは正答されないという保証を求める．従って，出題者は一回のゲームでコードを正答されるまでにより多くの手数を掛けさせるコードを設定したいと思っている．このようなコードは次のようにして調べられる．

1296 通りの各コードに対して 17280 通りの戦略を使用した場合の手数の最小値を調べた．この最小値は，そのコードを設定した場合に 17280 通りのどの戦略を使用されても，必ずその手数以上掛けさせることが可能な戦略となっている．従って，この最小値が最も大きいコードが有効なコードとなる．結果を図 3 の最短の手数の分布図と，表 2 に示した．

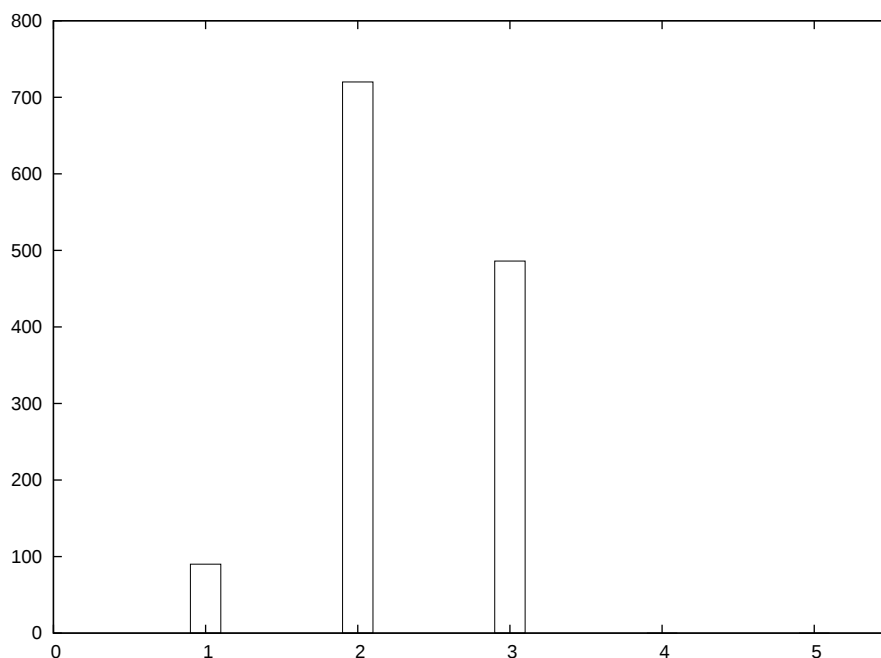


図 3 最短の手数の分布

図 3 の横軸は手数の最小値，縦軸はコードの数である．表 2 を見て分かるように，どの戦略を使用されたとしても 3 手以上かけさせることが可能なコードがあることが分かった．

表 2 正答されるまでの手数の最小値の分布

手数	コードの数
1	90
2	720
3	486
4	0
5	0

また、その数は 486 個存在することが分かった．このコードには傾向があり，aaaa, aaab, abcd(a, b, c, d は互いに異なる) となるシンボルの順列であり，aaaa のコードはシンボルの数が 6 種類より，6 通り，aaab のコードは

$$6 \times 5 \times \frac{4!}{3!} = 120 \quad (4)$$

通り，abcd のコードは，

$${}_6P_4 = 360 \quad (5)$$

通りとなる．以上より， $6 + 120 + 360 = 486$ 通りとなり，組み合わせが網羅されていることが証明できた．従って，以上のような 3 種類のコードの順列を考慮して出題者はコードを設定することにより，Knuth の戦略をもつ解答者に対して必ず正答に 3 手以上かけさせる保証を得ることが出来る．

5.2 情報理論的考察

情報理論的にマスターマインドを考察する．出題者は解答者がコードを正答するまでの手数の平均手数 (期待値) が多くなるようなコードを考える．そのため，各コードに対して，解答者が 17280 通りの戦略の中からひとつを等確率で使用する場合の正答までの手数を求める．この時 1296 通りのコードを辞書順に並べて小さい方から N 番とする． N 番目のコードを n 手で正答する戦略の数が X_{Nn} とすると求める確率 P_{Nn} は，

$$P_{Nn} = \frac{X_{Nn}}{17280} \quad (6)$$

である．この数を用いて平均手数は次の式から計算できる．

$$(\text{平均手数}) = \sum_{n=1}^5 n P_{Nn} \quad (7)$$

表 3 平均手数の分布

平均手数	分布数
4.167	6
4.378	90
4.450	120
4.474	720
4.528	360

で求められる。結果を表 3 に示した。

表 3 より平均手数の最大値は 4.528 手であり、コードは 360 通り存在することが分かる。このコードには傾向があり、 $abcd(a,b,c,d$ は互いに異なる) となるシンボルの順列となっている。これは第 5.1 節で有効とされるコードの一つである。従って、 $abcd$ となるコードは、ゲーム理論、情報理論どちらの観点においても Knuth の戦略に対して有効なコード設定である。

ここで、この結果と第 4.2 節で求めた式 (1) を比較することは無意味であることを注意しておく。ここでの平均手数は一つのコードを 17280 通りの戦略を等確率に使用して正答した場合の平均手数であり、式 (1) は全てのコードを等確率で使用する場合に一つの戦略で解答した場合の平均手数であるため、全く別の指標である。

6 まとめ

本研究では、Donald E. Knuth が示したマスターマインドの戦略に対して、実際にゲームに使用できるような戦略として補完を行った。その戦略に対して有効なコード設定をゲーム理論、情報理論の観点から考察を行い、どちらにも共通した傾向を持つコードを発見した。

7 今後の課題

Donald E. Knuth の戦略に対しての平均手数やコードの分布、有効なコードの設定を考察したが、最適なマスターマインドの戦略とされる K. Koyama と T. W. Lai の戦略におけるコードの分布や有効なコードの設定に関しては手付かずのままである。従って、K. Koyama と T. W. Lai の戦略についてそれらを明らかにすると共に、より正答するまでの手数が減る戦略の探索が課題となっている。

謝辞

本研究に当たり、細かく指導して下さった指導教員の西新幹彦准教授に感謝の意を表する。

参考文献

- [1] D. E. Knuth, “The Computer as Master Mind”, J. Recreational Mathematics, Vol. 9(1), pp.1-6, 1976-77.
- [2] David J. C. MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, pp.70-72, Fifth printing 2006.
- [3] K. Koyama and T. W. Lai, “An Optimal Master Mind Strategy”, Journal of Recreational Mathematics, pp.251-256, 1993.

