データマイニング入門（２）

Symbolic Systems *http://www.symbolics.jp/lecturenotes.html*

matsuda@symbolics.jp (2010.2.24)

データマイニングについて

データマイニングの本来の意味からすると既存データから「一定のルール」を見つけ、それを未知のデータに対し適用することが求められるが、しかし、データ間の傾向を視覚的にみせる「見せ方」やある種の「統計的処理」を施すことにより一定の情報予測が可能であるならば，これは広義の意味のデータマイニングと見なすことができる.

その意味で，すでに既知となっているデータ解析手法はデータマイニングにおいても有効なツールとなり得る.

ただし，一般的にデータマイニングという時，対象となるデータに欠損があったり，大量であったり，あるいは，従来のデータ解析手法では結果品質が劣る場合などに利用されることが想定される.

いずれにせよデータをなんらかの基準によって**グループ（クラスタ）に分割**することは，データマイニングの基本である．分割して終わりの場合もあれば，分割したもの同士を比較したり，あるいは分割したものからさらに別の分類を見つけたりすることもある.

クラスタ分割には通常はなんらかの基準が必要となる．一般的にはこの基準がわからないケースも多い．しかし，ここではある種の基準を常に仮定している.

SQLデータベース検索結果に対するクラスタ分類と特徴素の計算

データベースから抽出したレコード群を分類し，各クラスタの特徴を計算．分類にはレコード内のテキストパターンを使用.

ref.
Gazing at New Paradigms in Data Mining
Experiences, Results, and Questions
Ariel Sepúlveda
Hewlett Packard Corporation
@Wolfram Technology Conference 2005

■ 初期化

In[133]:=
```
Needs["Combinatorica`"];
Needs["MultivariateStatistics`"];
```

■ SQLSelect検査の結果サンプルデータの用意

SQLSelectの検索結果に対するランダムなレコードを１００００個用意．各レコードのフィールド数は６．そのうち５つはテキストデータ（その種類は３種類）で残り一つは数値データ.

テキストデータの番号は１から３(=NoOfClassesPerVariable) までの離散一様分布

In[135]:=
```
NoOfVariables = 5;
NoOfClassesPerVariable = 3;
TotalRows = 10 000;
data = Join[#, {Random[]}] & /@
    Map["Class" <> ToString[#] &, RandomInteger[DiscreteUniformDistribution[
       {1, NoOfClassesPerVariable}], {TotalRows, NoOfVariables}], {2}];
Take[data, 5] // TableForm
```

Out[139]//TableForm=

Class1	Class2	Class1	Class1	Class1	0.898828
Class1	Class3	Class1	Class2	Class3	0.920331
Class1	Class1	Class2	Class2	Class1	0.858976
Class1	Class2	Class1	Class3	Class1	0.829739
Class2	Class2	Class3	Class3	Class1	0.637911

テキストデータは離散一様分布を仮定．当然，実際のデータはこの通りの分布ではないかもしれず，その場合は理想的にはその分布を発見することもデータマイニングの仕事．

In[140]:=
```
Mean[DiscreteUniformDistribution[{1, NoOfClassesPerVariable}]]
```

Out[140]= 2

In[141]:=
```
RandomInteger[DiscreteUniformDistribution[{1, NoOfClassesPerVariable}], {3, 5}] //
 TableForm
```

Out[141]//TableForm=

2	3	3	1	1
2	3	2	1	2
3	3	3	3	2

■ ユニークなレコードサブリストに対し，それと一致する値の平均を計算．Excelでよくやる計算．

In[142]:=
```
ClusterByVars = Range[NoOfVariables]
```

Out[142]= {1, 2, 3, 4, 5}

In[143]:=
```
ClustersIDs = data[[All, ClusterByVars]] // Intersection;
ClustersIDs // Length
Take[ClustersIDs, 5] // TableForm
```

Out[144]= 243

Out[145]//TableForm=

Class1	Class1	Class1	Class1	Class1
Class1	Class1	Class1	Class1	Class2
Class1	Class1	Class1	Class1	Class3
Class1	Class1	Class1	Class2	Class1
Class1	Class1	Class1	Class2	Class2

下記のような使い方．Excel風に解釈すると第２引数の要素番号に対応した列すべてを取り出す操作に相当．

In[146]:=
```
data[[1 ;; 10, ClusterByVars]]
```

Out[146]= {{Class1, Class2, Class1, Class1, Class1}, {Class1, Class3, Class1, Class2, Class3},
 {Class1, Class1, Class2, Class2, Class1}, {Class1, Class2, Class1, Class3, Class1},
 {Class2, Class2, Class3, Class3, Class1}, {Class2, Class1, Class2, Class1, Class2},
 {Class1, Class1, Class3, Class1, Class3}, {Class3, Class1, Class2, Class3, Class2},
 {Class3, Class1, Class1, Class1, Class3}, {Class2, Class3, Class2, Class1, Class3}}

In[147]:=
```
data[[1 ;; 100, ClusterByVars]] // Intersection // Length
```

Out[147]= 87

Length[ClusterIDs]個のクラスタを構成し，それぞれのクラスタ内レコードに関し数値フィールドの平均値を計算.

In[148]:=
```
MeanResponseForCluster =
  Table[{ClustersIDs[[i]], Last /@ (Select[data, #[[ClusterByVars]] === ClustersIDs[[i]] &]) //
      Mean // N} // Flatten, {i, 1, Length[ClustersIDs]}];
Take[MeanResponseForCluster, 5] // TableForm
```

Out[149]//TableForm=
```
Class1 Class1 Class1 Class1 Class1 0.570936
Class1 Class1 Class1 Class1 Class2 0.564517
Class1 Class1 Class1 Class1 Class3 0.521281
Class1 Class1 Class1 Class2 Class1 0.551492
Class1 Class1 Class1 Class2 Class2 0.44874
```

上記作業をひとまとめ関数として定義

In[150]:=
```
CalcMeanResponseForClustersInVars101[data_, ClusterByVars_] := Module[{},
  ClustersIDs = data[[All, ClusterByVars]] // Intersection;
  MeanResponseForCluster =
   Table[{ClustersIDs[[i]], Last /@ (Select[data, #[[ClusterByVars]] === ClustersIDs[[i]] &]) //
       Mean // N} // Flatten, {i, 1, Length[ClustersIDs]}]
 ]
```

In[151]:=
```
ans = CalcMeanResponseForClustersInVars101[data, ClusterByVars] // Timing;
t101 = ans[[1]]; ans[[1]]
Take[ans[[2]], 5] // TableForm
```

Out[152]= 16.676

Out[153]//TableForm=
```
Class1 Class1 Class1 Class1 Class1 0.570936
Class1 Class1 Class1 Class1 Class2 0.564517
Class1 Class1 Class1 Class1 Class3 0.521281
Class1 Class1 Class1 Class2 Class1 0.551492
Class1 Class1 Class1 Class2 Class2 0.44874
```

■ 上記作業をMathematicaらしい方法で速度改善

上記方法は簡単だけどLength[CluserID]に比例した時間がかかる．もっと効率的な方法はないか．つまりExcel風ではなく，Mathematicaらしい方法は？

In[154]:=
```
Classify[s_, f_: Identity] :=
  Map[Last, Split[Sort[{f[#], #} & /@ s], SameQ[First[#1], First[#2]] &], {2}];

testdata = {{1, P, 3}, {1, Q, 2}, {1, R, 3}, {1, S, 2}, {1, R, 3}, {1, S, 2}};
gg[s_] := s[[1 ;; 2]]
Classify[testdata, gg]
```

Out[157]= {{{1, P, 3}}, {{1, Q, 2}}, {{1, R, 3}, {1, R, 3}}, {{1, S, 2}, {1, S, 2}}}

上記方法は下記の例をベースに考えられている．つまり関数Classifyはレコード内の部分フィールドのパターンによって分類する機能を提供.

In[158]:=
```
Sort[{gg[#], #} & /@ testdata]
```

Out[158]=
```
{{{1, P}, {1, P, 3}}, {{1, Q}, {1, Q, 2}}, {{1, R}, {1, R, 3}},
 {{1, R}, {1, R, 3}}, {{1, S}, {1, S, 2}}, {{1, S}, {1, S, 2}}}
```

In[159]:=
```
(*#〚1,ClusterByVars2〛の１はClassify計算によって得られる同一要素リストの先頭を取り出すため *)
CalcMeanResponseForClustersInVarsMath[data2_, ClusterByVars2_] := Module[{},
  Flatten[{#〚1, ClusterByVars2〛, Mean[Last /@ #]}] & /@
   Classify[data2, data2〚ClusterByVars2〛]
 ]
```

In[160]:=
```
ans = CalcMeanResponseForClustersInVarsMath[data, ClusterByVars] // Timing;
tMath = ans〚1〛; ans〚1〛
Print["tMath fraction of 101: ", tMath / t101 * 100, "%"]
Take[ans〚2〛, 5] // TableForm
```

Out[161]= 0.592

```
tMath fraction of 101: 3.55001%
```

Out[163]//TableForm=

Class1	Class1	Class1	Class1	Class1	0.0116393
Class1	Class1	Class1	Class1	Class1	0.015319
Class1	Class1	Class1	Class1	Class1	0.0529342
Class1	Class1	Class1	Class1	Class1	0.0604372
Class1	Class1	Class1	Class1	Class1	0.116015

関数CalcMeanResponseForClustersInVarsMathは次の働きを前提にMap関数をうまく使うことで，Length[CluserID]に比例した前段の計算が数十倍速くなる.

In[164]:=
```
gg[s_] := s〚ClusterByVars〛;
Sort[{gg[#], #} & /@ data] // Take[#, 5] &
```

Out[165]=
```
{{{Class1, Class1, Class1, Class1, Class1},
   {Class1, Class1, Class1, Class1, Class1, 0.0116393}},
  {{Class1, Class1, Class1, Class1, Class1},
   {Class1, Class1, Class1, Class1, Class1, 0.015319}},
  {{Class1, Class1, Class1, Class1, Class1},
   {Class1, Class1, Class1, Class1, Class1, 0.0529342}},
  {{Class1, Class1, Class1, Class1, Class1},
   {Class1, Class1, Class1, Class1, Class1, 0.0604372}},
  {{Class1, Class1, Class1, Class1, Class1},
   {Class1, Class1, Class1, Class1, Class1, 0.116015}}}
```

■ さらに速くするために

In[166]:=
```
SplitListByElementsOfCol[TheList_, TheCol_] := Module[{pos, sdata},
   TheNewList = TheList〚Ordering[TheList〚All, TheCol〛]〛;
   pos = FoldList[Plus, 0, Length /@ ((TheNewList〚All, TheCol〛) // Split)];
   Take[TheNewList, #] & /@ (# + {1, 0} & /@ (Partition[pos, 2, 1]))
  ];

CalcMeanResponseForClustersInVarsASC[data3_, ClusterByVars_] := Module[{},
  Flatten[{#〚1, ClusterByVars〛, Mean[Last /@ #]}] & /@
   SplitListByElementsOfCol[data3, ClusterByVars]
 ]
```

In[168]:=
```
ans = CalcMeanResponseForClustersInVarsASC[data, ClusterByVars] // Timing;
tASC = ans[[1]]; ans[[1]]
Print["tASC fraction of tMath: ", tASC / tMath * 100, "%"]
Take[ans[[2]], 5] // TableForm
```

Out[169]= 0.187

```
tASC fraction of tMath: 31.5878%
```

Out[171]//TableForm=

Class1	Class1	Class1	Class1	Class1	0.570936
Class1	Class1	Class1	Class1	Class2	0.564517
Class1	Class1	Class1	Class1	Class3	0.521281
Class1	Class1	Class1	Class2	Class1	0.551492
Class1	Class1	Class1	Class2	Class2	0.44874

Sort[]関数を使った場合の各要素が元データにおける位置を与える．たとえば次の例では，aは一番先頭にくるが元データにおいては２番目に位置してるので２が最初に来る.

