```
(*
Instructor : Hiroyuki Akama
ComplexNetworks*)
```

In[218]:=
```
<< Combinatorica`
```

In[219]:=
```
? KSubsets
? RandomSample
list1 = Part[KSubsets[Range[10], 2], #] & /@ RandomSample[Range[10 * 9 * 0.5], 20] // Sort
```

KSubsets[*l*, *k*] gives all subsets of set l containing exactly k elements, ordered lexicographically. ≫

RandomSample[{e_1, e_2, …}, *n*] gives a pseudorandom sample of *n* of the e_i.
RandomSample[{w_1, w_2, …} –> {e_1, e_2, …}, *n*]
gives a pseudorandom sample of *n* of the e_i chosen using weights w_i.
RandomSample[{e_1, e_2, …}] gives a pseudorandom permutation of the e_i. ≫

Out[221]=
```
{{1, 2}, {1, 6}, {1, 7}, {2, 4}, {2, 5}, {2, 7}, {2, 8}, {3, 6}, {3, 10}, {4, 5},
 {4, 7}, {4, 8}, {4, 9}, {4, 10}, {5, 6}, {5, 8}, {5, 10}, {6, 7}, {6, 8}, {6, 9}}
```

In[222]:=
```
adjacencymatrixbylist1 = list1 // FromOrderedPairs // ToAdjacencyMatrix
graphobjectbylist1 = adjacencymatrixbylist1 // FromAdjacencyMatrix
? FromOrderedPairs
? ToAdjacencyMatrix
? FromAdjacencyMatrix
```

Out[222]=
```
{{0, 1, 0, 0, 0, 1, 1, 0, 0, 0}, {0, 0, 0, 1, 1, 0, 1, 1, 0, 0},
 {0, 0, 0, 0, 0, 1, 0, 0, 0, 1}, {0, 0, 0, 0, 1, 0, 1, 1, 1, 1},
 {0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, {0, 0, 0, 0, 0, 0, 1, 1, 1, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
```

Out[223]=
```
- Graph:< 20,10,Undirected >-
```

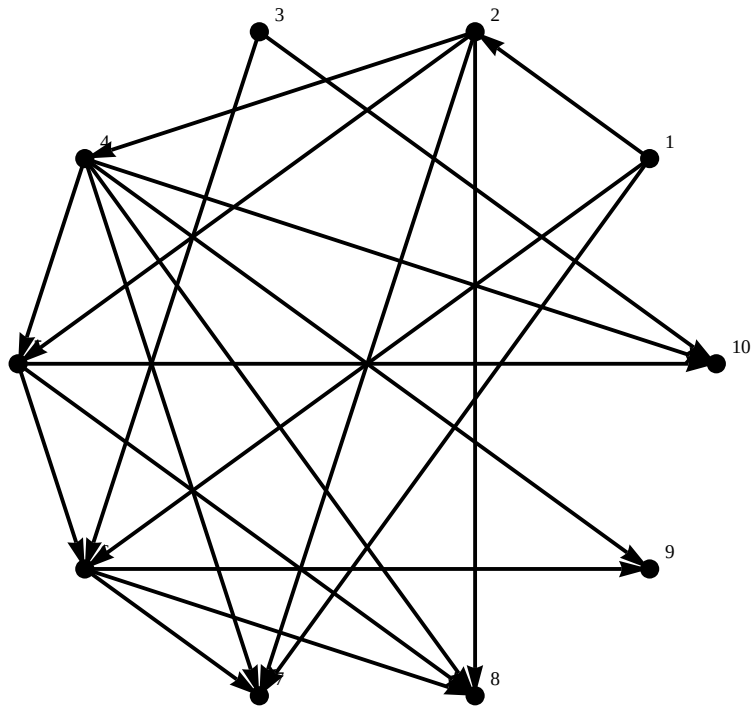
FromOrderedPairs[*l*] constructs an edge list
representation from a list of ordered pairs l, using a circular embedding.
FromOrderedPairs[*l*, *v*] uses v as the embedding for the resulting graph. The option Type that takes
on values Undirected or Directed can be used to affect the kind of graph produced. The default
value of Type is Directed. Type –> Undirected results in the underlying undirected graph. ≫

