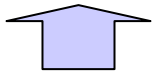
２．９　進化

“断続平衡（punctuated equilibrium）”



“共進化（co-evolution）”:　ある種の進化が相互に関連する他の種の進化に影響を及ぼす.

１．生態系：周期的境界条件，サイズ n の１次元格子
　　格子サイト（種族）の値 ＝ 乱数 ＝ 適合性の物指
　　＝ 進化の壁 ＝ 遺伝物質の量に関係
　　壁が高いほど種族は安定
２．最低値の格子サイトの位置をleastFitSites
　　最近接サイトの値を新しい乱数で置き換え
　　進化：突然変異，自然淘汰
　　与えられた時間ステップ，新しい適合レベル

```
In[1] := DarwinianEvolution[n_, t_ ] :=
          Module[{prebiotic, fitness, leastFitSites},
  初期条件:リストを空に
            leastFitSites = { };
            n 種族の初期生態系
            prebiotic = Table[Random[ ], {n}];
            無名関数の中で生態系の配置を表す
            fitness = Function[y, ReleaseHold[
            3つの隣接サイトの値が乱数で置きかえられる.
              ReplacePart[y, Hold[Random[ ]]
          適用される最低値のサイト，隣接サイト
              Join[# - 1/. 0->n,
                中心サイトの位置をleastFitSitesに置く
                  AppendTo[leastFitSites, #];
                  #,
                  # + 1/. (n+1) -> 1] & [Position[y, Min[y]]]

              ]]];                          適合性の値が最低の種族
```

無名関数 fitness を繰返し t 回，その生態系に適用

```
    Nest[fitness, prebiotic, t];
     Flatten[leastFitSites]
 ]
```

Hold 関数：　包括して，リストに代入される前に評価されない

２．１０　ランダム・ウォーク

物理学：　分子の輸送現象
生物学：　生物の移住のモデル化
経済学：　金融市場の動向のモデル化

［1次元］

```
In[1] := StepIncrements[n_ ] := Table[(-1)^Random[Integer, {n}]
In[2] := StepIncrements[10]
Out[2] = {-1, 1, -1, -1, -1, -1, 1, -1, 1, -1}
In[3] := FoldList[Plus, 0, %]
Out[3] = {0, -1, 0, -1, -2, -3, -4, -3, -4, -3, -4}
In[4] := Walk1D[n_ ] :=
              FoldList[Plus, 0, Table[(-1)^Random[Integer, {n}] ]
In[5] := Walk1D[10]
Out[5] = {0, 1, 2, 3, 2, 1, 2, 1, 2, 3, 2}
NestList[(# + (-1)^Random[Integer])&, 0, n]
```

［2次元］

```
In[6] := Walk2D[n_ ] := FoldList[Plus, {0, 0},
                                  {{0,1}, {1, 0}, {0, -1}, {-1, 0}}
                                [[Table[Random[Integer, {1, 4}], {n}] ]]
                              ]
In[7]:= Walk2D[10]
Out[7] = {{0, 0}, {1, 0}, {2, 0}, {2, -1}, {1, -1}, {1, -2},
            {1, -3}, {1, -2}, {1, -3}, {1, -2}, {1, -3}}
```

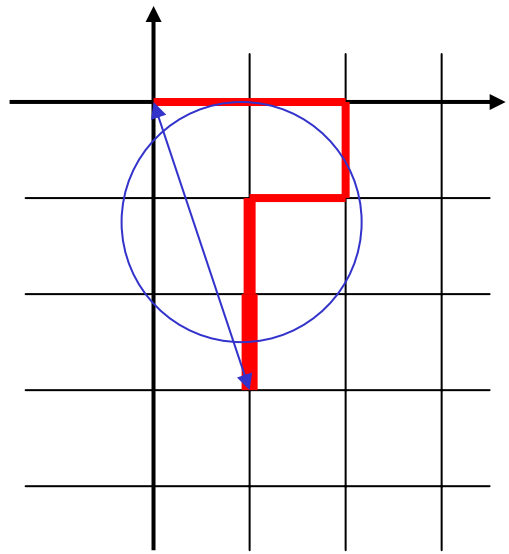
２次元格子ウォークの平均形状

平均２乗両端距離

（mean square end-to-end distance）

平均２乗回転半径

（mean square radius of gyration）



{0, 0} から出発して $\{x_f, y_f\}$ で終わる格子ウォークの２乗両端距離

Apply[Plus, {xf, yf}^2] ←——— $x_f^2 + y_f^2$

In[1] := SquareDistance[walk_List] := Apply[Plus, Last[walk]^2]