In[172]:=
```
Ordering[{c, a, b}]
```

Out[172]= {2, 3, 1}

In[173]:=
```
Ordering[data[[All, ClusterByVars]]] // Take[#, 50] &
```

Out[173]= {14, 70, 122, 246, 265, 459, 661, 752, 863, 1301, 1459, 1473, 1531, 2517,
2566, 3013, 3267, 3350, 3390, 3471, 3634, 3681, 3729, 3819, 4073, 4126,
4226, 4352, 4489, 4885, 5098, 5396, 5516, 5591, 5684, 5686, 5728, 5825,
5857, 5937, 6298, 6333, 6374, 6487, 6682, 6735, 7195, 7592, 7628, 7678}

次の例でもわかるように結局TheList[[Ordering[TheList[[All, TheCol]]]]はSort[data]を行ったのと同じ効果を持つ.

In[174]:=
```
{c, a, b}[[{2, 3, 1}]]
```

Out[174]= {a, b, c}

In[175]:=
```
With[{TheList = data, TheCol = ClusterByVars},
 TheNewList = TheList[[Ordering[TheList[[All, TheCol]]]]];
 pos = FoldList[Plus, 0, Length /@ ((TheNewList[[All, TheCol]]) // Split)]]
```

Out[175]= {0, 60, 98, 135, 178, 224, 261, 306, 331, 374, 413, 453, 491, 532, 579, 621, 664, 687, 732,
772, 824, 864, 898, 948, 1001, 1048, 1097, 1141, 1183, 1220, 1256, 1302, 1340, 1370,
1411, 1462, 1502, 1546, 1579, 1620, 1664, 1704, 1737, 1781, 1822, 1866, 1912, 1956,
2000, 2031, 2061, 2106, 2145, 2188, 2224, 2261, 2302, 2345, 2394, 2433, 2473, 2506,
2567, 2601, 2643, 2675, 2715, 2764, 2817, 2867, 2909, 2947, 2992, 3029, 3072, 3113,
3143, 3190, 3224, 3260, 3297, 3327, 3364, 3393, 3436, 3485, 3519, 3555, 3596, 3647,
3696, 3750, 3785, 3816, 3852, 3889, 3939, 3979, 4020, 4062, 4100, 4144, 4184, 4244,
4284, 4326, 4360, 4410, 4461, 4497, 4552, 4591, 4635, 4686, 4722, 4763, 4806, 4840,
4880, 4925, 4976, 5020, 5057, 5108, 5142, 5188, 5226, 5268, 5299, 5335, 5367, 5410,
5449, 5487, 5535, 5578, 5618, 5653, 5702, 5746, 5788, 5828, 5873, 5926, 5959, 6015,
6056, 6110, 6154, 6191, 6230, 6270, 6316, 6367, 6392, 6443, 6478, 6518, 6563, 6607,
6655, 6692, 6727, 6769, 6806, 6849, 6886, 6920, 6979, 7015, 7052, 7083, 7121, 7165,
7214, 7250, 7282, 7320, 7348, 7389, 7429, 7469, 7512, 7557, 7594, 7641, 7687, 7731,
7769, 7813, 7861, 7895, 7925, 7969, 8006, 8054, 8097, 8144, 8173, 8201, 8231, 8267,
8313, 8367, 8414, 8452, 8496, 8537, 8578, 8615, 8675, 8716, 8756, 8804, 8853, 8882,
8919, 8965, 9004, 9043, 9077, 9121, 9164, 9214, 9249, 9283, 9315, 9347, 9399, 9452,
9495, 9539, 9584, 9625, 9665, 9704, 9760, 9801, 9831, 9870, 9902, 9933, 9963, 10000}

In[176]:=
```
Partition[pos, 2, 1] // Take[#, 10] &
```

Out[176]= {{0, 60}, {60, 98}, {98, 135}, {135, 178}, {178, 224},
{224, 261}, {261, 306}, {306, 331}, {331, 374}, {374, 413}}

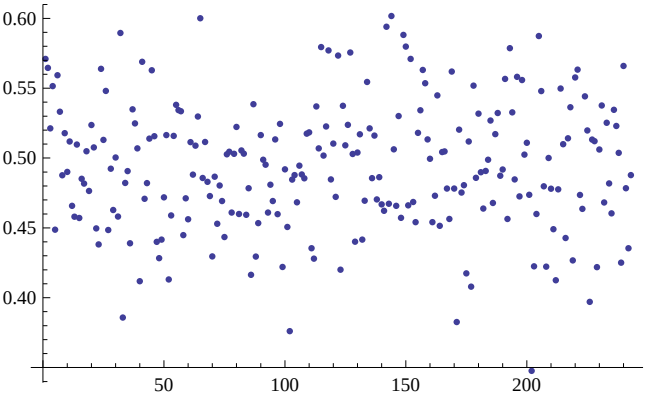
In[177]:=
```
(# + {1, 0} & /@ Partition[pos, 2, 1]) // Take[#, 10] &
```

Out[177]=
```
{{1, 60}, {61, 98}, {99, 135}, {136, 178}, {179, 224},
 {225, 261}, {262, 306}, {307, 331}, {332, 374}, {375, 413}}
```

仮にクラスタごとの平均値に関し散布図を表示してみたが元が一様分布なので当たり前であるが、特別何か情報が得られるわけではない。

In[178]:=
```
Last /@ MeanResponseForCluster // ListPlot
```

Out[178]=



■ 速度差比較

In[179]:=
```
NoOfVariables = 5;
NoOfClassesPerVariable = 3;
TotalRows = 10 000;
data = Join[#, {Random[]}] & /@
    Map["Class" <> ToString[#] &, RandomInteger[DiscreteUniformDistribution[
       {1, NoOfClassesPerVariable}], {TotalRows, NoOfVariables}], {2}];
```

In[183]:=
```
ClustersIDs = data[[All, ClusterByVars]] // Intersection;
```

In[184]:=
```
d = Table[
     Select[data, #[[ClusterByVars]] === ClustersIDs[[i]] &], {i, 1, Length[ClustersIDs]}]; //
 Timing
```

Out[184]=
```
{17.94, Null}
```

In[185]:=
```
Classify[s_, f_: Identity] :=
  Map[Last, Split[Sort[{f[#], #} & /@ s], SameQ[First[#1], First[#2]] &], {2}];
Classify[data, data[[ClusterByVars]]]; // Timing
```

Out[186]=
```
{0.297, Null}
```

In[187]:=
```
SplitListByElementsOfCol[TheList_, TheCol_] := Module[{pos, sdata},
    TheNewList = TheList[[Ordering[TheList[[All, TheCol]]]]];
    pos = FoldList[Plus, 0, Length /@ ((TheNewList[[All, TheCol]]) // Split)];
    Take[TheNewList, #] & /@ (# + {1, 0} & /@ (Partition[pos, 2, 1]))
   ];
d = SplitListByElementsOfCol[data, ClusterByVars]; // Timing
```

Out[188]=
```
{0.156, Null}
```

Cross-Sell

１０(possibleitems)種類の部品のうち５(itemsperorder)種類を使ってできる商品がある．この商品の購入履歴から，使用部品とクロスする部品のリストを見つける．

使う部品の種類は整数一様乱数を使用．以下の例では注文件数は１１件．ここでいうCross-Sellは注文ラインから関連部品の組み合わせ（たとえば部品Aは部品Bとよく使われる）の傾向を調べる意味で使われる．

In[189]:=
```
itemsperorder = 5;
possibleitems = 10;
byorderdata = Table[
   {RandomInteger[DiscreteUniformDistribution[{1, possibleitems}], {itemsperorder}],
     Table[i, {itemsperorder}]} // Transpose, {i, 1, 11}];
```

In[192]:=
```
byorderdata // TableForm
```

Out[192]//TableForm=

1	7	5	6	5
1	1	1	1	1
6	6	7	6	7
2	2	2	2	2
8	4	4	6	4
3	3	3	3	3
1	8	7	3	10
4	4	4	4	4
10	1	10	7	2
5	5	5	5	5
3	5	7	5	9
6	6	6	6	6
1	7	8	6	2
7	7	7	7	7
6	7	7	6	5
8	8	8	8	8
9	10	9	6	8
9	9	9	9	9
6	1	9	10	5
10	10	10	10	10
10	1	2	9	3
11	11	11	11	11

{部品番号，注文番号}のリスト

In[193]:=
```
Take[byorderdata[[1]], All]
Take[byorderdata[[2]], All]
```

Out[193]= {{1, 1}, {7, 1}, {5, 1}, {6, 1}, {5, 1}}

Out[194]= {{6, 2}, {6, 2}, {7, 2}, {6, 2}, {7, 2}}

注文１番の部品リスト

In[195]:=
```
d = byorderdata[[1]][[All, 1]]
```

Out[195]= {1, 7, 5, 6, 5}

部品リストの２を基準とするすべての組み合わせ

In[196]:=
```
KSubsets[d, 2]
```

Out[196]= {{1, 7}, {1, 5}, {1, 6}, {1, 5}, {7, 5}, {7, 6}, {7, 5}, {5, 6}, {5, 5}, {6, 5}}

組み合わせ内ペアを小さい順に並び替える

In[197]:=
```
Sort /@ KSubsets[d, 2]
```

Out[197]=
```
{{1, 7}, {1, 5}, {1, 6}, {1, 5}, {5, 7}, {6, 7}, {5, 7}, {5, 6}, {5, 5}, {5, 6}}
```

上記値と下記計算結果の一行目が等しいことを確認

In[198]:=
```
(Sort /@ KSubsets[#[[All, 1]], 2]) & /@ byorderdata
```

Out[198]=
```
{{{1, 7}, {1, 5}, {1, 6}, {1, 5}, {5, 7}, {6, 7}, {5, 7}, {5, 6}, {5, 5}, {5, 6}},
 {{6, 6}, {6, 7}, {6, 6}, {6, 7}, {6, 7}, {6, 6}, {6, 7}, {6, 7}, {7, 7}, {6, 7}},
 {{4, 8}, {4, 8}, {6, 8}, {4, 8}, {4, 4}, {4, 6}, {4, 4}, {4, 6}, {4, 4}, {4, 6}},
 {{1, 8}, {1, 7}, {1, 3}, {1, 10}, {7, 8}, {3, 8}, {8, 10}, {3, 7}, {7, 10}, {3, 10}},
 {{1, 10}, {10, 10}, {7, 10}, {2, 10}, {1, 10}, {1, 7}, {1, 2}, {7, 10}, {2, 10}, {2, 7}},
 {{3, 5}, {3, 7}, {3, 5}, {3, 9}, {5, 7}, {5, 5}, {5, 9}, {5, 7}, {7, 9}, {5, 9}},
 {{1, 7}, {1, 8}, {1, 6}, {1, 2}, {7, 8}, {6, 7}, {2, 7}, {6, 8}, {2, 8}, {2, 6}},
 {{6, 7}, {6, 7}, {6, 6}, {5, 6}, {7, 7}, {6, 7}, {5, 7}, {6, 7}, {5, 7}, {5, 6}},
 {{9, 10}, {9, 9}, {6, 9}, {8, 9}, {9, 10}, {6, 10}, {8, 10}, {6, 9}, {8, 9}, {6, 8}},
 {{1, 6}, {6, 9}, {6, 10}, {5, 6}, {1, 9}, {1, 10}, {1, 5}, {9, 10}, {5, 9}, {5, 10}},
 {{1, 10}, {2, 10}, {9, 10}, {3, 10}, {1, 2}, {1, 9}, {1, 3}, {2, 9}, {2, 3}, {3, 9}}}
```

KSubsets[L,n] Lからn個選ぶすべての組み合わせ

In[199]:=
```
KSubsets[{1, 2, 3}, 2]
```

Out[199]=
```
{{1, 2}, {1, 3}, {2, 3}}
```

{頻度，データ}のペアを返す.