ToAdjacencyMatrix[*g*] constructs an adjacency matrix representation for graph g.
ToAdjacencyMatrix[*g*, EdgeWeight] returns edge weights
as entries of the adjacency matrix with Infinity representing missing edges. ≫

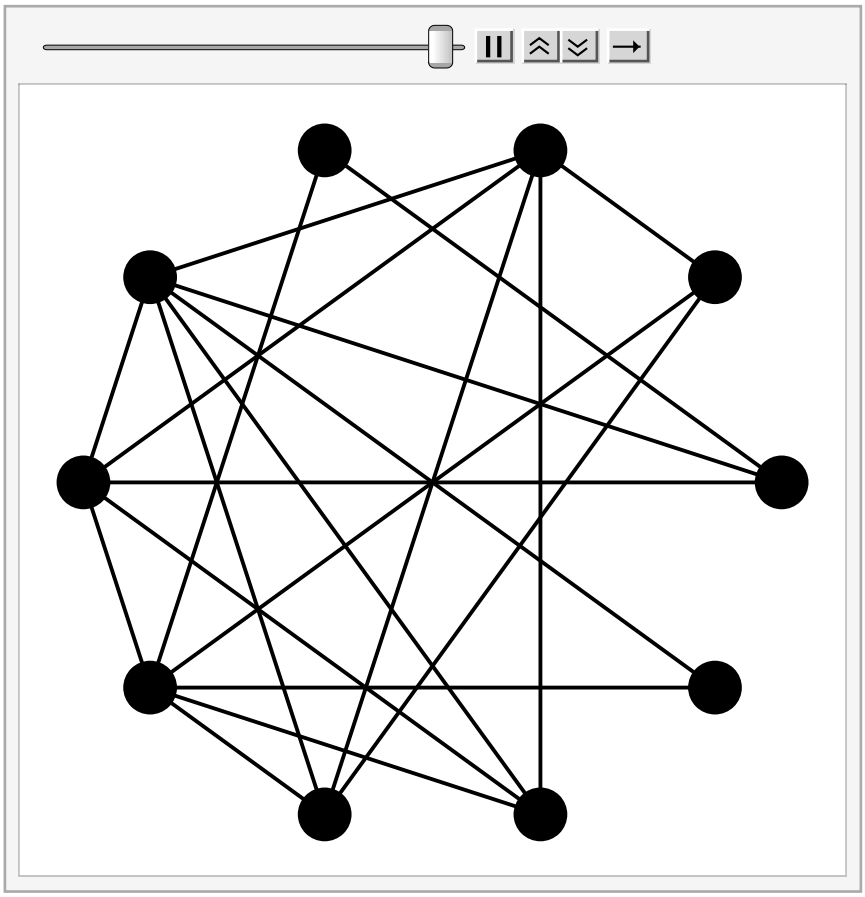
FromAdjacencyMatrix[*m*] constructs a graph from a given adjacency matrix m, using a circular embedding.
FromAdjacencyMatrix[*m*, *v*] uses v as the embedding for the resulting
graph. An option Type that takes on the values Directed or Undirected can be used
to affect the type of graph produced. The default value of Type is Undirected.
FromAdjacencyMatrix[*m*, EdgeWeight] interprets the entries in m as edge weights, with infinity
representing missing edges, and from this constructs a weighted graph using a circular embedding.
FromAdjacencyMatrix[*m*, *v*, EdgeWeight] uses v as the embedding for the resulting
graph. The option Type can be used along with the EdgeWeight tag. ≫

