きみろん Comp. 第6章

—「Q 学習」による最短距離探索問題 —

学習するコンピューター

皆さん、とうとうこの「きみろん Comp.」も最終章にやってきました。この最後の章では、コンピューターに学習させる方法を考えることにします。コンピューターの使い方を学ぶのではなく、コンピューターに学習させる方法を学ぶのです。

コンピューターというのは不思議な道具です。自分で考えたアルゴリズムでコードを書き、それが動いた！という体験を持つと、コンピューターそのものが何か「知性」を持っているように思えてきます。

1950 年代、コンピューターが「電子計算機」として大量の計算を必要とする研究所に普及し始めた頃、多くの計算機科学をやっていた人が共通に感じた感覚もそうだったと思います。そしてその予感は、この先コンピューターはヒトレベルの知能を持てるようになるという確信に変わり、人工的に知性というものを創ろうとする研究者が現れるようになりました。その人工的な知性は、人工知能（**artificial intelligence**）略して **AI**（エーアイ）と呼ばれ、その研究は今も続いています。**AI** の研究はしたがって 70 年ほどの歴史があることになります。

2012 年に **Google**（グーグル）が「**AI** が自発的に猫を認識することに成功した」と発表します。猫？と突っ込みたくなりますが、**AI** が認識する相手は猫に限らず、牛でも豚でもいいのです。動画や静止画を見て「この猫、かわいい！」ということができるのはそれまでヒトにしかできないことでした。（ヒト以外でも、例えば牛や豚も猫がかわいいと思っているのかもしれませんが、ここでは生命体ではないものが猫を認識したという意味です。）これは「**AI** が視覚を獲得した」ということを意味しているという研究者もいます。

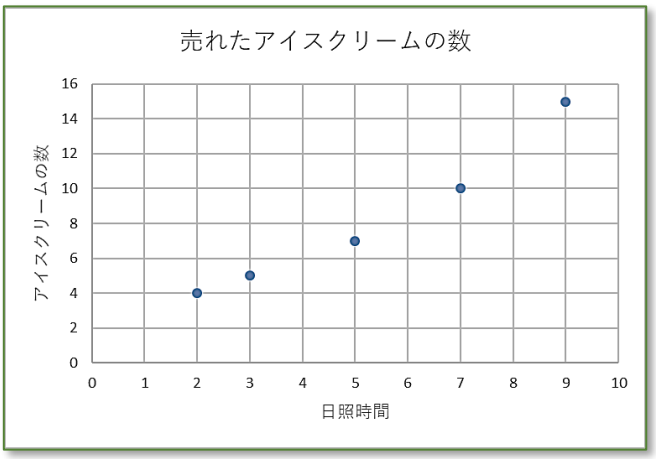
2017 年 5 月には、**Google** に買収されたイギリスのベンチャー企業 **DeepMind** 社製の **AlphaGo**（アルファ碁）という囲碁ソフトが、世界の頂点にいる棋士と目されていた中国の柯潔九段を 3 戦全勝で破るというとんでもないことがおこります。このころから、**AI** は人間を超えたといわれるようになり、これまでの人工知能を研究していた研究成果が一気に花開き始めました。その中でも有名なのが **Deep Learning （深層学習）**という学習方法です。聞いたことのある人もいるでしょう。さらには、ネットの巨大データ（**Big Data**）の中から **AI** を利用して有益な情報を見つけ出す研究者は **Data Scientist** と呼ばれるようになりました。やっと**数学 OK ∩ コンピューターOK** の人が高給取になれる時代が来たのです（笑）。

本講座では、**AI** の初歩を学ぶことにします。本テキストは、「実際にプログラムを組んで学ぶ」という哲学で書かれています。できるだけ難しい数式でごまかすようなことはしないようにと考えています。講座では「コンピューターマウスがゴールへの最短距離を見つけられるか」という課題に挑みましょう。

日照時間	アイスクリームの数
2	4
3	5
5	7
7	10
9	15

長岡科学技術大学のサイトより引用

図 1



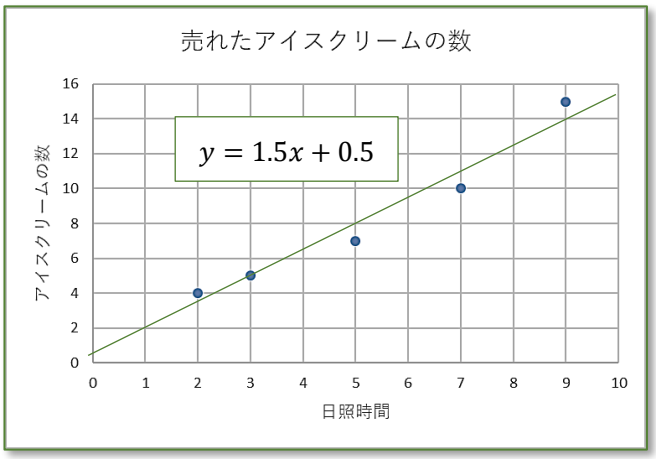


図 2

最小二乗法 回帰モデル

この講座の第二章で、鼻デカ坊やが振動してそのデータをとる話を覚えていますか。そのときばねの「**ばね定数**」 k を決定させた方法が「**最小二乗法**」でした。実はこの方法はコンピューターがデータを使って「考え」、予測を導き出す時の、最も基本的な手法になっています。

データには「**誤差**」が存在し、その「**誤差**」と理論値との差の二乗の和が最小になる値を見つけ出す方法が最小二乗法でした。この方法は「**統計学**」や人工知能の研究分野である「**機械学習分野**」では、「**回帰分析**」と呼ばれています。ゴルトンという生物学者が個体ごとの大きさや質量の差を研究しているときに名付けたそうです。生物のサイズはたいてい誤差が「**正規分布**」(きみろんテキスト参照) になり、最小二乗法を使って研究できることが分かりました。彼は、生物のサイズは世代を重ねると「平均的な値に回帰していく」と主張しました。この「**回帰**」の言葉がちょっとロマンチックに残ったのです。

さて、**図 1** を見てください。ちょっとほんとかよ！というデータです。こんな研究ができたら楽しいですね。さて、このデータをグラフにすると**図 2** のようになります。

もう、みんなだったらこれを最小二乗法を使って直線に近似しようとしますね。これを「**線形化する**」といいます。直線じゃない近似は「非線形近似」。別名「メンドクサイノデヤメトケ」という近似です。第 2 章ではこれをエクセルを使って計算させながら最小値を求めました。アイスクリームの例でやってみたところ、日照時間を *x*、売れたアイスクリームの数を y とすると**図 2** の下のような線形の近似式になります。これを人工知能の分野では「**回帰方程式**」と呼びます。

	日照時間x	曜日w	アイスクリームの数y
SAT	3	7	20
SUN	6	6	23
MON	4	5	17
TUE	7	4	20
WED	5	3	15
THU	0	2	4
FRI	7	1	13

図３

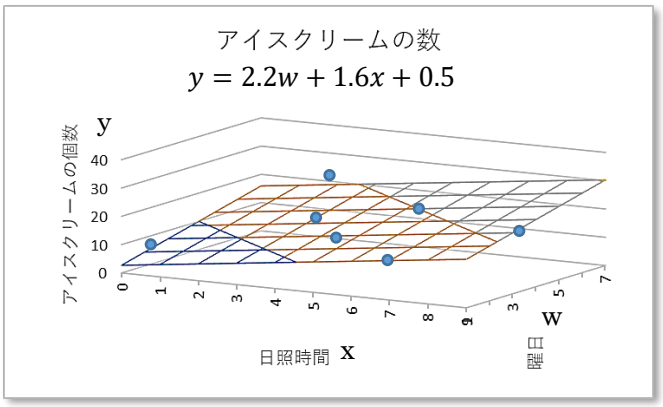


図４

重回帰モデル

データを使って学習するコンピューターは、どうも「**回帰方程式**」を作って理解し、それで予測するという思考方法を持っていそうです。

それでは図３をみてください。

A「**前ページ**のデータは、アイスクリームは日が照ってるほうが売れるってわけだけど、**図３**の表は・・・？。」

B「アイスクリームって、もちろん曜日にも関係するんじゃない？」

C「土曜日とか休日は当然売れるよね」

A「それじゃ、曇った日の土曜日は？」

BC「えっ・・・ん？！」

実はこれも最小二乗法を使った回帰分析で回帰方程式を求めることができます。詳しい方法は、コンピューターに任せることにしますが、**図４**のように、今度は直線ではなく平面になります。

この平面の上側や下側に実際のデータが分散しているのです。

この平面のような次元の高い回帰方程式を求める方法を「**重回帰分析**」といいます。

データの種類が多くなればなるほど、回帰分析によって導かれる回帰方程式は変数だらけの恐ろしいものになりそうです。計算時間も半端じゃないでしょう。もちろん直線や平面より次元の高いｎ次元曲面です。

これをコンピューターで解く方法は、これまでいろんな手法が開発されてきました。ディープラーニングもその一つといっていいでしょう。ディープラーニングは、演算の仕組みそのものを動物の脳の仕組みをモデル化した「ユニット」（人工ニューロン）と呼ぶ演算子によって行います。この構造全体をニューラルネットワーク（神経細胞ネットワーク）と呼んでいます。

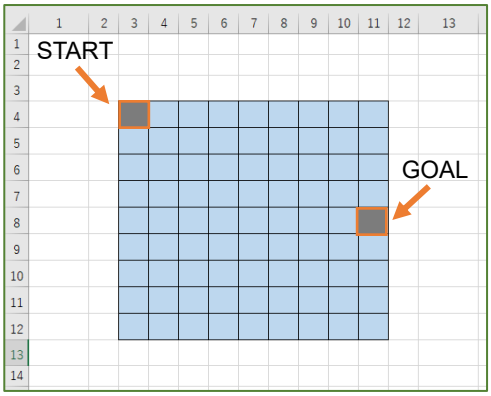


図 Q-1

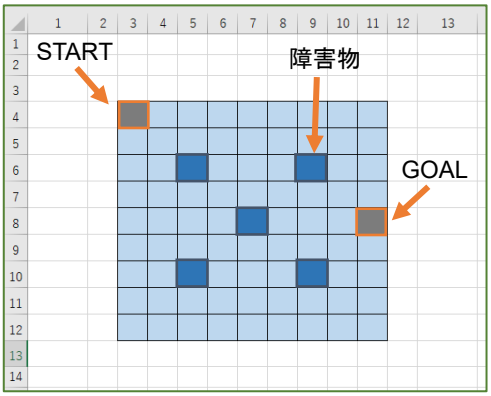


図 Q-2

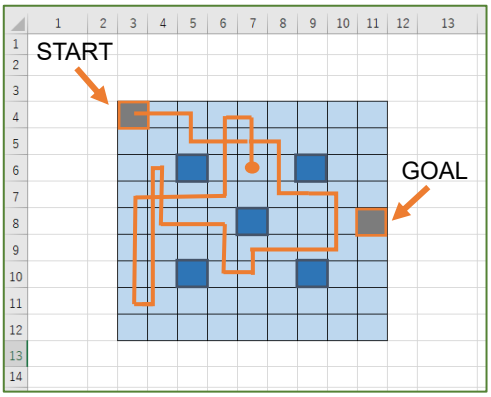


図 Q-3

経路探索問題

今回こんなプログラムを作ってみようと思います。図 Q-1 を見てください。9×9 のマス目を Sheet1 に作ります。これが今回の舞台です。例えば左上のマス目から隣のマス目にサイコロを振って移動していけるとします。サイコロを振って「右」が出たら右に移動できます。サイコロは「上」「下」「左」「右」の 4 つの目が出るとします。ただし、START 時のマス目は「上」や「左」にはいけませんね。このようにしてランダムに進んだとしたら、例えば右端の中央に設定した GOAL にどのぐらいでたどり着けるのでしょうか。これも**モンテカルロ法**の手法ですね。

このプログラムには、図 Q-2 のように障害物も自由に設定できるとします。もちろん障害物のほうには進むことはできません。

このように設定したプログラムを 100step だけ走らせてみます。100 回サイコロを振るということです。100step というのはマス目が 9×9 ですから、理屈上すべてのマス目上を辿れることになります。でもひょっとして狭い範囲で行ったり来たりで終わる可能性も十分あります。それとは逆に短い step であっという間に GOAL する場合もあるかもしれません。