In[200]:=
```
Tally[{a, b, a, b, c}]
```

Out[200]=
```
{{a, 2}, {b, 2}, {c, 1}}
```

In[201]:=
```
Frequencies[data_] := {#[[2]], #[[1]]} & /@ Tally[data];
```

In[202]:=
```
crosssellbyorder = {#[[1, 2]], (Sort /@ KSubsets[#[[All, 1]], 2]) // Frequencies // Sort} & /@
    byorderdata; // Timing
```

Out[202]=
```
{0., Null}
```

In[203]:=
```
crosssellbyorder[[1]]
```

Out[203]=
```
{1, {{1, {1, 6}}, {1, {1, 7}}, {1, {5, 5}}, {1, {6, 7}}, {2, {1, 5}}, {2, {5, 6}}, {2, {5, 7}}}}
```

■ 実験

In[204]:=
```
itemsperorder = 100;
possibleitems = 10;
byorderdata = Table[
   {RandomInteger[DiscreteUniformDistribution[{1, possibleitems}], {itemsperorder}],
     Table[i, {itemsperorder}]} // Transpose, {i, 1, 1000}];

crosssellbyorder = {#[[1, 2]], (Sort /@ KSubsets[#[[All, 1]], 2]) // Frequencies // Sort} & /@
    byorderdata; // Timing
```

Out[207]=
```
{41.793, Null}
```

In[208]:=
```
crosssellbyorder[[2]] // Short
```

Out[208]//Short=
```
{2, {{10, {5, 5}}, {28, {7, 7}}, {28, {8, 8}}, {28, {9, 9}},
  <<47>>, {160, {1, 3}}, {182, {2, 6}}, {208, {1, 6}}, {224, {1, 2}}}}
```

より速いバージョン

```
In[209]:= crosssellbyorder2 = {#[[1, 2]],
            Clear[freq];
            (* 一列目(部品番号)の頻度リスト *)
            IDFreqs = #[[All, 1]] // Frequencies;
            (* freq[部品番号]=頻度 *)
            (freq[#[[2]]] = #[[1]]) & /@ IDFreqs;

            Join[
              (* 部品番号の部分組み合わせに対しその頻度をかけ算する. *)
              {Times @@ freq /@ #, #} & /@ KSubsets[IDFreqs[[All, 2]], 2],
              (* 部品番号の部分組み合わせに対し二項分布の個数を計算 *)
              Select[{Binomial[freq[#], 2], {#, #}} & /@ IDFreqs[[All, 2]], #[[1]] =!= 0 &]]
             // Sort
           } & /@ byorderdata; // Timing

Out[209]= {1.544, Null}
```

- **速度差比較**

```
In[210]:= itemsperorder = 100;
          possibleitems = 10;
          byorderdata = Table[
             {RandomInteger[DiscreteUniformDistribution[{1, possibleitems}], {itemsperorder}],
               Table[i, {itemsperorder}]} // Transpose, {i, 1, 1000}];

In[213]:= {#[[1, 2]], (Sort /@ KSubsets[#[[All, 1]], 2]) // Frequencies // Sort} & /@ byorderdata; // Timing

Out[213]= {44.258, Null}

In[214]:= crosssellbyorder2 = {#[[1, 2]],
            Clear[freq];
            IDFreqs = #[[All, 1]] // Frequencies;
            (freq[#[[2]]] = #[[1]]) & /@ IDFreqs;
            Join[
              {Times @@ freq /@ #, #} & /@ KSubsets[IDFreqs[[All, 2]], 2],
              Select[{Binomial[freq[#], 2], {#, #}} & /@ IDFreqs[[All, 2]], #[[1]] =!= 0 &]]
             // Sort
           } & /@ byorderdata; // Timing

Out[214]= {1.545, Null}
```

国語の近接関係

ref.
"Communities of Nations Bridged by Language Similarity" from The Wolfram Demonstrations Project
http://demonstrations.wolfram.com/CommunitiesOfNationsBridgedByLanguageSimilarity/

- **国名情報　allCountries**
- **国語情報　allLanguages, allLanugaes0**
- **国記号　allUNCodes**

languageVector

In[222]:=
```
Take[allLanguages0, 5] (* 国別．母国語（複数）以外に使用される外国語も含む  *)
```

Out[222]=
```
{{FarsiEastern, Hazaragi, UzbekSouthern, PashtoSouthern, Turkmen, Aimaq, Brahui,
  BalochiWestern, PashayiSouthwest, PashayiSoutheast, PashayiNortheast, Shughni,
  Kati, Tangshewi, Darwazi, Wakhi, GawarBati, Warduji, Grangali, ArabicTajikiSpoken,
  Kamviri, Munji, Uyghur, Savi, Malakhel, Pahlavani, WotapuriKatarqalai, Kazakh,
  Karakalpak, Gujari, Waigali, Jakati, Parya, Ashkun, Tregami, Shumashti,
  SanglechiIshkashimi, Prasuni, Kirghiz, Parachi, Mogholi, Tirahi, Ormuri},
 {AlbanianTosk, AlbanianGheg, RomaniVlax, Greek, RomanianMacedo, Macedonian},
 {ArabicAlgerianSpoken, Kabyle, Tachawit, French, ArabicAlgerianSaharanSpoken, Tumzabt,
  Taznatit, TamahaqTahaggart, TamazightTidikelt, TamazightTemacine, Tagargrent, Chenoua},
 {Samoan, English}, {CatalanValencianBalear, Spanish, French}}
```

In[223]:=
```
Take[allLanguages, 5] (* 全言語リスト *)
```

Out[223]=
```
{Aari, Aasax, Abadi, Abaga, Abanyom}
```

すべての言語に対し，その言語が使われている国の対応表（行は国インデックス，列は言語インデックスで対応する
場所に 1 がたつ疎行列）を作成.

In[224]:=
```
languageVector =
  SparseArray[Flatten[MapIndexed[Thread[List[#2[[1]], Position[allLanguages, #][[1, 1]]] &,
      allLanguages0, {2}], 1] → 1, {Length[allCountries], Length[allLanguages]}];
```

In[225]:=
```
Clear[a, b, c, d, e];
```

In[226]:=
```
MapIndexed[f, {{a, b}, {c, d, e}}, {2}]
```

Out[226]=
```
{{f[a, {1, 1}], f[b, {1, 2}]}, {f[c, {2, 1}], f[d, {2, 2}], f[e, {2, 3}]}}
```

In[227]:=
```
SparseArray[{{1, 1} → 1, {2, 2} → 2, {3, 3} → 3, {1, 3} → 4}] // MatrixForm
```

Out[227]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

この疎行列を関数Normalによって正規化し，距離値を計算（方法は下記を参照）．この距離値がクラスタ分類に利用される．また，距離の差異をどこで線引きするかによってクラスタの構成が異なってくる.

In[228]:=
```
dm = HierarchicalClustering`DistanceMatrix[
   Normal@languageVector, DistanceFunction → JaccardDissimilarity];
```

In[229]:=
```
dm[[All, 1]] // N
```

Out[229]=
```
{0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.98513,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.997717, 1., 0.91954, 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 0.955556, 1., 1., 1., 1., 0.977778, 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.960784,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 0.886598, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
 1., 1., 1., 1., 1., 0.893617, 1., 1., 1., 1., 1., 1., 1., 1., 0.954545, 1., 1., 1.,
 0.979592, 0.979167, 1., 1., 1., 1., 0.979167, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.}
```

Normal@languageVector 疎行列に対し正規密度行列を作成
JaccardDissimilarity　論理値（あるいは１，０データ）からなるベクトル同士の非類似度検査関数
DistanceMatrix　次の例では差の２乗を２つの要素間の距離とする．しかし，この問題ではJaccardDissimilarityを使用.

In[230]:=
```
HierarchicalClustering`DistanceMatrix[{1, 2, 4}] // MatrixForm
```

Out[230]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 9 \\ 1 & 0 & 4 \\ 9 & 4 & 0 \end{pmatrix}$$

In[231]:=
```
HierarchicalClustering`DistanceMatrix[{1, 0, 0, 1},
  DistanceFunction → JaccardDissimilarity] // MatrixForm
```

Out[231]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

距離行列は非類似度値からなるので，値が大きいほど関係が弱いとみる．したがって，下記でz値を大きくすればするほど関係性の薄いもの同士も抽出される．しかし，だからといって問題なのではなく，別のメトリックスを使えば，そこに大きな関係が見いだせるかもしれない．そのために，zのようなパラメータは必要.

In[232]:=
```
Select[Flatten[dm], # > 0.6 && # < 0.7 &] // Length
```

Out[232]= 986

In[233]:=
```
Select[Flatten[dm], # > 0.2 && # < 0.3 &] // Length
```

Out[233]= 2

In[234]:=
```
z = 0.6; (* z=0.2 ... 0.75 *)
(Sort /@ Rule @@@ Position[Map[If[0 <= # < z, 1, 0] &, dm, {2}], 1, {2}]) // Short
```

Out[235]//Short=

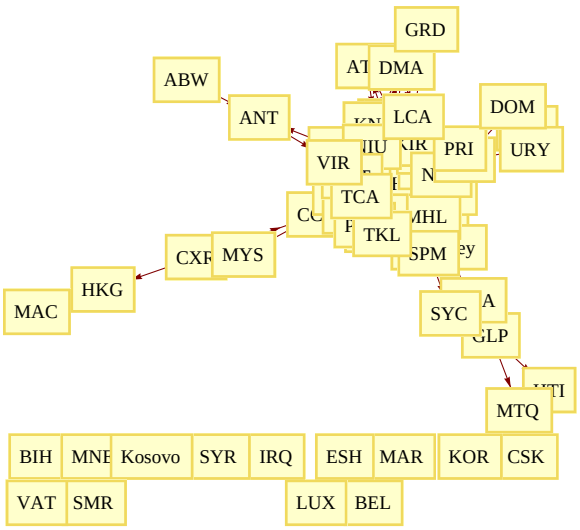
```
{1 → 1, 2 → 2, 3 → 3, 4 → 4, 4 → 23, 4 → 38, 4 → 66, 4 → 79, <<626>>, 232 → 232,
 233 → 233, 234 → 234, 143 → 235, 235 → 235, 236 → 236, 237 → 237, 238 → 238}
```

In[236]:=
```
g =
 DeleteCases[Union@(Sort /@ Rule @@@ Position[Map[If[0 <= # < z, 1, 0] &, dm, {2}], 1, {2}]) /.
   i_Integer :> allUNCodes[[i]], _[a_, a_]]
```

