信州大学工学部

学士論文

同時質問マスターマインドの最適質問数は 6 である

指導教員　西新 幹彦 准教授

学科　電子情報システム工学科
学籍番号　19T2002B
氏名　秋好 伸之輔

2023 年 3 月 27 日

目次

1　はじめに

本論文ではマスターマインドという 2 人でプレイする対戦型推理ゲームの最適戦略について論じる．

このゲームは古くから存在し，Bulls and Cows や Hit and Blow，そしてマスターマインドなど様々な名前で呼ばれている．1970 年代前半にイギリスのインヴィクタ社からマスターマインドという商品名でボードゲームとして発売され，その後アメリカでハスブロ社から発売されるなど，世界中で発売された．また，マスターマインドのような対戦型推理ゲームは様々な番組になって楽しまれてきた．例えば，1946 年からは，アメリカで Twenty Questions，日本では，これをモデルにした二十の扉というゲーム番組が放送されていた．このゲームは，解答者がヒントによって対象を絞り込んでいくという点がマスターマインドと共通している．マスターマインドと同じルールのゲーム番組としては，2011 年から 2014 年までフジテレビで放送されていた NumerOn[1] がある．

マスターマインドは，出題者と解答者の 2 人のプレイヤーによって行われる．出題者が秘密のコードというものを決め，解答者が秘密のコードを言い当てるために推測を行い，それを出題者に質問する．出題者は，解答者の推測が秘密のコードにどのくらい近いかヒントを与える．解答者が質問してから出題者がヒントを返すまでの手順を 1 手とし，推測したコードが秘密のコードに一致するまで，この手順を繰り返し行う．2 人のプレイヤーは出題者と解答者の役目を 1 回ずつ行い，手数が小さいプレイヤーの勝ちとなる．

解答者の振るまいをあらかじめ記述したものを戦略と呼ぶ．戦略を決めると，秘密のコードごとにそれを当てるまでの手数が決まる．特に関心があるのは，秘密のコードを言い当てるまでの最大手数である．これをその戦略の最悪手数と呼ぶ．最悪手数に関して，これまでに様々な研究が行われてきている．1976 年に D.E.Knuth[2] は，最悪手数が 5 手の戦略を 1 つ示し，かつ，最悪手数が 5 手より小さな戦略は存在しないことを示している．

本研究では，マスターマインドのルールを変更し，1 度に複数のコードを出題者に提示し，それらに対するヒントをまとめて受け取ることで，秘密のコードを明らかにすることを考える．すると，どのようなコードの組み合わせを提示すれば良いかが問題となる．秘密のコードを特定するために必要な最小のコードの個数を最適質問数と呼ぶ．

従来研究 [3] によって，最適質問数が 5 または 6 であることが明らかにされ，秘密のコードを特定できる 6 つのコードの組み合わせが 1 つ示されている．その組み合わせは，1122, 3344, 2455, 6564, 5313, 2623 である．本研究では，最適質問数が 5 つにならないことを示すことによって，最適質問数が 6 つになることを明らかにした．

2　マスターマインドのルール

ここでは，通常のマスターマインドのルールを説明する．

2.1　ゲームのルール

マスターマインドのルールは次の通りである．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．数字の重複を許して 4 つの数字を並べたものをコードと呼ぶ．したがってコードは全部で $6^4 = 1296$ 種類存在する．プレイヤーは，出題者と解答者に分かれ，以下の手順でゲームが行われる．

1. 出題者は，コードを 1 つ選んで解答者に知られないようにする．これを秘密のコードと呼ぶ．
2. 解答者は，秘密のコードを推測して 1 つのコードを出題者に提示する．これを出題者への質問と呼ぶ．
3. 出題者は解答者の推測したコードが，どの程度秘密のコードに近いかをヒントというもので示す．ヒントは，2 つのコードを比べて，位置と数字が合っている数と，位置は違うが使われている数字のみ合っている数の 2 つで表される．
4. 位置と数字が合っている数が 4 のとき，つまり推測したコードが秘密のコードと一致したとき，ゲームは終了する．このことを秘密のコードを正答すると言うことにする．正答しなければ手順 2 に戻って解答者は改めて秘密のコードを推測する．

2 から 3 までの手順を 1 手と数え，この手順を繰り返した回数を手数と言うことにする．2 人のプレイヤーは，出題者と解答者の役割を交互に行い，秘密のコードを正答するまでの手数の小さいプレイヤーの勝ちとなる．

2.2　ヒント

前節でヒントについて簡単に説明したが，本節でヒントに関する厳密な定義を与える．秘密のコード $c = c_1c_2c_3c_4$ と推測 $q = q_1q_2q_3q_4$ に対して，両者の近さを表すヒントを $h(c, q)$ と書く．ヒントは 2 つの整数 r, w を用いて (r, w) という形式で表される．この r と w は次のように決まる．

1. $c_j = q_j$ を満たす j の数を r とする．
2. $c_j = i$ を満たす j の数を m_i ，$q_j = i$ を満たす j の数を n_i とし， w を

$$w = \sum_{i=1}^{6} \min(m_i, n_i) - r$$

と定義する．

秘密のコードと推測は入れ替えても同じヒントが得られる．例えば，$h(1123, 2413) = h(2413, 1123) = (1, 2)$ である．実際にヒントとして取り得る値は

$$(r, w) = (0, 4), (0, 3), (0, 2), (0, 1), (0, 0), (1, 3), (1, 2),$$
$$(1, 1), (1, 0), (2, 2), (2, 1), (2, 0), (3, 0), (4, 0)$$