何回に 1 回 GOAL できると思いますか。

GOAL を探し当てたときちょっとコンピューターと一緒に喜びたくなります。しかしその後の探索も、コンピューターは試行錯誤の繰り返しです。

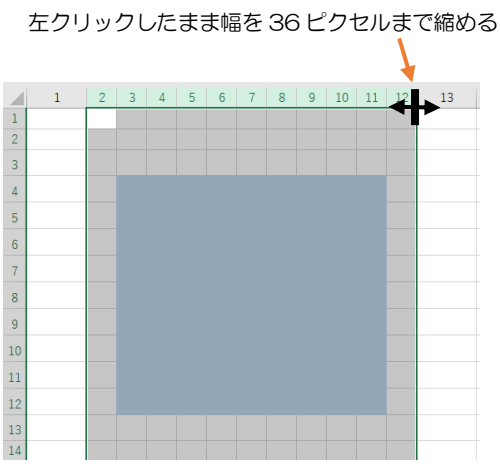
どうやったらコンピューターに GOAL した成功体験を学習させてやることができるのでしょうか。

これが今回のテーマです。

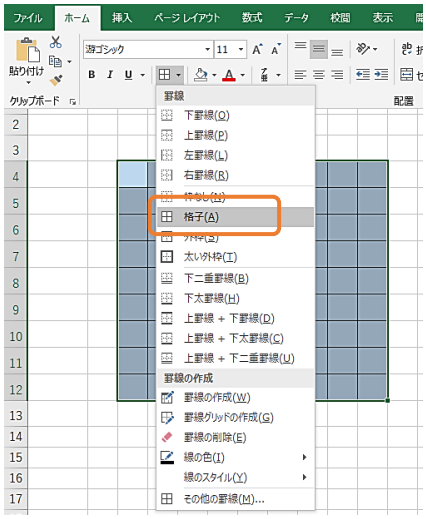
9×9 のマス目

Excel の Sheet1 に左図 Q-4 のような流れで **9×9＋外側**のマス目を作っていきます。

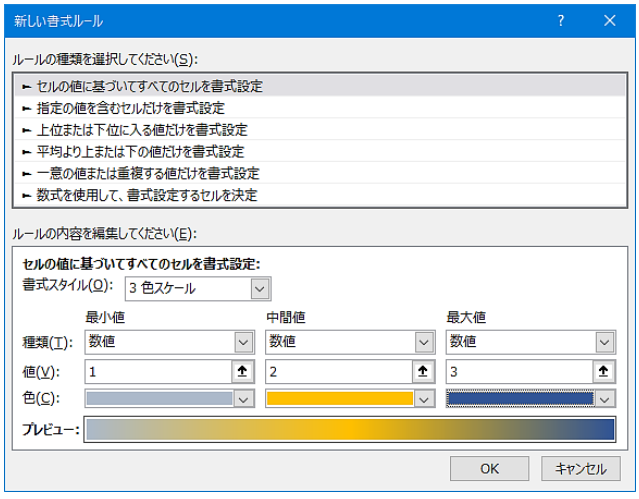
1．9×9 のマス目に線を引く
2．9×9 のマス目の外側のセルは障壁にする。
3．マス目に数字の 1 を入れると背景の色が現れる。
4．マス目に数字の 2 を入れるとマス目の中を動く何か（生命体）の色になる。
5．マス目に数字の 3 を入れると壁になる。



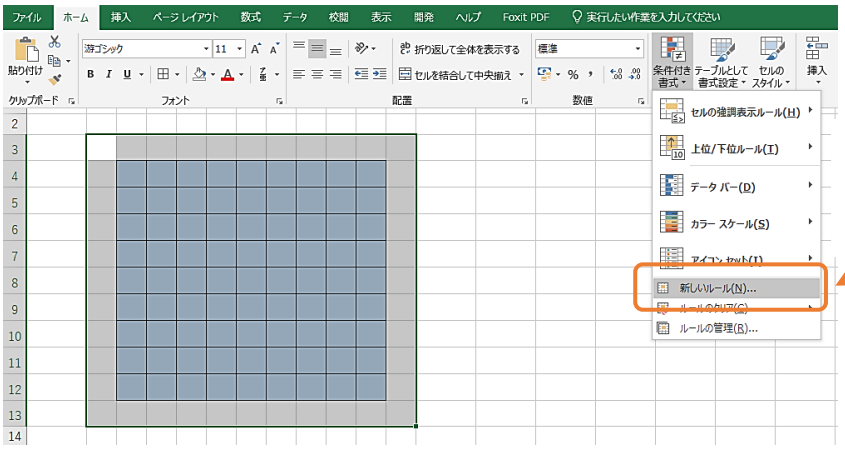
横 36 ピクセル×縦 31 ピクセル程度のマス目をセル（3,2）からセル（13，12）のレンジ（範囲）に作る。



9×9マス目板全体を選択して、「ホーム」タブから「罫線」を選び「格子」をクリック

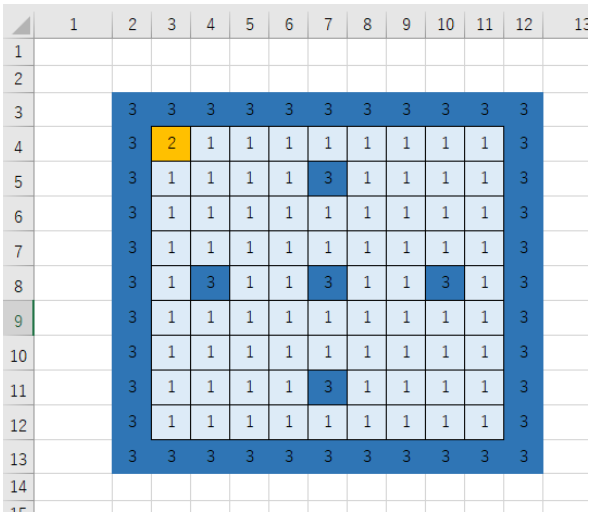


数値の値が1，2，3のときにどんな色にするか決める。



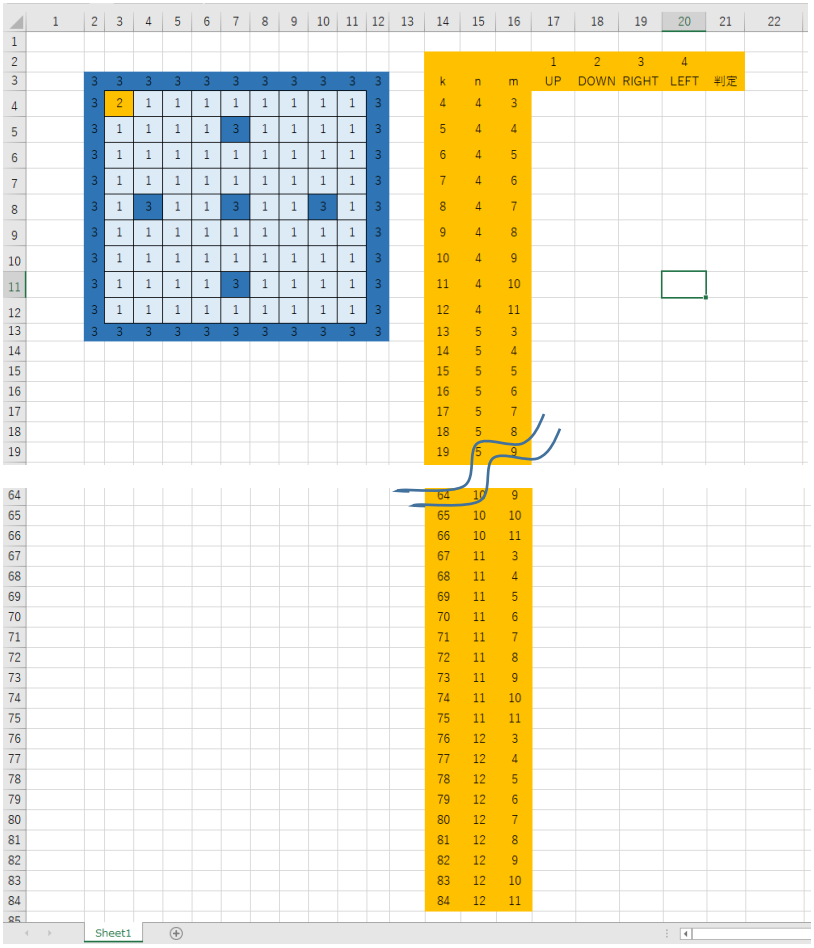
格子の外側のセルも選んでドラッグし「ホーム」タブの「条件付き書式」から「新しいルール」を選ぶ

図 Q-4



	1	2	3	4	5	6	7	8	9	10	11	12	13
1													
2													
3		3	3	3	3	3	3	3	3	3	3	3	
4		3	2	1	1	1	1	1	1	1	1	3	
5		3	1	1	1	1	3	1	1	1	1	3	
6		3	1	1	1	1	1	1	1	1	1	3	
7		3	1	1	1	1	1	1	1	1	1	3	
8		3	1	3	1	1	3	1	1	3	1	3	
9		3	1	1	1	1	1	1	1	1	1	3	
10		3	1	1	1	1	1	1	1	1	1	3	
11		3	1	1	1	1	3	1	1	1	1	3	
12		3	1	1	1	1	1	1	1	1	1	3	
13		3	3	3	3	3	3	3	3	3	3	3	
14													

図 Q-5



	k	n	m	1 UP	2 DOWN	3 RIGHT	4 LEFT	判定
4	4	4	3					
5	5	4	4					
6	6	4	5					
7	7	4	6					
8	8	4	7					
9	9	4	8					
10	10	4	9					
11	11	4	10					
12	12	4	11					
13	13	5	3					
14	14	5	4					
15	15	5	5					
16	16	5	6					
17	17	5	7					
18	18	5	8					
19	19	5	9					
64	64	10	9					
65	65	10	10					
66	66	10	11					
67	67	11	3					
68	68	11	4					
69	69	11	5					
70	70	11	6					
71	71	11	7					
72	72	11	8					
73	73	11	9					
74	74	11	10					
75	75	11	11					
76	76	12	3					
77	77	12	4					
78	78	12	5					
79	79	12	6					
80	80	12	7					
81	81	12	8					
82	82	12	9					
83	83	12	10					
84	84	12	11					

Sheet1

図 Q-6

サイコロの記録

図 Q-5 を見てください。9×9 のマス目板の外側に壁ができています。またマス目板の中にも 5 つの障壁を作りました。障壁は、どこにいくつ作ってもかまいません。このプログラムが完成した後も、自由に障壁を作ることができるようにプログラミングすることにします。

スタートとゴールの位置は、とりあえず決めておきます。スタートはセル（4,3）、ゴールはセル（8, 11）です。これも最終的には自由に設定できるようにします。

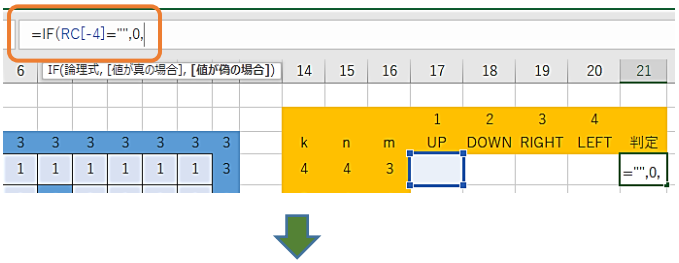
移動するものはサイコロを担いだマウスMということにしておきます。まずこのマウスがどこのセルにいたとき振ったサイコロの目はどれだけだったかを記録する表を図 Q-6 のように作っておきます。

セル（n,m）の列は、セル (4,3) から始まって 9×9 マス目板を横にセル(4,11)まで行った後、次にセル(5,3) からまた右にセル(5,11) にいって、次もセル(6,3)からという具合に最後はセル(12,3)からセル(12,11)まで書いてあります。数値を間違えないよう作ってください。