Out[236]=
```
{ASM → BMU, ASM → CYM, ASM → FLK, ASM → GIB, ASM → SHN, ASM → WSM, AIA → ATG, AIA → BMU,
 AIA → CYM, AIA → DMA, AIA → FLK, AIA → GIB, AIA → MSR, AIA → SHN, AIA → KNA, ATG → MSR,
 ATG → KNA, ABW → ANT, BHS → BMU, BHS → CYM, BHS → FLK, BHS → GIB, BHS → SHN, BRB → BMU,
 BRB → CYM, BRB → FLK, BRB → GIB, BRB → SHN, BEL → LUX, BMU → VGB, BMU → CYM, BMU → CCK,
 BMU → FLK, BMU → GIB, BMU → GUM, BMU → Guernsey, BMU → IsleOfMan, BMU → Jersey, BMU → KIR,
 BMU → MLT, BMU → MHL, BMU → MSR, BMU → NRU, BMU → NZL, BMU → NIU, BMU → NFK, BMU → PAN,
 BMU → PCN, BMU → PRI, BMU → SHN, BMU → KNA, BMU → LCA, BMU → SPM, BMU → VCT, BMU → WSM,
 BMU → TKL, BMU → TCA, BMU → VIR, BIH → MNE, VGB → CYM, VGB → FLK, VGB → GIB, VGB → ANT,
 VGB → SHN, VGB → VIR, CYM → CCK, CYM → FLK, CYM → GIB, CYM → GUM, CYM → Guernsey,
 CYM → IsleOfMan, CYM → Jersey, CYM → KIR, CYM → MLT, CYM → MHL, CYM → MSR, CYM → NRU,
 CYM → NZL, CYM → NIU, CYM → NFK, CYM → PAN, CYM → PCN, CYM → PRI, CYM → SHN, CYM → KNA,
 CYM → LCA, CYM → SPM, CYM → VCT, CYM → WSM, CYM → TKL, CYM → TCA, CYM → VIR, CXR → CCK,
 CXR → HKG, CCK → FLK, CCK → GIB, CCK → MYS, CCK → SHN, CUB → PAN, CUB → PRI, CUB → URY,
 DMA → GRD, DMA → MSR, DMA → KNA, DMA → LCA, DOM → PAN, DOM → PRI, FLK → GIB, FLK → GUM,
 FLK → Guernsey, FLK → IsleOfMan, FLK → Jersey, FLK → KIR, FLK → MLT, FLK → MHL,
 FLK → MSR, FLK → NRU, FLK → NZL, FLK → NIU, FLK → NFK, FLK → PAN, FLK → PCN, FLK → PRI,
 FLK → SHN, FLK → KNA, FLK → LCA, FLK → SPM, FLK → VCT, FLK → WSM, FLK → TKL, FLK → TCA,
 FLK → VIR, GIB → GUM, GIB → Guernsey, GIB → IsleOfMan, GIB → Jersey, GIB → KIR,
 GIB → MLT, GIB → MHL, GIB → MSR, GIB → NRU, GIB → NZL, GIB → NIU, GIB → NFK, GIB → PAN,
 GIB → PCN, GIB → PRI, GIB → SHN, GIB → KNA, GIB → LCA, GIB → SPM, GIB → VCT,
 GIB → WSM, GIB → TKL, GIB → TCA, GIB → VIR, GRD → LCA, GLP → Guernsey, GLP → HTI,
 GLP → MTQ, GLP → SPM, GUM → SHN, Guernsey → RWA, Guernsey → SHN, Guernsey → SPM,
 Guernsey → SYC, HKG → MAC, IRQ → SYR, IsleOfMan → SHN, Jersey → SHN, KIR → SHN,
 Kosovo → MNE, MLT → SHN, MHL → SHN, MSR → SHN, MSR → KNA, MAR → ESH, NRU → SHN,
 ANT → VIR, NZL → SHN, NIU → SHN, NFK → PCN, NFK → SHN, CSK → KOR, PAN → PRI, PAN → SHN,
 PAN → URY, PCN → SHN, PRI → SHN, PRI → URY, RWA → SPM, RWA → SYC, SHN → KNA, SHN → LCA,
 SHN → SPM, SHN → VCT, SHN → WSM, SHN → TKL, SHN → TCA, SHN → VIR, SPM → SYC, SMR → VAT}
```

In[237]:=
```
GraphPlot[g, VertexLabeling → True, DirectedEdges → True]
```

Out[237]=



In[238]:=
```
languages[codes_] := Module[{pos},
  pos = (Position[allUNCodes, #][[1, 1]]) & /@ codes;
  Thread[{allCountries[[pos]], allLanguages0[[pos]]}]]
```

In[239]:=
```
languages[{"MSR", "KNA", "AIA"}]
```

Out[239]=
```
{{Montserrat, {AntiguaBarbudaCreoleEnglish, English}},
 {SaintKittsNevis, {AntiguaBarbudaCreoleEnglish, English}},
 {Anguilla, {AntiguaBarbudaCreoleEnglish, English}}}
```

In[240]:=
```
csp = GraphUtilities`CommunityStructurePartition[g]
```

Out[240]=
```
GraphUtilities`CommunityStructurePartition[
 {ASM → BMU, ASM → CYM, ASM → FLK, ASM → GIB, ASM → SHN, ASM → WSM, AIA → ATG, AIA → BMU,
  AIA → CYM, AIA → DMA, AIA → FLK, AIA → GIB, AIA → MSR, AIA → SHN, AIA → KNA, ATG → MSR,
  ATG → KNA, ABW → ANT, BHS → BMU, BHS → CYM, BHS → FLK, BHS → GIB, BHS → SHN, BRB → BMU,
  BRB → CYM, BRB → FLK, BRB → GIB, BRB → SHN, BEL → LUX, BMU → VGB, BMU → CYM, BMU → CCK,
  BMU → FLK, BMU → GIB, BMU → GUM, BMU → Guernsey, BMU → IsleOfMan, BMU → Jersey, BMU → KIR,
  BMU → MLT, BMU → MHL, BMU → MSR, BMU → NRU, BMU → NZL, BMU → NIU, BMU → NFK, BMU → PAN,
  BMU → PCN, BMU → PRI, BMU → SHN, BMU → KNA, BMU → LCA, BMU → SPM, BMU → VCT, BMU → WSM,
  BMU → TKL, BMU → TCA, BMU → VIR, BIH → MNE, VGB → CYM, VGB → FLK, VGB → GIB, VGB → ANT,
  VGB → SHN, VGB → VIR, CYM → CCK, CYM → FLK, CYM → GIB, CYM → GUM, CYM → Guernsey,
  CYM → IsleOfMan, CYM → Jersey, CYM → KIR, CYM → MLT, CYM → MHL, CYM → MSR, CYM → NRU,
  CYM → NZL, CYM → NIU, CYM → NFK, CYM → PAN, CYM → PCN, CYM → PRI, CYM → SHN, CYM → KNA,
  CYM → LCA, CYM → SPM, CYM → VCT, CYM → WSM, CYM → TKL, CYM → TCA, CYM → VIR, CXR → CCK,
  CXR → HKG, CCK → FLK, CCK → GIB, CCK → MYS, CCK → SHN, CUB → PAN, CUB → PRI, CUB → URY,
  DMA → GRD, DMA → MSR, DMA → KNA, DMA → LCA, DOM → PAN, DOM → PRI, FLK → GIB, FLK → GUM,
  FLK → Guernsey, FLK → IsleOfMan, FLK → Jersey, FLK → KIR, FLK → MLT, FLK → MHL,
  FLK → MSR, FLK → NRU, FLK → NZL, FLK → NIU, FLK → NFK, FLK → PAN, FLK → PCN, FLK → PRI,
  FLK → SHN, FLK → KNA, FLK → LCA, FLK → SPM, FLK → VCT, FLK → WSM, FLK → TKL, FLK → TCA,
  FLK → VIR, GIB → GUM, GIB → Guernsey, GIB → IsleOfMan, GIB → Jersey, GIB → KIR,
  GIB → MLT, GIB → MHL, GIB → MSR, GIB → NRU, GIB → NZL, GIB → NIU, GIB → NFK, GIB → PAN,
  GIB → PCN, GIB → PRI, GIB → SHN, GIB → KNA, GIB → LCA, GIB → SPM, GIB → VCT,
  GIB → WSM, GIB → TKL, GIB → TCA, GIB → VIR, GRD → LCA, GLP → Guernsey, GLP → HTI,
  GLP → MTQ, GLP → SPM, GUM → SHN, Guernsey → RWA, Guernsey → SHN, Guernsey → SPM,
  Guernsey → SYC, HKG → MAC, IRQ → SYR, IsleOfMan → SHN, Jersey → SHN, KIR → SHN,
  Kosovo → MNE, MLT → SHN, MHL → SHN, MSR → SHN, MSR → KNA, MAR → ESH, NRU → SHN,
  ANT → VIR, NZL → SHN, NIU → SHN, NFK → PCN, NFK → SHN, CSK → KOR, PAN → PRI, PAN → SHN,
  PAN → URY, PCN → SHN, PRI → SHN, PRI → URY, RWA → SPM, RWA → SYC, SHN → KNA, SHN → LCA,
  SHN → SPM, SHN → VCT, SHN → WSM, SHN → TKL, SHN → TCA, SHN → VIR, SPM → SYC, SMR → VAT}]
```

