モノポリにおける最も稼げる都市について数値計算

東京大学理学系研究科天文学専攻 M1 チン タン

2010/9/27

第 I 部
カードの効果なし

1　目的

モノポリというボードゲームは 20 世紀初頭にアメリカ合衆国で生まれた。プレイヤーはサイコロをふたつ振り、出た目の数に応じて 40 個のマスを正方形の周囲を回るようにして回り、その途中途中で都市を買い、そのマスに止まった他のプレイヤーから高額なレンタル料をとるゲームである。このゲームの目的は自分以外のプレイヤーを破産に追い込むことである。そこで重要なことが、限りある資金を使って、どの都市を買うか、もしくは交渉をし、手に入れるかである。都市によって、買うために必要な費用も違えば、レンタル料も異なってくる。そこで、どの都市が一番儲かる都市なのかを計算することはモノポリをプレイする上では非常に大事なことである。またモノポリ自体は政治や教育上の試みとして、製作された原型をもつ。この原型からもわかるように私はモノポリをプレイすることで、社会で生きていくために必要な、投資力、判断力、さらには交渉力が身につくと確信しております。そこで今回は非常に簡単なモデルを提示し、そのモデル内でどの都市が最も儲かるかを計算した。

2　モデル

まず今回計算したモデルを示す。

モノポリには全部で 40 個のマスがあり、その内 22 マスが都市と呼ばれるマスである。プレイヤー一人が「GO」マスからサイコロをふたつ振り、40 個のマスを順々に巡り、41 マス目に「GO」に戻ってくるとする。これを何周もする。（実際は数万周させた。）ただし、毎回「GO」から始まるのではなく、「GO」マスを飛び越えての進みも可能である。このときの各都市のマスに一体何回止まったかを計算する。

各都市において、相手プレイヤーから取れるレンタル料が、その都市の購入金額に比例すると仮定した。この仮定により、各都市に止まった回数に都市の購入金額を乗じた値が、すなわちその都市の「期待値」である。（ここでは各都市間の比較をしたいだけなのでこの「期待値」が分かれば十分である。）

ただしここで注意したいのは、「GO TO JAIL」というマスである。このマスに止まると「JAIL」マスに戻ってしまい。保釈金を払うか、サイコロを振りゾロ目がでない限り最高 2 回足止めされる。今回の計算では、簡単のために、保釈金は払わないとして、「GO TO JAIL」に止まってしまった場合には、即時に「JAIL」に行き、サイコロを振り、ゾロ目が出れば、その分だけ先に進む、出なければ最高 2 回まで足止めされるとした。この仮定をしたことで、「GO TO JAIL」マス以降のマスは確率的に止まりにくいと考えられる。

3　プログラムソース

以下にソースを載せる。

プログラムソースここから＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

```
c       2010/09/26 Chen Dan
c       モノポリのどのマスに一番コマが止まるかを計算する
c       さらにそのコマで支払うお金も計算する
```

```
program monopoly
  implicit none
  real ran, ran1, ran2
  real d1, d2, d, s, m
  integer i(3), seed, j, k, r
  integer n(40), mon(40), seki

  s=0.0
  r = 4
  call itime(i)
  seed = i(3) * 131
  ran= rand(seed)
  do j=1, 40, 1
     n(j) = 0
  end do
  open (1, file="out.data", status='replace')
  open (3, file="ran.data", status='replace')
  do j=0, 100000, 1
     !ダイスを振る
     ran = ran *1000.0 + 43.0 * real(j)**0.5
     ran = rand(int(ran))
     ran = ran *1000.0
     ran1 = ran
     d1 = mod(int(ran), 6) + 1 !一個目のダイス決定
     ran = ran + 37.0 * real(j)**0.5
     ran = rand(int(ran))
     ran = ran *1000.0
     ran2 = ran
     d2 = mod(int(ran), 6) + 1 !二個目のダイス決定
     d = d1 + d2
     !ダイスを振った (d1, d2)。d はその合計、つまり進む数を決定

     if (r>=2) then
        s = s + d
        m = mod(s, 40) + 1.0
        n(int(m)) = n(int(m)) + 1
        if (mod(m, 30)==0 ) then
           s = s + 20.0
           r = 0
        end if
        !牢屋に入ってなければコマがどこに進んだかを計算。ただし変数型は real

     else if (d1 == d2) then !牢屋入っていれば、ゾロ目かどうかを判断
        s = s + d !ゾロ目なら進む。
        r = 2 !牢屋からもでる。
        m = mod(s, 40) + 1.0
        n(int(m)) = n(int(m)) + 1
        if (mod(m, 30)==0 ) then
```

```
            s = s + 20.0
            r = 0
         end if
      else !牢屋に入っていてかつゾロ目でなければ、進まずに牢屋に入った数を数える。
         r = r + 1
      end if

      write(1, "(2i10, 5f20.10)") j, r, d, s, m
      write(3, "(i10, 2f20.10)") j, ran1, ran2
   end do
   close(3)
   close(1)

   open(10, file="mon.data", status='old')
   do j=1, 40, 1
      read(10,*) mon(j)
   end do
   close(10)

   open(2, file="n.data", status='replace')
   do j = 1, 40, 1
      seki = n(j) * mon(j)
      write(2, "(4i10)") j, n(j), mon(j), seki
   end do
   close(2)

end program monopoly
```