付録 A 補完した戦略

戦略の使用方法を説明する。解答者はまず一行目を見る。一行目は以下のようになっている。

```
[000] 1122:(2211),[001],[006],[043],[121];*,[198],[213],[284];[362],[363],[372];[417];#
```

戦略の各行は以下のような構造となっている。

[ラベル] テストパターン:

```
(0,4),(0,3),(0,2),(0,1),(0,0);(1,3),(1,2),(1,1),(1,0);(2,2),(2,1),(2,0);(3,0);(4,0)
```

コロンの降にはヒントが記載されており、解答者が次に行動すべきことが、ヒントごとに分けられて記されている。

最初にラベル [000] の行に記載されている 1122 のテストパターンを解答する。判定により得られたヒントの場所には以下のことが記載されているので、その指示に従う。

例えば 1122 と解答し、ヒントが (0,4) となった場合、戦略を見ると (2211) と明記されている。解答者は、次に 2211 と解答すると正答となる。

このように、(xxxx) の場合、次のテストパターンを xxxx とし、次の手で xxxx が正答となる。

ヒントが (0,3) の場合、戦略を見ると [001] と明記されている。解答者は、次に [001] の行に移り、そこに明記されているテストパターンを解答する。

このように、[yyy] の場合、yyy と同じ数字のラベルの行に記載されているテストパターンを解答する。

ヒントが (1,3) の場合、戦略を見ると*と明記されている。この場合、次に解答するテストパターンは存在しないので、出題者の提示したヒントが間違っている、もしくは解答者が参照する場所を間違えている。

ヒントが (4,0) の場合、戦略を見ると#と明記されている。この場合、#と記載されている行のテストパターンで正答である。

```
[000] 1122:(2211),[001],[006],[043],[121];*,[198],[213],[284];[362],[363],[372];[417];#
[001] 1213:*,*,*,*(2311),[002],[003],*(3211),[004],[005];(2213);*
[002] 1415:*,*,*(2231),*,*(2511),*(2611),*,*(2411),*,*
[003] 1145:*,*(2251),*(2261),*,*(2241),*,*,*,*
[004] 4115:*,*,*,*,*(5211),*(6211),*,*(4211),*,*
[005] 1145:*,*(2214),*(2216),*,*(2215),*,*,*,*
[006] 2344:*,[007],[008],[014],[019];*,[020],[021],[028];*,[035],[036];[042];*
[007] 4213:*,*,*,*,*(4231),*,*,*
[008] 3215:*,*,*,*(5231),[009],*(4261),*(4411);[010],[011],[012];[013];#
[009] 4251:*,*,*,*,*(6231),*,*
[010] 3251:*,*,*,*(5213),*,*,*,*
[011] 3231:*,*,*,*(6213),*,*,*,*(3261);#
[012] 3411:*,*,*,*(4216),*,*,*,*
[013] 3213:*,*,*,*(4215),*(3216);#
[014] 5215:*,*,*,*(6251),[015],[016];(5251),(5261),[017];[018];#
[015] 3511:*,*,*,*(6261),*,*,*(4511);#
```

[illegible]

```
[085] 1356:*,*(3541),(4641),(4441),*,*,*,*(4351),*,*#
[086] 4361:*,*(6254),(5254),*,*,*(5264),*,*,*(4331);#
[087] 1635:*,*(2563),(2553),(6264),*,*(2536),(2533),*,*(2535);*#
[088] 4541:(5414),*,*,*,*,*,*,*,*#
[089] 4315:*,*,*,*,*(6414),*,*,*,*#
[090] 3614:*,*,*,*(4316),(4313),*,*,*,*#
[091] 4414:*,*,*,*,*,*,*(3514),(4614);#
[092] 3315:*,*[093],[094],[2666];(5331),[095],[096],[097];[098],[099],[6316];[100];#
[093] 5641:*,*,*,*,*,*,*(6541),*,*(5541);#
[094] 2566:*,*(6641),*,*,*,*(2656),*,*(2556);#
[095] 5361:*,*,*,*,*,*(6351),(6331),*(5351);#
[096] 5614:*,*(6361),*,*,*,*(6514),*(5514);#
[097] 6614:*,*,*(2565),(2555),*,*(2665),(2655),*,*,*#
[098] 3351:*,*,*,*(5313),*,*,*,*#
[099] 3331:*,*,*,*,*(6313),(5316),*,*,*(3361);#
[100] 3316:*,*,*,*,*,*(6315),(5315);(3313);#
[101] 2434:*,*,*,*(3244),*,*(2443),*,*,*#
[102] 2425:*,*(5244),[103],[104],*,*[105],[106],*,*(2454),[107];(2445);*
[103] 4244:*,*,*,*,*,*(6244);#
[104] 4314:*,*,*,*,*(4341),*,*,*#
[105] 2534:*,*,*,*,*(2543),*,*,*#
[106] 2634:*,*,*,*,*(2643),*,*,*#
[107] 2446:*,*,*,*,*(2464),*,*,*#
[108] 1545:*,*(5314),[109],[110],[111],*,*[112],[113],[114],*,*[115];(2545);*
[109] 2654:*,*,*,*,*(6314),(3314),*,*,*#
[110] 2353:*,*,*,*,*(2664),*,*(2356);#
[111] 1136:*,*(2363),*,*,*(2366),(2333),*,*(2336);*#
[112] 2554:*,*(5341),*,*,*,*,*,*#
[113] 2564:*,*(6341),(3341),*,*(2355),*,*,*,*#
[114] 2335:*,*,*,*,*(2646),*,*(2365);#
[115] 2546:*,*,*,*,*(2645),*,*,*#
[116] 1335:*,*(2544),[117],*,*[118],[119],*,*[120];*#
[117] 1445:*,*,*,*,*,*(2644);#
[118] 2343:*,*,*,*,*(2354),*,*,*#
[119] 2346:*,*,*,*,*(2364),*,*,*#
[120] 2334:*,*,*,*,*(2345),*,*,*#
[121] 3345:[122],[123],[131],[142],[6666];[148],[149],[158],[169];[174],[176],[182];[191];#
[122] 4533:*,*,*,*,*(5433),*,*,*#
[123] 4653:[124],[125],*,*,*[126],[127],[5533],*,*[128],[129],[4433];[130];#
[124] 5436:*,*,*,*(6534),*,*,*,*,*#
[125] 5434:*,*,*,*,*(5534);#
[126] 4536:(5463),*,*,*(5634),*,*,*,*,*#
[127] 4534:*,*,*,*,*(6433),*,*(6533);*#
[128] 4563:*,*,*,*(6453),*,*,*,*,*#
[129] 5453:*,*,*,*,*(5633);*#
[130] 4453:*,*,*,*,*(4633);(4553);#
[131] 6634:*,*[132],[133],[5553],*,*[134],[135],[136],[137];[138],[139],[140];[141];#
[132] 4566:*,*,*,*,*(5466),*(4463);*#
[133] 4556:*,*,*,*,*(5563),*,*(5456),*,*(4456);#
[134] 4663:*,*,*,*,*(6463),*,*,*#
[135] 4656:*,*(6563),*,*,*(5663),*,*(6456),*(4436);*#
[136] 5653:*,*(4564),*,*(5536),(5564),(5464),*(6553),*,*,*#
[137] 1444:*,*,*,*,*(4554),(5554),*(4454),(5454);*#
[138] 4636:*,*,*,*,*(6436),*,*,*#
[139] 5636:*,*(6564),*,*(6653),*,*(6536),(5664),*,*,*#
[140] 4654:*,*,*,*,*(6454),(6554),(4434);(5654);#
[141] 1413:*,*(4634),(6654),*,*(6434),(6633),*,*,*,*#
[142] 6646:*,*(4464),*,*,*[143],[5566],[5556];[144],[145],[146];[147];*#
[143] 1416:*,*(4664),*,*,*(6464),*,*(4466);*#
[144] 1416:*,*(6664),*,*,*(4666),*,*,*(6466);*#
[145] 5666:*,*,*,*,*(6566),(6663),*,*,*#
[146] 5656:*,*,*,*,*(6556),*,*,*#
[147] 6636:*,*,*,*,*(6656);#
[148] 3453:(5334),*,*,*(3534),*,*(4353),*,*,*#
[149] 3454:(4543),[150],[151],*,*,*[152],[153],[154],*(4354),[155],[156];[157];#
[150] 4535:*,*(5643),*,*(5543),(6543),*,*,*,*(4635);#
[151] 1436:*,*(4363),(5363),*,*,*(4333),(5333),*(4336),(5336);*#
[152] 4435:*,*,*,*(5443),*,*,*,*,*#
[153] 4356:(6435),(5435),*,*,*(5364),*,*,*(4334);*#
```