zの値を低くすれば強い関係のものだけが抽出されるので，自然，この値は高くなる．

In[241]:=
```
q = GraphUtilities`CommunityModularity[g, csp]
```

Out[241]=
```
GraphUtilities`CommunityModularity[
 {ASM → BMU, ASM → CYM, ASM → FLK, ASM → GIB, ASM → SHN, ASM → WSM, AIA → ATG, AIA → BMU,
  AIA → CYM, AIA → DMA, AIA → FLK, AIA → GIB, AIA → MSR, AIA → SHN, AIA → KNA, ATG → MSR,
  ATG → KNA, ABW → ANT, BHS → BMU, BHS → CYM, BHS → FLK, BHS → GIB, BHS → SHN, BRB → BMU,
  BRB → CYM, BRB → FLK, BRB → GIB, BRB → SHN, BEL → LUX, BMU → VGB, BMU → CYM, BMU → CCK,
  BMU → FLK, BMU → GIB, BMU → GUM, BMU → Guernsey, BMU → IsleOfMan, BMU → Jersey, BMU → KIR,
  BMU → MLT, BMU → MHL, BMU → MSR, BMU → NRU, BMU → NZL, BMU → NIU, BMU → NFK, BMU → PAN,
  BMU → PCN, BMU → PRI, BMU → SHN, BMU → KNA, BMU → LCA, BMU → SPM, BMU → VCT, BMU → WSM,
  BMU → TKL, BMU → TCA, BMU → VIR, BIH → MNE, VGB → CYM, VGB → FLK, VGB → GIB, VGB → ANT,
  VGB → SHN, VGB → VIR, CYM → CCK, CYM → FLK, CYM → GIB, CYM → GUM, CYM → Guernsey,
  CYM → IsleOfMan, CYM → Jersey, CYM → KIR, CYM → MLT, CYM → MHL, CYM → MSR, CYM → NRU,
  CYM → NZL, CYM → NIU, CYM → NFK, CYM → PAN, CYM → PCN, CYM → PRI, CYM → SHN, CYM → KNA,
  CYM → LCA, CYM → SPM, CYM → VCT, CYM → WSM, CYM → TKL, CYM → TCA, CYM → VIR, CXR → CCK,
  CXR → HKG, CCK → FLK, CCK → GIB, CCK → MYS, CCK → SHN, CUB → PAN, CUB → PRI, CUB → URY,
  DMA → GRD, DMA → MSR, DMA → KNA, DMA → LCA, DOM → PAN, DOM → PRI, FLK → GIB, FLK → GUM,
  FLK → Guernsey, FLK → IsleOfMan, FLK → Jersey, FLK → KIR, FLK → MLT, FLK → MHL,
  FLK → MSR, FLK → NRU, FLK → NZL, FLK → NIU, FLK → NFK, FLK → PAN, FLK → PCN, FLK → PRI,
  FLK → SHN, FLK → KNA, FLK → LCA, FLK → SPM, FLK → VCT, FLK → WSM, FLK → TKL, FLK → TCA,
  FLK → VIR, GIB → GUM, GIB → Guernsey, GIB → IsleOfMan, GIB → Jersey, GIB → KIR,
  GIB → MLT, GIB → MHL, GIB → MSR, GIB → NRU, GIB → NZL, GIB → NIU, GIB → NFK, GIB → PAN,
  GIB → PCN, GIB → PRI, GIB → SHN, GIB → KNA, GIB → LCA, GIB → SPM, GIB → VCT,
  GIB → WSM, GIB → TKL, GIB → TCA, GIB → VIR, GRD → LCA, GLP → Guernsey, GLP → HTI,
  GLP → MTQ, GLP → SPM, GUM → SHN, Guernsey → RWA, Guernsey → SHN, Guernsey → SPM,
  Guernsey → SYC, HKG → MAC, IRQ → SYR, IsleOfMan → SHN, Jersey → SHN, KIR → SHN,
  Kosovo → MNE, MLT → SHN, MHL → SHN, MSR → SHN, MSR → KNA, MAR → ESH, NRU → SHN,
  ANT → VIR, NZL → SHN, NIU → SHN, NFK → PCN, NFK → SHN, CSK → KOR, PAN → PRI, PAN → SHN,
  PAN → URY, PCN → SHN, PRI → SHN, PRI → URY, RWA → SPM, RWA → SYC, SHN → KNA, SHN → LCA,
  SHN → SPM, SHN → VCT, SHN → WSM, SHN → TKL, SHN → TCA, SHN → VIR, SPM → SYC, SMR → VAT},
 GraphUtilities`CommunityStructurePartition[{ASM → BMU, ASM → CYM, ASM → FLK,
   ASM → GIB, ASM → SHN, ASM → WSM, AIA → ATG, AIA → BMU, AIA → CYM, AIA → DMA, AIA → FLK,
   AIA → GIB, AIA → MSR, AIA → SHN, AIA → KNA, ATG → MSR, ATG → KNA, ABW → ANT, BHS → BMU,
   BHS → CYM, BHS → FLK, BHS → GIB, BHS → SHN, BRB → BMU, BRB → CYM, BRB → FLK, BRB → GIB,
   BRB → SHN, BEL → LUX, BMU → VGB, BMU → CYM, BMU → CCK, BMU → FLK, BMU → GIB, BMU → GUM,
   BMU → Guernsey, BMU → IsleOfMan, BMU → Jersey, BMU → KIR, BMU → MLT, BMU → MHL, BMU → MSR,
   BMU → NRU, BMU → NZL, BMU → NIU, BMU → NFK, BMU → PAN, BMU → PCN, BMU → PRI, BMU → SHN,
   BMU → KNA, BMU → LCA, BMU → SPM, BMU → VCT, BMU → WSM, BMU → TKL, BMU → TCA, BMU → VIR,
   BIH → MNE, VGB → CYM, VGB → FLK, VGB → GIB, VGB → ANT, VGB → SHN, VGB → VIR, CYM → CCK,
   CYM → FLK, CYM → GIB, CYM → GUM, CYM → Guernsey, CYM → IsleOfMan, CYM → Jersey,
   CYM → KIR, CYM → MLT, CYM → MHL, CYM → MSR, CYM → NRU, CYM → NZL, CYM → NIU, CYM → NFK,
   CYM → PAN, CYM → PCN, CYM → PRI, CYM → SHN, CYM → KNA, CYM → LCA, CYM → SPM, CYM → VCT,
   CYM → WSM, CYM → TKL, CYM → TCA, CYM → VIR, CXR → CCK, CXR → HKG, CCK → FLK, CCK → GIB,
   CCK → MYS, CCK → SHN, CUB → PAN, CUB → PRI, CUB → URY, DMA → GRD, DMA → MSR, DMA → KNA,
   DMA → LCA, DOM → PAN, DOM → PRI, FLK → GIB, FLK → GUM, FLK → Guernsey, FLK → IsleOfMan,
   FLK → Jersey, FLK → KIR, FLK → MLT, FLK → MHL, FLK → MSR, FLK → NRU, FLK → NZL, FLK → NIU,
   FLK → NFK, FLK → PAN, FLK → PCN, FLK → PRI, FLK → SHN, FLK → KNA, FLK → LCA, FLK → SPM,
   FLK → VCT, FLK → WSM, FLK → TKL, FLK → TCA, FLK → VIR, GIB → GUM, GIB → Guernsey,
   GIB → IsleOfMan, GIB → Jersey, GIB → KIR, GIB → MLT, GIB → MHL, GIB → MSR, GIB → NRU,
   GIB → NZL, GIB → NIU, GIB → NFK, GIB → PAN, GIB → PCN, GIB → PRI, GIB → SHN, GIB → KNA,
   GIB → LCA, GIB → SPM, GIB → VCT, GIB → WSM, GIB → TKL, GIB → TCA, GIB → VIR, GRD → LCA,
   GLP → Guernsey, GLP → HTI, GLP → MTQ, GLP → SPM, GUM → SHN, Guernsey → RWA, Guernsey → SHN,
   Guernsey → SPM, Guernsey → SYC, HKG → MAC, IRQ → SYR, IsleOfMan → SHN, Jersey → SHN,
   KIR → SHN, Kosovo → MNE, MLT → SHN, MHL → SHN, MSR → SHN, MSR → KNA, MAR → ESH, NRU → SHN,
   ANT → VIR, NZL → SHN, NIU → SHN, NFK → PCN, NFK → SHN, CSK → KOR, PAN → PRI, PAN → SHN,
   PAN → URY, PCN → SHN, PRI → SHN, PRI → URY, RWA → SPM, RWA → SYC, SHN → KNA, SHN → LCA,
   SHN → SPM, SHN → VCT, SHN → WSM, SHN → TKL, SHN → TCA, SHN → VIR, SPM → SYC, SMR → VAT}]]
```

言語の重要度をその国内で使用される言語の一番目に来てるという基準でさらに絞り込みを行う.

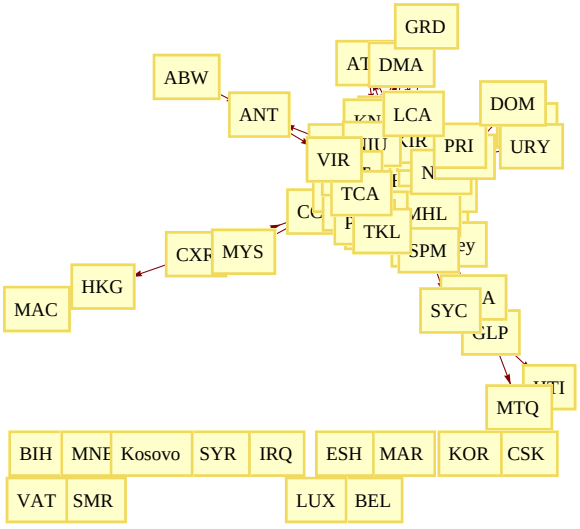
In[242]:=
```
cutInterCommunityLinks[g_, csp_] :=
  Select[g, First@First@Position[csp, #[[1]]] === First@First@Position[csp, #[[2]]] &];
```

In[243]:=
```
cc = cutInterCommunityLinks[g, csp]
```

Out[243]=
```
{ASM → BMU, ASM → CYM, ASM → FLK, ASM → GIB, ASM → SHN, ASM → WSM, AIA → ATG, AIA → BMU,
 AIA → CYM, AIA → DMA, AIA → FLK, AIA → GIB, AIA → MSR, AIA → SHN, AIA → KNA, ATG → MSR,
 ATG → KNA, ABW → ANT, BHS → BMU, BHS → CYM, BHS → FLK, BHS → GIB, BHS → SHN, BRB → BMU,
 BRB → CYM, BRB → FLK, BRB → GIB, BRB → SHN, BEL → LUX, BMU → VGB, BMU → CYM, BMU → CCK,
 BMU → FLK, BMU → GIB, BMU → GUM, BMU → Guernsey, BMU → IsleOfMan, BMU → Jersey, BMU → KIR,
 BMU → MLT, BMU → MHL, BMU → MSR, BMU → NRU, BMU → NZL, BMU → NIU, BMU → NFK, BMU → PAN,
 BMU → PCN, BMU → PRI, BMU → SHN, BMU → KNA, BMU → LCA, BMU → SPM, BMU → VCT, BMU → WSM,
 BMU → TKL, BMU → TCA, BMU → VIR, BIH → MNE, VGB → CYM, VGB → FLK, VGB → GIB, VGB → ANT,
 VGB → SHN, VGB → VIR, CYM → CCK, CYM → FLK, CYM → GIB, CYM → GUM, CYM → Guernsey,
 CYM → IsleOfMan, CYM → Jersey, CYM → KIR, CYM → MLT, CYM → MHL, CYM → MSR, CYM → NRU,
 CYM → NZL, CYM → NIU, CYM → NFK, CYM → PAN, CYM → PCN, CYM → PRI, CYM → SHN, CYM → KNA,
 CYM → LCA, CYM → SPM, CYM → VCT, CYM → WSM, CYM → TKL, CYM → TCA, CYM → VIR, CXR → CCK,
 CXR → HKG, CCK → FLK, CCK → GIB, CCK → MYS, CCK → SHN, CUB → PAN, CUB → PRI, CUB → URY,
 DMA → GRD, DMA → MSR, DMA → KNA, DMA → LCA, DOM → PAN, DOM → PRI, FLK → GIB, FLK → GUM,
 FLK → Guernsey, FLK → IsleOfMan, FLK → Jersey, FLK → KIR, FLK → MLT, FLK → MHL,
 FLK → MSR, FLK → NRU, FLK → NZL, FLK → NIU, FLK → NFK, FLK → PAN, FLK → PCN, FLK → PRI,
 FLK → SHN, FLK → KNA, FLK → LCA, FLK → SPM, FLK → VCT, FLK → WSM, FLK → TKL, FLK → TCA,
 FLK → VIR, GIB → GUM, GIB → Guernsey, GIB → IsleOfMan, GIB → Jersey, GIB → KIR,
 GIB → MLT, GIB → MHL, GIB → MSR, GIB → NRU, GIB → NZL, GIB → NIU, GIB → NFK, GIB → PAN,
 GIB → PCN, GIB → PRI, GIB → SHN, GIB → KNA, GIB → LCA, GIB → SPM, GIB → VCT,
 GIB → WSM, GIB → TKL, GIB → TCA, GIB → VIR, GRD → LCA, GLP → Guernsey, GLP → HTI,
 GLP → MTQ, GLP → SPM, GUM → SHN, Guernsey → RWA, Guernsey → SHN, Guernsey → SPM,
 Guernsey → SYC, HKG → MAC, IRQ → SYR, IsleOfMan → SHN, Jersey → SHN, KIR → SHN,
 Kosovo → MNE, MLT → SHN, MHL → SHN, MSR → SHN, MSR → KNA, MAR → ESH, NRU → SHN,
 ANT → VIR, NZL → SHN, NIU → SHN, NFK → PCN, NFK → SHN, CSK → KOR, PAN → PRI, PAN → SHN,
 PAN → URY, PCN → SHN, PRI → SHN, PRI → URY, RWA → SPM, RWA → SYC, SHN → KNA, SHN → LCA,
 SHN → SPM, SHN → VCT, SHN → WSM, SHN → TKL, SHN → TCA, SHN → VIR, SPM → SYC, SMR → VAT}
```

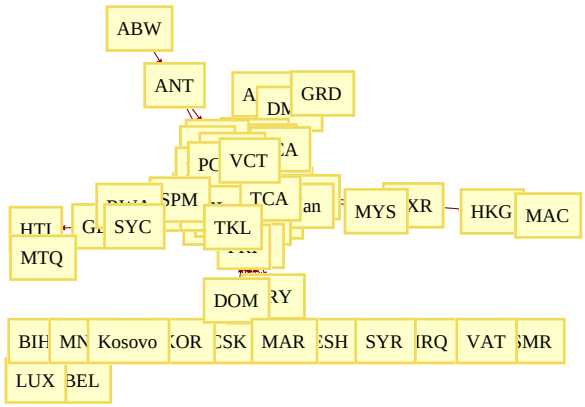
In[244]:=
```
GraphPlot[cc, VertexLabeling → True, DirectedEdges → True]
```