でもよい.

```
In[2] := MeanSquareDistance[n_Integer, m_Integer] :=
          Module[{walk2D},
            walk2D[s_ ] := FoldList[Plus, {0, 0},
                                    {{0,1}, {1, 0}, {0, -1}, {-1, 0}}
                                  [[Table[Random[Integer, {1, 4}], {s}] ]] ];
          n ステップのランダム・ウォークを m 通りで行う
            N[Sum[Apply[Plus, Last[walk2D[n]]^2], {m}/m]
          ]
```

平均2乗回転半径

```
In[3] := MeanSquareRadiusGyration[m_Integer, n_Integer] :=
          Module[{squareRadiusGyration},
            squareRadiusGyration[s_Integer] :=
              Module[{locs, cm, choices = {{1, 0}, {-1, 0},
                                           {0, 1}, {0, -1}}},
                locs = FoldList[Plus, {0, 0},
                          choices[[Table[Random[Integer, {1, 4}],
                                         {s}]]]];
                cm = N[Apply[Plus, locs]/(s + 1)];
                Apply[Plus,
                      Flatten[(Transpose[locs] – cm)^2]]/(s + 1)
                ];
            N[Sum[squareRadiusGyration[n], {m}]/m]]
```

位置のリスト

重心

リスト　$\{\{(x_0-x_{cm})^2, ---, (x_0-x_{cm})^2, (x_0-x_{cm})^2\}\{\{(y_0-y_{cm})^2, ---, \}\}$