[154] 3536:(6353),(5353),*,*,*,(6334),*,*(3563),*,*(3533);#
 [155] 3564:*,*,*,*,(6354),(5354),*,*,*,*,*,*;
 [156] 3463:*,*,*,*,(3634),*,*,*,(3436),(3653),(3553);(3433);#
 [157] 3456:*,*,*,*,*,*,*,(3654),(3554),(3434);*,*;
 [158] 3636:(6363),(6364),[159],[160],[161];*,[162],[163],[164];[165],[166],[167];[168];#
 [159] 4364:(6443),*,*,*,*,*,(6465),*,*,*,*,*,*;
 [160] 4565:*,*,*,*,*,(6455),(6544),*,(4443);(5465),*,*(4465);#
 [161] 4544:*,*,(5455),*,*,*,(4455),*,*(5444),*,(4555);(5544);#
 [162] 4366:(6643),*,*,*,*,*,(6333),*,*(6356),*(5366);#
 [163] 1565:*,(5356),*,(4643),*,*,*,(6546),(3464),*(6535),(4665);*,*;
 [164] 4546:*,*,*,*,(5644),(4655),*,(5535);(5446),*,*(5546);#
 [165] 3663:(6336),*,*,*,*,*,*,*,*,*,*,*,*;
 [166] 3466:*,(6635),*,*,*,*,*,*,(3664),*,*(3566);#
 [167] 3556:(5635),*,*,*,*,*,(5646),*,*,*,*,*,*;
 [168] 3633:*,*,*,*,*,*,*,*,*,*,*,*,(3656);*,*;
 [169] 3656:*,(6565),(5565),(6444),(4444);*,[170],[171],[172];*,(6655),[173];(3666);*
 [170] 5665:*,*,*,*,*,*,(6366),*,*,*,*,*,(6665);#
 [171] 6446:*,*,*,*,*,*,(6555);(6644),*,*,*,*;
 [172] 4446:*,*,*,*,(5555);*,*,*,(4644),*,*,*,*;
 [173] 4646:*,*,*,*,*,*,(5655);*,*,*,(6646);#
 [174] 3435:(5343),*,*,*,*,[175],*,*,*,(4335),*,*,*,*;
 [175] 3354:*,*,*,*,*,(3543),*,*,*,*,*,*,*;
 [176] 3443:*,*,[177],*,*,*,[178],[179],*(4343),[180],[181];(3643);#
 [177] 4355:*,*,*,*,*,*,*,*,*,*,(5335),(6335);(4365);#
 [178] 3334:*,*,*,*,*,*,*,*,*,*,(5344);(3364);#
 [179] 3356:*,*,*,*,*,(3635),(3535),*,*,*,(5346),*,*,*;
 [180] 3544:*,*,*,*,*,*,(6343),*,*,*,*,*,*;
 [181] 3455:*,*,*,*,*,(3546),*,*,*,*,(3353);(3465);#
 [182] 3636:*,(6365),[183],[184],[185];*,[186],[187],[188];(3366),(3665),[189];[190];*
 [183] 5365:*,*,*,*,*,*,(6344),*(6355),*,*,*,*;
 [184] 6445:*,*,*,*,*,*,(4344),(5355);*,*,*,(6545);#
 [185] 1444:*,*,*,*,*,*,(4545),(5545);*,(4445),(5445);*,*;
 [186] 3363:*,*,*,*,*,*,(6346),*,*,*,*,*,*;
 [187] 3565:*,*,(4346),*,*,*,*,(6645),*,*,*,*,*,*;
 [188] 4645:*,*,*,*,*,*,(3444),(3555);*,*,*,(5645);#
 [189] 3446:*,*,*,*,*,*,(3655),(3333);(3644),*,*,*,*;
 [190] 3336:*,*,*,*,*,*,*,*,*,*,(3646);*,*;
 [191] 3446:*,*,*,*,*,*,[192],[193],[194];*,[195],[196];[197];*
 [192] 4345:*,*,*,*,*,*,*,*,*,*,*,*,(6345);#
 [193] 3365:*,*,*,*,*,*,*,*,*,*,(5345);*,*;
 [194] 3335:*,*,*,*,*,*,*,*,*,*,(3355);#
 [195] 3344:*,*,*,*,*,*,*,*,*,*,(3645);*,*;
 [196] 3343:*,*,*,*,*,*,*,*,*,*,(3545);*,*;
 [197] 3346:*,*,*,*,*,*,*,*,*,*,(3445);*,*;
 [198] 1213:(2131),[199],[200],*,*,*,[201],[203],*,[206],[207],[208];[211];#
 [199] 1145:*,*,*,(2321),*,*,(2151),(2161),*,*,(2141),*,*,*;
 [200] 1415:*,*,(2521),(2621),*,*,*,(2421),*,*,*,*,*,*;
 [201] 1114:*,*,*,(3221),*,*,*,(2312),*,(2111),[202];(2114);*
 [202] 2115:*,*,*,*,*,*,*,*,*,*,(2116);#
 [203] 2412:(4221),[204],*,*,*,*,(2221),*,*,*,*,*,*;
 [204] 5221:*,*,*,*,*,*,*,*,*,*,(6221);#
 [205] 2512:*,*,*,*,*,*,*,*,*,*,(2612);#
 [206] 1231:(2113),*,*,*,*,*,*,*,*,*,*,*,*;
 [207] 1145:*,*,*,(3212),*,*,(1251),(1261),*,*,(1241),*,*,*;
 [208] 1145:*,*,[209],[210],*,*,*,*,*,*,*,*,*,*;
 [209] 4212:*,*,*,*,*,*,*,*,*,*,*,*,(5212);#
 [210] 2212:*,*,*,*,*,*,*,*,*,*,(6212);#
 [211] 1114:*,*,*,*,*,*,*,*,*,*,(1211),[212];(1214);*
 [212] 1215:*,*,*,*,*,*,*,*,*,*,*,*,(1216);#
 [213] 1134:*,[214],[215],[224],[234];[242],[243],[252],[263];[271],[272],[278];[283];*
 [214] 1312:*,(3421),*,*,*,*,(4321),*,*,*,(3412),*(4312);*
 [215] 3521:(5312),[216],[217],*,*,*,*,[218],[219],*,[220],[221],[222];[223];#
 [216] 5412:*,*,*,*,*,*,*,*,*,*,(6312);*,*;
 [217] 4612:*,*,*,*,*,*,(2342),*(6412),*,*(4412);#
 [218] 3312:*,*,*,*,*,*,*,*,*,*,(4512);(3612);#
 [219] 2423:(3242),*,*,*,*,*,*,*,(4223),*,*,*,*;
 [220] 3512:(5321),*,*,*,*,*,*,*,*,*,*,*,*;
 [221] 5421:*,*,*,*,*,*,*,*,*,*,*,(6321);*,*;
 [222] 4621:*,*,*,*,*,*,*,*,*,*,(6421),*,*(4421);#

