正十二面体上のランダム・ウォークに関する入試問題のコンピュータによる検証：３進数の利用

東京薬科大学生命科学部　高須 昌子[1]、小島 正樹[2]

１．はじめに

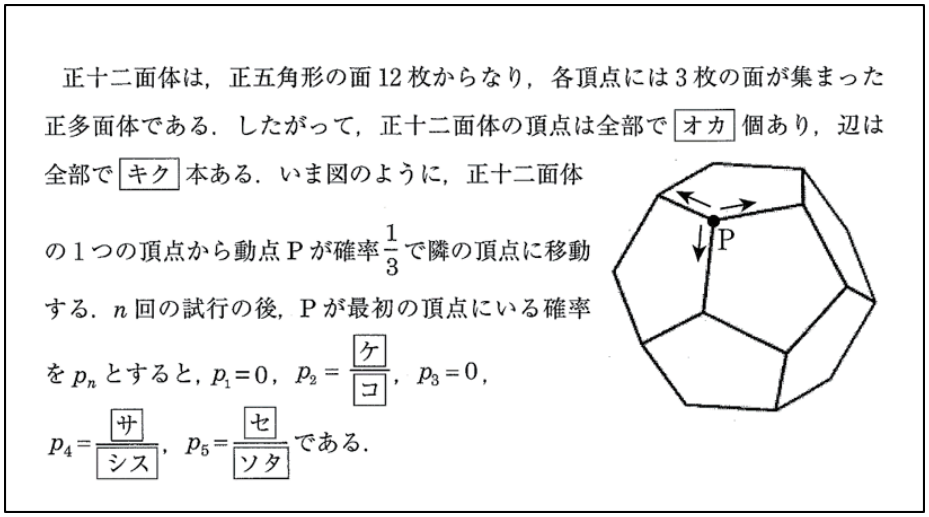
様々な大学の入試問題を見ると、その大学の教育方針が浮かび上がってくる。単純そうに見える問題が、実は奥の深い内容である場合もある。よい問題は、数年間は参考書などに掲載されることもある。あるいは、他の大学で類似の問題を出すこともあるだろう。

本稿では、東京薬科大学生命科学部の数学の入試問題の中から、興味深い問題をとりあげ、特に解答をコンピュータで検証する方法を述べる。コンピュータ実習の課題としても使える問題である。

２．正十二面体とランダム・ウォーク

図１にあげた問題は、2016年3月に東京薬科大学生命科学部の一般Ｃ方式入学試験の数学で出題された問題である。解答を付録に掲載した。

この問題は、「正十二面体」という点と、「ランダム・ウォーク」である点の２つの点で興味深い。以下に解説する。



正十二面体は，正五角形の面 12 枚からなり，各頂点には３枚の面が集まった正多面体である．したがって，正十二面体の頂点は全部で［オカ］個あり，辺は全部で［キク］本ある．いま図のように，正十二面体の１つの頂点から動点Ｐが確率$\frac{1}{3}$で隣の頂点に移動する．n 回の試行の後，Ｐが最初の頂点にいる確率を p_n とすると，$p_1=0$，$p_2=\frac{[ケ]}{[コ]}$，$p_3=0$，$p_4=\frac{[サ]}{[シス]}$，$p_5=\frac{[セ]}{[ソタ]}$ である．

図１　東京薬科大学生命科学部　入試問題[1)]

2.1 正十二面体

図１の入試問題は、「正十二面体」という正多面体の１つを扱っている。正多面体については、中学１年の数学の教科書[2)]で扱われている。古くは、プラトンの『ティマイオス』[3)]に述べられていて、古代ギリシャの頃から知られていた。さらに詳しく知りたい方には、『正多面体を解く』[4)]（一松信著）の中の「正多面体小史」が参考になる。