2．11　自己回避ウォーク（self-avoiding walk, SAW）

一度通過した位置を避けて進む

高分子物理学：　共有化学結合で結びついた多数の分子からなる長い鎖状分子

ゼロ成分の強磁性体：　N → 0 でのN-ベクトル・モデル

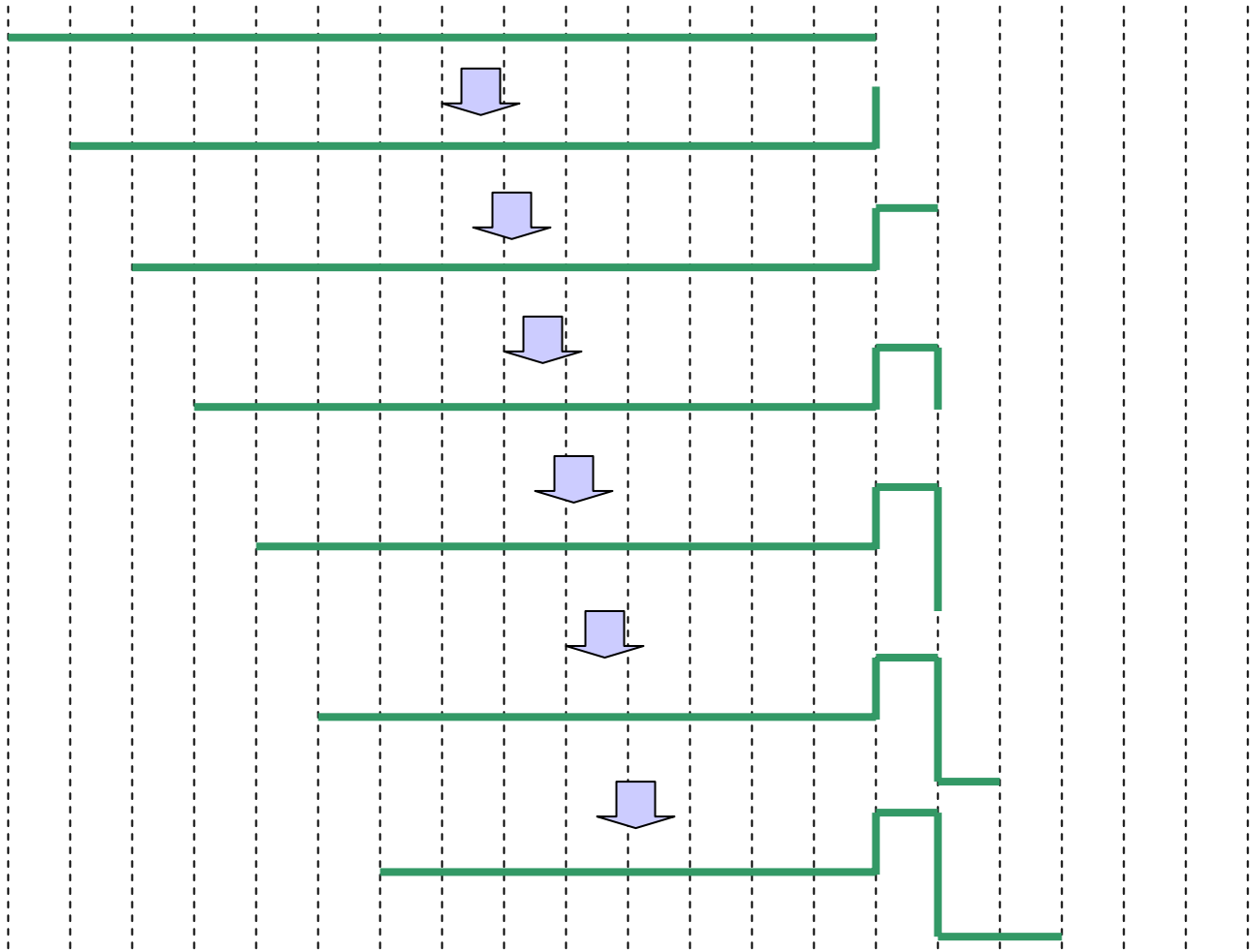
一般的な臨界現象

① スリザリング・スネーク・アルゴリズム（slithering snake, 滑るように歩く蛇）

② 枢軸変換（pivot）アルゴリズム

2．11．1　スリザリング・スネーク・アルゴリズム

初期



```
In[1] := SlitheringSAW[n_, m_ ] := Module[{squaredist, snake},
```
初期両端2乗距離
```
        squaredist = n^2
        snake = (Module[{newpt, path,
                                choice = {{1, 0}, {-1, 0},
                                          {0, 1}, {0, -1}}}
```
1ステップの増加．新しいステップ位置
```
          newpt = Last[#] + choice[[Random[Integer, {1, 4}]]];
```
newpt が（最初のステップを除いた）任意のステップと一致するか？
```
          If [MemberQ[Rest[#], newpt]
```
重複があれば，# を逆転　→　path
```
             path = Reverse[#],
```
重複がなければ，# の最初のステップを取り除き，newpt を加える．
```
             path = Join[Rest[#], {newpt}]];
```

新しいSAWの配置 path の両端2乗距離を squaredist に加える.

```
squaredist + =
        Apply[Plus, (First[path] – Last[path])^2];
path])& ;
```

初期配置　　m 回繰り返す

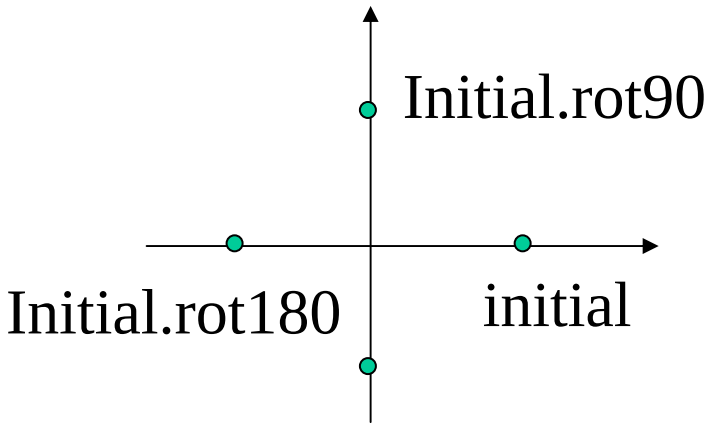
```
Nest[snake, Table[{i, 0}, {i, 0, n}], m];
```