[223] 3321:*,*,*,*,*,*,*,*,*,*,*,*(4521);#
 [224] 2352:[225],[226],[227],[6621],*(2523),[228],[229],[6612];[230],[231],[232];[233];#
 [225] 3225:*,*,*,*,*,*,*,*,*,*(5223),*,*,*,*;
 [226] 3226:*,*,*,*,*,*,*,*,*,*(6223),*(4225);(3223);#
 [227] 5621:*,*,*,*,*,*,*(4226),*(6521),*,*,*(5521);#
 [228] 2223:*,*,*,*,*,*(3262),(5242),*,*,*(2425);(2623);#
 [229] 6242:(2426),*,*,*,*,*,*(5612),(5512);(4262),*(6512);(4242);#
 [230] 2325:(3252),*,*,*,*,*,*,*,*,*,*;
 [231] 2323:*,*,*(4252),*,*,*,*(2542),*,*,*,*(2326);#
 [232] 2462:*,*,*,*,*,*,*,*,*(2642),(2242),*(2442);#
 [233] 2362:*,*,*,*,*,*,*,*,*(2452);#
 [234] 2525:(5252),[235],[6262],*,*,*,[236],[237],*,[238],[239],[240];[241];#
 [235] 5262:*,*,*,*,*,*,*,*,*(6252),*,*,*,*;
 [236] 2252:*,*,*,*,*,*(5226),*,*,*,*(2652);#
 [237] 2262:*,*,*,*,*,*(6226),*,*,*,*(2662);#
 [238] 2552:(5225),*,*,*,*,*,*,*,*,*,*;
 [239] 2562:(6225),*,*,*,*,*,*,*,*,*,*;
 [240] 2226:*,*,*,*,*,*,*,*,*(2626);#
 [241] 2225:*,*,*,*,*,*,*,*(2526),*(2625);#
 [242] 1341:(4113),*,*,*,*(1413),*,*,*(3141),*,*,*;
 [243] 1315:(3151),[244],[245],*,*,*(5113),[246],[247],*,[248],[249],[250];[251];#
 [244] 4151:*,*,*,*,*,*,*(5141),*(3161),*,*;
 [245] 4161:*,*,*,*,*,*(2143),*(6141),*,*(4141);#
 [246] 6451:*,*,*(3116),(3113),*,*(1541),(6113),(3111),*,*,*(1451);*
 [247] 1461:(4116),*,*,*,*(4111),(1243),*(1641),*,*(1441);#
 [248] 1351:(3115),*,*,*,*(1513),*,*,*,*,*,*;
 [249] 1361:*,*,*(4115),*,*(1613),*,*,*,*,*,*;
 [250] 1411:*,*,*,*,*,*,*,*,*(1416);#
 [251] 1113:*,*,*,*,*,*,*(1311),(1316),(1415);(1313);*
 [252] 1516:[253],[254],[255],*,[256];[257],[258],[259],*,[260],[261],[1246];[262];#
 [253] 5161:*,*,*,*,*,*,*(6151),*,*,*,*;
 [254] 5151:*,*,*,*,*,*,*,*(6161),*,*;
 [255] 2145:*,*,*,*,*,*,*(2153),(2163),*,*;
 [256] 2324:(4232),*,*,*,*(2432),*,*,*(3224),*,*,*;
 [257] 1651:(6115),*,*,*,*,*,*,*,*,*,*;
 [258] 1661:*,*,*(5115),*,*,*(6111),(5111),*,*,*,*;
 [259] 1245:*,*,*,*,*(2146),*,*,*(1253),(1263),*,*;
 [260] 1561:(5116),*,*,*,*(1615),*,*,*,*,*,*;
 [261] 1551:*,*,*(6116),*,*,*,*,*,*(1611),*,*;
 [262] 1511:*,*,*,*,*,*,*,*(1616);(1515);#
 [263] 1256:(2165),*,[264],[265],*,*,[266],[267],[268],[269],*,*,[270];#
 [264] 2524:*,*,*,*,*,*(2632),*,*(2532),*(2624);#
 [265] 2332:*,*,*,*,*(2424),*,*,*,*,*;
 [266] 2155:*,*,*,*,*,*,*,*(2166),*,*;
 [267] 5224:*,*,*,*,*,*(6232),*,*(5232),*(6224);#
 [268] 2224:*,*,*,*,*(3232),*,*(2232),*(4224);#
 [269] 1265:(2156),*,*,*,*,*,*,*,*,*,*;
 [270] 1255:*,*,*,*,*,*,*,*(1266),*,*;
 [271] 1314:(4131),*,*,*,*(1431),*,*,*(3114),*,*,*;
 [272] 1315:(5131),[273],*,*,*(1531),[274],[275],*,*,[276],[277],*,*;
 [273] 3131:*,*,*,*,*,*,*,*(6131);#
 [274] 1631:*,*,*(5114),*,*,*,*,*,*,*;
 [275] 4114:*,*,*,*,*,*,*,*(6114);#
 [276] 1331:*,*,*,*,*(1514),*,*,*,*,*;
 [277] 1414:*,*,*,*,*,*,*,*(1614);#
 [278] 1235:*,*(2154),[279],*,*,*,[280],*,*(2135),(1254),[281];[282];#
 [279] 2144:*,*,*,*,*,*,*,*(2164);#
 [280] 2133:*,*,*,*,*,*,*(2136);#
 [281] 1244:*,*,*,*,*,*,*(1264);#
 [282] 1233:*,*,*,*,*,*,*(1236);#
 [283] 1234:*,*,*,*,*,*(2134),*,*,*,*;
 [284] 1344:*,[285],[286],[297],[307];[312],[313],[322],[333];[342],[343],[349];[357];#
 [285] 1335:*,*(4153),(4163),(4423),*,*(4133),(4136),(4432),*(4135),*,*,*;
 [286] 3135:*,*,[287],[288],[289],*,[290],[291],[292];[293],[294],[295];[296];#
 [287] 4523:*,*,*,*,*,*(5423),*,*,*,*;
 [288] 4623:*,*,*,*,*(4452),*(6423),*,*,*,*;
 [289] 4426:*,*,*,*,*(4462),*,*,*,*;
 [290] 5163:*,*,*,*,*(6153),*,*(5153);#
 [291] 3423:*,*(4532),*(4156),*,*(5432),*(6163),*(3452),*,*,*;