正十二面体を利用した商品としては、さいころと、ルービック・キューブに似たパズルがある（図２）。正十二面体のさいころは、12 枚の面に１から 12 までの数字が、反対側の面との数字の合計が 13 になるように、書かれている。また、立方体のルービック・キューブに類似した正十二面体のキロミンクス（図３(a)）やメガミンクス（図 3(b)）がアマゾン[5)]やトライボックス[6)]のサイトで販売されている。さらに細かいパーツに分けたものもある。他にも、正十二面体の五角形を曲線的に切るようなタイプもある。世界キューブ協会[7)]や、日本ルービックキューブ協会[8)]では、毎年大会が開かれていて、メガミンクスも正式種目に含まれている。

1 東京薬科大学生命科学部 生命物理科学研究室
2 東京薬科大学生命科学部 生物情報科学研究室

図２　正十二面体のサイコロ

(a)　　(b)

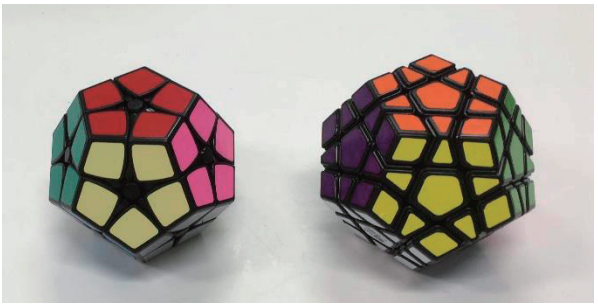


図３　正十二面体を利用した、キロミンクス(a)とメガミンクス(b)

2.2 ランダム・ウォーク

この問題は、「ランダム・ウォーク」[9]と呼ばれる研究分野と関係があり興味深い。ランダム・ウォークとは、どの方向に進むかがランダムな動きである。酔っ払いの歩き方と似ていることから、酔歩の問題とも呼ぶ。

ランダム・ウォークは「ブラウン運動」として統計力学の分野でも研究されている[10]。また近年は、株価の変動を確率的に扱うことから、ファイナンスの分野でも使われている[11]。

ランダム・ウォークの軌跡を１本のひもと考えると、高分子の形と見ることもできる。高分子とランダム・ウォークの関係については、ノーベル賞を受賞したフランスの物理学者ド・ジャンヌの著書『高分子の物理学』[12]に詳しい。特に、既に通った場所を避けるランダム・ウォークを「自己排除（self-avoiding）ランダム・ウォーク」と呼ぶことがある。本稿で扱う問題は、自己排除ではなく、どこでも等確率で移動する、普通のランダム・ウォークである。また、統計力学のランダム・ウォークは、正方格子や立方格子など無限に広がった空間で考えることが多いが、この問題では、有限個の点の上を動く。

３．コンピュータによる検証

この問題は通常の入試のように手で計算して確率を求めることができるが、コンピュータを使って確率を求めることもできる。本節では、プログラムを考える手順を述べた後、実際のプログラムについて解説する。

3.1 頂点の番号を決める

正十二面体の頂点は全部で 20 個ある。全ての場合を数え上げるプログラムを書く場合、どの頂点を通った経路なのか、区別する必要がある。そこで、頂点に番号をつけた。

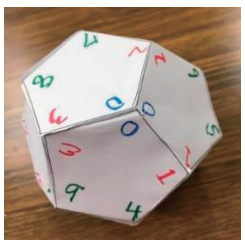


図４　頂点に番号をつけた正十二面体

番号を振るために、正十二面体の展開図[13]を利用した。白い紙で立体を作成してから、ペンで番号を記入した（図４）。

頂点の番号はどのようにつけてもよいのだが、今回は、図４の写真のように、最初の点を 0 番とし、0 番に連結している３つの頂点を 1, 2, 3 番にした。次に 1 番に連結していてまだ番号のついていない頂点２つを 4 番と 5 番にした。次は 2 番と 3 番についても、連結している点をそれ