平均2乗両端距離

```
N[squaredist/(m + 1)]
]
```

２．１１．２　枢軸変換アルゴリズム

正準集団中に d 次元のSAWを生成
効率のよい動的アルゴリズム
d 次元格子上
$2^d d!$ の対称（回転と反射）操作から1つをランダムに選択
（2次元の場合）8つの対称操作のうち3つ（±90°, 180°）を
　　考えるだけで充分



3つの2次元回転行列

```
In[1] := rot90 = {{0, 1}, {-1, 0}}
In[1] := rot180 = {{-1, 0}, {0, -1}}
In[1] := rot270 = {{0, -1}, {1, 0}}
```

初期位置

```
In[4] := initial = {{1, 0}}
```

回転は行列の掛け算

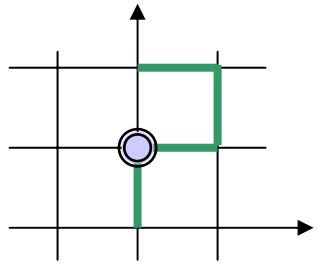
```
In[5] := {initial.rot270, initial.rot90, initial.rot180}
Out[5] = {{{0, -1}}, {{0, 1}}, {{-1, 0}}}
```

In[6] := chain ={{0, 0}, {0, 1}, {1, 1}, {1, 2}, {0, 2}}

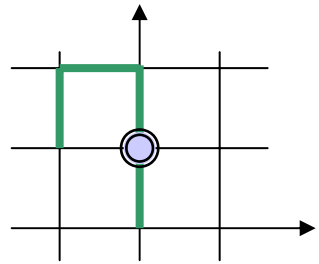
枢軸点の位置 pivot

In[7] := pivot = chain[[2]]

Out[7] = {0, 1}



chain



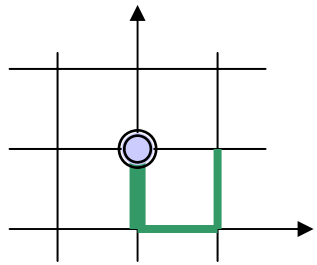
Plus90RotSAW

枢軸点から見た i 番目のステップの chainの配置の配位→ relative

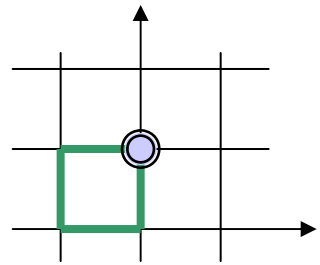
relative = chain[[i]] – pivot

pivot まわりの回転

move = pivot + relative.roti



Minus90RotSAW



Plus180RotSAW

In[8] := Plus90RotSAW =
{chain[[1]],
chain[[2]],
chain[[2]] + (chain[[3]] - chain[[2]]).rot90,
chain[[2]] + (chain[[4]] - chain[[2]]).rot90,
chain[[2]] + (chain[[5]] - chain[[2]]).rot90}

```
In[1] := Pivot2DSAW[n_Integer, m_Integer] :=
          Module[{squaredist, twistAndShout},
```
２乗両端距離
```
            squaredist = n^2 ;
            twistAndShout =
              (Module[{ball, fixsec, movesec, rotchoice,
                       newsec, newconfig,
```
回転行列
```
                       rot = {{{0, -1}, {1, 0}},
                              {{0, 1}, {-1, 0}},
                              {{-1, 0}, {0, -1}}},
```
← rot270
← rot90
← rot180

枢軸点をランダムに選ぶ
```
                 ball = Random[Integer, {1, n-1}] ;
```
ball を用いて，# を2つの部分に分ける

固定部分＝始めから枢軸点まで
```
                 fixsec = Take[#, ball] ;
```
回転部分＝枢軸点から最後まで
```
                 movesec = Take[#, ball – (n+1)] ;
```