プログラムソースここまで＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

4　プログラムのポイント

このプログラムは変数 s をサイコロを振った分だけ進め、これを 40 で割って 1 足した数をそのときに止まったマスであるとした。途中の if 分では「JAIL」に入っているかどうかの判断をし、結果に応じて処理（「JAIL」に入ってないと判断されれば先に進む処理、そうでなければ進まないで「JAIL」に入ったときの回数計算処理、もしくはゾロ目が出ていれば「JAIL」から出て先に進む処理）を行う。

本プログラムで最も苦労した点は乱数の発生である。関数 rand() に時刻の秒数を入れて、乱数を発生させているが、全プログラムの計算時間は数秒しかないので、これだけでは同じ乱数が何度も発生してしまう。そこで、初回に発生させた乱数を次の乱数の seed にした。だが、それだけでは、実は乱数に周期的な波が出てきてしまった。そこで次の乱数の seed にするときに do 文を回している変数 j の乗数 (0.5 乗) を seed の中にいれた。こうすることで周期的な波を消した。ここで 0.5 乗としたのには理由がある。それは j は 0 から 100000 まで回すために、例えば 2 乗では最高で 10000000000 という数が出てきてしまい、どこかの変数がサチってしまう。そこで j の乗数が最高でもあまり上がらないように、0.5 乗とした。

5　結果１

まずは乱数が少なくとも見ためでたしかに乱数っぽくなっているかどうかを確認する。ここで χ^2 検定などを使用してその乱数っぷりを示してあげるのが筋であると思うが、眠いのでそれは割愛した。発生させたサイコロの目の直前の乱数を以下に示す。