Out[244]=



In[245]:=
```
GraphPlot[Union@(Sort /@ cutInterCommunityLinks[g, csp]),
 VertexLabeling → True, DirectedEdges → True]
```

Out[245]=



Shakespear劇発言リストのみからキーパーソンを発見する

- **作品名->各シーン別発言者リスト speechSequences**
- **計算**

In[247]:=
```
("The Tragedy of Antony and Cleopatra" /. speechSequences)[[3]] (* Scene3 *)
```

Out[247]=
```
{CLEOPATRA, CHARMIAN, CLEOPATRA, CHARMIAN, CLEOPATRA, CHARMIAN, CLEOPATRA,
 CHARMIAN, CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY,
 CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY, CLEOPATRA,
 MARK ANTONY, CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY,
 CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY,
 CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY, CLEOPATRA, MARK ANTONY}
```

In[248]:=
```
First /@ speechSequences (* all plays listed in speedSquences *)
```

Out[248]=
```
{The Tragedy of Antony and Cleopatra, A Midsummer Night's Dream,
 The Tragedy of Hamlet, Prince of Denmark, The Tragedy of Julius Caesar,
 The Tragedy of Macbeth, The Merchant of Venice,
 The Tragedy of Othello, the Moor of Venice, The Tragedy of Romeo and Juliet}
```

In[249]:=
```
playNetwork[play_ → sequenceList_List, n_Integer] := play → Flatten[Apply[Rule, Map[
     Map[DeleteCases[Tuples[#, 2], {a_, a_}] &, Partition[#, n, 1]] &, sequenceList], {3}]]
```

In[250]:=
```
slist2 = Cases[{speechSequences[[3]]}, Rule[play_, sequenceList_] → sequenceList][[1]];
(* {...}の中身 *)
slist2 // First (* scene1 *)
```

Out[251]=
```
{BERNARDO, FRANCISCO, BERNARDO, FRANCISCO, BERNARDO, FRANCISCO, BERNARDO,
 FRANCISCO, BERNARDO, FRANCISCO, BERNARDO, FRANCISCO, HORATIO, MARCELLUS,
 FRANCISCO, MARCELLUS, FRANCISCO, MARCELLUS, BERNARDO, HORATIO, BERNARDO,
 MARCELLUS, BERNARDO, MARCELLUS, HORATIO, BERNARDO, HORATIO, BERNARDO,
 MARCELLUS, BERNARDO, MARCELLUS, BERNARDO, HORATIO, BERNARDO, MARCELLUS, HORATIO,
 MARCELLUS, BERNARDO, HORATIO, MARCELLUS, BERNARDO, HORATIO, MARCELLUS, HORATIO,
 MARCELLUS, HORATIO, MARCELLUS, HORATIO, BERNARDO, HORATIO, MARCELLUS, HORATIO,
 BERNARDO, HORATIO, MARCELLUS, BERNARDO, HORATIO, MARCELLUS, HORATIO, MARCELLUS}
```

空間的に配置された「もの」は近傍において関連性が強いという仮説。近傍の長さnによってn-gramという定義がされる．テキストにおけるキーワード抽出理論においてはもっともらしいと認められている。ただし、仮説であることは忘れないこと。

In[252]:=
```
Partition[slist2 // First, 2, 1](* n-gram, here n==2 *)
```

Out[252]=
```
{{BERNARDO, FRANCISCO}, {FRANCISCO, BERNARDO}, {BERNARDO, FRANCISCO},
 {FRANCISCO, BERNARDO}, {BERNARDO, FRANCISCO}, {FRANCISCO, BERNARDO},
 {BERNARDO, FRANCISCO}, {FRANCISCO, BERNARDO}, {BERNARDO, FRANCISCO},
 {FRANCISCO, BERNARDO}, {BERNARDO, FRANCISCO}, {FRANCISCO, HORATIO},
 {HORATIO, MARCELLUS}, {MARCELLUS, FRANCISCO}, {FRANCISCO, MARCELLUS},
 {MARCELLUS, FRANCISCO}, {FRANCISCO, MARCELLUS}, {MARCELLUS, BERNARDO},
 {BERNARDO, HORATIO}, {HORATIO, BERNARDO}, {BERNARDO, MARCELLUS},
 {MARCELLUS, BERNARDO}, {BERNARDO, MARCELLUS}, {MARCELLUS, HORATIO},
 {HORATIO, BERNARDO}, {BERNARDO, HORATIO}, {HORATIO, BERNARDO}, {BERNARDO, MARCELLUS},
 {MARCELLUS, BERNARDO}, {BERNARDO, MARCELLUS}, {MARCELLUS, BERNARDO},
 {BERNARDO, HORATIO}, {HORATIO, BERNARDO}, {BERNARDO, MARCELLUS}, {MARCELLUS, HORATIO},
 {HORATIO, MARCELLUS}, {MARCELLUS, BERNARDO}, {BERNARDO, HORATIO}, {HORATIO, MARCELLUS},
 {MARCELLUS, BERNARDO}, {BERNARDO, HORATIO}, {HORATIO, MARCELLUS}, {MARCELLUS, HORATIO},
 {HORATIO, MARCELLUS}, {MARCELLUS, HORATIO}, {HORATIO, MARCELLUS}, {MARCELLUS, HORATIO},
 {HORATIO, BERNARDO}, {BERNARDO, HORATIO}, {HORATIO, MARCELLUS}, {MARCELLUS, HORATIO},
 {HORATIO, BERNARDO}, {BERNARDO, HORATIO}, {HORATIO, MARCELLUS}, {MARCELLUS, BERNARDO},
 {BERNARDO, HORATIO}, {HORATIO, MARCELLUS}, {MARCELLUS, HORATIO}, {HORATIO, MARCELLUS}}
```

In[253]:=
```
s = {{a, b, c, d, a, e, f}}
```

Out[253]=
```
{{a, b, c, d, a, e, f}}
```

In[254]:=
```
Flatten[
 Apply[Rule, Map[Map[DeleteCases[Tuples[#, 2], {a_, a_}] &, Partition[#, 3, 1]] &, s], {3}]]
```

Out[254]=
```
{a → b, a → c, b → a, b → c, c → a, c → b, b → c, b → d, c → b,
 c → d, d → b, d → c, c → d, c → a, d → c, d → a, a → c, a → d, d → a,
 d → e, a → d, a → e, e → d, e → a, a → e, a → f, e → a, e → f, f → a, f → e}
```

In[255]:=
```
p = Partition[s // First, 3, 1]
```

Out[255]=
```
{{a, b, c}, {b, c, d}, {c, d, a}, {d, a, e}, {a, e, f}}
```

In[256]:=
```
Tuples[#, 2] & /@ p
```

Out[256]=
```
{{{a, a}, {a, b}, {a, c}, {b, a}, {b, b}, {b, c}, {c, a}, {c, b}, {c, c}},
 {{b, b}, {b, c}, {b, d}, {c, b}, {c, c}, {c, d}, {d, b}, {d, c}, {d, d}},
 {{c, c}, {c, d}, {c, a}, {d, c}, {d, d}, {d, a}, {a, c}, {a, d}, {a, a}},
 {{d, d}, {d, a}, {d, e}, {a, d}, {a, a}, {a, e}, {e, d}, {e, a}, {e, e}},
 {{a, a}, {a, e}, {a, f}, {e, a}, {e, e}, {e, f}, {f, a}, {f, e}, {f, f}}}
```

In[257]:=
```
Map[DeleteCases[#, {a_, a_}] &, %]
```

Out[257]=
```
{{{a, b}, {a, c}, {b, a}, {b, c}, {c, a}, {c, b}},
 {{b, c}, {b, d}, {c, b}, {c, d}, {d, b}, {d, c}},
 {{c, d}, {c, a}, {d, c}, {d, a}, {a, c}, {a, d}},
 {{d, a}, {d, e}, {a, d}, {a, e}, {e, d}, {e, a}},
 {{a, e}, {a, f}, {e, a}, {e, f}, {f, a}, {f, e}}}
```

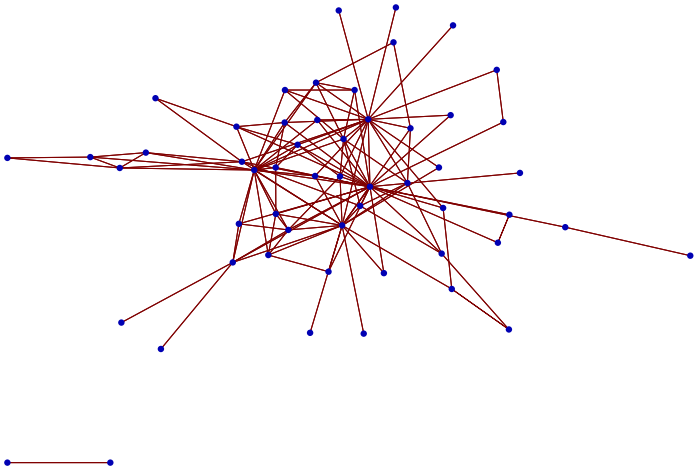
In[258]:=
```
(gmap = playNetwork[speechSequences[[1]], 2]) // Short
```

Out[258]//Short=
```
The Tragedy of Antony and Cleopatra →
 {PHILO → CLEOPATRA, <<2270>>, OCTAVIUS CAESAR → First Guard}
```

In[259]:=
```
Labeled[GraphPlot[gmap[[2]], MultiedgeStyle → None, ImageSize → {400, 300},
  AspectRatio → 2 / 3], "character networks of " <> gmap[[1]], Top]
```

character networks of The Tragedy of Antony and Cleopatra

Out[259]=



In[260]:=
```
Needs["GraphUtilities`"];
```

In[261]:=
```
s = Sort[PageRanks[gmap[[2]]], #1[[2]] > #2[[2]] &]
```

Out[261]=
```
{MARK ANTONY → 0.0966069, CLEOPATRA → 0.074289, OCTAVIUS CAESAR → 0.0657274,
 DOMITIUS ENOBARBUS → 0.0632638, CHARMIAN → 0.0293005, POMPEY → 0.0276778,
 LEPIDUS → 0.0264183, All → 0.0248077, DOLABELLA → 0.0211258, IRAS → 0.0210554,
 First Guard → 0.0208008, EROS → 0.0203142, CANIDIUS → 0.0201775, Messenger → 0.0200906,
 AGRIPPA → 0.0199768, First Soldier → 0.0190637, SILIUS → 0.0185185,
 VENTIDIUS → 0.0185185, Soldier → 0.0178888, MECAENAS → 0.0173297, SCARUS → 0.0169656,
 Third Soldier → 0.0159111, Second Guard → 0.0158937, PROCULEIUS → 0.0155726,
 Second Soldier → 0.0151008, MENAS → 0.014567, Soothsayer → 0.0145042,
 THYREUS → 0.0143466, ALEXAS → 0.0143228, DERCETAS → 0.0135169, First Servant → 0.0126367,
 DIOMEDES → 0.0125472, OCTAVIA → 0.011597, Attendant → 0.0114061, PHILO → 0.0100219,
 Third Guard → 0.00998497, DEMETRIUS → 0.00996979, Second Attendant → 0.00993129,
 First Attendant → 0.00993129, Fourth Soldier → 0.00939971, GALLUS → 0.0090698,
 Captain → 0.00883123, SELEUCUS → 0.00872515, MARDIAN → 0.00871743,
 EUPHRONIUS → 0.00865093, Second Servant → 0.00814834, VARRIUS → 0.00613865,
 MENECRATES → 0.00613865, Clown → 0.00578472, Guard → 0.00578472, Attendants → 0.00578472,
 Egyptian → 0.00571821, TAURUS → 0.00571821, Second Messenger → 0.00571049}
```