3つの回転行列(-90°，90°，180°の回転)の1つをrotchoiceでランダムに選ぶ

```
rotchoice = rot[[Random[Integer, {1, 3}]]] ;
```

movesec の # を回転する.

```
newsec = Map[Function[y, #[[ball]] +
                                (y - #[[ball]]).rotchoice], movesec] ;
```

newsec と fixsec が一致していないかを調べる

```
If [Intersection[newsec, fixsec] = = { },
```

もし重なるステップがなければ，fixsec と newsec をつなぎ，新しいSAW配置を作り，newconfig と名づける．２乗両端距離を計算し，squaredist に加える.

```
newconfig = Join[fixsec, newsec] ;
squaredist += Apply[Plus, Last[newconfig]^2] ;
newconfig,
```

重なるステップがあれば，前のSAW

```
        squaredist += Apply[Plus, Last[#]^2] ;
        #]])& ;
```

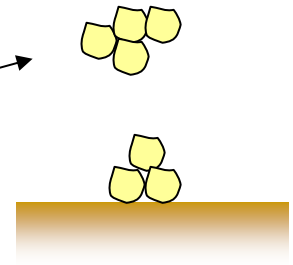
初期SAW配置

```
Nest[twistAndShout, Table[{j, 0}, {j, 0, n}], m] ;
N[sqauredist/(m + 1)]]
```

2．12　付着（accretion）

粒子が他の粒子に衝突，“合体する”まで動き回る過程
⇒　クラスター

凝集（aggregation）：　クラスターが自由に浮遊
堆積（deposition）：　クラスターが地面に着く



“拡散律則凝集（diffusion-limited aggregation, DLA）”
粒子は他の浮遊クラスターに接触するまでブラウン運動

- 多様な自然現象の基礎
  結晶化，コロイドと重合体の縮体，煤の形成，絶縁破壊

“弾道堆積（ballistic deposition）”
固体の基盤，表面から1つの粒子が出発　→
鉛直下方の軌道　→　表面に到着

- 材料開発
  薄膜形成，気相成長，スパッタリング，分子線エピタキシ

2．12．1　DLA

DLAプログラム

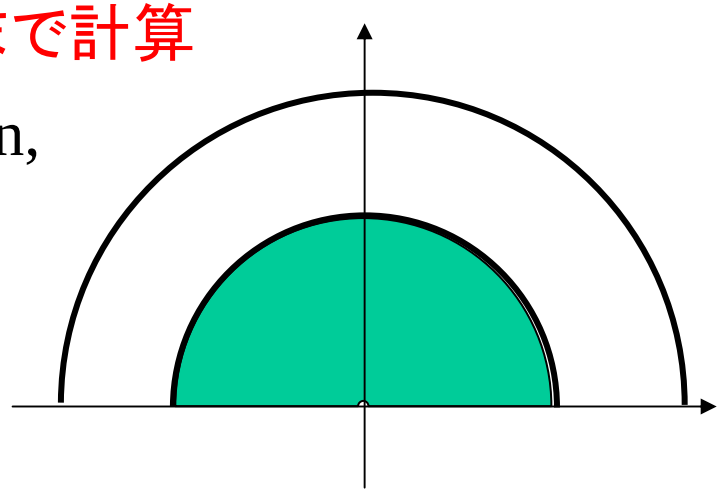
```
In[1] := DLA[s_Integer, n_Integer] :=
          Module[{loc, rad, particleCount = 0,
                  stepChoices = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}},
```

“核（シード）”となるサイトを含んだリスト

```
          occupiedSites = {{0, 0}};
```

occupiedSites の長さが n になるまで計算

```
          While[Length[occupiedSites] < n,
              + +particleCount ;
```