ぞれ６番と７番、８番と９番とした。この方法で全部の頂点に番号をつけた。

```c
/* 正12面体上のランダム・ウォーク     */

#include <stdio.h>
#define NSTEP 5   /* ランダムウォークのステップ数*/
#define NPATH 243  /* 経路の場合の数   3 ^ NSTEP */
        /*   N=5: 243, N=4, 81, N=3, 9  */
#define ISTART 0  /* 出発する頂点。普通は0を入れておく  */

int connect[20][3]={      /*正20面体の連結性を示す。固定配列*/
 /* 頂点の番号はトップが0, 次の段が1,2,3（上から見て反時計周り）
    それぞれの下に、(4,5), (6,7), (8,9)
    それぞれの下に  10,11, 12, 13, 14, 15
    (15,10)の下に16, (11,12)の下に17, (13,14)の下に18.
    残り（一番下）が19. */
      1,  2,  3,                          /* 0番の接続先*/
      0,  4,  5,    0,  6,  7,    0,  8,  9,  /* 1,2,3の接続先*/
      1,  9, 10,    1,  6, 11,    2,  5, 12,  /* 4,5,6の接続先*/
      2,  8, 13,    3,  7, 14,    3,  4, 15,  /* 7,8,9の接続先*/
      4, 11, 16,    5, 10, 17,    6, 13, 17,  /* 10,11,12の接続先*/
      7, 12, 18,    8, 15, 18,    9, 14, 16,  /* 13,14,15の接続先*/
     10, 15, 19,   11, 12, 19,   13, 14, 19,  /* 16,17,18の接続先*/
     16, 17, 18};                         /* 19の接続先*/

int nter[NPATH][NSTEP]; /*３進数の各桁が0,1,2のどれか 固定配列*/

int pos[NPATH][NSTEP];/*ある経路のあるステップでの場所。0-19 */

int stat[NSTEP][20];  /* 各ステップでの頂点にいる個数の統計*/

/*   3進数を入れておく。最初に入れると後は変化なし*/
void InitTer(){
  int iter, iter1, iter2, idigit;

   /*  初期化しておく  */
  for (iter = 0; iter < NPATH; iter++){
    for (idigit = 0; idigit < NSTEP; idigit++){
       nter[iter][idigit] = 0;
    }
  }

  /* 3進数を入れる */
  for (iter = 0; iter < NPATH; iter++){
     iter2 = iter;
     idigit = NSTEP-1;
     while (iter2 >0){
       iter1 = iter2 % 3; /* ３で割った余り */
       nter[iter][idigit] = iter1;
       iter2 = iter2 / 3; /* 商を次の計算に使う*/
       idigit--;
     }
  }
}

/* 3進数を書いてみる。デバック用 */
void WriteTer(){
  int iter, idigit;

  for (iter = 0; iter < NPATH; iter++){
     printf(" iter = %d ", iter);
     for (idigit = 0; idigit < NSTEP ; idigit++){
        printf (" %d", nter[iter][idigit]);
     }
     printf(" ¥n");
  }
}

/* 途中を出力する場合 */
void WriteRoute(){
  int iroute, istep;
  for (iroute = 0; iroute < NPATH; iroute++){
      printf(" ¥n iroute = %d ", iroute);
      for (istep = 0; istep < NSTEP; istep++){
        printf(" %d", pos[iroute][istep]);
      }
  }
  printf(" ¥n");
}

/* 場合の数を出力する。*/
void WriteStat(){
  int istep, ipos1;
  double prob;
  printf("  step数    戻る確率       各頂点に来た回数  ");
  for (istep = 0; istep < NSTEP; istep++){
       printf("¥nistep= %d", istep+1);
       prob =  (double) stat[istep][0]/NPATH;
       printf("  prob= %2.4f", prob);
    for (ipos1 = 0; ipos1 < 20; ipos1++){
       printf("%3d", stat[istep][ipos1]);
    }
  }
  printf(" ¥n");
}
/*   メイン  */
int main(void){
  int pos1, pos2, ipos1, ter1, istep, iroute;

  InitTer();
  /* WriteTer(); */  /*デバッグ用*/

/* 統計量の初期化 */
  for (istep = 0; istep < NSTEP; istep++){
    for (ipos1 = 0; ipos1 < 20; ipos1++){
       stat[istep][ipos1] = 0;
    }
  }

/* 動かす */
  for (istep = 0; istep < NSTEP ; istep ++){
    for (iroute = 0; iroute < NPATH ; iroute ++){
       if (istep == 0){
         pos1 = ISTART; /*初期値*/
       }else{
         pos1 = pos[iroute][istep-1];   /* 現在の場所 */
       }
       ter1 = nter[iroute][istep];     /* 行く方向：0,1,2 */
       pos2= connect[pos1][ter1];      /*次の場所*/
       stat[istep][pos2] ++;           /*滞在の統計*/
       pos[iroute][istep]= pos2;
    }
  }

/* 途中の場所を出力*/
 /*   WriteRoute(); *//* 必要な場合のみ */

/* 場合の数を出力する。*/
  WriteStat();
}
```