の 14 通りである．特に，ヒント (4,0) が出題者から提示された場合，解答者は秘密のコードに正答したということを表している．

3　同時質問マスターマインド

ここでは，同時質問マスターマインドのルールと秘密のコードを特定するための原理について説明する．

3.1　ルールの変更点

本研究では，マスターマインドのルールに変更を加える．通常のマスターマインドでは，解答者は 1 回の質問で 1 つのコードを提示する．そして，それに応じたヒントをもとにして次の質問を行う．一方，本研究では解答者に許される質問の提示回数を 1 回のみとし，その代わり，複数のコードを同時に質問できるものとする．出題者はそれらに対応するヒントを解答者に一度に与える．そして，解答者の目的は，出題者から (4,0) のヒントを得ることではなく，得られたヒントから秘密のコードを特定することとする．秘密のコードを特定するために使う最も少ないコード数を最適質問数と呼ぶことにする．この最適質問数を求めることが本研究の目的である．

3.2　同時質問で秘密のコードを特定する原理

秘密のコードを特定できる組み合わせとはどのようなものなのか，以下で説明する．

図 1 の左側の集合には，1296 個の全てのコードが入っている．これは，出題者が選ぶことのできるコード全体を表している．今，これらの全てのコードに対して，1122 というコード (推測) に対するヒントを求める．そして得られるヒントごとにコードをまとめると，いくつかの部分集合が得られる．ヒントは 14 種類あるので，部分集合の数は高々 14 である．この作業

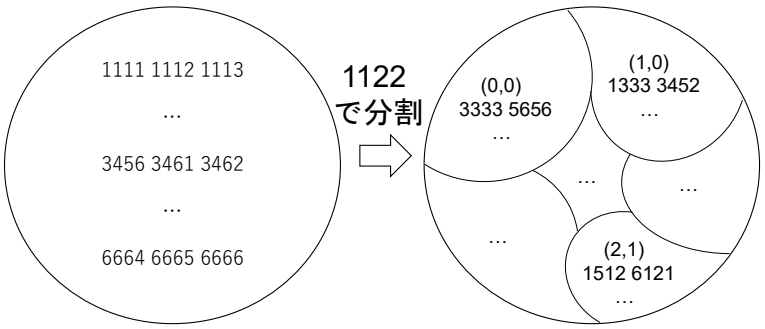

図 1　1122 による分割

を，1122 による分割と呼ぶことにする．2 つ目のコードを用いてさらに分割を行うと，コード全体は高々 14^2 個の部分集合に分割される．一般に，コードを n 個用いると，秘密のコードの集合は，高々 14^n 個の部分集合に分割される．十分多くのコードを用いれば，分割した結果の各部分集合の要素数を 1 以下にすることができる．このとき，分割に使用した全てのコードを出題者に提示すれば，それらに対応するヒントから秘密のコードを特定することができる．すなわち，これらのコードの組み合わせが，解答者の提示すべき質問となる．

4　探索範囲の削減

本研究の目的は，秘密のコードを特定できる質問を探索することに帰着するが，全ての組み合わせを探索するのは効率が悪い．ここでは，最適質問数を効率的に求めるために行った探索範囲の削減方法を，等価性と代表的な推測という 2 つの考え方を使って説明する．

4.1　コードの等価性

n 個のコードの全ての組み合わせを探索すると， 1296^n 通りの質問の組み合わせを考えなければならない．ここまで多くの組み合わせについて考えるのは効率的ではない．幸い，以下で指摘するようなことから，探索範囲を削減することができる．

図 2 は，1331 が 1122 へ変換される様子を表している．1331 に対して，左から 2 番目の「3」という数字と左から 4 番目の「1」という数字を入れ替えると 1133 というコードになる．これを位置の置換と呼ぶ．さらに，数字の「3」と数字の「2」を入れ替えると、1122 というコードになる．これを数字の置換と呼ぶ．一方，置換によって互いに変換できないコードも存在する．例えば，1331 というコードに対して位置と数字の置換を行っても，5555 というコードに変換することはできない．このように，位置と数字の置換によって互いに変換できるコードとできないコードが存在する．