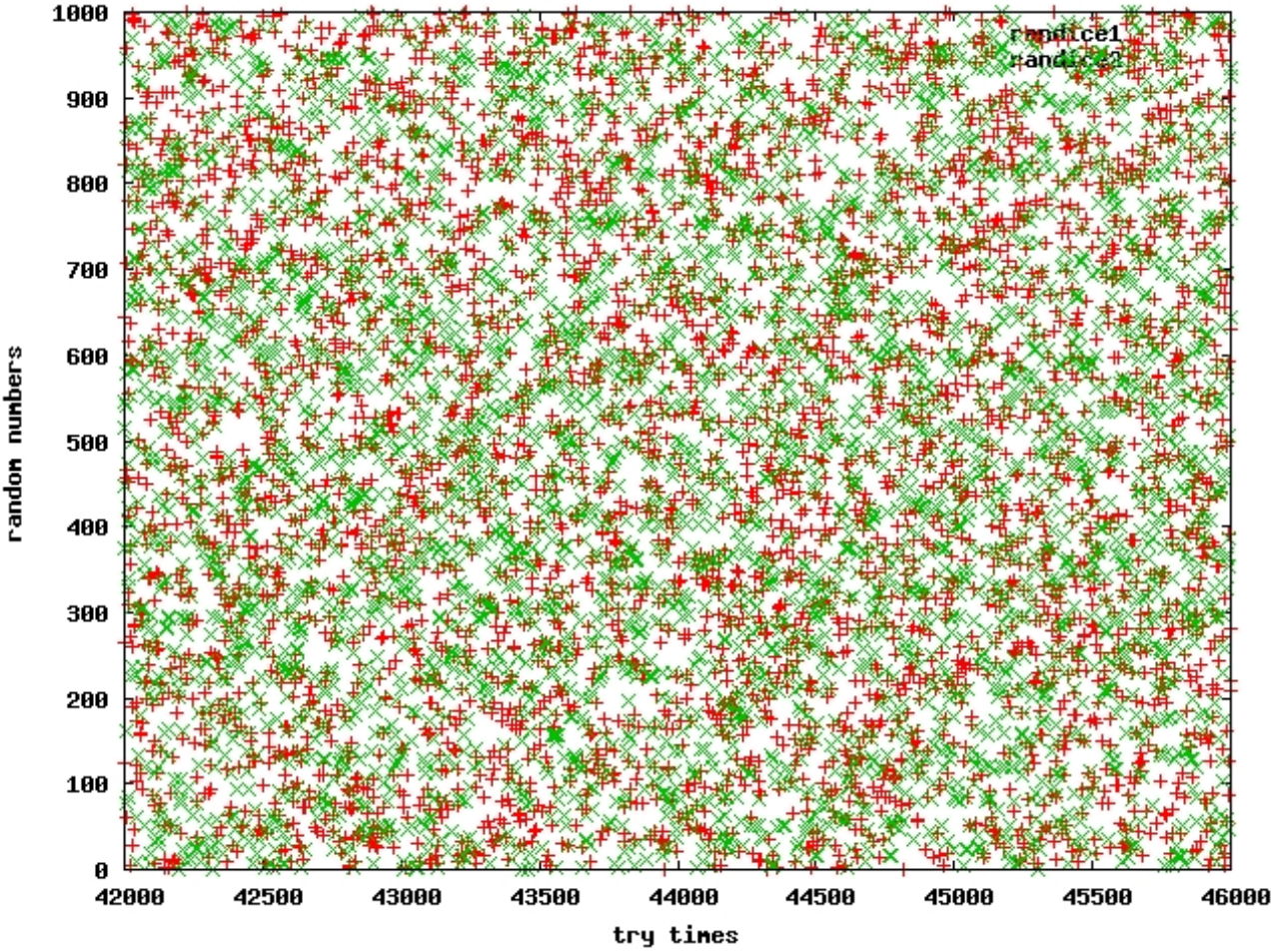

図 1: サイコロ 1,2 を決めるための乱数 1,2。ぱっと見、特に気になる周期的な変化もなければ、一様に分布しているように見える。

サイコロを振るのってこんなに苦労するんだなぁって思った。

6 結果２

次に本結果を示す。その中でまずはどのマスにどれくらい止まったかの計算結果を示す。なお乱数による効果の変化を見るために結果は二つ示す。

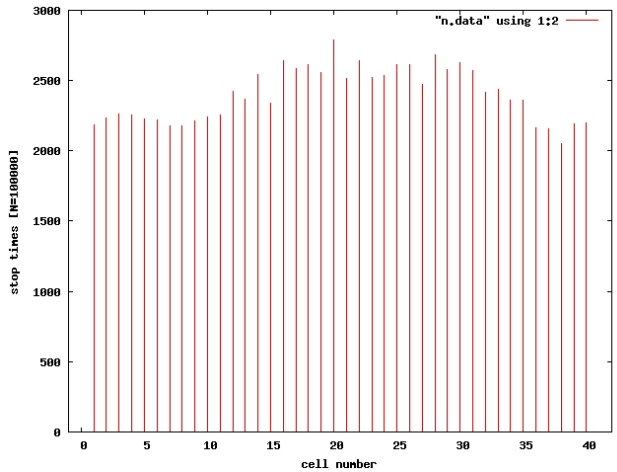

図 2: 各マスに何回止まったかを表すグラフ１。

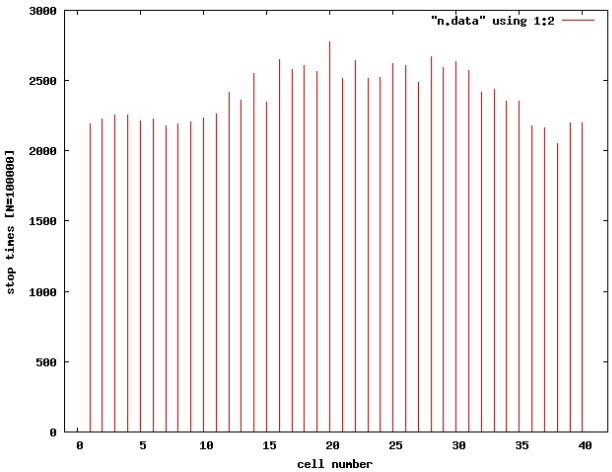

図 3: 各マスに何回止まったかを表すグラフ２。

上記のグラフで横軸は各マスに振り分けられた番号。1 番は「GO」、11 番は「JAIL」、21 番は「FREE PARKING」、31 番は「GO TO JAIL」である。サイコロを振った数、つまり母数は 100000 回である。