In[227]:=
```
ShowGraph[graphobjectbylist1, VertexLabel → True, EdgeDirection → True]
AnimateGraph[graphobjectbylist1, Range[10]]
? ShowGraph
? AnimateGraph
```

Out[227]=



Out[228]=

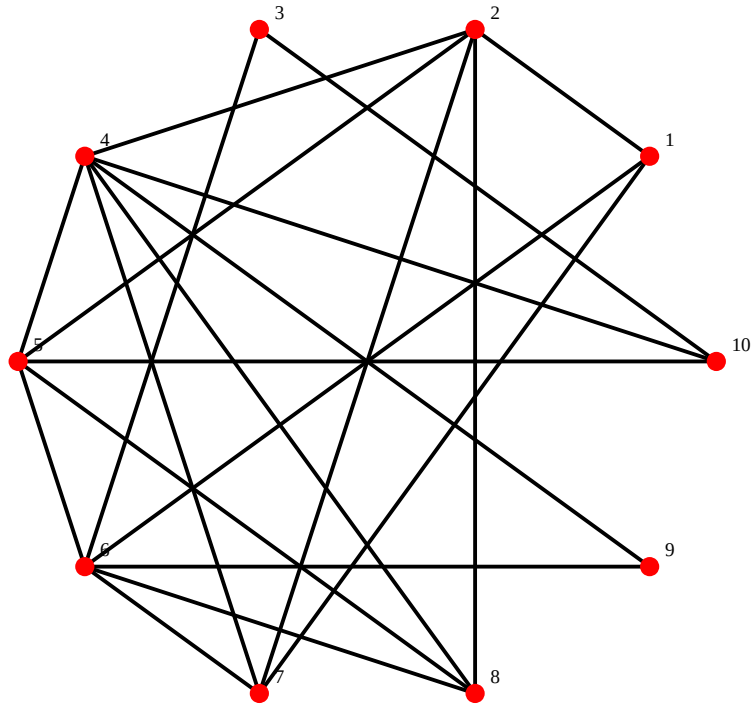


ShowGraph[*g*] displays the graph g.
ShowGraph[*g*, Directed] is obsolete and it is currently identical to ShowGraph[*g*]. ≫

AnimateGraph[*g*, *l*] displays graph g with each element in the list l successively highlighted. Here l is a list containing vertices and edges of g. An optional flag, which takes on the values All and One, can be used to inform the function about whether objects highlighted earlier will continue to be highlighted or not. The default value of flag is All. All the options allowed by the function Highlight are permitted by AnimateGraph, as well. See the usage message of Highlight for more details. ≫

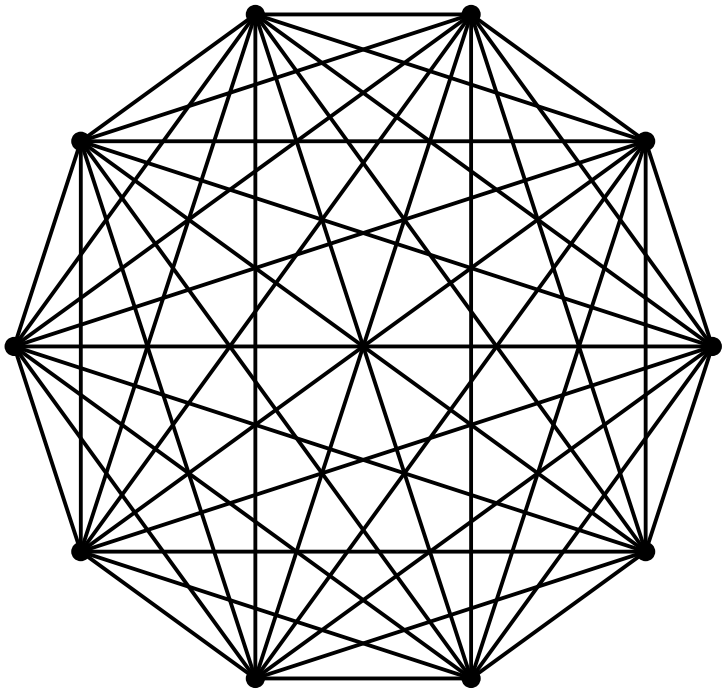
In[231]:=
```
ShowGraph[FromAdjacencyMatrix[ToAdjacencyMatrix[FromUnorderedPairs[list1]]],
 VertexLabel → True, VertexColor → Red]
```