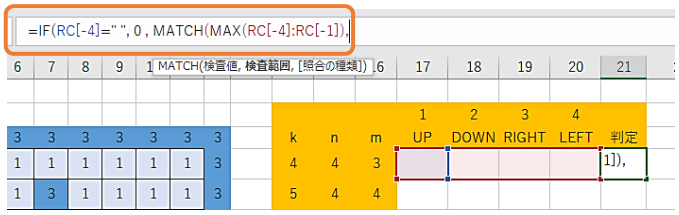
もし何回も同じセルン戻ってきた場合は、最後にセルを出たときの記録が残ります。

セル(4,21)をクリックして式を書いていく。

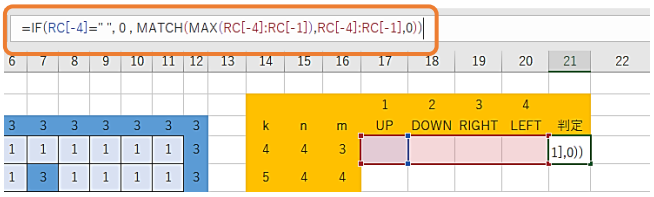
① **=IF(RC[-4]=”” ,0**
意味　もしここ（セル 4,21）から左に 4 番目のセルが
空白 “ ”（数字キー２の上にあるダブルクオーテーションを
２回押して書きます）ならここに０って書いてください。



② **=IF(RC[-4]=”” ,0,MATCH(MAX(RC[-4]:RC[-1])**
意味　そうじゃなかったら、左 4 番目から左隣までのセルの中
で一番大きい(MAX)ヤツは・・・



③=IF(RC[-4]=”” ,0,MATCH(MAX(RC[-4]:RC[-1]),RC[-4]:RC[-1],0))
意味　その 4 つの並びの中で左から何番目になるか、



④　1 から 4 の番号で書きなさい。

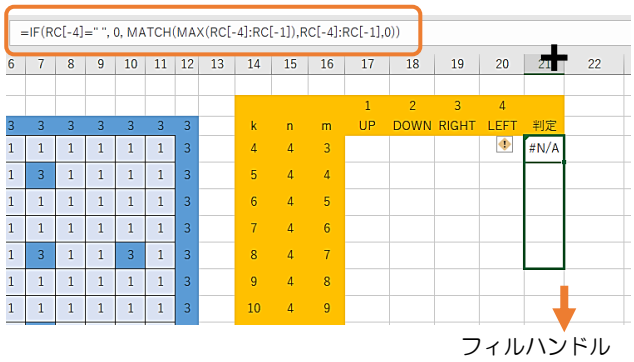


図 Q-7

MATCH 関数

ここで今まで使ったことのない Excel のシート上で使う MATCH 関数というものを使ってみます。例で紹介します。

例

MATCH(MAX(A1:D1), A1:D1, 0)

これはセルの A1 から D1 までの中で最大値があるセルは A1 から何番目にあるかを返してくれる関数です。

実際の Sheet1 に書きながら見ていきます。この表の UP , DOWN , RIGHT , LEFT の欄には、そのセルにマウスMが来た時に転がしたサイコロの値が乱数として記入されるようにします。そしてその 4 つの乱数のうち最も大きいものの向きに従うこととするのです。UP , DOWN , RIGHT , LEFT には番号を 1,2,3,4 とつけておきます。するとどっちに進むかは番号で示されることになります。

例えばセル (4,17) からセル (4,20)に出てくる乱数の値のうち一番大きいものを選び、それを 1,2,3,4 の番号に直してセル(4,21)に書いてくれるわけです。

ついでにマウスMが訪れなかったセルは 0 を返してくれるようにします。

#N/A の意味

「左の 4 つのセルが空白なんですけど、#N/A って出てきます？！」

「ああ「空白」を認識していないようです。」

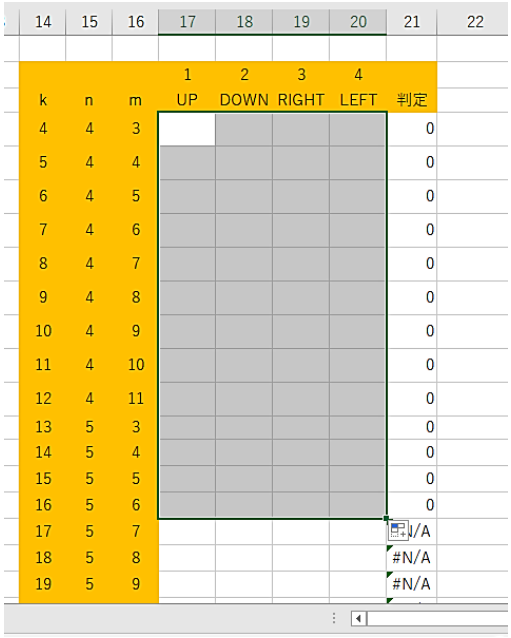
「一番上の 4 つのセルの中にスペースキーを打って「空白」を入れください。入れたら最下部までそうなるようフィルハンドルしてください。」

「あ、全部 0 になりました！」

⑤　#N/A って出てきているんですけど・・・

⑥　一番上の 4 つのセルに「スペースキー」で空白を入れてみてください。#N/A がなくなって数字の0になるはずです。

⑦　上の 4 つのセルをフィルハンドルしていくと #N/A が 0 になっていきます。



			1	2	3	4	
k	n	m	UP	DOWN	RIGHT	LEFT	判定
4	4	3					0
5	4	4					0
6	4	5					0
7	4	6					0
8	4	7					0
9	4	8					0
10	4	9					0
11	4	10					0
12	4	11					0
13	5	3					0
14	5	4					0
15	5	5					0
16	5	6					0
17	5	7					#N/A
18	5	8					#N/A
19	5	9					#N/A

⑧　適当な場所にダミーの数字を入れて、確かに最大値のあるセルの場所を「判定」でちゃんと書いてくれるか確認しましょう。

			1	2	3	4	
k	n	m	UP	DOWN	RIGHT	LEFT	判定
4	4	3					0
5	4	4					0
6	4	5					0
7	4	6	23	12	1	5	1
8	4	7	8.9	9	9.1	8.8	3
9	4	8					0
10	4	9					0
11	4	10					0
12	4	11					0
13	5	3					0
14	5	4					0
15	5	5					0
16	5	6					0
17	5	7					0
18	5	8					0
19	5	9					0

図 Q-8

プログラミングへ

まだどうやってプログラミングするのか不思議だと思います。実際に製作したプログラムでは、はじめこの表はありませんでした。

プログラミングから先に始めたのです。そのため、マウスがモンテカルロ法でランダムに動くようにはなりましたが、そこから、コンピューターが学習する方法で行き詰ってしまったのです。

学習させる方法とプログラミングがうまくつながりません。製作中のある日曜日の朝、妻が入れてくれたミルクティーに力をもらい、思い切ってそれまでに作ったプログラムを書き直しました。

この表を先に作り、そしてプログラムを改良したのです。そこから完成へと進むことができました。

この講座では、できるだけプログラミングのブラックボックス化をなくそうと思っています。ブラックボックスというのは、例えば MATCH 関数のように中身がわからないものです。これもプログラミングでうまく作れると思うのですが、ちょっとズルをしてしまいました。

このテキストを作る前に最近人気の Python で作れないか検討するためにいろいろな本を読みました。しかし、いろんな人が開発したブラックボックスがあちこちに出てきて「こりゃだめじゃ」ということになりました。VBA でやった後、大学で Python や C⁺言語をやるといいと思います。C⁺言語は大学でよく使われています。

VBA の良さは、表計算のシートが横にあるということです。いちいち作らずに済みます。言語は一つ知っていれば他の言語はすぐ使えるようになります。ニューラルネットワークにも表が有効という本もあります。

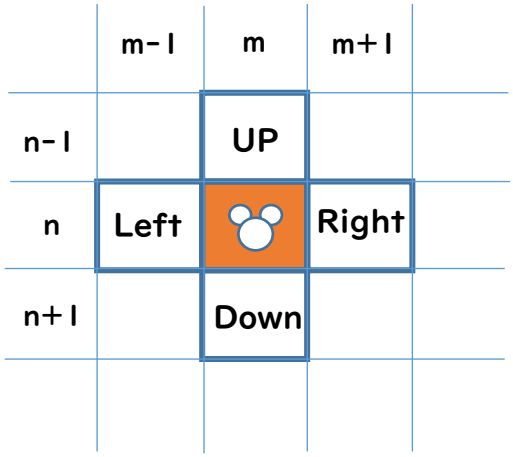


図 Q-9

```
Dim n , m , k , step As Integer
Dim U, D, R, L As Single

Sub 経路発見()
    '出発地点
        n = 4
        m = 3
        Cells(n, m) = 2
For step = 1 To 100
    '表の行番号 k を決定する
    If n = 4 Then
        k = 1 + m
    ElseIf n = 5 Then
        k = 10 + m
    ElseIf n = 6 Then
        k = 19 + m
    ElseIf n = 7 Then
        k = 28 + m
    ElseIf n = 8 Then
        k = 37 + m
    ElseIf n = 9 Then
        k = 46 + m
    ElseIf n = 10 Then
        k = 55 + m
    ElseIf n = 11 Then
        k = 64 + m
    ElseIf n = 12 Then
        k = 73 + m
    Else
    End If
```

マウスMよ 動け！

マウスM が Cells(n,m) にいるとします。そのとき上のセル UP の場所は左図を見ればすぐわかりますね。

UP = Cells (n - 1 , m)

となります。同様にして

Down = Cells (n + 1 , m)
Right = Cells (n , m + 1)
Left = Cells (n , m - 1)

ここからがプログラミングのスタートです。実際に作った私のプログラミングも、まず白い紙に図 Q-9 を書いたところから始まりました。

プログラムを読み解いていく

それでは一緒にプログラムを読み解いていきます。プログラムのコードを書く白いシートを出して、意味の分かったものから少しずつコードを書いていきます。左のコードを見てください。ちょっと長く 3 頁にわたっています。

```
Sub 経路発見()
    [          ]
End Sub
```

まず「**経路発見**」という名前のプログラムなんだなとみて、End Sub がどこに書かれているかを探します。3 頁目ですね。

次に

```
    '出発地点
```

から後の 3 行を読み、スタート地点のセルが**セル** (4,3) つまり右上でセルの値を「2」にしなさい。つまりマウスM がそこにいるということにしなさいという意味であると読みます。

「'」(シングルクオーテーション　7 のキーの上) をはじめに書くとコードとは認識されません。プログラムの内容を書いておくと後から改良していくとき便利です。

次の IF から End If までは例えば**セル**(8,8)は Sheet1 の表に書いてあるセルの通し番号 k の何番になるかを調べるコードです。

問題　IF から End If **までの**コードを読んで**セル**(8,8)の場所は k がいくつになるか答えよ。

(解答は次の次のページ)

'UP DOWN RIGHT LEFT **のサイコロの値を記録**

```
If Cells(n - 1, m) = 1 Then
  U = Rnd()
Else
  U = 0
End If

If Cells(n + 1, m) = 1 Then
  D = Rnd()
Else
  D = 0
End If

If Cells(n, m + 1) = 1 Then
  R = Rnd()
Else
  R = 0
End If

If Cells(n, m - 1) = 1 Then
  L = Rnd()
Else
  L = 0
End If
```

a) 進める可能性のあるセルの調査

（マウスMはセル(n,m)にいます。）
もし上のセル(n-1,m)の値が 1 だったら（上（UP）に行くことが可能だから）サイコロを振ってその値を UP を意味する U の箱に入れます。それ以外の場合は(Else) U＝0としときます。それ以外というのは、このとき上のセルが 3 という値なら移動可能性は0ということです。

以下「下のとき（Down)」「右のとき（Right)」「左のとき(Left)」が同じように書いてあります。

これで U,D,R,L という箱の中にサイコロの値が入りました。

この値の中で一番大きいものを探す必要があります。最初に Sheet 1 に MATCH 関数を使って準備をしておいたので、表に入れてやれば自動で計算結果を出してくれるはずです。

```
  Cells(k, 17) = U
  Cells(k, 18) = D
  Cells(k, 19) = R
  Cells(k, 20) = L

  Cells(n, m) = 1
```

b) 調査結果を表に報告

まだマウスMはセル(n,m)にいます。その場所のｋの値もすでに計算済みですね。そこで

Sheet1 のサイコロの値の表のセル(k,17)UP のセルに U の値を入れときます。同様にしてセル(k,18)DOWN に D の値、セル(k,19)RIGHT に R の値,セル(k,20)LEFT に L の値を入れときます。