[292] 3426:(4632),*,,*,*(6432),*(4166),*(3462),*(4425);*#
 [293] 3153:*,,*,*(5133),*,,*#
 [294] 3156:*,,*,*(6133),*(5136),*(3163),*,,*#
 [295] 1436:*(4165),*(4155),*,,*(3166),*(3425),*,,*(6136),*(3432);*#
 [296] 1536:(3165),*(3155),*,,*(6135),*(5135),*(3133),*,,*(3136),*,,*#
 [297] 3526:(5632),[298],[299],*(5155),*,[300],[301],[302],*(6166);[303],[304],[305];[306];#
 [298] 4652:*,,*,*(5462),*,,*(6452),*(6632);*#
 [299] 6155:*,,*(4662),*,,*(5452),*(6462);*(5165),*,,*(6165);#
 [300] 5623:(6532),*,,*,*(3652),*,,*,*,,*#
 [301] 1462:*(4625),*(6623),*,,*(6425),*(3632),*(5532);*(4562),*(3662);*#
 [302] 4552:(5425),*,,*(5166),*,,*(5156),*(6156),*,,*#
 [303] 1123:*,,*(3562),*,,*(3625),*,,*(6523);*#
 [304] 3552:*,,*(5426),*,,*(5523),*(3623),*,,*(3532);#
 [305] 4525:*,,*,*(6426),*,,*(4626);*#
 [306] 1145:*,,*(4526),*(3523),*(3626),*,,*(3525),*,,*#
 [307] 5525:*,,*(6652),*(6662),*,,*[308],[309],*(6626);*(5552),*(5562),[310];[311];#
 [308] 5652:*,,*,*(6552),*,,*#
 [309] 5662:*,,*,*(6562),*,,*#
 [310] 1516:*,,*(6625),*,,*(5626),*,,*(6526);*#
 [311] 1516:*,,*(5625),*,,*(6525),*,,*(5526);*#
 [312] 4134:*,,*,*(4143),*,,*#
 [313] 3145:(1453),[314],*,,*[315],[316],[317],*,[318],[319],[320];[321];#
 [314] 1436:*,,*,*(1463),*,,*(1433);#
 [315] 1435:*,,*,*(5134),*,,*#
 [316] 4154:*,,*,*(6134),*,,*#
 [317] 3424:*,,*,*(4164),*,,*#
 [318] 3154:*,,*,*(5143),*,,*#
 [319] 3134:*,,*,*(6143),*,,*(3164);#
 [320] 3442:*,,*,*(4146),*,,*#
 [321] 3143:*,,*,*(4145),*(3146);#
 [322] 4524:(5442),*(6442),[323],[324],[325],*,[326],[327],[328];[329],[330],[331];[332];#
 [323] 1456:(6145),*(5145),*(3642),*,,*(5146),*,,*(1465),*,,*(1455);#
 [324] 1635:*,,*(6146),*,,*(1466),*(1653),*,,*#
 [325] 1636:*,,*,*(1663),*,,*(1633);#
 [326] 1356:*,,*(3542),*(4642),*(4442),*,,*(4352);*#
 [327] 4362:*,,*(6154),*(5154),*,,*(5164),*,,*(4332);#
 [328] 1336:*,,*(6164),*,,*(1563),*(1553),*,,*(1533),*(1535);*(1536);*#
 [329] 4542:(5424),*,,*,*,,*#
 [330] 4325:*,,*,*(6424),*,,*#
 [331] 3624:*,,*,*(4326),*(4323),*,,*#
 [332] 4424:*,,*,*(3524),*(4624);#
 [333] 3325:*,,*[334],[335],*(1666);*(5332),[336],[337],[338];[339],[340],*(6326);[341];#
 [334] 5642:*,,*,*(6542),*,,*(5542);#
 [335] 1566:*,,*(6642),*,,*(1656),*,,*(1556);#
 [336] 5362:*,,*,*(6352),*(6332),*,,*(5352);#
 [337] 5624:*,,*(6362),*,,*(6524),*,,*(5524);#
 [338] 6624:*,,*(1565),*(1555),*,,*(1665),*(1655),*,,*#
 [339] 3352:*,,*,*(5323),*,,*#
 [340] 3332:*,,*,*(6323),*(5326),*,,*(3362);#
 [341] 3326:*,,*,*(6325),*(5325);*(3323);#
 [342] 1434:*,,*,*(3144),*,,*(1443),*,,*#
 [343] 1415:*,,*(5144),[344],[345],*,,*[346],[347],*,,*(1454),[348];*(1445);*#
 [344] 4144:*,,*,*(6144),*,,*#
 [345] 4324:*,,*,*(4342),*,,*#
 [346] 1534:*,,*,*(1543),*,,*#
 [347] 1634:*,,*,*(1643),*,,*#
 [348] 1446:*,,*,*(1464),*,,*#
 [349] 1415:*,,*[350],[351],*,,*[352],[353],[354];*,[355],[356];*#
 [350] 5324:*,,*,*(5342),*,,*#
 [351] 3324:*,,*,*(6342),*,,*(3342),*,,*(6324);#
 [352] 1546:*,,*,*(1654),*,,*(1564),*(1554),*,,*#
 [353] 1356:*,,*,*(1664),*,,*(1646);*(1353);#
 [354] 1136:*,,*,*(1363),*,,*(1366),*(1333);*(1336);*#
 [355] 1545:*,,*,*(1645);#
 [356] 1136:*,,*,*(1365),*(1355),*,,*(1335);*#
 [357] 1335:*,,*,*(1544),[358],*,[359],[360];[361];*#
 [358] 1444:*,,*,*(1644);#
 [359] 1343:*,,*,*(1354),*,,*#
 [360] 1346:*,,*,*(1364),*,,*#