図５　正十二面体上のランダム・ウォークのプログラム

3.2 頂点の連結を定義する

頂点の番号付けができたら、次にどの頂点とどの頂点が辺を作るかを知る必要がある。たとえば、頂点 0 にいるときは、次のステップで、頂点 1,2,3 に移ることができる。これを、あらかじめ配列に入れておく。図５のプログラムの 9-22 行で配列 connect を定義している。たとえば、5 番の頂点と連結している３つの頂点は、connect[5][j] (j=0, 1, 2)で得られる。

3.3 全ての場合を出すには

各頂点において、３通りの行き先がある。次のステップでも同じように３通りある。そうすると、全ての場合を記述するには、３進数を使うことができる。

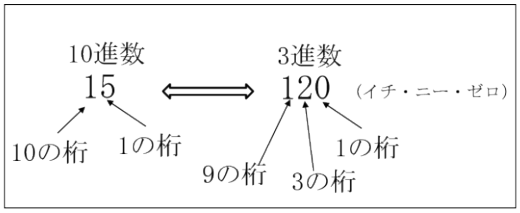


図６ ３進数の表記

３進数は0, 1, 2を用いる。10進数の3は、3進数では10（イチゼロ）と書く。たとえば、10進数の15を３進数で表すには、15を3で割ると5余り0、5を3で割ると1余り2である。３進数では120となる。逆に３進数の120を10進数で表すと、0×1+2×3+1×9=6+9=15である(図６)。

本稿の問題では、各頂点で３つのうちの１つの行先を決めるので、各ステップで、３進数を１桁決める（0,1,2の３つのうち１つを選ぶ）と見ることができる。

そうすると、たとえば４ステップあるなら、３進数が４桁なので、3の４乗で81通りになる。一般にnステップの場合は、全ての場合を考えるというのは、この3^n通りの場合を考えればよいことになる。

逆に0から3^n-1までの10進数を与えれば、経路が一意的に決まる。これをあらかじめ計算して、配列nter[iter][idigit]に入れている。変数iterは、経路番号であり、0から3^n-1までの数をとる。変数idigitは0からn-1までの数をとり、３進数の何桁目かを表す。たとえば、iter=15は先程述べたように、３進数で120であるので、nter[15][0]=0, nter[15][1]=2, nter[15][2]=1が入る。