今のマウスのいるセルはマウスのいない空きセル 1 にしておきます。（だってもうそのセルは次に空き部屋になります。）

すると、表計算の MATCH 関数が最も確率の高いセルを見つけてくれます。

それがセル(k,21)です。

```
'サイコロの目が最も大きかったセルに移動する
    Select Case Cells(k, 21).Value

        Case Is = 1
          Cells(n - 1, m) = 2
          n = n - 1

        Case Is = 2
          Cells(n + 1, m) = 2
          n = n + 1

        Case Is = 3
          Cells(n, m + 1) = 2
          m = m + 1

        Case Is = 4
          Cells(n, m - 1) = 2
          m = m - 1

    End Select

    'もしゴールに達した時の処理
        If Cells(8, 11) = 2 Then
            Cells(8, 12) = 2
            step = 100
          Else

        End If

Next step
        Cells(n, m) = 2
End Sub
```

c) 一番可能性の大きいセルを知る

どれが一番サイコロの目が大きいかの判定がセル(k,21)に出ています。このセルの値が 1 のときは UP つまり上に行けということだから、上のセル(n-1,m)を 2 としてマウスが移動したことにします。このとき新しく n-1 を n と置き直して新しいマウスの位置(n,m)とします。

同様にして値が 2 の時は、DOWN で、下のセルに移動します。3 の時は RIGHT で右のセルに、4 の時は LEFT で左のセルに移動します。移動したら新しいマウスの位置(n,m)とします。

このようにしてモンテカルロ法でマウスがうろうろしていたら、100Step の間にこちらがゴールと設定しているセル(8,11)に達することがあるかもしれません。そのときの処理の仕方を作っておく必要があります。

d) マウスが偶然ゴールに来たら！

もしマウスがゴールであるセル(8,11)に達していたらセルのマス目板の外側のセル(8,12)を 2 にして色を変えマス目から出た感じを作ります。そし Step 数を 100 にして Step の 100 回繰り返しを途中でやめさせます。また最後マウスのいるセル Cells(n,m)も値を 2 にして終わりにします。

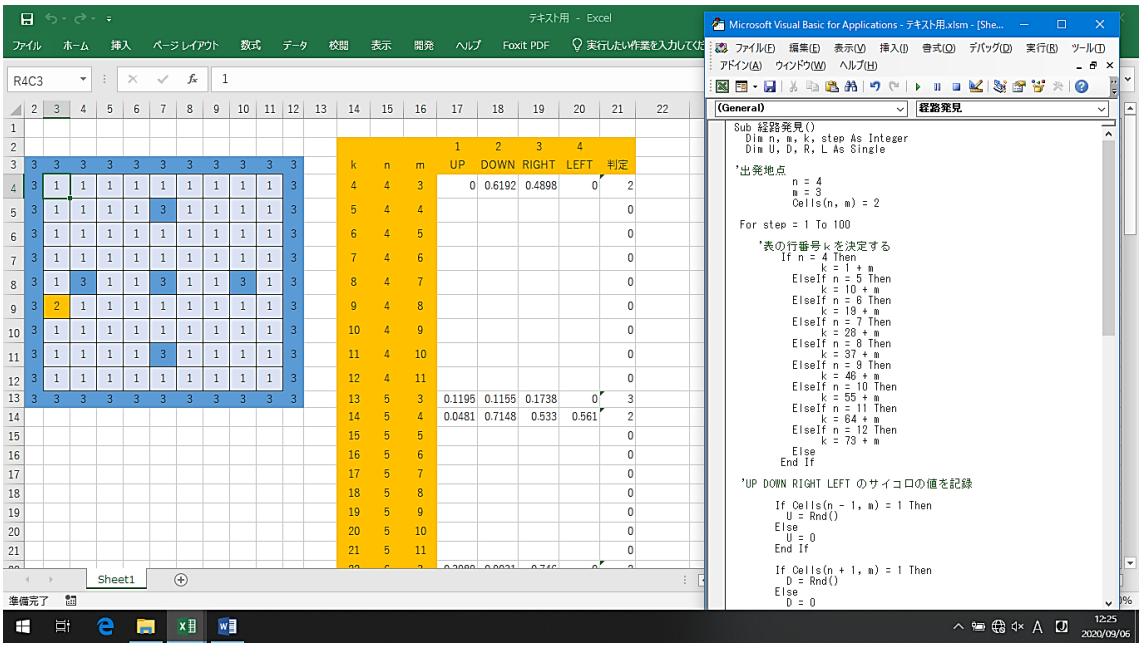


図 Q-10

図では Dim 文が Sub 経路発見()の中に入っているが、これは下のように外に出しておくほうがよい。

```
Dim n , m , k , Step As Integer
Dim U , D , R , L As Single
Sub 経路発見()
```

F8 で走らせる

コードを理解し**図 Q-10** のように書けたら、F8 を使って一行ずつ実行させながら、思い通りに動くかやってみましょう。まず表の K=4 のところにサイコロの目が書かれ、判定が 2 もしくは 3 になりマウスMが動き出したら第一段階クリアです。

次に F8 を押しっぱなしにするとマウスが行ったり来たりうろうろしているのが観測できます。表には、飛び飛びでサイコロの目が記録されていきます。同じところに戻ると、新しいものに上書きされます。

次に■でクリアし ▶ を使って、一気にプログラムを実行させたいのですが、ここで注意です。まだ製作途中ですので、新しくスタートさせるときには次の２つのことを手動でする必要があります。

１．**表の乱数の部分をすべて空白する。**つまりマウスで空白をフィルハンドルしながら乱数を消す。

２．**マス目板**のマウスMの最後の位置 2 を 1 にしておく。

マウスMが…！！

ところが、▶ で動かすとあっという間に 100 Step 動き回って我々には最後のマウスMの位置しか確認できません。F8 を使ってマウスを動かしたときは、ウロウロしてゆっくり動いてかわいかったのが、急に F1 マシンに乗ったマウスみたいになってしまいます。これではどう動いているのかわかりません。これから改良していきましょう。学習させるまえに、はや過ぎるマウスの動きをとらえる必要が出てきました。

	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
		サイコロの値と判定			1	2	3	4			マウスの経路				障害物の位置		
		k	n	m	UP	DOWN	RIGHT	LEFT	判定		STEP	n	m		n	m	
		4	4	3	0	0.6192	0.4898	0	2								
		5	4	4					0								
		6	4	5					0								
		7	4	6					0								
		8	4	7					0								
		9	4	8					0								
		10	4	9					0								
		11	4	10					0								
		12	4	11					0								
		13	5	3	0.1195	0.1155	0.1738	0	3								
		14	5	4	0.0481	0.7148	0.533	0.561	2								
		15	5	5					0								
		16	5	6					0								
		17	5	7					0								
		18	5	8					0								
		19	5	9					0								
		20	5	10					0								

図 Q-11

```
Dim n, m, k, step As Integer
Dim U, D, R, L As Single

Sub 経路発見()

    '出発地点
        n = 4
        m = 3
        Cells(n, m) = 2
    For step = 1 To 100

        '表の行番号 k を決定する
        If n = 4 Then
                k = 1 + m
        ElseIf n = 5 Then
                k = 10 + m
        ElseIf n = 6 Then
                k = 19 + m
                = 7 Then
                8 + m
                Then
```

nが－nとマイナスがついていることに注意

```
Cells(4, 23) = 0
Cells(4, 24) = -n
Cells(4, 25) = m
```

グラフで追跡

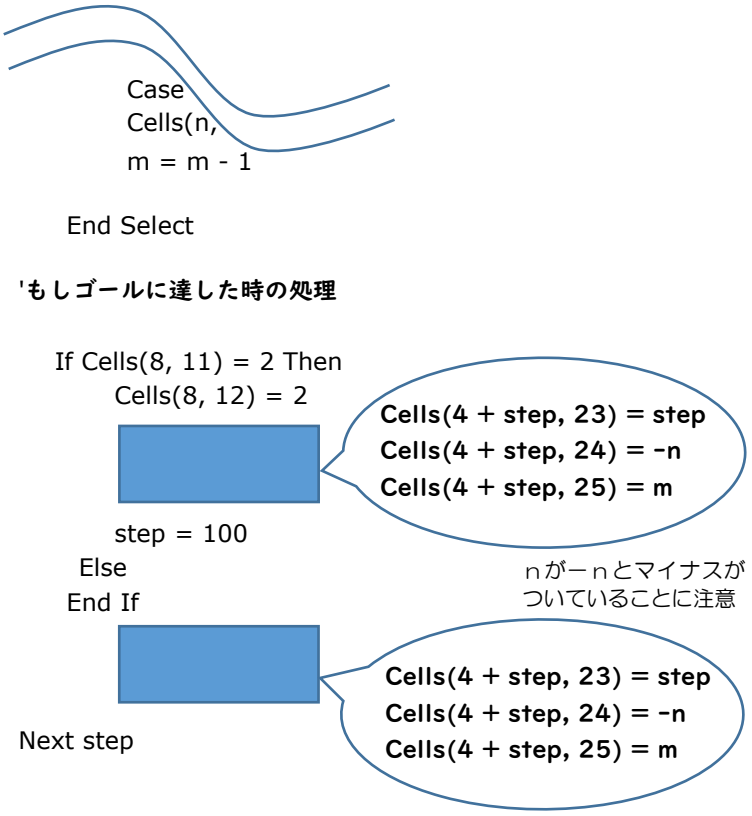
マウスMがどのように動いているのかは 9×9 マス目板ではうまくわからないということが分かりました。

それでグラフで動きを**トレース**することにします。トレースとは軌跡を線で表すということです。

まず Sheet1 に図 Q-11 のように新しい表を作ります。一つは**マウスの経路を** STEP 毎に記録する表です。これを**散布図グラフ**にしてトレースすれば動きが分かるかもしれません。

もう一つは、その散布図グラフに障害物の位置も記録できるようにするための「**障害物データ表**」です。

ここにプログラムでデータが入るようにしましょう。左のコードを見てください。新しく挿入するコードを吹き出しで、入れる場所を長方形で示しています。



挿入するコードは、まず step = 0 としてスタート地点セル(4,3) の値を (−4,3)として n を−n として入れておきます。マイナスにするのはグラフを描くためです。

最後に左のコードに書かれている吹き出しを見てもらうと分かるように、step の順番が 1 のときのマウス M の位置、2 のときの位置・・・というふうに Sheet1 に作った表「**マウスの経路**」の中に数値が入っていくことになります。

コードを追加したら、F8 を使って手動で表の中に数値が入っていくか確認します。何かバグがあると動きません。

うまく動くと**図 Q-12** のように表に数値が入っていきます。この図のときはたまたま step68 でゴールに達しました。

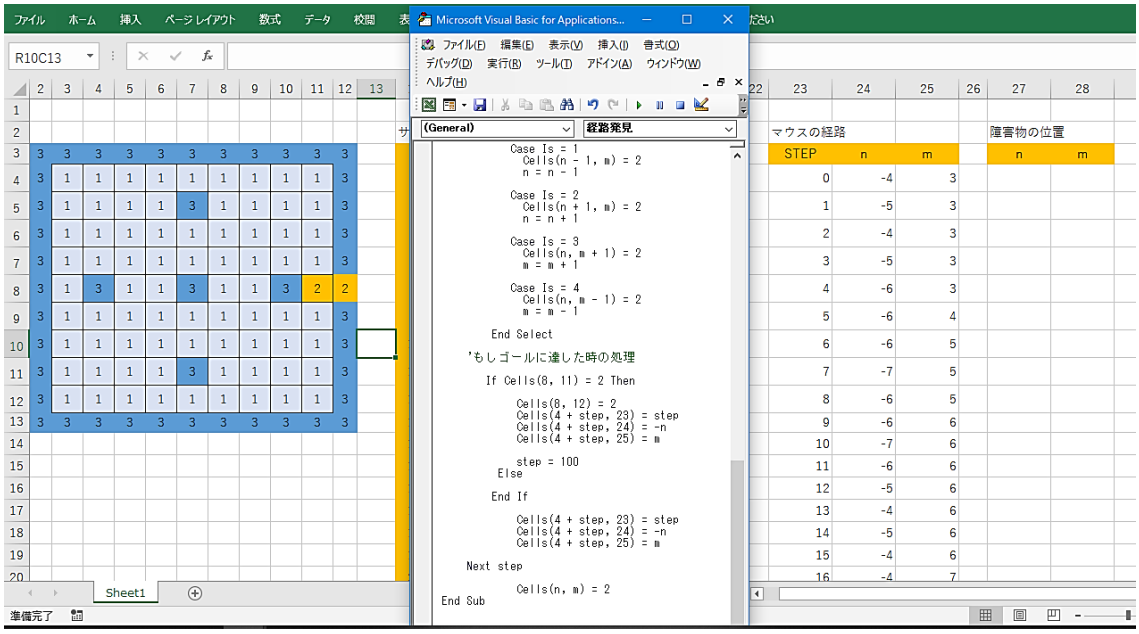


図 Q-12

図 Q-13

経路図を描く

100step の試行の後、図 Q-13 のような経路図が出ていたら、どのような動きをしたか一目でわかります。例に上げたグラフはなんと 68 ステップで GOAL に達しています。これは「**散布図**」の中の各点を曲線で結ぶグラフを選択したものです。