次に「期待値」を示す前に、各都市を購入するための金額をグラフにした。この値を上の値に乗じて「期待値」を算出した。

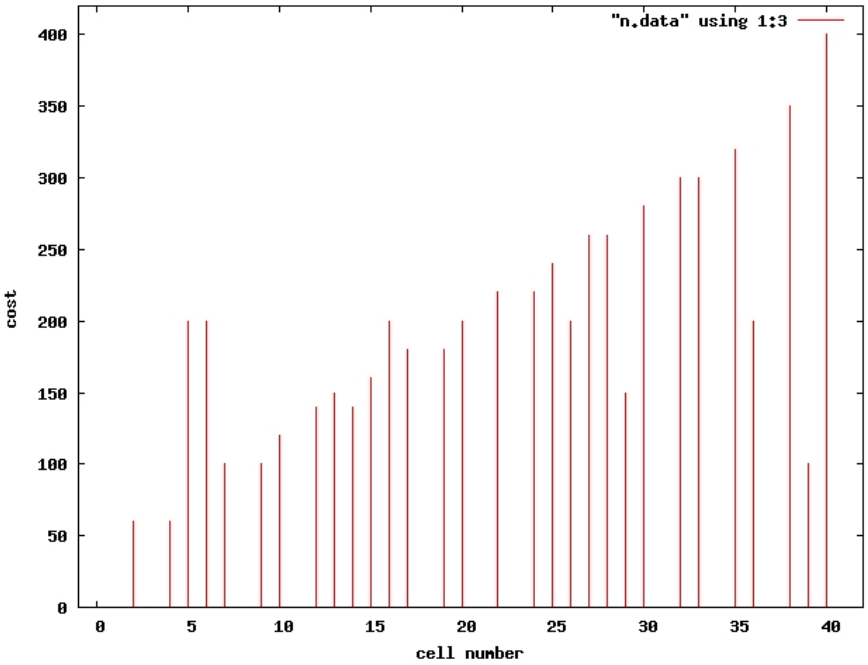

図 4: 各都市を購入するための金額。都市以外の場合には、購入金額、もしくは罰金金額、もしくは 0 とした。

最後に「期待値」を示す。これがそれぞれの都市の価値だと考えてよい。

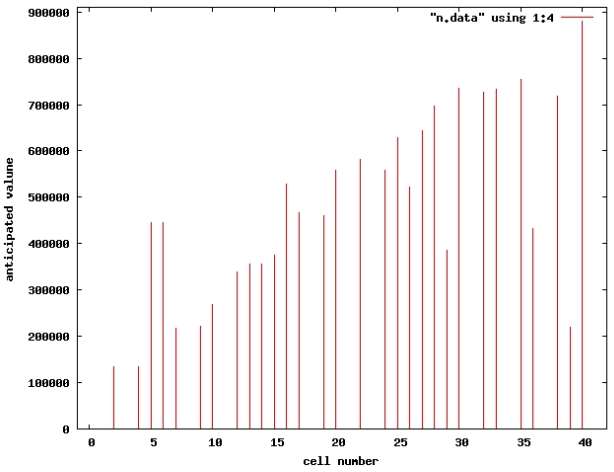

図 5: 各都市の期待値１。

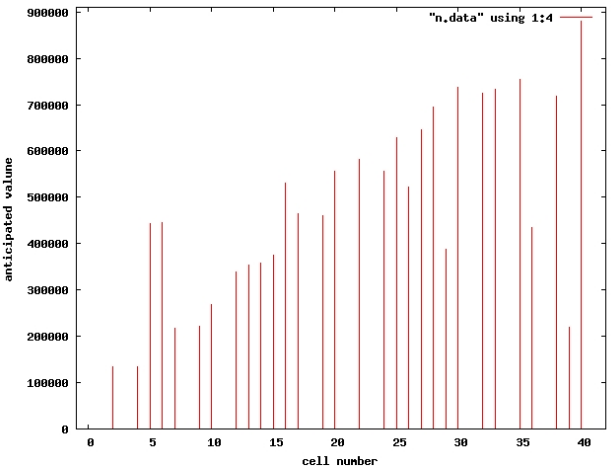

図 6: 各都市の期待値２。

上記のグラフで縦軸の値そのものに特に意味はなく、その相対的な差に注目したい。横軸は各マスに振り分けられた番号。1 番は「GO」、11 番は「JAIL」、21 番は「FREE PARKING」、31 番は「GO TO JAIL」である。