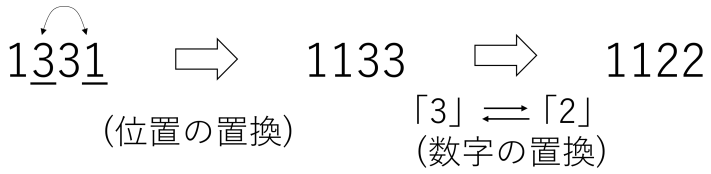

図 2　1331 を 1122 に置換

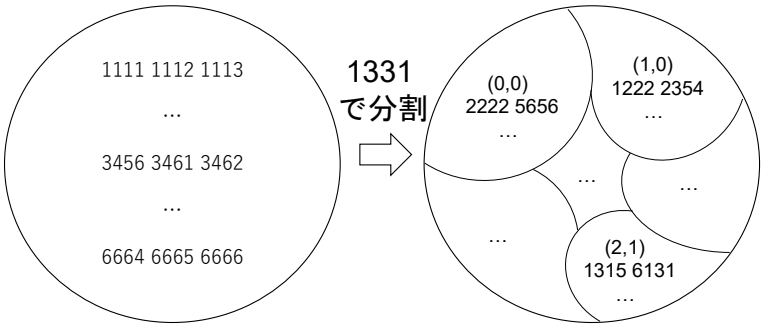

図 3　1331 による分割

これをふまえて，1331 による分割を考える (図 3)．先ほど行った 1331 を 1122 に変換する位置と数字の置換を，1331 で分割された集合の各要素に適用すると，図 1 のような 1122 で分割された集合と同じものが得られる．例えば，1331 で分割されたヒント (2,1) の要素である 1315 というコードに着目する．先ほどの置換同様，左から 2 番目の「3」と左から 4 番目の「5」を入れ替えると 1513 になる．そして，「3」と「2」を入れ替えることで，1512 というコードができあがる．この 1512 は，1122 で分割した時のヒント (2,1) を満たしていることがわかる．このことから，2 つのコードが位置と数字の置換で互いに変換できるとき，それらのコードは互いに等価であると言うことにする．等価であるコード同士は置換を行うことで，お互いの分割結果を求めることができる．よって，等価なコードの中から 1 つだけ選んで探索を行えば良い．ここで，等価であるコードの中でコードの並びを 4 桁の整数と見なしたときに，最も値が小さいコードを代表的な推測と呼ぶことにする．例えば，1122 と等価なコードの中の代表は 1122 である．

4.2　代表的な推測と代表的な質問

代表的な推測を求めるには，位置と数字に関して全通りの置換を行う必要がある．具体的には，位置の置換が 4! 回，数字の置換が 6! 回行われ，さらにこれを分割に使うコードの数だけ行うことになる．

全てのコードに対して，等価になるコードごとに集めると，5 つのグループに分けられる．それぞれのグループの中で最も値が小さいコードは，1111，1112，1122，1123，1234 の 5 つとなり，この 5 つのコードが代表的な推測となる．この 5 つのグループに，それぞれ ① から ⑤ までの番号を割り当てる．

代表的な推測の考え方は，2 つ以上のコードからなる質問にも応用できる．まず，2 つのコードからなる質問への応用を考える．1 つ目のコードは，先ほど述べた 5 つのコードのいずれかにすれば十分である．一方，2 つ目のコードとしては任意のコードを考える．その上で，2 つのコードからなる質問の等価性を以下のように導入する．例えば，1234, 2222 の組み合わせと，1234, 5555 の組み合わせを考える．前者について，左から 1 番目の数字と左から 2 番目の数字に関して位置の置換を行うと，2134, 2222 という組み合わせが出てくる．さらに，「2」と「1」に関して数字の置換を行うと，1234, 1111 という組み合わせが出てくる．一方，後者について，どのような置換を行っても 1234, 1111 という組み合わせは出てこない．2 つ目のコードのみに着目すると，2222 と 5555 は「2」と「5」を置換することで，互いに変換できるため，等価なコードである．しかし，2 つ組のコードに関して等価性を考えると，1234，2222 の組み合わせと 1234，5555 の組み合わせは等価ではない．このように，代表的な推測と任意のコードを組にすることによって，2 つ組の等価性を調べる．等価になるコードの組ごとにグループを作り，2 つのコードをそれぞれ 4 桁の整数と見なした時に，その和が最も小さくなるものを代表的な質問とする．一般に n 個組のコードに関しても同様に，$(n-1)$ 個組の代表に対して n 個目のコードを追加しながら質問の等価性を調べることによって，代表的な質問を選ぶ．

さらに，複数のコードからなる質問ではコードの順番を考える必要がないので，これを探索範囲の削減に利用できる．具体的には，表 1 のように代表的な質問を求める際に使うコードの候補を削減する．例えば，1 つ目のコードが ③ の場合，2 つ目のコードは， ① から ③ のグループの中から選び，2 つ組の等価性を調べて代表的な質問を見つける．一般に，n 個組のコードに関して n 個目のコードは，$(n-1)$ 個目のコードの番号までのグループの中から選び，n 個組の等価性を調べて代表的な質問を見つけることによって，探索範囲の削減を行う．

表 1　2 つのコードを使った質問

1 つ目のコード	2 つ目のコード
① 1111	① に属するコード
② 1112	① から ② に属するコード
③ 1122	① から ③ に属するコード
④ 1123	① から ④ に属するコード
⑤ 1234	任意のコード

5　代表の個数と探索結果

5.1　コード数が 4 以下における探索

第 4 章で述べた方法で，質問に含まれるコード数ごとに代表的な質問を求め，その代表的な質問を使って秘密のコードの集合の分割を行った．その結果，4 つのコードを使った代表的な質問を用いて秘密のコードの集合を分割しても，全ての部分集合の要素数は 1 以下にならなかった．これより，最適質問数が 4 にならないことが確認できた．このとき，コード数ごとの代表的な質問の数とそれに要した時間は表 2 のようになった．