[361] 1334:*,*,*,*,*,*,*,*,*(1345),*,*;
 [362] 1213:*,*(2121),*,*,*,*(2112),*,*,*(1221),*,*(1212);*
 [363] 1223:(2132),[364],[365],*,*,*,[366],[367],*,[368],[369],[370];[371];#
 [364] 2145:*,*,*,*,*,*(3112),*,*(2152),*(2162);*(2142);*
 [365] 4115:*,*,*,*,*,*,*,*(5112),*(6112);*(4112);*
 [366] 2145:*,*(1312),*,*,*(3121),*,*(2124),*(2126);*(2125);*
 [367] 4512:*,*(5121),*(6121),*,*,*(4121),*,*(1412),*(1612);*(1512);*
 [368] 1232:(2123),*,*,*,*,*,*,*,*,*;
 [369] 1245:*,*,*,*,*,*(1321),*,*(1252),*(1262);*(1242);*
 [370] 1415:*,*,*,*,*,*(1521),*(1621),*,*(1421),*,*;
 [371] 1145:*,*,*,*,*,*(1224),*(1226),*,*(1225),*,*;
 [372] 1234:[373],[374],[379],[386],*,[387],[388],[396],[405],[411],[412],[413];*(1134);*
 [373] 3142:*,*,*,*,*(4123),*,*,*,*,*,*;
 [374] 1325:(3152),[375],[376],*,*(5123),[377],[378],*,*(3125),*(4125),*(4322);*;
 [375] 4125:*,*,*,*,*(5142),*(3162),*(4152),*,*,*;
 [376] 4162:*,*,*,*,*(6142),*,*(4142);#
 [377] 3126:*,*,*,*,*(6123),*,*(3123);#
 [378] 3422:*,*,*,*,*(4126),*,*,*,*;
 [379] 1325:*,[380],*(6162),*,*,[381],[382],[383],*,[384],[385];*;
 [380] 5162:*,*,*,*,*(6152),*,*(5152);#
 [381] 3522:*,*,*,*,*(5126),*,*,*,*;
 [382] 4522:*,*,*,*,*(6126);*(5422),*(3622);*;
 [383] 4622:*,*,*,*,*(6422),*(2422),*(4422);#
 [384] 5125:*,*,*,*,*(5322);*(6125);#
 [385] 2116:*,*(6322),*(3322),*,*,*(2322),*,*,*;
 [386] 2515:*,*(5622),*(6622),*,*(5522),*(6522),*(2622);*,*(2522);*;
 [387] 1323:*,*(4132),*,*,*(3124),*,*(1342),*(1423);*
 [388] 1352:*,*(5124),[389],*,[390],[391],[392],*(1325),[393],[394];[395];#
 [389] 4124:*,*,*,*,*(6124);#
 [390] 1523:(5132),*,*,*,*,*,*,*;
 [391] 1623:(6132),*(3132),*,*,*,*(1425);*;
 [392] 1143:*,*,*,*,*(1426),*,*,*,*;
 [393] 1323:*,*,*,*,*(1542),*,*,*(1326);#
 [394] 1462:*,*,*,*,*(1642),*,*(1442);#
 [395] 1362:*,*,*,*,*(1452);*;
 [396] 2156:[397],[398],*,[399],*,[400],[401],[402],[403],*,[404];*;
 [397] 1562:*,*,*,*(1625),*,*,*,*;
 [398] 1525:*,*,*,*,*(1662),*,*,*,*;
 [399] 3222:*,*,*,*,*(4222);#
 [400] 1526:*,*,*,*(1652),*,*,*,*;
 [401] 1552:*,*,*,*,*(1626),*,*,*,*;
 [402] 1145:*,*,*,*,*(1163);*;
 [403] 1113:*,*,*,*,*(1141),*,*;
 [404] 1146:*,*,*,*,*(1153);*;
 [405] 1315:*,*,*(5222),[406],*,[407],[408],*,[409],[410];*(1115);*
 [406] 2222:*,*,*,*,*(6222);#
 [407] 1151:*,*,*,*,*(1156);#
 [408] 1161:*,*,*,*,*(1166);#
 [409] 1155:*,*,*,*,*(1165);#
 [410] 1111:*,*,*,*,*(1116);#
 [411] 1324:*,*,*,*(1432),*,*,*,*;
 [412] 3526:*,*(1632),*(1332),*,*(1532),*(1624),*(1424);*,*(1524);*;
 [413] 1536:*,*,*,*,*(4141),[415],*(1135),[416];*(1136);*
 [414] 1154:*,*,*,*,*(1164);#
 [415] 1114:*,*,*,*,*(1144);#
 [416] 1131:*,*,*,*,*(1133);#
 [417] 1223:*,*,*,*(3122),[418],[421],*(1322),[423],[424];[426];*
 [418] 1145:*,*,*,*,[419],[420],*,*(1132);*;
 [419] 4122:*,*,*,*,*(5122);#
 [420] 2122:*,*,*,*,*(6122);#
 [421] 1114:*,*,*,*,*(1142),[422];*(1112);*
 [422] 1152:*,*,*,*,*(1162);#
 [423] 1415:*,*,*,*,*(1522),*(1622),*,*(1422);*;
 [424] 1114:*,*,*,*,*(1121),[425];*(1124);*
 [425] 1125:*,*,*,*,*(1126);#
 [426] 1123:*,*,*,*,*(1222);*;