1つの粒子がランダム・ウォーク

を開始する円の半径

```
          rad = Max[Abs[occupiedSites]] + s ;
```

ランダム・ウォーク後の位置

```
loc = FixedPoint[
```

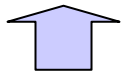
次の新たなステップ

```
(# + stepChoices[[Random[Integer, {1, 4}]]] ) &,
 Round[rad{Cos[#], Sin[#]}] & [Random[Real,
                                      {0, N[2Pi]}]],
 SameTest -> (Apply[Plus, #2^2] > (rad + s)^2 ||
                Intersection[occupiedSites,
                  Map[Function[y, y + #2],
                    stepChoices]] ! = { } &)
  ] ;
 If [Apply[Plus, loc^2] < rad^2,
  occupiedSites = Join[occupiedSites, {loc}]]
] ;
Print[“The number of particles released was”,
     particleCount] ;
occupiedSites
```

```
In[2] := ShowDLA[sites_, opts_ _ _] :=
            Module[{structure, s = Length[sites]},
                structure = { };
                Map[(AppendTo[structure,
                                    {Hue[.99 #/s],
                                     Rectangle[sites[[#]] – {0.5, 0.5},
                                          sites[[#]] + {0.5, 0.5}],
                                     RGBColor[1, 1, 1],
                                     Text[#, sites[[#]]]}] ) &,
                      Range[s]] ;
               Show[Graphics[structure],
                    opts,
                    Axes -> None,
                    AspectRatio -> Automatic,
                    PlotRange -> All
                ]
```

2．12．2　DLAのフラクタル次元

DLA成長　→　凝集体，クラスターの形が不規則，希薄



∵　遮蔽効果：　ふらつく粒子がDLAの剥き出しの外面の一部に接する可能性が増す

- フラクター次元　＝　DLAの密集度の尺度
- クラスターの回転半径のクラスター・サイズ依存性
- DLAのサイズ　↑　⇒　フラクター次元　↓

フラクタル次元のプログラム

```
In[3] := FractalDimension[occupiedSites_List] :=
          Module[{occSiteDensity, fractalDataList, fractaldim},
            occSiteDensity[t_Integer] :=
                N[Count[occupiedSites, {x_?(Abs[#] <= t &),
                                y_? (Abs[#] <= t &)}]]/(2t + 1)^2 ;
```

順番になった対の fractalDataList

```
            fractalDataList = Table[{2s, occSiteDensity[s]},
                                    {s, Max[Abs[occupiedSites]]}] ;
```

クラスター中のサイトのリスト

DLA構造のフラクタル次元

```
            fractaldim = Fit[N[Log[fractalDataList]], {1, x}, x] ;
            Print["The fractal dimension of the DLA is      ",
                  Coefficient[fractaldim, x]] ;
          ]
```

２．１２．３　弾道堆積

弾道堆積プログラム

```
In[4] := MolecularDeposition[n_, t_] :=
           Module[{init, newLat, nnColumns,
                   depositRow, emitLayer},
初期格子　0 or 1
      init = Transpose[Table[{0, 1}, {n}]] ;
粒子が落ちる場所
      newLat =
格子の列
        (depositColumn = Random[Integer, {1, n}] ;
選択された列と2つの最近接の列
         nnColumns =
          Transpose[#][[{depositColumn – 1/. 0 -> n,
                         depositColumn,
                         depositColumn + 1/. (n+1) -> 1}]]&;
```

落ちてくる粒子が止まる格子の行

```
            depositRow =
                Min[Flatten[Map[Function[y,
                    First[Position[y, 1]]],
                    nnColumns[#]]] – {0, 1, 0}]& ;

            ReplacePart[#, 1, {depositRow[#],
                        depositColumn}]])& ;

emitLayer = If [#[[1]]  ! = Table[0, {n}],
                    Prepend[#, Table[0, {n}]], #]& ;

Nest[emitLayer[newLat[#]]&, init, t]
]
```

２．１３　伝播現象

伝播：　物体を接する領域を取りこむことで範囲を広げながら拡張する過程

接触成長（kinetic growth, KG）モデル：　自然現象の過程の様々な多様性

腫瘍の成長，伝染病の伝播，ゲル化，噂の広まり，
多孔質媒体中の液体の流れ

KGモデルの原点：　イーデン・モデル，２次元正方格子

１つの初期のクラスター・リスト（シード・サイト（初期配置サイト））
⇒　ノイマン近傍の１つのサイトを選ぶ　⇒　クラスター・リストに
⇒　この過程をサイズ n まで続ける