5.2　5 つのコードにおける探索

表 2 の通り，コード数が 4 のときに約 2 日かかったので，コード数が 5 のときにどのくらい時間がかかるか次のように試算した．コード数が 2 から 3 に増えると代表的な質問の個数は，2 桁増えている．これより，プログラムが終わるまでにかかる時間が，約 6^2 倍されていると予想した．同じように，コード数が 3 から 4 に増えると代表的な質問の個数は，3 桁増えている．これより，プログラムが終わるまでにかかる時間が，約 6^3 倍されていると予想した．同じように，コード数が 4 個から 5 個に増えると代表的な質問の個数は，4 桁増えるのではないかと予想し，約 6^4 倍の時間がかかってしまうだろうと考えた．つまり，このままのアルゴリズムだとプログラムに要する時間が，

$$2 \times 6^4 = 2592$$

より約 2600 日かかってしまうだろうと予想された．そこで，プログラムの高速化を模索した．

ここで，コードの数が 1 つ増えると，代表的な質問の個数は高々 1296 倍にしかならないことに注意する．コードは全部で 1296 種類なので，1296 倍よりも大きくなることはない．表 2 より，コード数が 1 から 2 に増えた時，代表的な質問の個数が 5 から 194 に増えているため，約 39 倍になっている．2 から 3 に増えた時は約 329 倍になっており，3 から 4 に増えた時は約 656 倍になっている．この時点で，1296 の半分以上の倍率で増えている．これより，コード数が 4 から 5 に増えた時には 656 倍よりも大きくなることが予想された．これより，コードの数が多いと，コードが 1 つ増えた時の代表的な質問の個数が 1296 倍に近い数値になると考え，等価な質問を探索するメリットが薄いと考えた．コード数が少ない時には，等価な質問を集めたグループに含まれるコードは多いが，コード数が増えると，各グループに含まれるコードは少なくなる．つまり，探索範囲を小さくするために時間をかけて等価な質問を見つけようとしても，探索範囲はそれほど小さくならないと考えた．よって，5 つのコードを使った代表的な質問を求めずに，5 つ目のコードの候補となる全てのコードに対して探索を行った．これ

表 2　コード数ごとの代表的な質問の個数と探索にかかった時間

コード数	代表的な質問の個数	探索にかかった時間
1 個	5 個	約 1 秒
2 個	194 個	約 20 秒
3 個	63903 個	約 12 分
4 個	41923954 個	約 2 日

によって調べる質問の数は確実に 4 つ組の代表数の 1296 倍になるが，等価性を網羅的に調べるための計算を省くことができる．

さらに，5 つのコードにおける探索を行う際には，4 つの代表的な質問の中で部分集合の要素数の最大値が 14 以下のものだけを取り出せば良い．部分集合の中に 15 個以上の要素がある集合が存在している場合，5 つ目のコードでヒントごとに分割しても，ヒントは 14 種類なので，必ず要素数が 2 以上の集合が存在することになる．このことを使うと，元々の 4 つの代表的な質問が 41923954 個だったのに対して，35242748 個にまで絞り込むことができ，

$$\frac{35242748}{41923954} \approx 0.84$$

より 4 つの代表的な質問の数を約 16 % 削減することができる．

このように，等価性を確認するための計算を省略し，さらに 4 つの代表的な質問を絞り込んだ結果，約 4 日で結果を求めることができた．そしてその結果，5 つのコードで全ての部分集合の要素数が 1 以下になることがなかった．すなわち，最適質問数が 5 にならないことが判明した．従来研究 [3] より同時質問に使用する最適質問数が 5 または 6 であることがわかっているので，最適質問数が 6 であることが判明した．

6　まとめ

本研究ではマスターマインドのルールを変更し，1 度に複数のコードを提示した場合に，秘密のコードを特定するためのコード数がいくつになるのかを調べた．その際に，コードの等価性や代表的な質問，といった効率良く最適質問数を求めるための考え方を実行した．その結果，同時質問マスターマインドの最適質問数が 6 であることが判明した．一方，通常のマスターマインドでは，D.E.Knuth[2] によって，秘密のコードを特定するために必要なコード数が 4 であることがわかっている．したがって，まとめて質問する場合と 1 つずつ質問する場合では，まとめて質問する方が秘密のコードを特定するのに，多くのコードが必要であることが明らかになった．

謝辞

本研究にあたり，細かく指導してくだ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]「ヌメロン - フジテレビ」，https://www.fujitv.co.jp，2022 年 12 月閲覧．

[2] D. E. Knuth, “The Computer as Master Mind,” J.Recreational Mathematics, Vol.9(1), pp.1–6, 1976–77.