In[262]:=
```
Take[s, 5]
```

Out[262]=
```
{MARK ANTONY → 0.0966069, CLEOPATRA → 0.074289,
 OCTAVIUS CAESAR → 0.0657274, DOMITIUS ENOBARBUS → 0.0632638, CHARMIAN → 0.0293005}
```

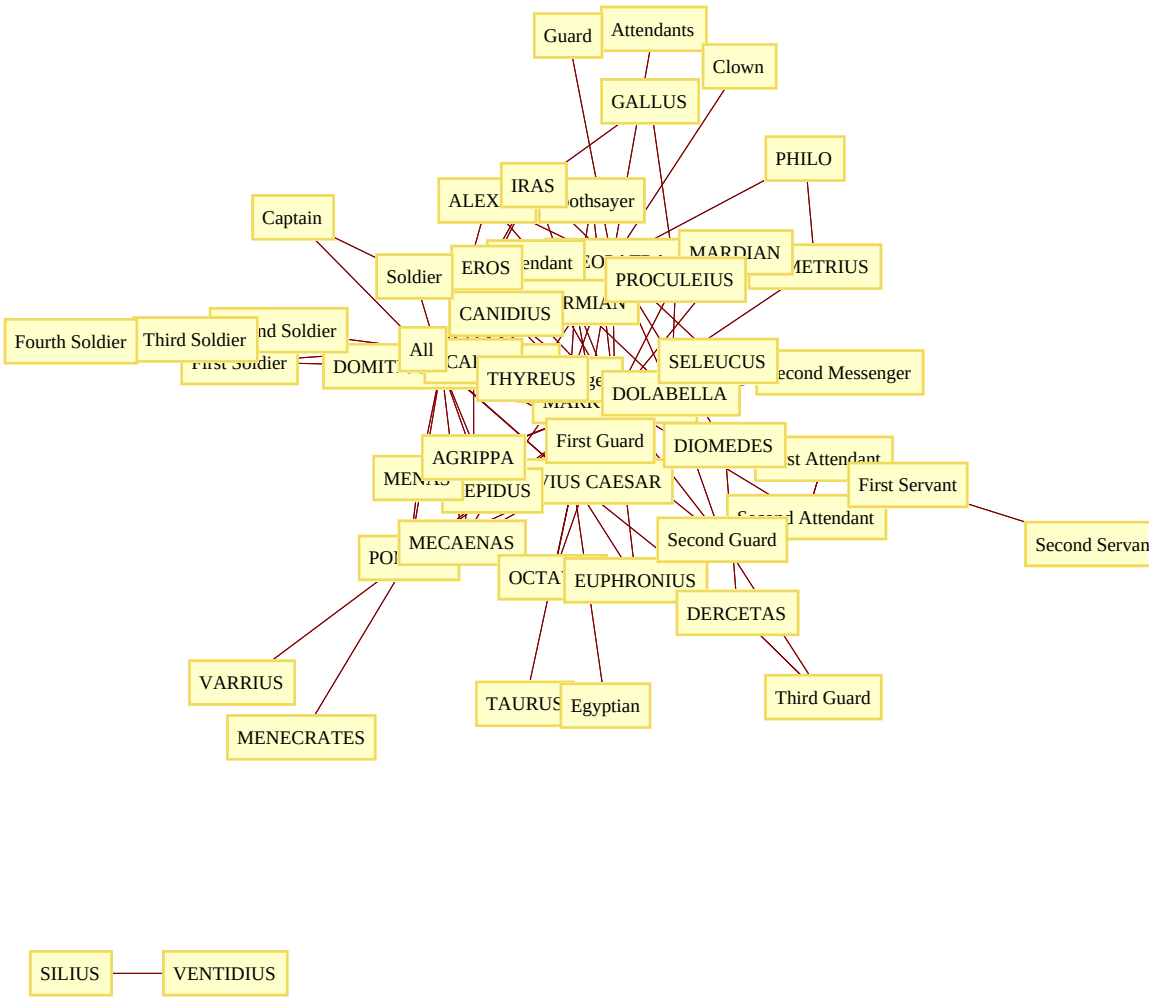
In[263]:=
```
groups = CommunityStructurePartition[gmap[[2]], Weighted → True]
```

Out[263]=
```
{{PHILO, DEMETRIUS}, {CLEOPATRA, Messenger, CHARMIAN, IRAS, ALEXAS, DOLABELLA, Soothsayer,
  Clown, PROCULEIUS, THYREUS, MARDIAN, SELEUCUS, Guard, Attendants, GALLUS},
 {MARK ANTONY, EROS, OCTAVIUS CAESAR, OCTAVIA, LEPIDUS, EUPHRONIUS, SCARUS,
  DOMITIUS ENOBARBUS, MENAS, POMPEY, AGRIPPA, MECAENAS, CANIDIUS, Soldier,
  DERCETAS, DIOMEDES, Second Messenger, MENECRATES, Egyptian, First Attendant,
  Second Attendant, VARRIUS, TAURUS, Attendant}, {First Servant, Second Servant},
 {VENTIDIUS, SILIUS}, {All, First Soldier, Second Soldier, Third Soldier,
  Fourth Soldier, Captain, First Guard, Second Guard, Third Guard}}
```

In[264]:=
```
(* g=GraphPlot[gmap[[2]],VertexLabeling→ True,MultiedgeStyle→None,
  ImageSize→ 600,VertexRenderingFunction→({Black,Text[#2,#1]}&)] *)
```

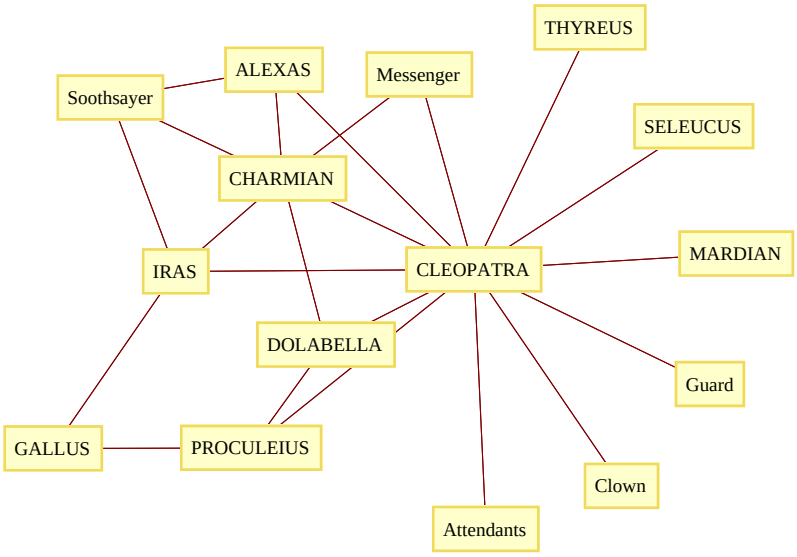
In[265]:=
```
g = GraphPlot[gmap[[2]], VertexLabeling → True, MultiedgeStyle → None, ImageSize → 600]
```

Out[265]=



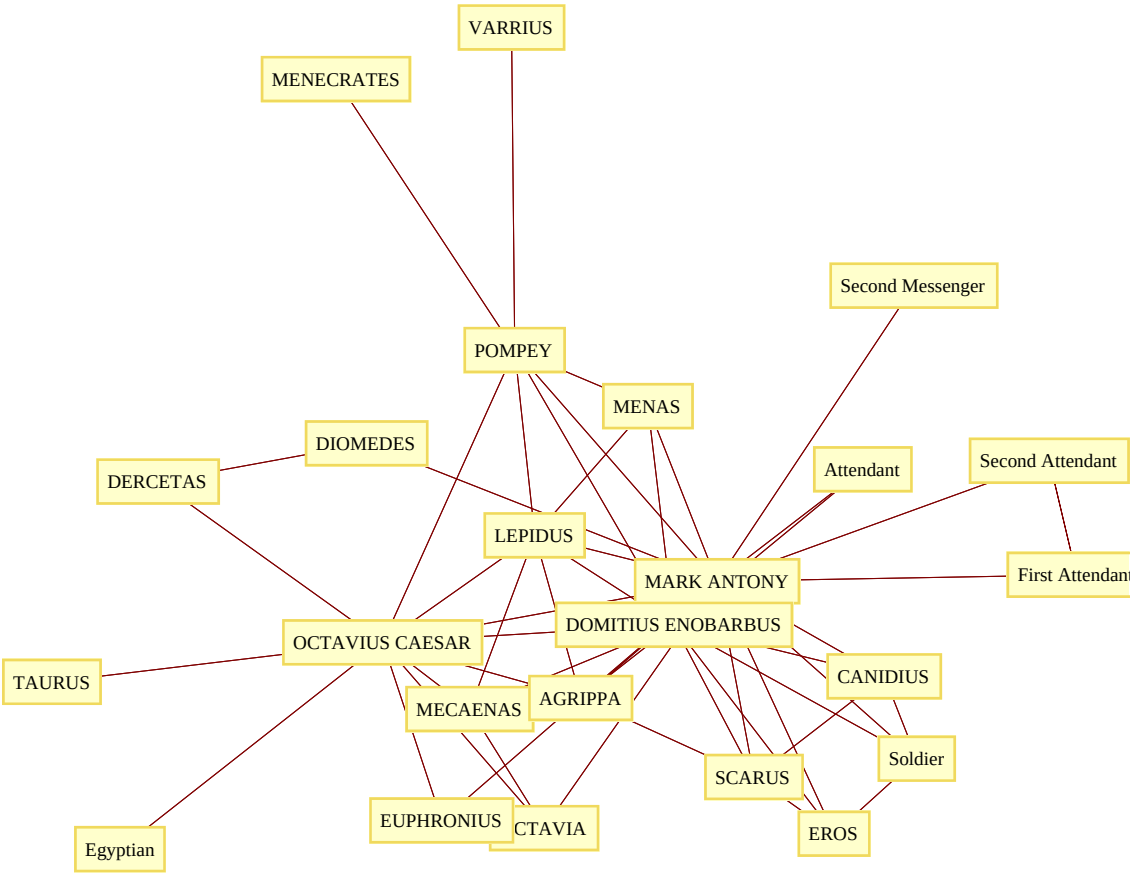
In[266]:=
```
keypersons1 = groups[[2]]; keypersons2 = groups[[3]];
sub1 = Select[gmap[[2]], MemberQ[keypersons1, #[[1]]] && MemberQ[keypersons1, #[[2]]] &];
GraphPlot[sub1, VertexLabeling → True, MultiedgeStyle → None, ImageSize → 400]
```

Out[268]=



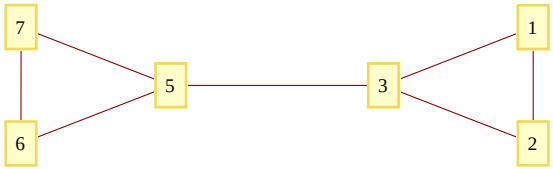
In[269]:=
```
sub2 = Select[gmap[[2]], MemberQ[keypersons2, #[[1]]] && MemberQ[keypersons2, #[[2]]] &];
GraphPlot[sub2, VertexLabeling → True, MultiedgeStyle → None, ImageSize → 600]
```