二回実行した結果を見比べて、乱数の不定性による結果の依存性は低いと考えられる。そこで以下に「期待値」の結果をもっと大きく示す。今度は都市以外は全て 0 にした。

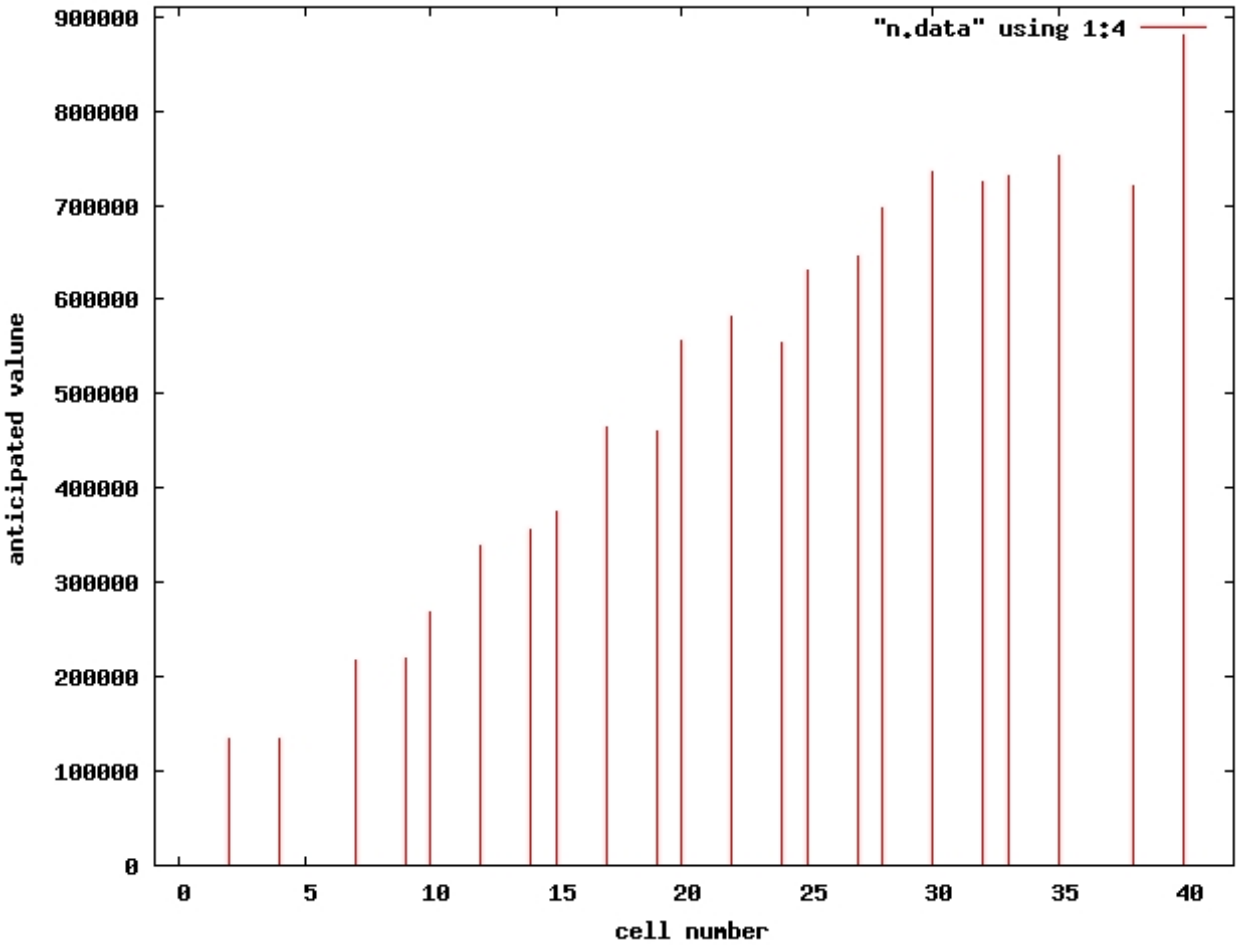

図 7: 各都市の「期待値」

7　考察

まずは各マスの止まりやすさについて。結果から見ると、cell number = 20 の「NEW YORK AVENUE:ニューヨーク通り」と cell number = 28 の「VENTNOR AVENUE:ベントノール通り」を中心に二つ、もしくは三つの山になっていることがわかる。さらに cell number = 31 の「GO TO JAIL」を堺にしてそれ以降、cell number = 11 の「JAIL」までは比較的低い。これはまさに牢屋の効果である。

最終結果から、まず考えられるのは、右に行けば、行くほどに期待値が上がっていることである。つまり基本的には購入が高額な都市ほどに儲けも大きいということである。ここからゲーム自体の設定として、各マスにおおよそ同じ確率で止まるようになっていることが言える。

各都市の中でも特に目を引くのは最高額な都市 cell number = 40 の「BOARDWALK:ボードウォーク」である。他の都市から飛び抜けて期待値が高い。次に注目したいのはやはり、cell number = 32, 33, 35 の「PENNSYLVANIA:ペンシルバニア通り、NORTH CAROLINA AVENUE:ノースキャロライナ通り、PACIFIC AVENUE:パシフィック通り」の地域である。ここの三つの都市はともに、期待値が高く、ここを独占できれば、トップに立つ可能性が高くなるだろう。

逆に目を引くのが cell number = 2, 4 の「MEDITER-RANEAN AVENUE:地中海通り、BALTIC AVENUE:バルティック通り」である。ここの地域は非常に期待値が低い。

以上では、各都市を購入するのに必要な経費を考慮にはいれておらず、上記にある考えでゲームを進めた場合に勝てる確率は増えるとは考えられるが、確実というわけには行かない。なぜなら序盤と終盤での各都市の発展速度や、所持金の多少、交渉の成功失敗などは考慮していないからである。それでもどのマスが止まりやすく、儲けやすいかを計算できた。

第 II 部
カードの効果有り

8　目的

今度はカード (チャンス、共同基金) の効果を入れて、計算してみた。なおカードはそれぞれ 16 枚ずつある。また前回のプログラム中では牢屋で足踏しているときに牢屋に止まった数には入れていなかったが、今回は足踏も、入った数として計算する。つまり牢屋で 2 回足踏をすれば、その分だけ牢屋に止まったと数えるのである。

9　モデル

今回のモデルを示す。

チャンスカード、共同基金カードの二種類のカードを予め適当に並べ、カードのマスに止まる度に次のカードを見て、その効果を反映する。ただし今回反映したカードの効果は移動カードのみである。つまり牢屋へ飛ぶとか、3 マス戻るとか、その類。

ルール上使用したカードは即座にカードの山札の一番下に潜り込ませる。そのため、今回初期のカードの順番は適当にしたが、何周もすれば、どのカードをどこのマスで引くかはランダムになるはずで、初期のカード順はほとんど影響しないはずである。