この３進数を使って移動する例を、図７に示す。

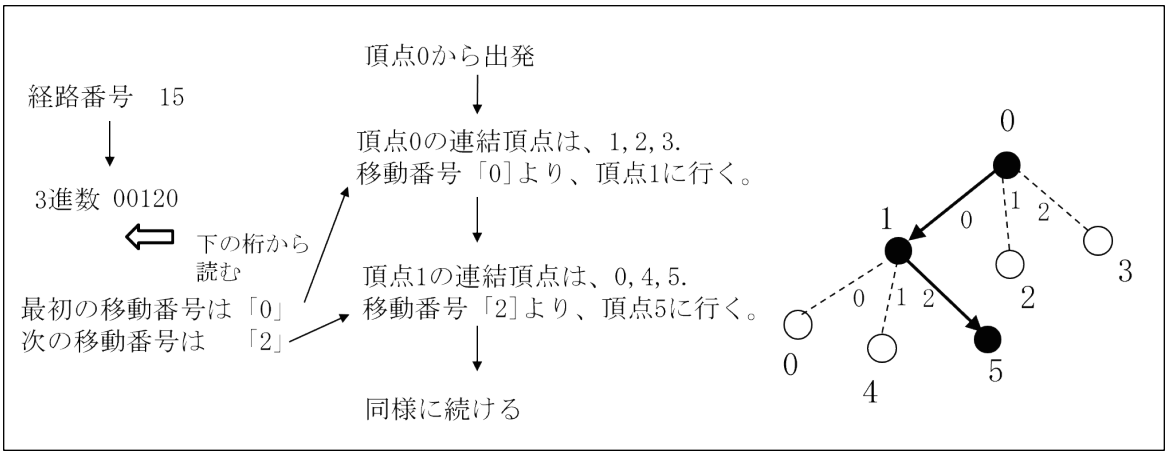


図７ ３進数を正12面体上の移動に使う

図７において、経路番号15から得られる移動番号「0」より、頂点0から頂点1に動き、移動番号「2」により、頂点5に動く。

3.4 メインループの計算

図８にメインループの計算の概略を示す。一番外側のループは、変数istepが変化する部分である。全ステップ数には、この入試問題に合わせて、NSTEP=5が入っている。

外側から２番めのループは、変数irouteが変化する部分である。全経路数NPATHには、問題に合わせて、3の５乗である243が入っている。

変数 pos1 には現在の位置（頂点番号）が入っている。初期状態 istep=0 の場合は、pos1=0 を代入する。初期状態でない場合（istep ＞ 0）の場合は、１つ前のステップにおける位置、pos[iroute][istep-1]の中身を pos1 に代入する。

次に行く場所を調べるために、配列 nter[iroute][istep]を変数 ter1 に入れる。配列 nter には、既に述べたように、経路番号 iroute のステップ istep における方向 (0,1,2 のどれか) が入っている。実際どの頂点に動けばよいかは、現在の場所 pos1 に依存する。行先の頂点番号 connect[pos1][ter1]を変数 pos2 に代入する。ステップ istep で頂点 pos2 に来たという統計をとるために、stat[istep][pos2]に 1 を加える。また pos[iroute][istep]に現在の場所 pos2 を代入する。

ある経路に関する操作が終わったら、次の経路について同様のことを行う。あるステップに関して全ての経路を計算したら、次のステップの操作を行う。

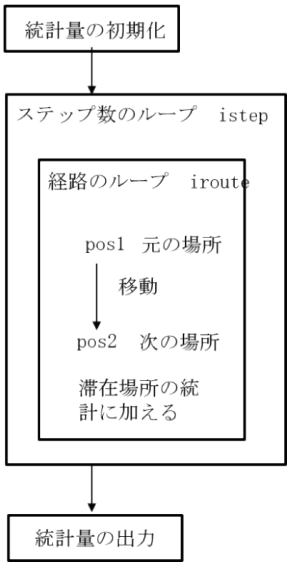


図８　メインループの概略

3.5 出力例

上記のプログラムは、 ノートパソコン MacBook Air のターミナル上で、 gcc を用いてコンパイルし実行している。 出力例を図９に示す。この例では、ステップ数５の全ての経路について求めている。

最終状態のステップ５の場合は、図９の出力によると、6 個の経路が元の場所に戻っている。全体の経路数は、3 の 5 乗で 243 なので、6/243=2/81 が元の場所に戻る確率である。