Out[270]=



In[271]:=
```
g = {3 → 2, 2 → 1, 1 → 3, 3 → 5, 5 → 6, 6 → 7, 7 → 5};
GraphPlot[g, VertexLabeling → True]
```

Out[272]=



In[273]:=
```
CommunityStructurePartition[g]
```

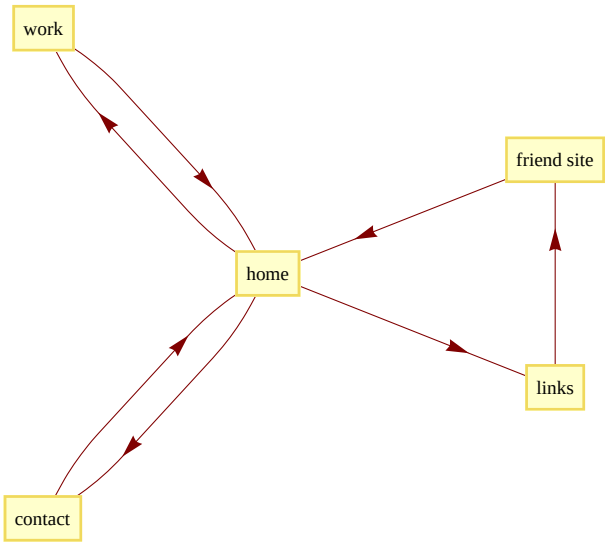
Out[273]=
```
{{3, 2, 1}, {5, 6, 7}}
```

http://ja.wikipedia.org/wiki/ページランク
http://en.wikipedia.org/wiki/LinkRank

In[274]:=
```
g = {"home" → "contact", "home" → "links", "home" → "work", "links" → "friend site",
    "friend site" → "home", "work" → "home", "contact" → "home"};
```

In[275]:=
```
GraphPlot[g, VertexLabeling → True, DirectedEdges → True]
```

Out[275]=



In[276]:=
```
{VertexList[g], AdjacencyMatrix[g] // MatrixForm}
```

Out[276]=
$$\left\{\{\text{home}, \text{contact}, \text{links}, \text{work}, \text{friend site}\}, \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}\right\}$$

In[277]:=
```
m = AdjacencyMatrix[g];
```

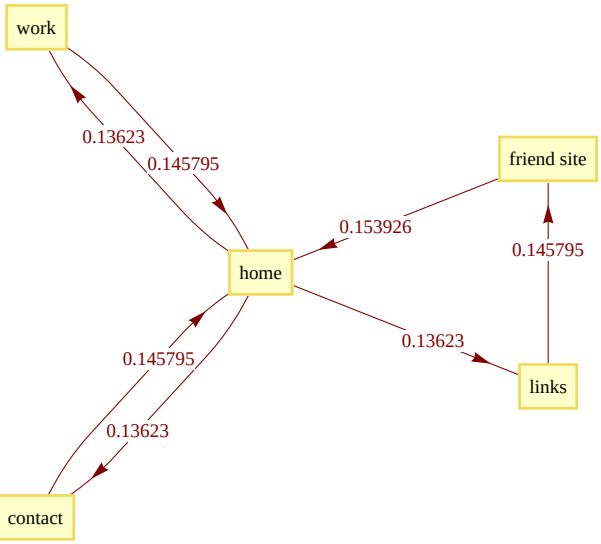
In[278]:=

In[279]:=
```
LinkRankMatrix[g] // MatrixForm
```

Out[279]//MatrixForm=
$$\begin{pmatrix} 0 & 0.13623 & 0.13623 & 0.13623 & 0 \\ 0.145795 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.145795 \\ 0.145795 & 0 & 0 & 0 & 0 \\ 0.153926 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[280]:=
```
GraphPlot[g, VertexLabeling → True,
 EdgeRenderingFunction → ({RGBColor[0.5, 0., 0.], Arrowheads[{{0.03, 0.8}}],
     Arrow[#1], Text[LinkRankMatrix[g][[First[#4], Last[#4]]],
      LineScaledCoordinate[#1, .6], Background → White]} &), DirectedEdges → True]
```

Out[280]=



In[281]:=
```
PageRanks[g]
```

Out[281]=
```
{home → 0.408689, contact → 0.145795,
 links → 0.145795, work → 0.145795, friend site → 0.153926}
```

In[282]:=
```
Total[#[[2]] & /@ PageRanks[g]]
```

Out[282]=
```
1.
```

同一対象に対する複数人の判断傾向を調べる

ref.
"Correlated Binary Decision Rules" from The Wolfram Demonstrations Project
http://demonstrations.wolfram.com/CorrelatedBinaryDecisionRules/

In[283]:=
```
judges = 6;
decisionfactors = 5;
clusters = 0.1; (* 0..1 *)
```

判断するのに参考にした３つの要因に関し，それぞれYes/Noの選択があり，それぞれに選択に対しさらにYes/Noと選択を行うと仮定すると判断に関し2^3の組み合わせが存在する

In[286]:=
```
SeedRandom[82 409]; ri = RandomInteger[{0, 1}, {judges, 2^decisionfactors}]
```

Out[286]=
```
{{1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0},
 {0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0},
 {1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1},
 {0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1},
 {1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1},
 {0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0}}
```

非構造的クラスタ分類においては，あらかじめクラスタ数を指定することが多い．ただし一般にどう決めればいいかの定義は存在しない．そのため，下記では以下のように定めた．

```
In[287]:= clusterCount[judges_, clusters_] :=
            Round@Rescale[Sqrt@clusters, {0, 1}, {0.51, judges + 0.49}];

In[288]:= clusterCount[judges, clusters]

Out[288]= 2
```

xの値を{a, b}にスケールし{c, d}の間の数に写像する。

```
In[304]:= Clear[x];

In[305]:= Rescale[x, {a, b}, {c, d}]

Out[305]= -(-b c + a d)/(-a + b) + (-c + d) x/(-a + b)
```

非構造的クラスタ分類では最初に仮のクラスタを作る．そのためここではFindCluster[]を使用し仮分類しておく．

```
In[290]:= fc = FindClusters[ri, clusterCount[judges, clusters], DistanceFunction → HammingDistance]

Out[290]= {{{1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0},
             {0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0},
             {0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0}},
           {{1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1},
             {0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1},
             {1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1}}}
```

相関係数は何らかの方法であらかじめ求めておくことも可能だが，ここでは仮設定を行って使用する．

```
In[291]:= k = 0.5; (*相関係数", 0..1,0.01 *)
```

■ mean

あらかじめクラス分けした際の元データは一様乱数で作成。それに対し、合議形成に集団の平均的判断があると仮定して仮のジャッジを補正．

```
In[292]:= clusteredJudges =
            Map[Round, Map[With[{μ = Mean[#]}, Map[k μ + (1 - k) # &, #]] &, fc], {2}] // N

Out[292]= {{{1., 1., 0., 0., 0., 0., 1., 1., 0., 1., 1., 1., 0., 0.,
               0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0.},
             {0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1.,
              1., 1., 0., 1., 0., 0., 1., 1., 1., 1., 0.}, {0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 1.,
              1., 1., 0., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 0.}},
           {{1., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0.,
              0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1.}, {0., 0., 1., 0., 0., 1., 1., 1., 1., 0.,
              1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 1.},
             {1., 0., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 0., 1., 1., 1.,
              1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.}}}
```

以下の結果を見ればわかるように平均的距離を保ってる．

```
In[293]:= clusterLengths = Length /@ clusteredJudges;
          dm = HierarchicalClustering`DistanceMatrix[
             Flatten[clusteredJudges, 1], DistanceFunction → HammingDistance] // MatrixForm

Out[294]//MatrixForm=
          0.  15  13  19  15  15
          15  0.  14  24  20  16
          13  14  0.  20  12  16
          19  24  20  0.  18  14
          15  20  12  18  0.  12
          15  16  16  14  12  0.
```

leader

合議形成にリーダの判断が影響したと仮定し仮のジャッジを補正.

In[295]:=
```
takeProportion[z_List, k_ /; 0 ≤ k ≤ 1] :=
  Take[z, Round@Rescale[k, {0, 1}, {-0.49, Length[z] + 0.49}]];
```

In[296]:=
```
clusterSweepTowards[cluster_, leader_, indicesList_, k_] := ReplacePart[cluster, Join @@
    Map[# → cluster[[leader, #[[2]]]] &, Map[takeProportion[#, k] &, indicesList], {2}]];
(* リーダと異なる判断を行った部分の位置リストを作成 *)
clusterSweepTowards[cluster_, leader_, k_] :=
  clusterSweepTowards[cluster, leader, GatherBy[Position[
     MapIndexed[#1 === cluster[[leader, #2[[2]]]] &, cluster, {2}], False, {2}], First], k];
```

In[298]:=
```
clusteredJudges = clusterSweepTowards[#, 2, k] & /@ fc
```

Out[298]=
```
{{{0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0},
  {0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0},
  {0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0}},
 {{0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1},
  {0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1},
  {0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1}}}
```

以下の結果を見ればわかるように判決に関しいくつかのグループに分かれることがわかる.

In[299]:=
```
clusterLengths = Length /@ clusteredJudges;
dm = HierarchicalClustering`DistanceMatrix[
    Flatten[clusteredJudges, 1], DistanceFunction → HammingDistance] // MatrixForm
```

Out[300]//MatrixForm=

$$\begin{pmatrix} 0 & 7 & 6 & 18 & 15 & 13 \\ 7 & 0 & 7 & 19 & 20 & 16 \\ 6 & 7 & 0 & 18 & 15 & 17 \\ 18 & 19 & 18 & 0 & 9 & 9 \\ 15 & 20 & 15 & 9 & 0 & 6 \\ 13 & 16 & 17 & 9 & 6 & 0 \end{pmatrix}$$

In[301]:=
```
GatherBy[{1, 2, 3, 4, 5}, OddQ]
```

Out[301]=
```
{{1, 3, 5}, {2, 4}}
```

```
{{1, 3, 5}, {2, 4}}
```

```
{{1, 3, 5}, {2, 4}}
```

In[302]:=
```
With[{leader = 1, cluster = clusteredJudges[[1]]},
 MapIndexed[#1 === cluster[[leader, #2[[2]]]] &, cluster, {2}]]
```

Out[302]=
```
{{True, True, True, True, True, True, True, True, True,
  True, True, True, True, True, True, True, True, True, True, True,
  True, True, True, True, True, True, True, True, True, True, True},
 {True, True, True, True, True, True, True, True, True, True, True, True,
  True, True, True, False, False, True, False, True, True, False,
  False, True, True, True, False, True, True, True, False, True},
 {True, True, True, True, True, True, True, True, True, True, True, True,
  True, True, True, False, True, True, False, False, True, True,
  True, True, False, False, True, True, True, True, False, True}}
```

In[303]:=
```
With[{leader = 1, cluster = clusteredJudges[[1]]},
 Position[MapIndexed[#1 === cluster[[leader, #2[[2]]]] &, cluster, {2}], False, {2}]]
```

Out[303]=
```
{{2, 16}, {2, 17}, {2, 19}, {2, 22}, {2, 23}, {2, 27},
 {2, 31}, {3, 16}, {3, 19}, {3, 20}, {3, 25}, {3, 26}, {3, 31}}
```


```
{{2, 16}, {2, 17}, {2, 19}, {2, 22}, {2, 23}, {2, 27},
 {2, 31}, {3, 16}, {3, 19}, {3, 20}, {3, 25}, {3, 26}, {3, 31}}
```

```
{{2, 16}, {2, 17}, {2, 19}, {2, 22}, {2, 23}, {2, 27},
 {2, 31}, {3, 16}, {3, 19}, {3, 20}, {3, 25}, {3, 26}, {3, 31}}
```