10　プログラムソース

以下にソースを載せる。

プログラムソースここから＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

```
c     2010/09/26 Chen Dan
c     モノポリのどのマスに一番コマが止まるかを計算する
c     さらにそのコマで支払うお金も計算する

      program monopoly
        implicit none
        real ran, ran1, ran2
        real d1, d2, d, s, m, sold
        integer i(3), seed, j, k, r
        integer n(40), mon(40), seki
        real card, kyoudou, chance
        integer c1, c2

        c1 = 1
        c2 = 1
        s=0.0
        r = 4
        call itime(i)
        seed = i(3) * 131
        ran= rand(seed)
        do j=1, 40, 1
           n(j) = 0
        end do
        open (1, file="out.data", status='replace')
```

```
open (3, file="ran.data", status='replace')
do j=0, 100000, 1
   !ダイスを振る
   ran = ran *1000.0 + 43.0 * real(j)**0.5
   ran = rand(int(ran))
   ran = ran *1000.0
   ran1 = ran
   d1 = mod(int(ran), 6) + 1 !一個目のダイス決定
   ran = ran + 37.0 * real(j)**0.5
   ran = rand(int(ran))
   ran = ran *1000.0
   ran2 = ran
   d2 = mod(int(ran), 6) + 1 !二個目のダイス決定
   d = d1 + d2
   !ダイスを振った (d1, d2)。d はその合計、つまり進む数を決定

   if (r>=2) then
      s = s + d
      m = mod(s, 40) + 1.0
      n(int(m)) = n(int(m)) + 1
      if (mod(m, 31)==0 ) then !m が「31:GO TO JAIL」を示してれば「11:JAIL」にいく。
         s = s + 20.0
         r = 0
      end if
      !牢屋に入ってなければコマがどこに進んだかを計算。ただし変数型は real
      sold = s
      s = card(s, m, c1, c2)
      if (sold/=s) then
         m = mod(s, 40) + 1.0
         if (mod(m, 11)==0) then
            r = 0
         else
            n(int(m)) = n(int(m)) + 1
         end if
      end if
    else if (d1 == d2) then !牢屋入っていれば、ゾロ目かどうかを判断
      s = s + d !ゾロ目なら進む。
      r = 2 !牢屋からもでる。
      m = mod(s, 40) + 1.0
      n(int(m)) = n(int(m)) + 1
      if (mod(m, 30)==0 ) then
         s = s + 20.0
         r = 0
      end if
      sold = s
      s = card(s, m, c1, c2)
      if (sold/=s) then
         m = mod(s, 40) + 1.0
         if (mod(m, 11)==0) then
```

```
            r = 0
          else
            n(int(m)) = n(int(m)) + 1
          end if
        end if
      else        !牢屋に入っていてかつゾロ目でなければ、進まずに牢屋に入った数を数える。
        r = r + 1
        m = mod(s, 40) + 1.0
        n(int(m)) = n(int(m)) + 1
      end if

      write(1, "(2i10, 3f20.10, 2i10)") j, r, d, s, m, c1, c2
      write(3, "(i10, 2f20.10)") j, ran1, ran2
  end do
  close(3)
  close(1)

  open(10, file="mon2.data", status='old')
  do j=1, 40, 1
     read(10,*) mon(j)
  end do
  close(10)

  open(2, file="n.data", status='replace')
  do j = 1, 40, 1
     seki = n(j) * mon(j)
     write(2, "(4i10)") j, n(j), mon(j), seki
  end do
  close(2)

end program monopoly




real function card(s, m, c1, c2)
  implicit none
  real s, m, ds
  integer c1, c2
  real kyoudou, chance
  if ((m==3).or.(m==18).or.(m==34)) then !共同基金
     ds = kyoudou(m, c1)
  else if ((m==8).or.(m==23).or.(m==37)) then !チャンス
     ds = chance(m, c2)
  else !どちらでもない
     ds = 0.0
  end if
  card = s + ds
end function card
```

```
real function kyoudou(m, c1)
  implicit none
  real m
  integer c1

  if (c1==1) then !刑務所に行く
     if (m==3) then
        kyoudou = 8.0
     else if (m==18) then
        kyoudou = 33
     else if (m==34) then
        kyoudou = 17
     end if
  else if (c1==2) then !GO に行く
     if (m==3) then
        kyoudou = 38.0
     else if (m==18) then
        kyoudou = 23.0
     else if (m==34) then
        kyoudou = 7.0
     end if
  else !なにもしない
     kyoudou = 0.0
  end if

  if (c1 <= 15) then !カードを回す
     c1 = c1 + 1
  else
     c1 = 1
  end if
end function kyoudou




real function chance(m, c2)
  implicit none
  real m
  integer c2

  if (c2==1) then !3 マス戻る
     chance = 38.0
  else if (c2==2) then !セントチャールズプレースへ進む
     if (m==8) then
        chance = 4.0
```

```
   else if (m==23) then
      chance = 29.0
   else if (m==37) then
      chance = 15.0
   end if
else if (c2==3) then !ボードウォークへ進む
   if (m==8) then
      chance = 32.0
   else if (m==23) then
      chance = 17.0
   else if (m==37) then
      chance = 3.0
   end if
else if (c2==4) then !刑務所へ進む
   if (m==8) then
      chance = 3.0
   else if (m==23) then
      chance = 28.0
   else if (m==37) then
      chance = 14.0
   end if
else if (c2==5) then !リーディング鉄道へ進む
   if (m==8) then
      chance = 39.0
   else if (m==23) then
      chance = 23.0
   else if (m==37) then
      chance = 9.0
   end if
else if (c2==6) then !イリノイへ進む
   if (m==8) then
      chance = 17.0
   else if (m==23) then
      chance = 2.0
   else if (m==37) then
      chance = 28.0
   end if
else if (c2==7) then !GO へ進む
   if (m==8) then
      chance = 33.0
   else if (m==23) then
      chance = 18.0
   else if (m==37) then
      chance = 4.0
   end if
else if (c2==8) then !次の鉄道会社に進む
   if (m==8) then
      chance = 8.0
   else if (m==23) then
```

```
            chance = 3.0
         else if (m==37) then
            chance = 9.0
         end if
      else if (c2==9) then !次の水道会社か電力会社に進む
         if (m==8) then
            chance = 5.0
         else if (m==23) then
            chance = 6.0
         else if (m==37) then
            chance = 16.0
         end if
      else if (c2==10) then !次の鉄道会社に進む
         if (m==8) then
            chance = 8.0
         else if (m==23) then
            chance = 3.0
         else if (m==37) then
            chance = 9.0
         end if
      else
         chance = 0.0
      end if

      if (c2 <= 15) then !カードを回す
         c2 = c2 + 1
      else
         c2 = 1
      end if
   end function chance
```