このグラフがいいのは、行ったり来たりが曲線を使っておおよそわかるという点です。

このグラフを自分で作ることができますか？簡単そうですが、縦軸が −n、横軸が m になっていますのでちょっと工夫がいります。卒業試験だと思って挑戦してみましょう。

さらに、この**経路図**には障害物が描かれていないために、もう一つ分かりにくいものになっています。そこで障害物も自動で入るようにしておきます。

左のコードを読んでみてください。もうずいぶん読めるようになったのではないでしょうか。

```
Sub 探索開始前の初期設定()
        Cells(8, 12) = 3
        k = 4
    For n = 4 To 12
        For m = 3 To 11
            Select Case Cells(n, m).Value
                Case Is = 1

                Case Is = 2
                    Cells(n, m) = 1

                Case Is = 3
                    Cells(k, 27) = -n
                    Cells(k, 28) = m
                    k = k + 1
            End Select
        Next m
    Next n
End Sub
```

k=k+1 というのは初め k=4 のセル(4,27)(4,28)に障害物の位置を書きますが、次の障害物の位置は、k の値を一つ増やして一つ下のセル(5,27)(5,28)に書くようにするためです。

e) 探索前のフィールドの掃除

ゴールした場合 2 に代わっているセル(8,12)を 3 にして壁に戻します。k の値は k=4 とおきます。

次はマス目のセルが 2 のものは 1 に直し、セルの値が 3 のものは障害物として Sheet1 の表に記録させます。まず n=4 としてマス目板の 4 行目の並びを m=3 から 11 までセルを一つずつチェックしていきます。それが済んだら今度はマス目板の n=5 で 5 行目の横の並びのセルをチェック。12 行目までそれをやるということになります。つまりマス目のすべてのマスを初期化します。

```
Dim n, m, k, step As Integer
Dim U, D, R, L As Single

Sub 探索開始前の初期設定()
            Cells(8, 12) = 3
            k = 4
        For n = 4 To 12
            For m = 3 To 11
                Select Case Cells(n, m).Value
                    Case Is = 1
                    Case Is = 2
                        Cells(n, m) = 1
                    Case Is = 3
                        Cells(k, 27) = -n
                        Cells(k, 28) = m
                        k = k + 1
                End Select
            Next m
        Next n
End Sub

Sub 経路発見()
'出発地点
            n = 4
            m = 3
            Cells(n, m) = 2
              ls(4, 23) = 0
                4, 24) = -n
                  25) = m
```

プログラムチェック

「探索開始前の初期設定」というプログラムができたら、左のようにプログラムを大きく 2 つ書いたことになりますから、変数の定義である

```
Dim n, m, k, step As Integer
Dim U, D, R, L As Single
```

は 2 つのプログラムの外に出し、2 つのプログラム共通の定義だということにします。

図 Q-14 のようにコードを書いたら F8 で「探索開始前の初期設定」のプログラムの最初にカーソルをもってきて F8 でチェックしていきます。うまく走ると、9×9 マス目板には 2 がなくなりすっきりした盤面になります。また右の「障害物の位置」には障害物の座標が入っていき、プログラムがマス目板を一番上から順に横のほうに点検していくのがリアルに分かります。「おう、読んでる読んでる」という感じでしょうか。プログラムを書くときはこんな実感が大事なのです。

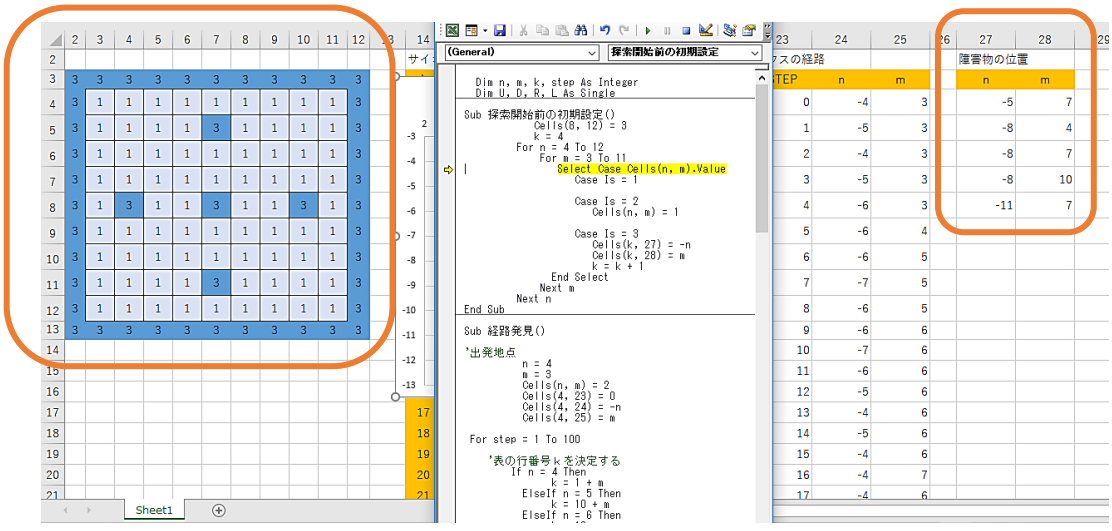


図 Q-14

図 Q-15-a

図 Q-15 のようにグラフ「**マウス M の経路**」の中に障害物も入れました。ずいぶん分かりやすくなりましたね。

それでは今まで作ったプログラムにスタートボタンも作り、マウス M がサイコロを振りながらランダムにゴールを目指して動き回るプログラム「**マウスの冒険**」α版 を完成させましょう。図 Q-15-b のように**プログラム「マウスの冒険」α版の特徴**をまとめましたので見てください。

障壁は自由に作れますので、完成したらいろいろ障壁のバリエーションを変えて実験してみましょう。

プログラム「マウスの冒険」α版 の特徴

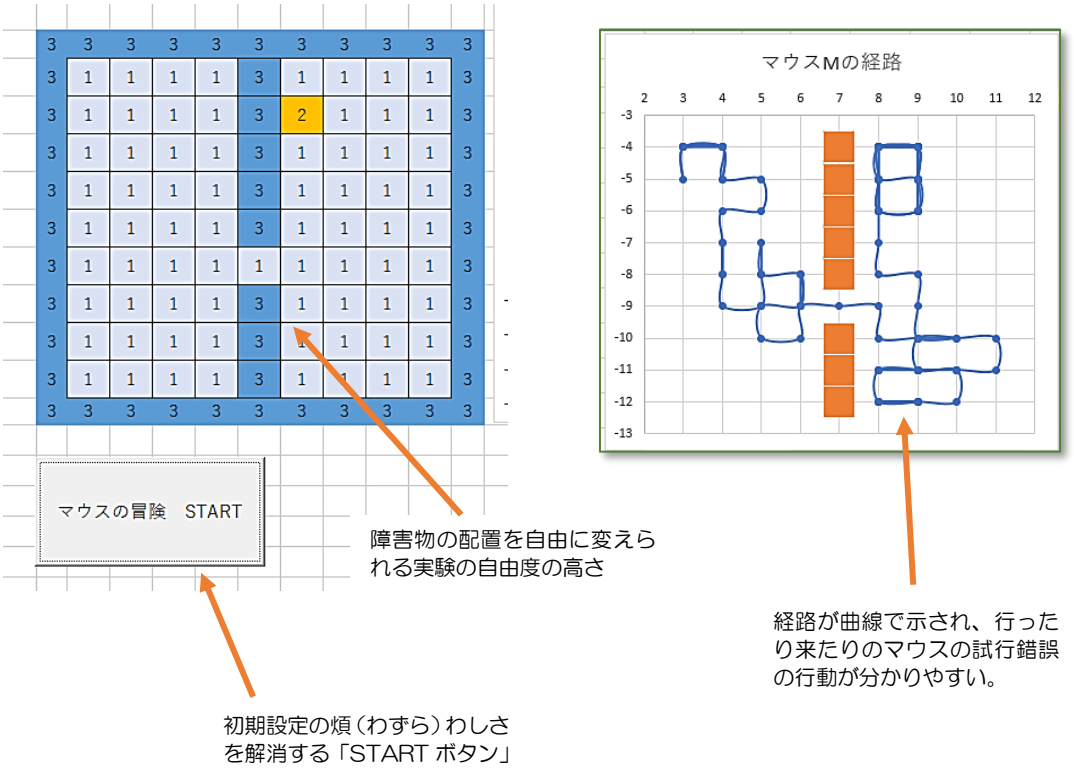


図 Q-15-b

プログラム「マウスの冒険」α版

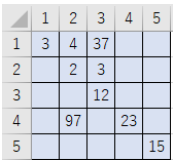
```
Dim n, m, k, step As Integer
Dim U, D, R, L As Single
```

```
Sub マウスの冒険()
    探索開始前の初期設定
    経路発見
End Sub
```

「マウスの冒険」というメインプログラムは、2 つのプログラム「探索開始前の初期設定」と「経路発見」からできている。

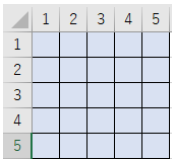
```
Sub 探索開始前の初期設定()
    'サイコロの値を初期化
        Range(Cells(4, 17), Cells(84, 20)).Value = " "
    'マウスの経路を初期化
        Range(Cells(4, 23), Cells(104, 25)).Clear
    '障害物を初期化
        Range(Cells(4, 27), Cells(84, 28)).Clear
    '9×9 マス目板の初期化
            Cells(8, 12) = 3
            k = 4
        For n = 4 To 12
            For m = 3 To 11
                Select Case Cells(n, m).Value
                    Case Is = 1
                    Case Is = 2
                        Cells(n, m) = 1
                    Case Is = 3
                        Cells(k, 27) = -n
                        Cells(k, 28) = m
                        k = k + 1
                End Select
            Next m
        Next n
End Sub
```

サイコロの初期化コード
空白「“”」を入れて初期化していることに注意
Range（レンジ）というのは「範囲」ということ。
Range(Cells(1,1),Cells(5,5))というのは下図の範囲