①単一パーコレーション（single percolation）クラスター・モデル
②侵入型パーコレーション（invasion percolation）モデル

2．13．1　単一パーコレーション・クラスター・モデル

病気の伝染を記述

ランダムに選ばれた周辺サイト：　クラスターにつながる確率 p

選択されたサイト：　クラスター中に存否に係わらず周辺リストから取り除かれる

プログラム

クラスターの長さ　　確率

```
In[1] := Epidemic[n_, p_] :=                    ノイマン近傍（初期リスト）

          Module[{choices = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}},
                          reject, pickAndChoose, select, newpers},
```

空のリスト

```
           reject = { }
```

1つのサイトを選ぶ，　{ #[[1]]: クラスターリスト，
　　#[[2]]: 周辺サイトのリスト

```
           pickAndChoose :=
             (select = #[[2, Random[Integer, {1, Length[#[[2]]]}]]];
```

乱数≦p

```
                  If [Random[ ] <= p,
```

新しいperimeterリスト

```
            newPers =
```

selectがperimeter listから取り除かれ　⇒
“#[[1]], #[[2]], reject ”にないselectの最近接サイト　→　newPers

```
        Complement[Union{Map{Function{y, y+select],
                                choices], #[[2]], {select}, #[[1]], reject];
```

{select}が#[[1]]に置かれ　+　新しい周辺リストの番号の組

```
        {Join[#[[1]], {select}], newPers},
```

乱数＞p,　selectはrejectに

```
        reject = Join[reject, {select}];
```

selectはperimeter listから取り除かれ，変わっていないcluster list #[[1]]と新しいperimeter list #[[2]]からなる番号のついた組

```
        {#[[1]], Complement[#[[2]], {select}]}])&;
```

perimeterリスト#2[[2]]が空か，clusterリストが n になるまで繰り返し計算

seed サイト　　近傍サイト

```
FixedPoint[pickAndChoose, {{{0, 0}}, choices}, n,
                SameTest -> (#2[[2]] = = { } | |
                                Length[#2[[1]] = = n &)][[1]]
    ]
```

２．１３．２　侵入型パーコレーション・モデル

多孔質媒体中の流体の流れ
（例）　第３紀層石油回収（tertiary oil recovery）

周辺リストの各サイトは対応した乱数を持つ
最も抵抗の低い道筋を辿って伝播

プログラム

```
In[1] := Invasion[n_Integer] :=
          Module[{pickAndChoose, nn, newnn, newpers, newPerLis,
                  choices = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}},
#[[1]]: cluster list, #[[2]]: perimeter list
                pickAndChoose :=
乱数最低値のperimeter site ＝　#の第２部分を順番に分類　→
                                  分類されたリストの第１部分の第２要素
                 (newcluSite = Sort[#[[2]][[1, 2]];
nn = newcluSite の最近接サイト
                  nn = Map[Function[y, y + newcluSite], choices];
     nn = Map[(# + newcluSite}&, {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}]
cluster list(#[[1]])かperimeter list(#[[2]])のどちらかにあるサイトをnn
から取り除く
                  newnn = Complement[nn, #[[1]],
Transpose[#[[2]][[2]];
```

残りのサイトは乱数と組にする

```
newpers = Transpose[{Table[Random[ ],
                          {Length[newnn]}], newnn}];
```

newpersをperimeterに加え，対応した乱数を持つnewcluSiteをその周辺リストから取り除く

```
newPerLis = Join[DeleteCases[#[[2]],
                          { _ , newcluSite}], newpers]
```

新しいcluster listと新しいperimeter listから成る順序立った組を作る.

```
{Join[#[[1]], {newcluSite}], newPerLis})&;
```

seedサイトとその周辺サイトからなる順序立った組

```
Nest[pickAndChoose,
    {{{0, 0}}, Transpose[{Table[Random[ ], {4}],
     choices}]}, n][[1]]
]
```