Out[231]=



In[232]:=
```
CompleteGraph[10] // ShowGraph
?CompleteGraph
```

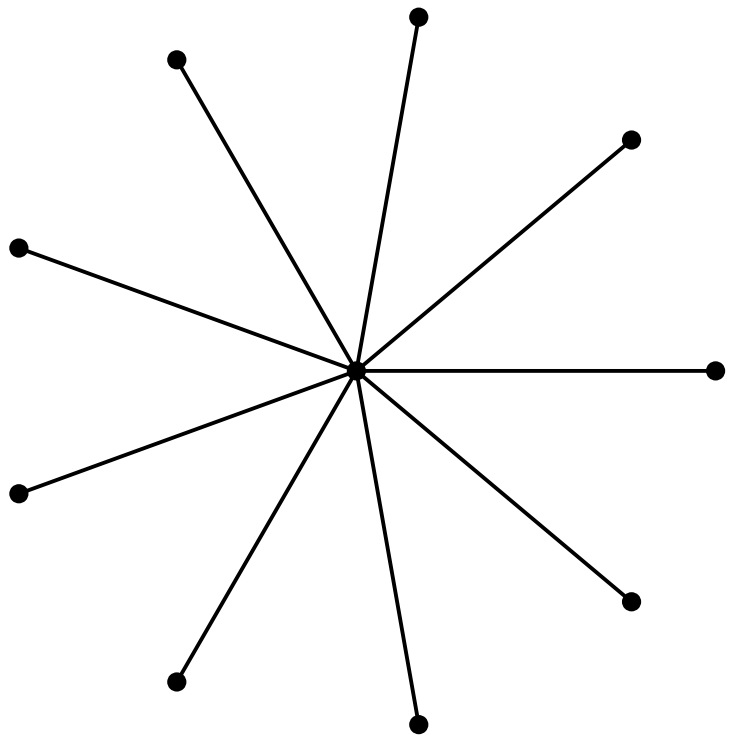
Out[232]=



CompleteGraph[*n*] creates a complete graph on n vertices. An option Type that takes on the values Directed or Undirected is allowed. The default setting for this option is Type –> Undirected.
CompleteGraph[*a*, *b*, *c*, …] creates a complete k–partite graph of the prescribed shape. The use of CompleteGraph to create a complete k–partite graph is obsolete; use CompleteKPartiteGraph instead. ≫

In[236]:=
```
Star[10] // ShowGraph
? Star
```

Out[236]=



Star[*x*, *y*, …] displays as *x*∗*y*∗… . ≫

In[238]:=
```
ToAdjacencyMatrix[Star[10]]
```

Out[238]=
```
{{0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 0}}
```

In[239]:=
```
? Degrees
Degrees[CompleteGraph[10]]
Degrees[Star[10]]
```

Degrees[*g*] returns the degrees of vertex 1, 2, 3, … in that order. ≫

Out[240]= {9, 9, 9, 9, 9, 9, 9, 9, 9, 9}

Out[241]= {1, 1, 1, 1, 1, 1, 1, 1, 1, 9}

In[242]:=
```
tmp = Table[Table[Random[BinomialDistribution[1, 0.2]], {i, 1, 30}], {i, 1, 30}];
adjacencymatrix2 = (tmp + Transpose[tmp]) /. x_ /; x ≠ 0 → 1
SymmetricQ[%]
? SymmetricQ
```