[3] 小澤泰雅，「同時質問マスターマインドにおける最適な質問」，信州大学大学院理工学系研究科修士論文（指導教員：西新幹彦），2022 年 3 月．

付録 A　ソースコード

A.1　1 つのコードの代表的な推測を求めるプログラム

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define CODELEN (4)
#define DIVPAT (14)
#define CODEKIND (6)
#define CODENUM (1296)

//int から文字に
void int_arr(int code, char pat[CODELEN]){
    for (int i=0;i<CODELEN;i++){
        pat[CODELEN-i-1] = '1' + code % 6;
        code /= 6;
    }
    return;
}

//文字から int に
int arr_int(char pat[CODELEN]){
    int code = 0;
    code = (pat[0] - '1')*6*6*6 + (pat[1] - '1')*6*6
            +(pat[2] - '1')*6 + (pat[3] - '1');
    return code;
}

//場所いれかえ
void perm(int n, char pat[CODELEN]){
    char tmp;
    for (int i=0;i<3;i++){
        int r = n % (4 - i);
        n /= (4 - i);
        tmp = pat[i];
        pat[i] = pat[i+r];
        pat[i+r] = tmp;
        //printf("%d %d\n",r,n);
    }
    return;
}

//数字いれかえ
void permn(int n, char pat[CODELEN]){
    char perm[6] = {'1','2','3','4','5','6'};
    char tmp;

    for (int i=0;i<6;i++){
        int r = n % (6 - i);
        n /= (6 - i);
        tmp = perm[i];
        perm[i] = perm[i+r];
        perm[i+r] = tmp;
    }
    for (int i=0;i<4;i++){
        pat[i] = perm[pat[i]-'1'];
    }
    return;
}

int main(void){
```

```
    //置換時の配列
    char pat1[CODELEN];

    //置換後
    int onepat[6];
    int n = 0; //代表コードの数
    int gr1[5]; //代表コードのグループ番号を格納

    //置換開始
    for (int i=0;i<CODENUM;i++){
        int code = 0;
        int min = 1300; //無限
        //場所いれかえ
        for (int j=0;j<24;j++){
            //数字いれかえ
            for (int k=0;k<720;k++){
                int_arr(i,pat1);
                perm(j,pat1);
                permn(k,pat1);
                code = arr_int(pat1);
                if(code < min){
                    min = code;
                }
            }
        }

        int p = 0; //flag
        for(int i=0;i<n;i++){
            if (onepat[i] == min){
                p = 1;
                break;
            }
        }
        if (p == 0) {
            onepat[n] = min;
            gr1[n] = n+1;
            n++;
        }
    }


    //プリント
    FILE *fp;
    fp = fopen("onecode.txt", "w");
    char pt[CODELEN];
    for (int i=0;i<n;i++){
        fprintf(fp," %d ",onepat[i]);
        fprintf(fp," %d \n",gr1[i]);
        int_arr(onepat[i],pt);
        printf(" %c%c%c%c ",pt[0],pt[1],pt[2],pt[3]);
        printf(" %d \n",gr1[i]);
    }
    fclose(fp);