	1	2	3	4	5
1	3	4	37		
2		2	3		
3			12		
4		97		23	
5					15

Range（Cells(1,1),Cells(5,5)）.Clear とすると下図のようになる。



	1	2	3	4	5
1					
2					
3					
4					
5					

```
Sub 経路発見()
    '出発地点
            n = 4
            m = 3
            Cells(n, m) = 2
            Cells(4, 23) = 0
            Cells(4, 24) = -n
            Cells(4, 25) = m
     For step = 1 To 100
        '表の行番号 k を決定する
            If n = 4 Then
                    k = 1 + m
            ElseIf n = 5 Then
```

```
        k = 10 + m

      ElseIf n = 6 Then
            k = 19 + m
      ElseIf n = 7 Then
            k = 28 + m
      ElseIf n = 8 Then
            k = 37 + m
      ElseIf n = 9 Then
            k = 46 + m
      ElseIf n = 10 Then
            k = 55 + m
      ElseIf n = 11 Then
            k = 64 + m
      ElseIf n = 12 Then
            k = 73 + m
      Else
    End If

'UP DOWN RIGHT LEFT のサイコロの値を記録
    If Cells(n - 1, m) = 1 Then
        U = Rnd()
    Else
        U = 0
    End If
    If Cells(n + 1, m) = 1 Then
        D = Rnd()
    Else
        D = 0
    End If
    If Cells(n, m + 1) = 1 Then
        R = Rnd()
    Else
        R = 0
    End If
    If Cells(n, m - 1) = 1 Then
        L = Rnd()
    Else
        L = 0
    End If
    Cells(k, 17) = U
    Cells(k, 18) = D
    Cells(k, 19) = R
    Cells(k, 20) = L
    Cells(n, m) = 1

'サイコロの目が最も大きかったセルに移動する
    Select Case Cells(k, 21).Value
          Case Is = 1
              Cells(n - 1, m) = 2
              n = n - 1
```

```
                    Case Is = 2
                        Cells(n + 1, m) = 2
                    n = n + 1
                Case Is = 3
                    Cells(n, m + 1) = 2
                    m = m + 1
                Case Is = 4
                    Cells(n, m - 1) = 2
                    m = m - 1
        End Select
    'もしゴールに達した時の処理
        If Cells(8, 11) = 2 Then
            Cells(8, 12) = 2
            Cells(4 + step, 23) = step
            Cells(4 + step, 24) = -n
            Cells(4 + step, 25) = m
            step = 100
          Else
        End If
            Cells(4 + step, 23) = step
            Cells(4 + step, 24) = -n
            Cells(4 + step, 25) = m
    Next step
            Cells(n, m) = 2
End Sub
```

```
Private Sub CommandButton1_Click()
    マウスの冒険
End Sub
```

これが「マウスの冒険」のスタートボタンが持つプログラム
「マウスの冒険」というプログラムを実行せよという意味。

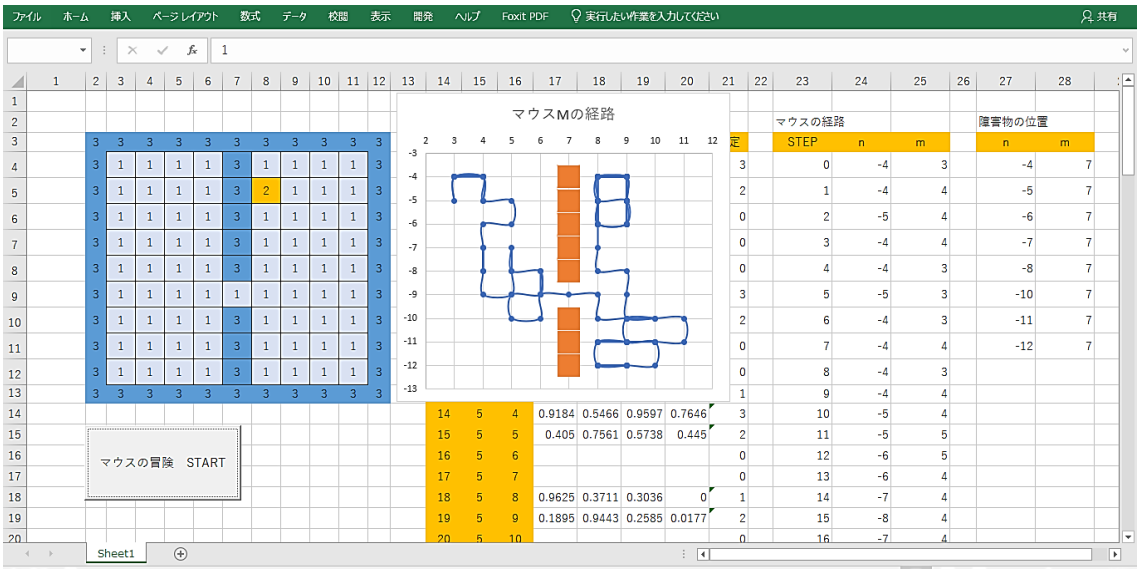


図 Q-16

学習するマウス

さて、何とかマウス M はランダムに経路を探索するようになりました。マウス M は、「**マウスの冒険」ボタン**を押すたびに文句も言わず探索してくれます。何回かやっていると、意外にゴールに達するパターンがあるのに気づきませんか。そりゃ、１００回も行ったり来たりすれば辿り着きますよね。

障害物の位置や数も色々変えてみましょう。いろいろのパターンがあることに気づくはずです。特に、突破する穴が１個しかない場合はわかりやすいですね。

こうやって探索させていると、こんなことを考えませんか。

『ゴールに達した時は、そのコースを覚えていてくれたらいいのに・・・』

そうですね。それならこうしてはどうでしょう。ゴールしたら、その時の判断はよかったとして評価してあげるのです。ゴールに達したのだったら、セルの中のマウスがサイコロで決めた一つ一つの向きには何らかの出口への可能性が他の向きより高いのかもしれません。

うまく言ったらほめてあげる、これは学習させる大事な方法です。

これを人工知能の研究では「**Q 学習**」と呼んでいます。なぜ「**Q 学習**」というのかは不思議ですが、この章の最後にこの「**Q 学習**」（**Q Learning**）の **Q** に込められた研究者の意図にちょっと触れたいと思います。

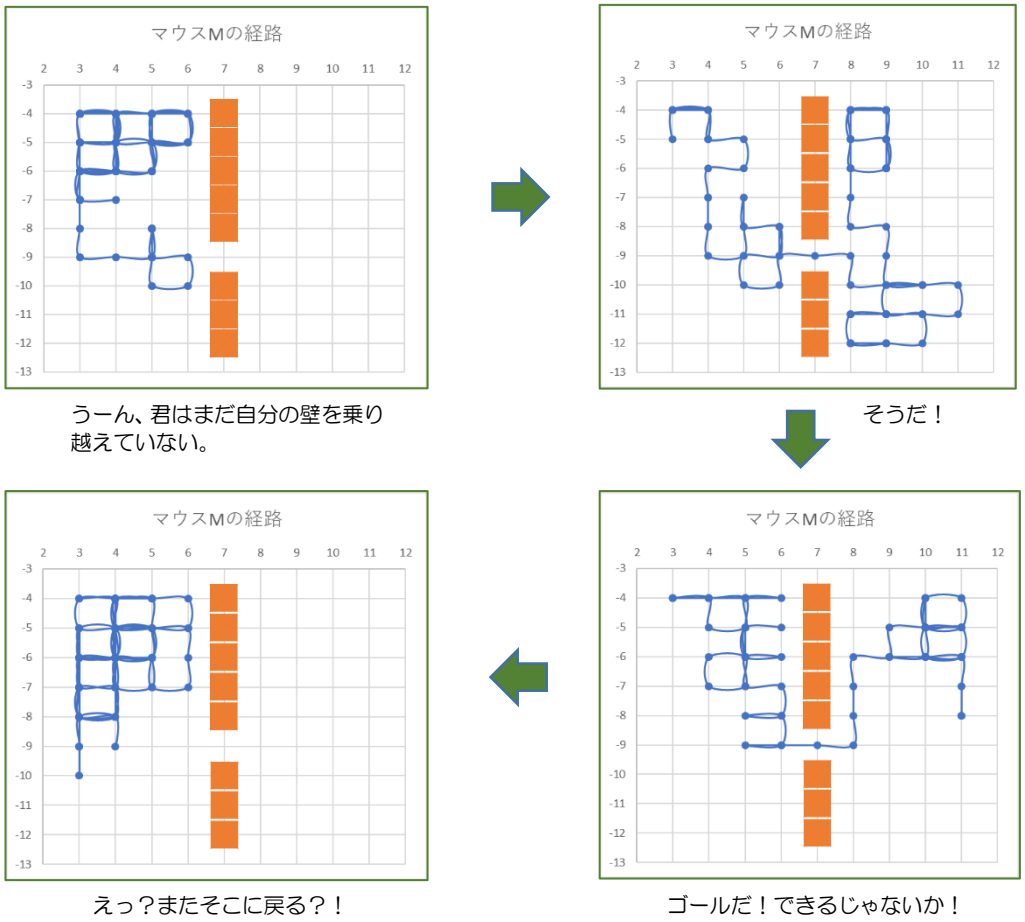


図 Q-17

新しいファイル「マウスの冒険 β 版」

まずこれまでに完成した作品を次の名前にして保存します。

　マウスの冒険 α 版

次に、今度は学習するマウスをこの α 版を改良しながら作っていきます。このファイルの名前を

　マウスの冒険 β 版

とすることにします。保存の仕方はわかりますか。「マウスの冒険 α 版」を開いたまま、「ファイル」タブから「名前をつけて保存」を開き、「現在のフォルダ」の中に、α を β に変えて保存するだけです。すると、これまでの α 版はしっかりフォルダにありますから、β 版でいろいろ改良しても安心ですね。

　まず、図 Q-16 のように「Q シート」というのを作りましょう。これは、マウスMがゴールに達した時の各セルの判定を記憶させるものです。α 版に少し知性を持ってもらうための頭脳に当たるのが「Q シート」です。「Q シート」という名前はちょっと変ですが、これから作っていく AI の学習方法を一般に Q 学習と呼んでいることから名付けました

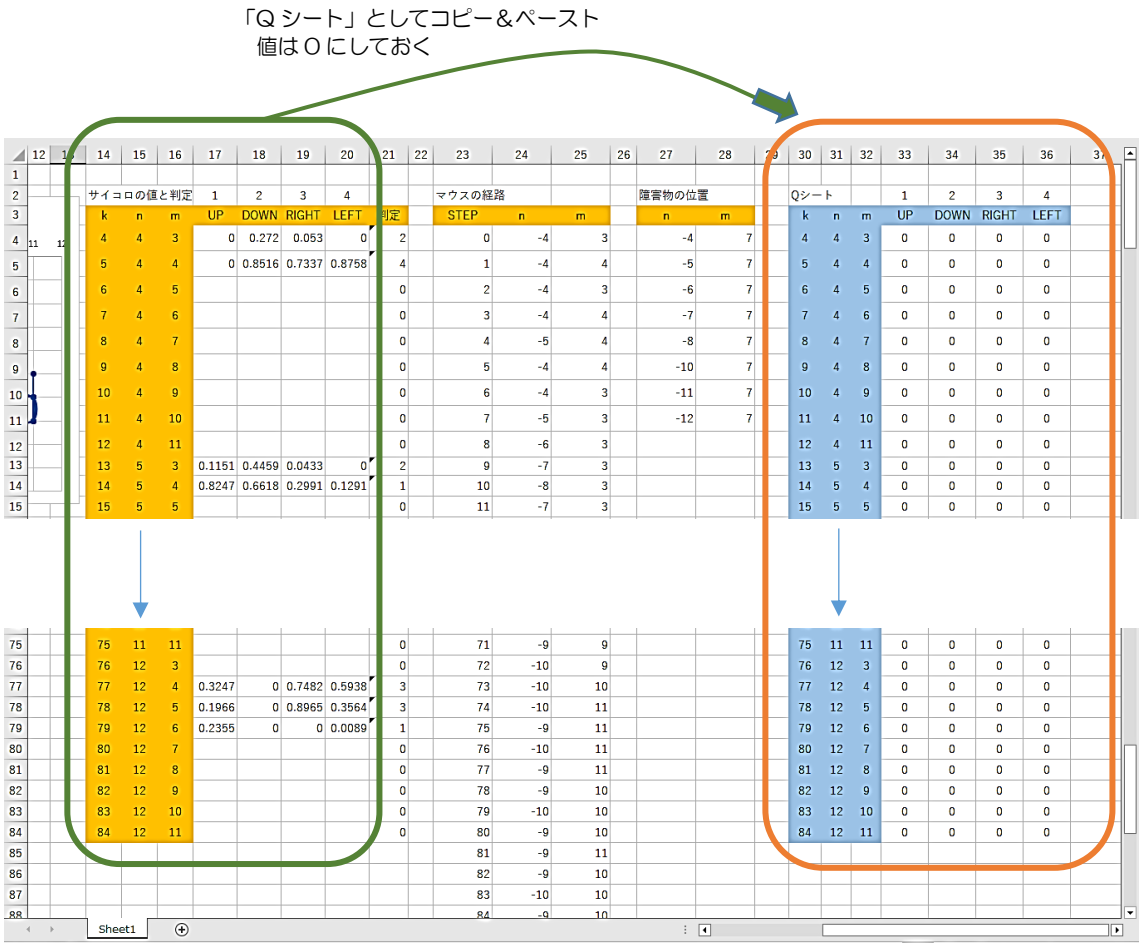


図 Q-18

```
Sub Qシート転写()
  For k = 4 To 84
    Select Case Cells(k, 21).Value
        Case Is = 1
          Cells(k, 33) = Cells(k, 33) + 1
        Case Is = 2
          Cells(k, 34) = Cells(k, 34) + 1
        Case Is = 3
          Cells(k, 35) = Cells(k, 35) + 1
        Case Is = 4
          Cells(k, 36) = Cells(k, 36) + 1
    End Select
  Next k
End Sub
```