Out[243]=
```
{{0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1},
 {0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1},
 {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1},
 {1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1},
 {0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0},
 {1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0},
 {0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0},
 {1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0},
 {1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0},
 {1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0},
 {1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1},
 {0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1},
 {0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0},
 {1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0},
 {1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0},
 {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0},
 {0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0},
 {0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1},
 {1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1},
 {0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1},
 {1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0},
 {1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1},
 {1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1},
 {0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0},
 {1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0},
 {0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0},
 {1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0},
 {1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1},
 {1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0}}
```

Out[244]= True

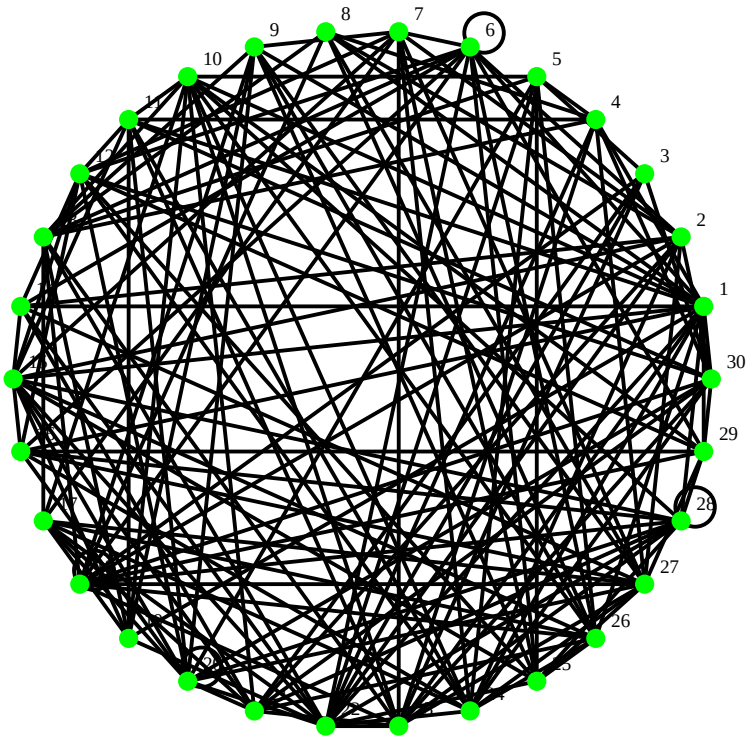
SymmetricQ[*r*] tests if a given square matrix r represents a symmetric relation.
SymmetricQ[*g*] tests if the edges of a given graph represent a symmetric relation. ≫

In[246]:=
```
listfromadjacencymatrix2 = ToUnorderedPairs[FromAdjacencyMatrix[adjacencymatrix2]]
```