    return 0;
}
```

A.2　2 つのコードの代表的な推測を求めるプログラム

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
```

```
#define CODELEN (4)
#define DIVPAT (14)
#define CODEKIND (6)
#define CODENUM (1296)

//int から文字に
void int_arr(int code, char pat[CODELEN]){
    for (int i=0;i<CODELEN;i++){
        pat[CODELEN-i-1] = '1' + code % 6;
        code /= 6;
    }
    return;
}

//文字から int に
int arr_int(char pat[CODELEN]){
    int code = 0;
    code = (pat[0] - '1')*6*6*6 + (pat[1] - '1')*6*6
            +(pat[2] - '1')*6 + (pat[3] - '1');
    return code;
}

//場所いれかえ
void perm(int n, char pat[CODELEN]){
    char tmp;
    for (int i=0;i<3;i++){
        int r = n % (4 - i);
        n /= (4 - i);
        tmp = pat[i];
        pat[i] = pat[i+r];
        pat[i+r] = tmp;
        //printf("%d %d\n",r,n);
    }
    return;
}

//数字いれかえ
void permn(int n, char pat[CODELEN]){
    char perm[6] = {'1','2','3','4','5','6'};
    char tmp;

    for (int i=0;i<6;i++){
        int r = n % (6 - i);
        n /= (6 - i);
        tmp = perm[i];
        perm[i] = perm[i+r];
        perm[i+r] = tmp;
    }
    for (int i=0;i<4;i++){
        pat[i] = perm[pat[i]-'1'];
    }
    return;
}

//グループ分け
int group(char pat[CODELEN]){
    int fn[CODEKIND] = {0,0,0,0,0,0};
    int ff[5] = {0,0,0,0,0};

    for (int i=0;i<CODELEN;i++){
        fn[pat[i] - '1']++;
    }

    for (int i=0;i<CODEKIND;i++){
        ff[fn[i]]++;
    }

    int m = ff[1] + ff[2]*5 + ff[3]*5*3 + ff[4]*5*3*2;
    int g;
```

```
    switch(m){
        case 4:
            g = 4; //group5
            break;

        case 7:
            g = 3; //group4
            break;

        case 10:
            g = 2; //group3
            break;

        case 16:
            g = 1; //group2
            break;

        case 30:
            g = 0; //group1
            break;

        default:
            printf("error");
            break;
    }
    return(g);
}

//プリント関数
void print(char pat[CODELEN]){
    printf(" %c%c%c%c ",pat[0],pat[1],pat[2],pat[3]);
    return;
}



int main(void){
    //グループ分け
    int gr[CODENUM]; //グループ 1~5 を順番に
    int gr_num[6] = {0,6,126,216,936,1296}; //代表コードのグループの始め
    char pat[CODELEN];
    int z[5] = {0,0,0,0,0};

    //順番に gr[] に格納
    for (int i=0;i<CODENUM;i++){
        int_arr(i,pat);
        int g = group(pat);
        if (g >= 0 && g <= 4){
            gr[gr_num[g] + z[g]] = i;
            z[g]++;
        } else {
            printf("error");
            break;
        }
    }



    //書き込み gr[]
    FILE *mfp;
    mfp = fopen("maingroup.txt", "w");
    for (int i=0;i<CODENUM;i++){
        fprintf(mfp," %d \n",gr[i]);
        //printf(" %d \n",gr[i]);
    }
    fclose(mfp);

    //書き込み gr_num[]
```

```
FILE *nfp;
nfp = fopen("groupnumber.txt", "w");
for (int i=0;i<6;i++){
    fprintf(nfp," %d \n",gr_num[i]);
    //printf(" %d \n",gr_num[i]);
}
fclose(nfp);

//置換時の配列
char pat1[CODELEN];
char pat2[CODELEN];

//置換後の配列
int twopat1[200];
int twopat2[200];
int g; //2 個目のコードのグループ番号
int gr2[200]; //2 個目のコードのグループ番号を格納
int n = 0; //代表コードの数
int u = 0;

//置換開始
for (int a=0;a<5;a++){
    for (g=0;g<(a+1);g++){
        for (int b=gr_num[g];b<gr_num[g+1];b++){
            int code1 = 0;
            int code2 = 0;
            int min = 1300;
            //場所いれかえ
            for (int c=0;c<24;c++){
                //数字いれかえ
                for (int d=0;d<720;d++){
                    int_arr(gr[gr_num[a]],pat1);
                    int_arr(gr[b],pat2);
                    perm(c,pat1);
                    perm(c,pat2);
                    permn(d,pat1);
                    permn(d,pat2);
                    code1 = arr_int(pat1);
                    code2 = arr_int(pat2);
                    if(gr[gr_num[a]] != code1){
                        continue;
                    }
                    if(code2 < min){
                        min = code2;
                    }
                }
            }

            if(gr[gr_num[a]] == min){
                u++;
                //printf("**%d\n",min);
                continue;
            }

            int p = 0; //flag
            for(int x=0;x<n;x++){
                if (twopat1[x] == gr[gr_num[a]] && twopat2[x] == min){
                    p = 1;
                    break;
                }
            }
            if (p == 0) {
                twopat1[n] = gr[gr_num[a]];
                twopat2[n] = min;
                gr2[n] = g+1;
                n++;
            }
        }
    }
```

```
    }

    //書き込み
    FILE *fp;
    fp = fopen("twocode.txt", "w");
    char pt1[CODELEN];
    char pt2[CODELEN];
    for (int i=0;i<n;i++){
        fprintf(fp," %d ",twopat1[i]);
        fprintf(fp," %d ",twopat2[i]);
        fprintf(fp," %d \n",gr2[i]);
        //int_arr(twopat1[i],pt1);
        //int_arr(twopat2[i],pt2);
        //print(pt1);
        //print(pt2);
        //printf(" %d \n",gr2[i]);
    }
    fclose(fp);
    printf("%d",n);

    return 0;
}
```

A.3　3 つのコードの代表的な推測を求めるプログラム

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define CODELEN (4)
#define DIVPAT (14)
#define CODEKIND (6)
#define CODENUM (1296)

//int から文字に
void int_arr(int code, char pat[CODELEN]){
    for (int i=0;i<CODELEN;i++){
        pat[CODELEN-i-1] = '1' + code % 6;
        code /= 6;
    }
    return;
}

//文字から int に
int arr_int(char pat[CODELEN]){
    int code = 0;
    code = (pat[0] - '1')*6*6*6 + (pat[1] - '1')*6*6
            +(pat[2] - '1')*6 + (pat[3] - '1');
    return code;
}

//場所いれかえ
void perm(int n, char pat[CODELEN]){
    char tmp;
    for (int i=0;i<3;i++){
        int r = n % (4 - i);
        n /= (4 - i);
        tmp = pat[i];
        pat[i] = pat[i+r];
        pat[i+r] = tmp;
        //printf("%d %d\n",r,n);
    }
    return;
```

```
}

//数字いれかえ
void permn(int n, char pat[CODELEN]){
    char perm[6] = {'1','2','3','4','5','6'};
    char tmp;

    for (int i=0;i<6;i++){
        int r = n % (6 - i);
        n /= (6 - i);
        tmp = perm[i];
        perm[i] = perm[i+r];
        perm[i+r] = tmp;
    }
    for (int i=0;i<4;i++){
        pat[i] = perm[pat[i]-'1'];
    }
    return;
}

//グループ分け
int group(char pat[CODELEN]){
    int fn[CODEKIND] = {0,0,0,0,0,0};
    int ff[5] = {0,0,0,0,0};

    for (int i=0;i<CODELEN;i++){
        fn[pat[i] - '1']++;
    }

    for (int i=0;i<CODEKIND;i++){
        ff[fn[i]]++;
    }

    int m = ff[1] + ff[2]*5 + ff[3]*5*3 + ff[4]*5*3*2;
    int g;

    switch(m){
        case 4:
            g = 4; //group5
            break;

        case 7:
            g = 3; //group4
            break;

        case 10:
            g = 2; //group3
            break;

        case 16:
            g = 1; //group2
            break;

        case 30:
            g = 0; //group1
            break;

        default:
            printf("error");
            break;
    }
    return(g);
}

//プリント関数
void print(char pat[CODELEN]){
    printf(" %c%c%c%c ",pat[0],pat[1],pat[2],pat[3]);
    return;
}
```