これらの 243 個の経路が、ステップ４でどのような場所にあったかというと、図９の出力によると 45 個の経路が元の場所に戻っている。全体の経路数は 243 なので、45/243=5/27 が元の場所に戻る確率になる。あるいは、全ステップ数が 4 であるとして、プログラムを動かしても、求めることができる。

```
  step数    戻る確率       各頂点に来た回数
istep= 1  prob= 0.0000  0 81 81 81  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
istep= 2  prob= 0.3333 81  0  0  0 27 27 27 27 27 27  0  0  0  0  0  0  0  0  0  0
istep= 3  prob= 0.0000  0 45 45 45  9  9  9  9  9  9  9  9  9  9  9  9  0  0  0  0
istep= 4  prob= 0.1852 45  6  6  6 21 21 21 21 21 21  6  6  6  6  6  6  6  6  6  0
istep= 5  prob= 0.0247  6 29 29 29 11 11 11 11 11 11 11 11 11 11 11 11  4  4  4  6
```

図９　出力例

3.6 プログラムの拡張性

元の問題が等確率で３つの頂点を選ぶものであった。行先の場所の個数に応じて、n 進数を使い分ければ同様にプログラムを作れる。

このプログラムは、１個の３進数が１個の経路に対応している。コンピュータが扱える整数には上限があるので、今回の問題のような５ステップなら問題ないが、ステップ数が多い場合は、3 進数を２個用意するなど、工夫が必要になる。ステップ数が非常に多い場合は、全部の経路を求めるのではなく、モンテカルロシミュレーションを用いるなど、一部の経路を使う方法がある。

４．おわりに

本稿では、入試問題を題材に、問題の背景を説明し、コンピュータによる検証プログラムを紹介した。

プログラミングの練習問題としても使える。本稿を書く途中で、いろいろな方向から入試問題を検討することができて楽しかった。

参考文献

1) 東京薬科大学ホームページ　平成28年度（2016年度）C方式入学試験、数学
https://www.toyaku.ac.jp/admissions/info/info-questions
2) 中学校　数学１、文部科学省検定済教科書、数研出版
3) ティマイオス クリティアス、プラトン著、岸見一郎訳、2015年、白澤社
4) 正多面体を解く、一松信著、2002年、東海大学出版会
5) アマゾン　http://www.amazon.co.jp/
6) トライボックス　http://store.tribox.com/
7) 世界キューブ協会　https://www.worldcubeassociation.org/
8) 日本ルービックキューブ協会　http://jrca.cc/
9) ランダム・ウォーク、乱れに潜む不思議な現象、津野義道著、2002年、牧野書店
10) 統計物理学（新装版　現在物理学の基礎５）、戸田盛和他、2011年、岩波書店
11) ランダムウォークと確率解析 ギャンブルから数理ファイナンスへ　藤田岳彦、2008年、日本評論社
12) 高分子の物理学、ド・ジャンヌ著、久保亮五監訳、1984年、吉岡書店
13) 数学教材の部屋　http://sintakenoko.la.coocan.jp/Note/note_11.pdf

付録　図１の問題の解答

・頂点の数：　１つの五角形が5個の頂点を持ち、12個の面があり、１つの頂点に面が３つ集まっているので、$\frac{5\times12}{3}=20$個である。

・辺の数：１つの五角形に辺が５個あり、１つの辺を２個の五角形が共有することから、 辺の数は$\frac{5\times12}{2}=30$個である。

・２回で戻ってくる場合：１回目に行った場所から戻る確率は1/3。p_2=1/3になる。

・４回で戻ってくる場合：2回目でいったん最初の点に戻って，新たに2ステップの行程を繰り返すのが3×3=9通り．2回の試行後に最初の点にいない場合は，残りの試行でいま来た道を戻るしかないので，3×2=6通り．合計15通り。これを全体の経路の数の3^4=81で割ると、p_4=15/81=5/27になる。

・５回で戻ってくる場合：最初の点が属する３面のうちの１つの正五角形を，時計回りまたは反時計回りに１周するしかないので，　3×2=6通り．全体の経路の数の 3^5で割ると、p_5=6/243=2/81になる。