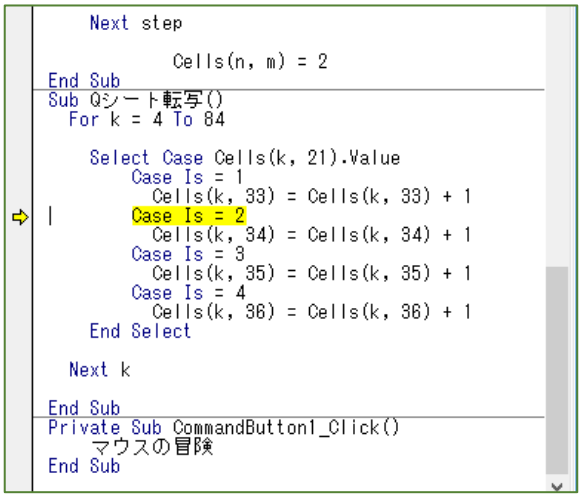


図 Q-19 - a

Qシートとは？

ゴールしたときの記憶がその後のマウスの動きに影響あたえるとどうなるのでしょう。この改良プログラムβ版では、ゴールに達した場合、その成功体験をマウスに与えることにします。

具体的には、ゴールした時の各セルでの向きを、その後のサイコロの値に少しだけ反映させるのです。まずその成功体験を記憶させる表を作ります。この表の名前を「**Qシート**」というのは前ページで触れましたね。

左の「**Qシート転写**」というプログラムを見てください。このプログラムの意味は図 Q-19-a のように実際にコードを入力して、F8 で走らせてみるとよく分かります。下の図 Q-17-b の 2 つの表を見比べながら動かして見てください。ゴールを果たしたときには、左の表から **Qシート**に結果が上から順に重ね書きの形で記録されていきます。

	14	15	16	17	18	19	20	21	2
	サイコロの値と判定			1	2	3	4		
	k	n	m	UP	DOWN	RIGHT	LEFT	判定	
	4	4	3	0	0.7731	0.3381	0	2	
	5	4	4	0	0.751	0.4199	0.4134	2	
	6	4	5	0	0.1265	0.2583	0.8793	4	
	7	4	6	0	0.594	0	0.9755	4	
	8	4	7					0	
	9	4	8					0	
	10	4	9					0	
	11	4	10					0	
	12	4	11					0	
	13	5	3	0.3764	0.8879	0.4144	0	2	
	14	5	4	0.6199	0.8771	0.1632	0.0695	2	
	15	5	5	0.2984	0.4115	0.3504	0.6366	4	
	16	5	6	0.0529	0.9159	0	0.8461	2	
	17	5	7					0	
	18	5	8					0	
	19	5	9					0	

	30	31	32	33	34	35	36	37
	Qシート			1	2	3	4	
	k	n	m	UP	DOWN	RIGHT	LEFT	
	4	4	3	0	1	0	0	
	5	4	4	0	1	0	0	
	6	4	5	0	0	0	1	
	7	4	6	0	0	0	1	
	8	4	7	0	0	0	0	
	9	4	8	0	0	0	0	
	10	4	9	0	0	0	0	
							0	
							0	
							0	
							1	
	16	5	6	0	1	0	0	
	17	5	7	0	0	0	0	
	18	5	8	0	0	0	0	
	19	5	9	0	0	0	0	

左の表ではセル(4,6)にマウスが来た時は、左（LEFT）向きに進んだことが分かる。。これを「Q シート」に記録する。セル(4,6)のときは左向き 1 回と数字の 1 を入れる。2 度の目のゴールでもこのセルから左に行った場合は、また 1 が足されて 2 となる。

図 Q-19 - b

```
'UP DOWN RIGHT LEFT のサイコロの値を記録
UR = Cells(k, 33).Value
DR = Cells(k, 34).Value
RR = Cells(k, 35).Value
LR = Cells(k, 36).Value
```

今マウスがいるセルの、Q シートの値を読み込む。
Q シートの値は、どちらに行けばゴールする確率が高いかを経験的に教えてくれる。

```
If Cells(n - 1, m) = 1 Then
  U = Rnd() + UR * γ
Else
  U = 0
End If

If Cells(n + 1, m) = 1 Then
  D = Rnd() + DR * γ
Else
  D = 0
End If

If Cells(n, m + 1) = 1 Then
  R = Rnd() + RR * γ
Else
  R = 0
End If

If Cells(n, m - 1) = 1 Then
  L = Rnd() + LR * γ
Else
  L = 0
End If
```

もし下側のセルが 1 でそっちに移動可能な時、サイコロを振る。そのサイコロの値は、乱数にそれまでの成功回数に 0.01 をかけたものを加えている。そのため、以前ゴールした向きのほうがその分だけ確率が上がるようにしている。

```
Cells(k, 17) = U
Cells(k, 18) = D
Cells(k, 19) = R
Cells(k, 20) = L
Cells(n, m) = 1
```

このときのそれぞれの向きの確率を、「サイコロの値と判定」の表に記録する。
また、マウスは移動してこのセル(n,m)は空になるので、数値を 1 としておく。
表では、一番確率の大きい向きが判定され、表の（k、21）に判定結果が示される。

Q シート利用

左のコードは α 版では、セル（n,m）にいるマウスがサイコロを振るプログラムでした。これを改良しています。説明します。

セルには k=4 から k=84 まで通し番号がついていることを思い出してください。このコードの前に、今マウス M がいるセル(n,m)の数値からセルの通し番号 k を計算しています。従って、縦に k の値が書かれている「Q シート」から、そのセルに以前マウスが来た時、どっちに行ったらゴールまで達している回数が多い向きかという情報が手に入ります。

最初の 4 行はその時の Q シートの情報を UP,DOWN,RIGHT,LEFT 別に読み取ってそれを UR,DR,RR,LR というところに入れときます。
次に

```
If Cells(n - 1, m) = 1 Then
    U = Rnd() + UR * γ
  Else
    U = 0
End If
```

は今マウスがいるセルの真上のセルがもし 1 で空だったら、真上に行ったほうが良い可能性を計算します。そのとき、これまで上に行ってゴールした回数 UR に記憶定数 γ をかけたものを、乱数 Rnd()に加えます。γ＝0.01 程度にすると乱数 Rnd()の値はあまり変化しませんが、成功回数が 100 回を超えると乱数の値より大きくなり、有利性が固定化します。それが

```
U = Rnd() + UR * γ
```

という計算式の意味です。このプログラムのキモになる計算になります。

```
Sub マウスの冒険()
    探索開始前の初期設定
    経路発見
        If Cells(8, 11) = 2 Then
            Q シート転写
        Else
        End If
End Sub
```

Sub マウスの冒険() ～ End Sub

このプログラムは短いけれどメインプログラムです。まずマウス M がゴールまでの探索を開始する初期設定をします。そして、探索に出かけます。
もし、セル(8,11)というゴールに達したのだったら、その探索を記憶するために Q シートにどのセルのときはどっちに行ったかを書き写しときます。その後、探索しゴールを見つけるたびに、Q シートに回数が加算されるようになります。

また、一回の探索で同じところを出たり入ったりした場合は、最後に出た向きだけが記録されるようになっています。

```
Private Sub CommandButton2_Click()
 Range(Cells(4, 33), Cells(84, 36)).Value = 0
End Sub
```

記憶RESET

PrivateSubCommandButton2_Click()
～End Sub

探索は連続して 350～400 回以上行いながらマウスに学習してもらいます。その間 Q シートの記憶は蓄積されます。

最初からやるには、Q シートの記憶をリセットする必要があります。

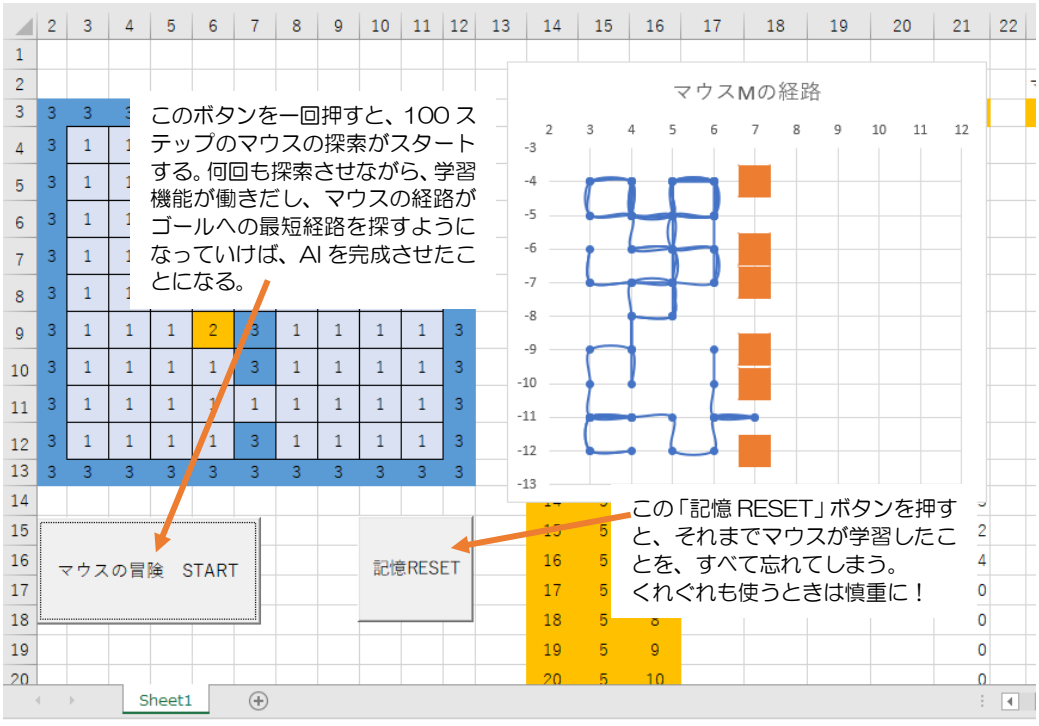


図 Q-20

プログラム「マウスの冒険」β 版

```
Dim U, D, R, L, UR, DR, RR, LR, γ As Single
Dim k, n, m, step As Integer
```

```
Sub マウスの冒険()
    探索開始前の初期設定
    経路発見
      If Cells(8, 11) = 2 Then
        Q シート転写
        Else
      End If
End Sub
```

```
Sub 探索開始前の初期設定()
    '記憶定数
      γ = 0.01
    'サイコロの値を初期化
      Range(Cells(4, 17), Cells(84, 20)).Value = " "
    'マウスの経路を初期化
      Range(Cells(4, 23), Cells(104, 25)).Clear
    '障害物を初期化
      Range(Cells(4, 27), Cells(84, 28)).Clear
    '9×9 マス目板の初期化
          Cells(8, 12) = 3
          k = 4
        For n = 4 To 12
            For m = 3 To 11
                Select Case Cells(n, m).Value
                    Case Is = 1
                    Case Is = 2
                        Cells(n, m) = 1
                    Case Is = 3
                        Cells(k, 27) = -n
                        Cells(k, 28) = m
                        k = k + 1
                End Select
            Next m
        Next n
End Sub
```

```
Sub 経路発見()
'出発地点
          n = 4
          m = 3
          Cells(n, m) = 2
```

AI プログラム

β 版が完成しました。左にその全コードを書いています。一つ一つチェックしていきましょう。

各**サブルーチン**（Sub() ～End Sub）は F8 でデバッグが済んでいるものが多いので、動き出すのは早いでしょう。「**探索開始前の初期設定**」の所に「**記憶定数**」として γ=0.01 としました。これは、このぐらいが最適なのではないかと考えて入れているものです。