```
int main(void){
    //読み込み
    int gr[CODENUM]; //グループ 1~5 を順番に
    FILE *mfp;
    mfp = fopen("maingroup.txt", "r");
    if(mfp == NULL){
        printf("error");
    } else {
        for (int i=0;i<CODENUM;i++){
            fscanf(mfp," %d ",&gr[i]);
        }
    }
    fclose(mfp);

    int gr_num[6]; //代表コードのグループの始め
    FILE *nfp;
    nfp = fopen("groupnumber.txt", "r");
    if(nfp == NULL){
        printf("error");
    } else {
        for (int i=0;i<6;i++){
            fscanf(mfp," %d ",&gr_num[i]);
        }
    }
    fclose(nfp);

    //置換時
    char pat1[CODELEN];
    char pat2[CODELEN];
    char pat3[CODELEN];

    //置換後の配列
    int threepat3[CODENUM]; //3 個目のコードを一時的に保存
    int g; //3 個目のコードのグループ番号
    int gr3[CODENUM]; //3 個目のコードのグループ番号を格納
    int n = 0; //代表コードの数
    int c_num[17280]; //置換のパターン保存
    int d_num[17280];
    int u = 0; //コードの重複数

    //書き込み用
    FILE *fp;
    fp = fopen("threecode.txt", "w");

    //読み込み
    FILE *twofp;
    twofp = fopen("twocode.txt", "r");
    int twopat1;
    int twopat2;
    int gr2;
    if(twofp == NULL){
        printf("error");
    } else {
        while ((fscanf(twofp, " %d %d %d ",&twopat1,&twopat2,&gr2)) == 3){
            int y = 0; //代表コードリセット
            //置換開始
            for (g=0;g<gr2;g++){
                for (int b=gr_num[g];b<gr_num[g+1];b++){
                    int code1 = 0;
                    int code2 = 0;
                    int r = 0; //c と d のパターンの数
                    //場所いれかえ
                    for (int c=0;c<24;c++){
                        //数字いれかえ
                        for (int d=0;d<720;d++){
                            int_arr(twopat1,pat1);
```

```
                int_arr(twopat2,pat2);
                perm(c,pat1);
                perm(c,pat2);
                permn(d,pat1);
                permn(d,pat2);
                code1 = arr_int(pat1);
                code2 = arr_int(pat2);
                if(code1 == twopat1 && code2 == twopat2){
                    c_num[r] = c;
                    d_num[r] = d;
                    r++;
                }
            }
        }

        int min = 1300;
        int code3 = 0;
        if (r != 0){
            for(int e=0;e<r;e++){
                int_arr(gr[b],pat3);
                perm(c_num[e],pat3);
                permn(d_num[e],pat3);
                code3 = arr_int(pat3);
                if(code3 < min){
                    min = code3;
                }
            }
        } else {
            printf("cd_error");
            continue;
        }

        if (twopat1 == min || twopat2 == min){
            u++;
            continue;
        }

        int p = 0; //flag
        for(int x=0;x<y;x++){
            if (threepat3[x] == min){
                p = 1;
                break;
            }
        }
        if (p == 0) {
            threepat3[y] = min;
            gr3[y] = g+1;
            y++;
            n++;
        }
    }
}
//書き込み
char pt1[CODELEN];
char pt2[CODELEN];
char pt3[CODELEN];
for (int i=0;i<y;i++){
    fprintf(fp," %d ",twopat1);
    fprintf(fp," %d ",twopat2);
    fprintf(fp," %d ",threepat3[i]);
    fprintf(fp," %d \n",gr3[i]);
    //int_arr(twopat1,pt1);
    //int_arr(twopat2,pt2);
    //int_arr(threepat3[i],pt3);
    //print(pt1);
    //print(pt2);
    //print(pt3);
    //printf(" %d \n",gr3[i]);
}
```

```
        }
    }
    fclose(fp);
    fclose(twofp);
    printf("%d\n",n);
    printf("%d",u);