付録 B 補完に用いたプログラム

Knuth の論文に掲載されている戦略を読み込み，補完に必要なコードを表示するプログラムである．

```
#include <stdio.h>
#include <stdlib.h>

struct guess {
    unsigned int pattern_1: 4;
    unsigned int pattern_2: 4;
    unsigned int pattern_3: 4;
    unsigned int pattern_4: 4;
    unsigned short next_node;
};

struct node {
    struct guess guess[14];
};

#define MAX_NODES 1300

struct guess first_guess;
struct node strategy[MAX_NODES];
int n_node = 0;

struct result{
    unsigned int check: 2;
    unsigned int black: 3;
    unsigned int white: 3;
};

#define MAX_DEPTH 5

struct result hit[MAX_DEPTH][1296];
struct guess guess[MAX_DEPTH];
int index[MAX_DEPTH];
struct result bw_table[14] = {
    {0, 0, 4},
    {0, 0, 3},
    {0, 0, 2},
    {0, 0, 1},
    {0, 0, 0},
    {0, 1, 3},
    {0, 1, 2},
    {0, 1, 1},
    {0, 1, 0},
    {0, 2, 2},
    {0, 2, 1},
    {0, 2, 0},
    {0, 3, 0},
    {0, 4, 0},
};

struct guess
read_guess()
{
    struct guess guess;
    int c;

    do {
```

```

        c = getchar();
        if (c == EOF){
            printf("error in %d\n", __LINE__);
            exit(1);
        }
    } while (c == ' ' || c == '\n' || c == '\t');

    switch (c){
    case '*':
        guess.pattern_1 = 0xf;
        guess.pattern_2 = 0xf;
        guess.pattern_3 = 0xf;
        guess.pattern_4 = 0xf;
        guess.next_node = 0xffff;
        break;
    case '#':
        guess.pattern_1 = 0xe;
        guess.pattern_2 = 0xe;
        guess.pattern_3 = 0xe;
        guess.pattern_4 = 0xe;
        guess.next_node = 0xffff;
        break;
    case '(':
        guess.pattern_1 = getchar() - '1';
        guess.pattern_2 = getchar() - '1';
        guess.pattern_3 = getchar() - '1';
        guess.pattern_4 = getchar() - '1';
        c = getchar();
        switch (c){
        case ')':
            guess.next_node = 0xffff;
            break;
        case ':':
            guess.next_node = n_node++;
            if (n_node > MAX_NODES){
                printf("error in %d\n", __LINE__);
                exit(1);
            }
            break;
        default:
            printf("error in %d\n", __LINE__);
            exit(1);
            break;
        }
        break;
    default:
        printf("error in %d\n", __LINE__);
        exit(1);
        break;
    }
    return(guess);
}

char *
code_string(int id)
{
    static char str[5];

    str[3] = (id % 6) + '1';
    id /= 6;
    str[2] = (id % 6) + '1';
    id /= 6;
    str[1] = (id % 6) + '1';
    id /= 6;
    str[0] = (id % 6) + '1';
    str[4] = '\0';
    return(str);
}

```

```
void
read_node(int node_index)
{
    char comma[15] = ",,,,,,,,)";
    int i;
    int c;

    for (i = 0; i < 14; i++){
        struct guess tmpguess = read_guess();
        strategy[node_index].guess[i] = tmpguess;
        if (tmpguess.next_node != 0xffff){
            read_node(tmpguess.next_node);
        }
        if ((c = getchar()) != comma[i]){
            printf("error in %d\n", __LINE__);
            exit(1);
        }
    }
    return;
}

struct result
hikaku(struct guess guess, int id){
    int i, j = 0;
    struct result result;
    int ma1[4];
    int br1[4];
    int ma2[4];
    int br2[4];
    int brf[4] = {0, 0, 0, 0};

    ma1[3] = id % 6;
    id /= 6;
    ma1[2] = id % 6;
    id /= 6;
    ma1[1] = id % 6;
    id /= 6;
    ma1[0] = id % 6;

    br1[0] = guess.pattern_1;
    br1[1] = guess.pattern_2;
    br1[2] = guess.pattern_3;
    br1[3] = guess.pattern_4;

    result.check = 1;
    result.black = 0;
    result.white = 0;

    for (i = 0; i < 4; i++){
        if (ma1[i] == br1[i]){
            result.black++;
        } else {
            ma2[j] = ma1[i];
            br2[j] = br1[i];
            j++;
        }
    }

    for (i = 0; i < 4 - (int)result.black; i++){
        for (j = 0; j < 4 - (int)result.black; j++){
            if ((ma2[i] == br2[j]) && (brf[j] == 0)){
                result.white++;
                brf[j] = 1;
                break;
            }
        }
    }
    return(result);
}
```

```
char comma[15] = ".,,:;.,,:;.,,:;";
int i;
int c;

for (i = 0; i < 14; i++){
    struct guess tmpguess = read_guess();
    strategy[node_index].guess[i] = tmpguess;
    if (tmpguess.next_node != 0xffff){
        read_node(tmpguess.next_node);
    }
    if ((c = getchar()) != comma[i]){
        printf("error in %d\n", __LINE__);
        exit(1);
    }
}

return;
```

```

struct result
hikaku(struct guess guess, int id){
    int i, j = 0;
    struct result result;
    int ma1[4];
    int br1[4];
    int ma2[4];
    int br2[4];
    int brf[4] = {0, 0, 0, 0};

    ma1[3] = id % 6;
    id /= 6;
    ma1[2] = id % 6;
    id /= 6;
    ma1[1] = id % 6;
    id /= 6;
    ma1[0] = id % 6;

    br1[0] = guess.pattern_1;
    br1[1] = guess.pattern_2;
    br1[2] = guess.pattern_3;
    br1[3] = guess.pattern_4;

    result.check = 1;
    result.black = 0;
    result.white = 0;

    for (i = 0; i < 4; i++){
        if (ma1[i] == br1[i]){
            result.black++;
        } else {
            ma2[j] = ma1[i];
            br2[j] = br1[i];
            j++;
        }
    }

    for (i = 0; i < 4 - (int)result.black; i++){
        for (j = 0; j < 4 - (int)result.black; j++){
            if ((ma2[i] == br2[j]) && (brf[j] == 0)){
                result.white++;
                brf[j] = 1;
                break;
            }
        }
    }

    return(result);
}