プログラムソースここまで＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

11　結果１

まずはカードがしっかり回っていることを示すために、プログラムを回し、カード番号を確かめる。プログラム中、カードには c1, c2 という変数をふっていて、この変数が 1 から始まり、一回カードを引くと 2 に変わり、次第に増えて 16 まで来たら 1 に戻るようにしてある。以下が結果である。

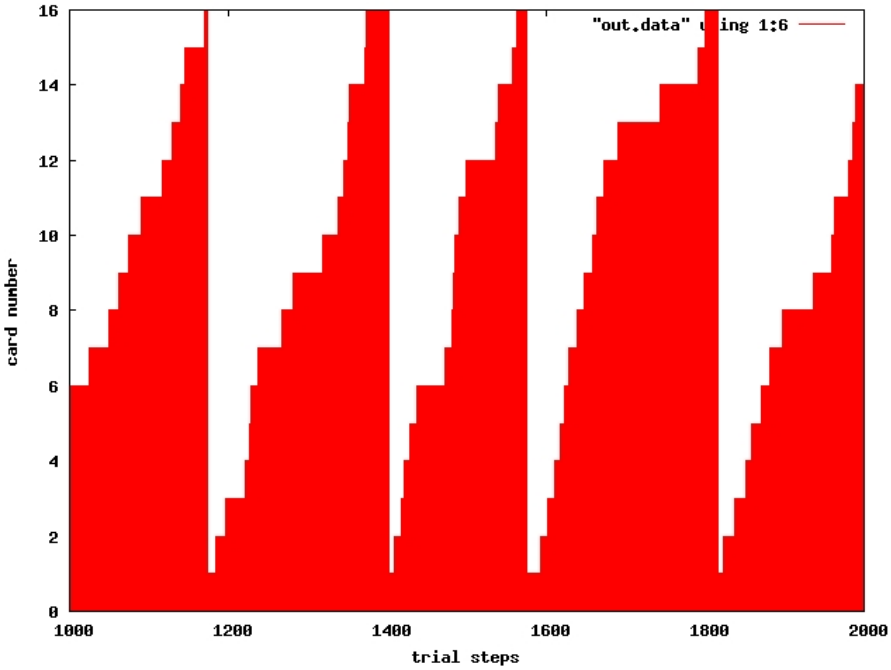

図 8: カードを示す変数の変化。横軸はサイコロを振った回数目であり、縦軸はカード変数 c1 の値である。

グラフからわかるように、おおよそ 200 回サイコロを振れば、使用したカードが一周 (つまり 16 枚使用) する。この結果からカードはしっかり回っていることがわかる。