    return 0;
}
```

A.4　4 つのコードの代表的な推測を求めるプログラム

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define CODELEN (4)
#define DIVPAT (14)
#define CODEKIND (6)
#define CODENUM (1296)

// int から文字に
void int_arr(int code, char pat[CODELEN])
{
    for (int i = 0; i < CODELEN; i++)
    {   pat[CODELEN - i - 1] = '1' + code % 6;
        code /= 6;
    }
    return;
}

//文字から int に
int arr_int(char pat[CODELEN])
{
    int code = 0;
    code = (pat[0] - '1') * 6 * 6 * 6 + (pat[1] - '1') * 6 * 6 + (pat[2] - '1') * 6 + (pat[3] - '1');
    return code;
}

//場所いれかえ
void perm(int n, char pat[CODELEN])
{

    char tmp;
    for (int i = 0; i < 3; i++)
    {
        int r = n % (4 - i);
        n /= (4 - i);
        tmp = pat[i];
        pat[i] = pat[i + r];
        pat[i + r] = tmp;
        // printf("%d %d\n",r,n);
    }
    return;
}

//数字いれかえ
void permn(int n, char pat[CODELEN])
{
    char perm[6] = {'1', '2', '3', '4', '5', '6'};
    char tmp;

    for (int i = 0; i < 6; i++)
```

```
    {
        int r = n % (6 - i);
        n /= (6 - i);
        tmp = perm[i];
        perm[i] = perm[i + r];
        perm[i + r] = tmp;
    }
    for (int i = 0; i < 4; i++)
    {
        pat[i] = perm[pat[i] - '1'];
    }
    return;
}

//プリント関数
void print(char pat[CODELEN])
{
    printf(" %c%c%c%c ", pat[0], pat[1], pat[2], pat[3]);
    return;
}

int main(void)
{
    //読み込み
    int gr[CODENUM]; //グループ 1~5 を順番に
    FILE *mfp;
    mfp = fopen("maingroup.txt", "r");
    if (mfp == NULL)
    {
        printf("error");
    }
    else
    {
        for (int i = 0; i < CODENUM; i++)
        {
            fscanf(mfp, " %d ", &gr[i]);
        }
    }
    fclose(mfp);

    int gr_num[6]; //代表コードのグループの始め
    FILE *nfp;
    nfp = fopen("groupnumber.txt", "r");
    if (nfp == NULL)
    {
        printf("error");
    }
    else
    {
        for (int i = 0; i < 6; i++)
        {
            fscanf(mfp, " %d ", &gr_num[i]);
        }
    }
    fclose(nfp);

    //置換時
    char pat1[CODELEN];
    char pat2[CODELEN];
    char pat3[CODELEN];
    char pat4[CODELEN];

    //置換後の配列
    int fourpat4[CODENUM]; // 4 個目のコードを一時的に保存
    int g;                 // 4 個目のコードのグループ番号
    int gr4[CODENUM];      // 4 個目のコードのグループ番号を格納
    int n = 0;             //代表コードの数
    int c_num[17280];      //置換のパターン保存
    int d_num[17280];
```

```
int q = 0; //コードの重複数

//書き込み用
FILE *fp;
fp = fopen("fourcode.txt", "w");

//読み込み
FILE *threefp;
threefp = fopen("threecode.txt", "r");
int threepat1;
int threepat2;
int threepat3;
int gr3;

if (threefp == NULL)
{
    printf("error");
}
else
{
    while ((fscanf(threefp, " %d %d %d %d",&threepat1,&threepat2,&threepat3,&gr3)) == 4){
        int y = 0; //代表コードリセット
        //置換開始
        for (g=0;g<gr3;g++){
            for (int b = gr_num[g]; b < gr_num[g + 1]; b++)
            {
                int code1 = 0;
                int code2 = 0;
                int code3 = 0;
                int r = 0; // c と d のパターンの数
                //場所いれかえ
                for (int c = 0; c < 24; c++)
                {
                    //数字いれかえ
                    for (int d = 0; d < 720; d++)
                    {
                        int_arr(threepat1, pat1);
                        int_arr(threepat2, pat2);
                        int_arr(threepat3, pat3);
                        perm(c, pat1);
                        perm(c, pat2);
                        perm(c, pat3);
                        permn(d, pat1);
                        permn(d, pat2);
                        permn(d, pat3);
                        code1 = arr_int(pat1);
                        code2 = arr_int(pat2);
                        code3 = arr_int(pat3);
                        if (code1 == threepat1 && code2 == threepat2 &&

                            code3 == threepat3)
                        {
                            c_num[r] = c;
                            d_num[r] = d;
                            r++;
                        }
                    }
                }

                int min = 1300;
                int code4 = 0;
                if (r != 0)
                {
                    for (int e = 0; e < r; e++)
                    {
                        int_arr(gr[b], pat4);
                        perm(c_num[e], pat4);
                        permn(d_num[e], pat4);
                        code4 = arr_int(pat4);
```

```
                        if (code4 < min)
                        {
                            min = code4;
                        }
                    }
                }
                else
                {
                    printf("cd_error");
                    continue;
                }

                if (threepat1 == min || threepat2 == min || threepat3 == min)
                {
                    q++;
                    continue;
                }

                int p = 0; // flag
                for (int x = 0; x < y; x++)
                {
                    if (fourpat4[x] == min)
                    {
                        p = 1;
                        break;
                    }
                }
                if (p == 0)
                {
                    fourpat4[y] = min;
                    gr4[y] = g + 1;
                    y++;
                    n++;
                }
            }
        }
        //書き込み
        char pt1[CODELEN];
        char pt2[CODELEN];
        char pt3[CODELEN];
        char pt4[CODELEN];
        for (int i=0;i<y;i++){
            fprintf(fp, " %d ", threepat1);
            fprintf(fp, " %d ", threepat2);
            fprintf(fp, " %d ", threepat3);
            fprintf(fp, " %d ", fourpat4[i]);
            fprintf(fp, " %d \n", gr4[i]);
        }
    }
}
fclose(fp);
fclose(threefp);
printf("%d\n", n);
printf("%d", q);
return 0;
}
```