動き出したら、まず「**記憶** RESET」ボタンを押して、**Q シート**をまっさらにリセットします。そして「**マウスの冒険** START」ボタンを押して、冒険させましょう。100 回や 200 回では中々最短距離を見つけ出すまでにはいきません。時々とんでもない冒険もしてくれます。これを Q 学習では explore と呼んでいます。せっかくいい方向でゴールを見つけたのに次のチャレンジでは全く違うことをしてしまう。学習ではこれがないと、だめなのかもしれません。一方で、以前成功したルートに頼ろうとする判断は exploit と呼ばれているそうです。その割合がこの **Q 学習**のキモです。

このプログラムでは、回を重ねるごとにルートが決まってくるようになります。つまり、はじめはほとんど冒険（explore）ですが、次第に確定のルートが巧みに利用される（exploit）ようになります。

「**記憶定数**」γ=0.01 がもっと小さいとマウスの冒険は収束するまでに時間がかかります。もっと大きいと、収束は早いのですが、それが最短距離に収束することはありません。これも実験してみてください。

```
        Cells(4, 23) = 0
        Cells(4, 24) = -n
        Cells(4, 25) = m
For step = 1 To 100
  '表の行番号 k を決定する
        If n = 4 Then
                k = 1 + m
          ElseIf n = 5 Then
                k = 10 + m
          ElseIf n = 6 Then
                k = 19 + m
          ElseIf n = 7 Then
                k = 28 + m
          ElseIf n = 8 Then
                k = 37 + m
          ElseIf n = 9 Then
                k = 46 + m
          ElseIf n = 10 Then
                k = 55 + m
          ElseIf n = 11 Then
                k = 64 + m
          ElseIf n = 12 Then
                k = 73 + m
          Else
        End If
 'UP DOWN RIGHT LEFT のサイコロの値を記録
        UR = Cells(k, 33).Value
        DR = Cells(k, 34).Value
        RR = Cells(k, 35).Value
        LR = Cells(k, 36).Value
      If Cells(n - 1, m) = 1 Then
          U = Rnd() + UR * γ
        Else
          U = 0
      End If
      If Cells(n + 1, m) = 1 Then
          D = Rnd() + DR * γ
        Else
          D = 0
      End If
      If Cells(n, m + 1) = 1 Then
          R = Rnd() + RR * γ
        Else
          R = 0
      End If
      If Cells(n, m - 1) = 1 Then
          L = Rnd() + LR * γ
        Else
          L = 0
    End If
```

大体、300 回から 400 回ぐらい試行させると最短距離を見つけ出しそうですが、まだどんなパターンがあるのかよく分かりません。皆さん、ちょっとマウスには突破できそうにない障壁を作ってチャレンジしてみてください。どんな場合でも最短距離を見つけ出せるのかはよくわかってないのです。

しかし不思議な気持ちです。プログラムを組んだ自分が言うのもなんですが、なにかコンピューターの中にマウスが存在しているようです。それも根気強さと冒険心を持ち、成功した時はメモ帳にコースを記録するクールな頭まで持っているマウスです。

これは AI なのでしょうか。それともこんなのはまだまだなのでしょうか。でも最短距離を発見するのは間違いないのです。

君たちの中から、これをきっかけに、AI の研究者やデータサイエンティストが出てくれば、このテキストを書いた甲斐があります。

Q 学習

そもそも**Q 学習**というのはコンピューターが学習する方法（**機械学習**）の一つで、1989 年にクリス・ワトキンズが理論的にまとめた方法といわれています。

Q という名前には、こんな話があります。彼の論文のキーワードはまず一つは Probability（**確率**），
もう一つは Reward（**報酬**）でした。
彼のまとめた **Q 学習法の** Q は The Q is the expected return from action at a given state**.**
（彼の論文からの抜粋らしい）
つまり「Q とは**与えられた状態における行動に見合った報酬**」というのが定義のようです。

```
            Cells(k, 17) = U
            Cells(k, 18) = D
            Cells(k, 19) = R
            Cells(k, 20) = L
            Cells(n, m) = 1

    'サイコロの目が最も大きかったセルに移動する
            Select Case Cells(k, 21).Value
                Case Is = 1
                  Cells(n - 1, m) = 2
                  n = n - 1
                Case Is = 2
                  Cells(n + 1, m) = 2
                  n = n + 1
                Case Is = 3
                  Cells(n, m + 1) = 2
                  m = m + 1
                Case Is = 4
                  Cells(n, m - 1) = 2
                  m = m - 1
            End Select

    'もしゴールに達した時の処理
            If Cells(8, 11) = 2 Then
                Cells(8, 12) = 2
                Cells(4 + step, 23) = step
                Cells(4 + step, 24) = -n
                Cells(4 + step, 25) = m
                step = 100
             Else
            End If
                Cells(4 + step, 23) = step
                Cells(4 + step, 24) = -n
                Cells(4 + step, 25) = m
    Next step
            Cells(n, m) = 2
End Sub
```

```
Sub Q シート転写()
  For k = 4 To 84
    Select Case Cells(k, 21).Value
        Case Is = 1
          Cells(k, 33) = Cells(k, 33) + 1
        Case Is = 2
          Cells(k, 34) = Cells(k, 34) + 1
        Case Is = 3
          Cells(k, 35) = Cells(k, 35) + 1
        Case Is = 4
          Cells(k, 36) = Cells(k, 36) + 1
    End Select
```

そこで行動の**確率 P** と**報酬 R** との関係の間の概念としてアルファベット順の **P** と **R** の間にある **Q** を名前として採用したということらしいです。ほんとかな？！でも、それっぽい感じもします。ちょっと遊び心のあるネーミングですね。

β版が動き出したら、再び障壁の位置や数を変え、いろんな状況で最短距離を探索させてください。このプログラムは、いろいろな条件に対応して最短距離を探すというちょっと生命体のような動きをします。

実際の「**機械学習**」のテキストを見ると、まず数学的な背景が出てきて、よほど数式に慣れた人じゃないと読み込めないようにできています。このテキストのプログラムはすべて作者のオリジナルですが、そんな数学的な背景が無茶苦茶あって作っているわけではありません。

まずプログラミングしてみること、そして動かすこと、この哲学で作られています。これを忘れないようにしてください。後から必要なら数学の言葉に翻訳すればいいのです。

このプログラムが正しいことは、**図 Q-21** が理屈を超えて証明してくれています。

```
  Next k
End Sub
```

```
Private Sub CommandButton1_Click()
    マウスの冒険
End Sub
```

```
Private Sub CommandButton2_Click()
   Range(Cells(4, 33), Cells(84, 36)).Value = 0
End Sub
```

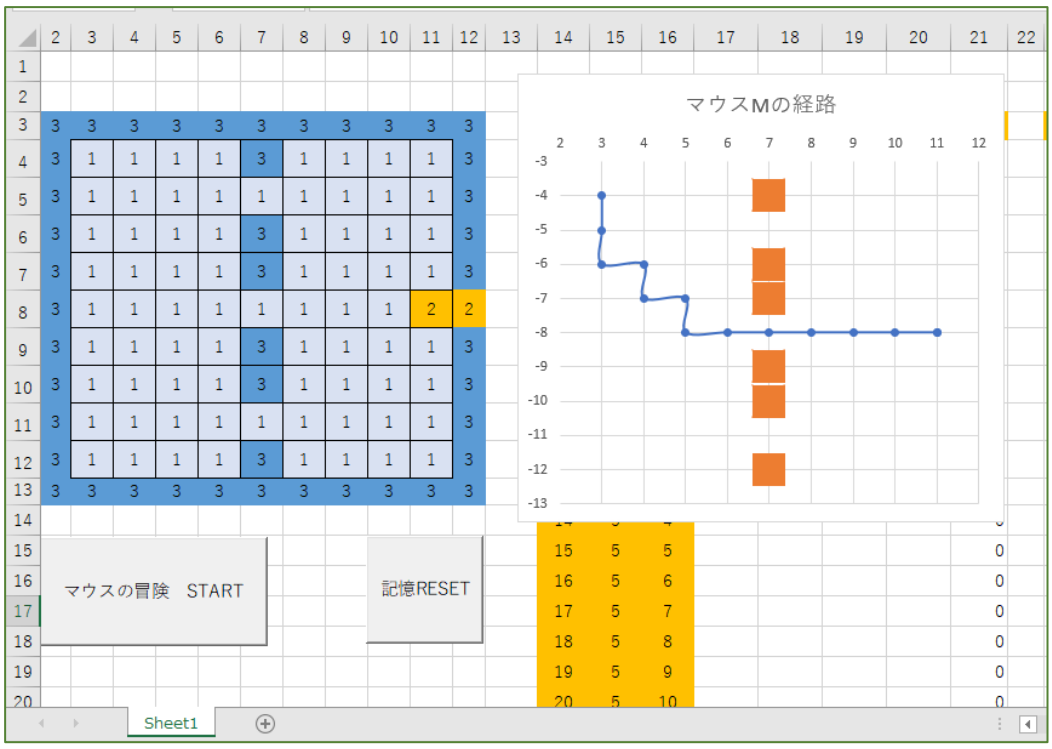


図 Q-21

課題Ⅷ　最短経路探索問題

マウスの冒険β版を利用して、障害物や実験エリアを工夫しオリジナルな最短経路探索問題を作成し実験せよ。

五目並べ競技会

さて、この講座でプログラマーの卵といわれる人材が育ったのか否かを世に問うために、「**五目並べ競技会**」を開催したいと思います。競技は、公開で行われ、コンピューター操作はその競技者のプログラムを預った第三者が行います。競技は参加者が二人以上集まったら開催するつもりです。

高校生にそんなことできないと思っている人もいるでしょう。実は、僕がある商業高校に勤めていたとき、パソコン部の顧問の先生と組んでこの「五目並べ競技会」を開催したことがあります。はじめ彼らの作るプログラムは、相手の石が 4 つ並んでいても、自信満々に全然違うところに打つという、お笑い芸人のボケをかましていました。相手になってもらった若い英語の先生は、はじめ笑いながら勝利を手にしていました。

ところが、改良が重ねられるにつれてその英語の先生から笑顔がなくなっていきました。

特に部長だった A 君のプログラムは強く、彼はとうとうそのプログラムを持って福岡の IT 企業の就職試験に臨んだのです。一発内定でした。あとで、高卒でその企業に入れることはあまりないと聞きました。当時は AI という概念も希薄でした。今にして思えば、A 君のプログラムを残しておけばよかったと後悔しています。実は、A 君は不慮の事故でいまこの世にはいません。

現在、プログラミングが一つのブームになっています。小学校からプログラミング教育が始まり、子供たちのプログラミングコンテストも盛んになっています。このブームの背景には、2000 年代半ばには AI 研究の主導権が完全にアメリカと中国に二分されるのでは、という国の危機感があります。つまり、政治的背景の強い、作られた人気なのが今の日本の AI ブームなのです。

中国で柯潔九段が AI に敗れたことは、中国共産党を震撼させることになりました。彼らは、アメリカがソビエト連邦（今は解体してロシア）に人工衛星で先を越された「スプートニクショック」に相当する衝撃を受けたと言われています。

現在中国は、2030 年までに AI の分野の主導権をとるという目標の元「中国脳計画」が始動し、何千人もの「機械学習」技術者が要請されているのです。現在のアメリカと中国とのファーウェイ（華為）や動画サイトの問題はその氷山の一角にすぎません。

しかし、人工知能の研究は、これまでアメリカを中心に人の知性に関わる大変面白い分野だと知って興味を持って人々はやってきました。研究者の中では、これまでの教育学、脳神経科学、機械学習、心理学が融合して新しい「**学習の科学**」が今、創造されたと受け取られています。この新しい学問を育てるのは若い君たちです。決して、国家の主導権争いの道具にしてはいけません。それは、かつての科学技術が原爆を生み出したような悲劇を生み出す可能性も持っています。「AI を誘導ミサイルに組み込み、AI を使って監視カメラで人々を追い、インターネットを検閲し、犯罪予測だけで人々の人権を蹂躙する」そんな未来を変えることができるのは君たちです。

だから、「**五目並べ競技会**」なのです。この呼びかけに賛同する人は、ぜひ、ボケかましの一手を見せるだけでもいいですから参加してください。