```

```

}

int
index_number(int black, int white){
    int i;
    for (i = 0; i < 14; i++){
        if (black == bw_table[i].black && white == bw_table[i].white){
            break;
        }
    }
    return(i);
}

void
strategy_check(){
    int depth, i, j;
    FILE *fp = fopen("output.txt", "wt");

    for (i = 0; i < 1296; i++){
        hit[0][i].check = 1;
    }
    guess[0] = first_guess;
    depth = 0;

    for (;;){
        if ((guess[depth].pattern_1 == 0xf && guess[depth].pattern_2 == 0xf &&
            guess[depth].pattern_3 == 0xf && guess[depth].pattern_4 == 0xf &&
            guess[depth].next_node == 0xffff) || (guess[depth].pattern_1 == 0xe &&
            guess[depth].pattern_2 == 0xe && guess[depth].pattern_3 == 0xe &&
            guess[depth].pattern_4 == 0xe && guess[depth].next_node == 0xffff)){
            if (index[depth-1] < 13){
                for (i = 0; i < 1296; i++){
                    if (hit[depth][i].check == 0){
                        continue;
                    }
                    for (j = 0; j < depth; j++){
                        fprintf(fp, "\"%d%d%d%d\" %d-%d(%d 番目) ",
                            guess[j].pattern_1 + 1, guess[j].pattern_2 + 1,
                            guess[j].pattern_3 + 1, guess[j].pattern_4 + 1,
                            bw_table[index[j]].black, bw_table[index[j]].white, index[j] + 1);
                    }
                    fprintf(fp, "深さ [%d], チェック付きコード\"%s\"(%d).\n",
                        depth + 1, code_string(i), i);
                }
                depth--;
            } else {
                int count = 0;
                for (i = 0; i < 1296; i++){
                    if (hit[depth][i].check > 0){
                        count++;
                    }
                }
                printf("[%d]depth=%d, count=%d\n", __LINE__, depth, count);
                if (count == 0){
                    if(guess[depth].pattern_1 != 0xf || guess[depth].pattern_2 != 0xf ||
                        guess[depth].pattern_3 != 0xf || guess[depth].pattern_4 != 0xf ||
                        guess[depth].next_node != 0xffff){
                        fprintf(fp, "4-0 になるものが無いが何かある.\n");
                        for (j = 0; j < depth; j++){
                            fprintf(fp, "\"%d%d%d%d\" %d-%d(%d 番目) ",
                                guess[j].pattern_1 + 1, guess[j].pattern_2 + 1,
                                guess[j].pattern_3 + 1, guess[j].pattern_4 + 1,
                                bw_table[index[j]].black, bw_table[index[j]].white, index[j] + 1);
                        }
                        fprintf(fp, "\n");
                    }
                }
            }
        }
    }
}

```

```

    } else if (count > 1){
        printf("プログラムエラー.\n");
    } else {
        if(guess[depth].pattern_1 != 0xe || guess[depth].pattern_2 != 0xe ||
            guess[depth].pattern_3 != 0xe || guess[depth].pattern_4 != 0xe ||
            guess[depth].next_node != 0xffff){
            for (j = 0; j < depth; j++){
                fprintf(fp, "\"%d%d%d%d\" %d-%d(%d 番目) ",
                    guess[j].pattern_1 + 1, guess[j].pattern_2 + 1,
                    guess[j].pattern_3 + 1, guess[j].pattern_4 + 1,
                    bw_table[index[j]].black, bw_table[index[j]].white, index[j] + 1);
            }
            fprintf(fp, "\n");
        }
        depth -= 2;
    }

    if (depth < 0){
        break;
    }
    index[depth]++;
} else {
    for (i = 0; i < 1296; i++){
        if (hit[depth][i].check == 1){
            hit[depth][i] = hikaku(guess[depth], i);
        }
    }
    if (guess[depth].next_node == 0xffff){
        for( i = 0; i < 1296; i++){
            if (hit[depth][i].check == 1){
                if (hit[depth][i].black != 4 || hit[depth][i].white != 0){
                    for (j = 0; j < depth; j++){
                        fprintf(fp, "\"%d%d%d%d\" %d-%d(%d 番目) ",
                            guess[j].pattern_1 + 1, guess[j].pattern_2 + 1,
                            guess[j].pattern_3 + 1, guess[j].pattern_4 + 1,
                            bw_table[index[j]].black, bw_table[index[j]].white, index[j] + 1);
                    }
                        fprintf(fp, "戦略 \"%d%d%d%d\"の時\n",
                            guess[depth].pattern_1 + 1, guess[depth].pattern_2 + 1,
                            guess[depth].pattern_3 + 1, guess[depth].pattern_4 + 1);

                        fprintf(fp, "4-0 になっていないので続きがある. %d-%d(%d 番目) \"%s\"(%d)\n",
                            hit[depth][i].black, hit[depth][i].white,
                            index_number(hit[depth][i].black, hit[depth][i].white) + 1,
                            code_string(i), i);
                    }
                }
            }
            depth--;
            if (depth < 0){
                break;
            }
            index[depth]++;
        }
    } else {
        index[depth] = 0;
    }
}
for (i = 0; i < 1296; i++){
    if (hit[depth][i].check == 0){
        hit[depth+1][i].check = 0;
        continue;
    }
    if (hit[depth][i].black == bw_table[index[depth]].black &&
        hit[depth][i].white == bw_table[index[depth]].white){
        hit[depth+1][i].check = 1;
    } else {

```

```

        hit[depth+1][i].check = 0;
    }
}

guess[depth+1] = strategy[guess[depth].next_node].guess[index[depth]];
depth++;
}
fclose(fp);
return;
}

int
main(){
    first_guess = read_guess();
    if (first_guess.next_node != 0xffff){
        read_node(0);
    }
    printf("[%d]", __LINE__);

    strategy_check();
    return(0);
}

```