12 結果２

次にまた各都市を買うのに必要な費用をグラフにした。

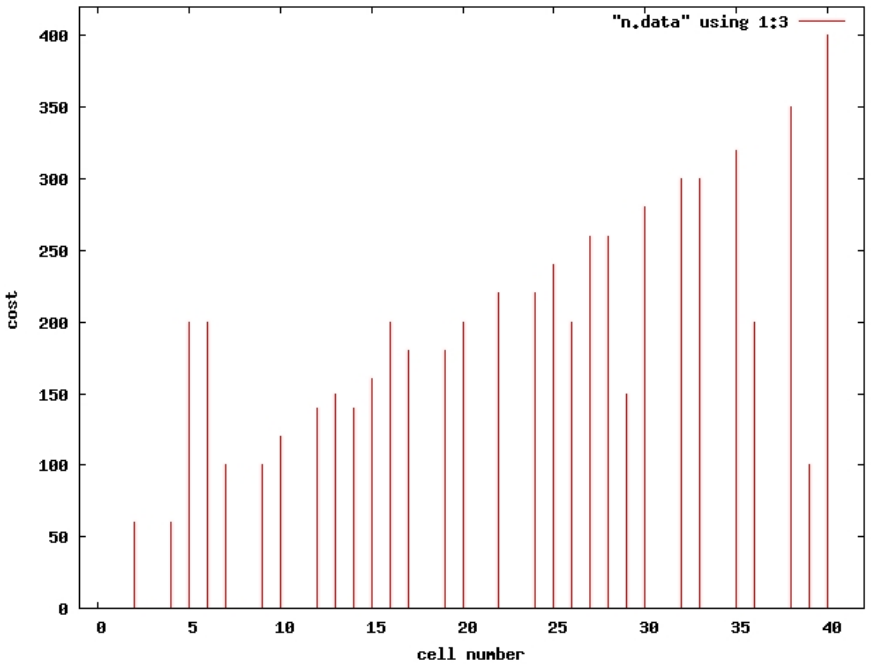

図 9: 各都市を購入するための金額。都市以外の場合には、購入金額、もしくは罰金金額、もしくは 0 にした。

いよいよ以下に止まった数の計算結果を示す。

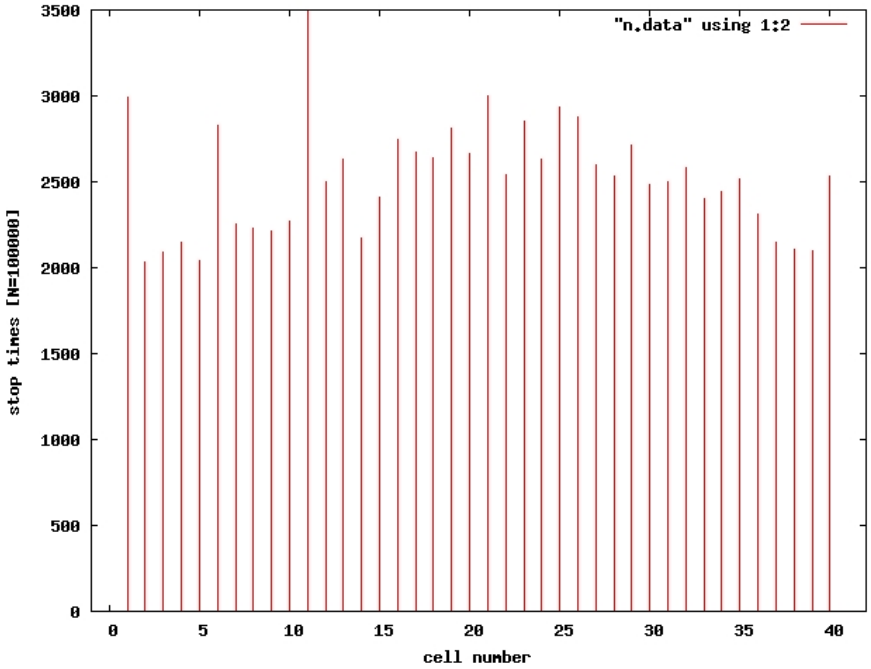

図 10: 各都市に止まった回数。サイコロを振った回数は 10 万回である。なお cell number=11 で飛んでいるのは牢屋でその値は 7300 くらいである。

次に「期待値」を示す。この値が実質的に、その都市が持つ価値、儲かるかどうかを表した値となる。

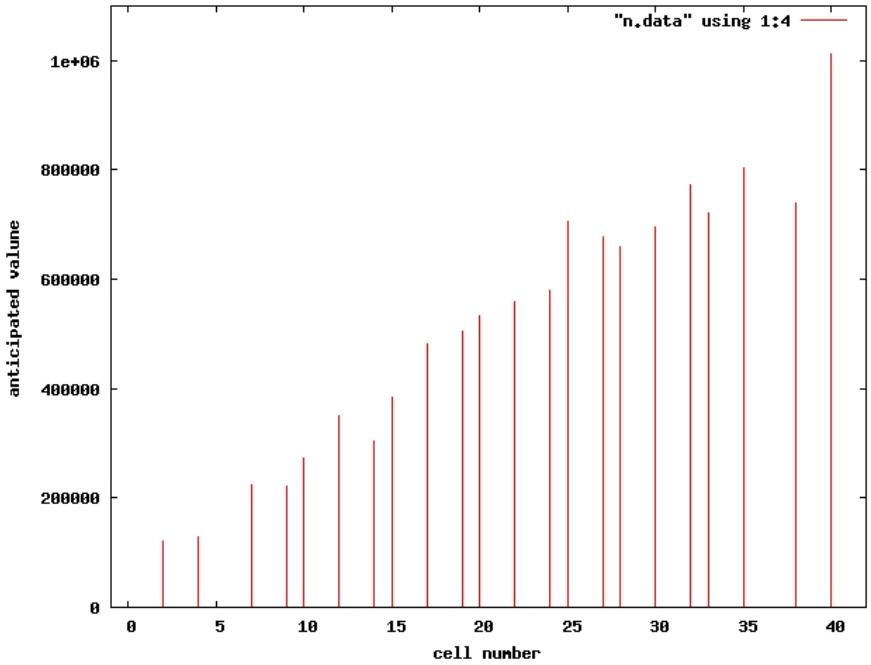

図 11: 各都市の「期待値」。止まる回数と初期費用を掛け合わせたものである。

結果を見ると、明らかに右に行く方が儲かることがわかる。これは止まる確率はどのマスも大して変わらないために、初期投資金額の影響が出ている。もっと詳しく見ていくと、「ボードウォークへ行く」というカードのおかげもあり、最後の都市ボードウォークの価値が高い。ただ、その手前の都市の価値は低く、独占できたときにはマスも多いグリーンの領域が儲かる可能性が高い。他にはやはりカードの効果が効いて、イリノイの価値が周りより高い。