Out[246]=
```
{{1, 4}, {1, 6}, {1, 8}, {1, 9}, {1, 10}, {1, 11}, {1, 14}, {1, 15}, {1, 16}, {1, 19},
 {1, 22}, {1, 23}, {1, 24}, {1, 26}, {1, 28}, {1, 29}, {1, 30}, {2, 6}, {2, 7}, {2, 8},
 {2, 14}, {2, 15}, {2, 18}, {2, 21}, {2, 22}, {2, 24}, {2, 27}, {2, 30}, {3, 5}, {3, 18},
 {3, 21}, {3, 22}, {3, 23}, {3, 30}, {4, 5}, {4, 8}, {4, 11}, {4, 13}, {4, 22}, {4, 23},
 {4, 27}, {4, 30}, {5, 10}, {5, 13}, {5, 20}, {5, 22}, {5, 24}, {5, 25}, {5, 26}, {6, 6},
 {6, 7}, {6, 12}, {6, 13}, {6, 14}, {6, 16}, {6, 18}, {6, 22}, {6, 25}, {6, 26}, {6, 29},
 {7, 9}, {7, 12}, {7, 15}, {7, 17}, {7, 23}, {7, 24}, {7, 25}, {7, 27}, {8, 13}, {8, 18},
 {8, 26}, {8, 27}, {9, 13}, {9, 18}, {9, 19}, {9, 20}, {9, 25}, {9, 26}, {9, 27},
 {10, 11}, {10, 13}, {10, 17}, {10, 19}, {10, 21}, {10, 23}, {10, 24}, {10, 26}, {10, 28},
 {10, 29}, {11, 12}, {11, 15}, {11, 19}, {11, 20}, {11, 23}, {11, 24}, {11, 30}, {12, 14},
 {12, 15}, {12, 16}, {12, 20}, {12, 24}, {12, 25}, {12, 29}, {12, 30}, {13, 17}, {13, 18},
 {13, 19}, {13, 22}, {13, 23}, {14, 15}, {14, 21}, {14, 22}, {14, 27}, {15, 16}, {15, 19},
 {15, 20}, {15, 21}, {15, 22}, {15, 25}, {15, 26}, {15, 28}, {16, 20}, {16, 24}, {16, 27},
 {16, 28}, {16, 29}, {17, 19}, {17, 20}, {17, 21}, {17, 22}, {17, 24}, {17, 26}, {17, 27},
 {18, 18}, {18, 23}, {18, 26}, {18, 27}, {18, 28}, {18, 29}, {18, 30}, {19, 22}, {20, 20},
 {20, 21}, {20, 22}, {20, 27}, {20, 28}, {20, 30}, {21, 23}, {21, 26}, {21, 28}, {21, 30},
 {22, 23}, {22, 24}, {22, 25}, {22, 28}, {23, 27}, {23, 28}, {23, 30}, {24, 26}, {24, 27},
 {24, 30}, {25, 26}, {25, 27}, {25, 28}, {27, 29}, {28, 28}, {28, 29}, {29, 30}}
```

In[247]:=
```
ShowGraph[FromAdjacencyMatrix[adjacencymatrix2], VertexLabel → True, VertexColor → Green]
```

Out[247]=



In[248]:=
```
? ShortestPath
```

ShortestPath[*g*, *start*, *end*] finds a shortest path between vertices *start* and *end* in graph g. ≫

In[249]:=
```
graphobject2 = FromAdjacencyMatrix[adjacencymatrix2];
ShortestPath[graphobject2, 1, 20]
```

Out[250]= {1, 9, 20}

In[251]:=
```
? NetworkFlow
NetworkFlow[graphobject2, 1, 5, Edge]
ShortestPath[graphobject2, 1, 5]
```

NetworkFlow[*g*, *source*, *sink*] returns the value of a maximum flow through graph g from *source* to *sink*.
NetworkFlow[*g*, *source*, *sink*, Edge] returns the edges in g that
have positive flow along with their flows in a maximum flow from *source* to *sink*.
NetworkFlow[*g*, *source*, *sink*, Cut] returns a minimum cut between *source* and *sink*.
NetworkFlow[*g*, *source*, *sink*, All] returns the adjacency
list of g along with flows on each edge in a maximum flow from *source* to *sink*. ≫

Out[252]=
```
{{{1, 4}, 1}, {{1, 6}, 1}, {{1, 9}, 1}, {{1, 10}, 1}, {{1, 15}, 1},
 {{1, 22}, 1}, {{1, 23}, 1}, {{1, 24}, 1}, {{1, 26}, 1}, {{3, 5}, 1}, {{4, 5}, 1},
 {{6, 13}, 1}, {{9, 20}, 1}, {{10, 5}, 1}, {{13, 5}, 1}, {{15, 25}, 1},
 {{20, 5}, 1}, {{22, 5}, 1}, {{23, 3}, 1}, {{24, 5}, 1}, {{25, 5}, 1}, {{26, 5}, 1}}
```

Out[253]= {1, 4, 5}

In[254]:=
```
Degrees[graphobject2]
```

Out[254]=
```
{17, 11, 6, 9, 9, 13, 10, 7, 9, 12, 10, 11, 11,
 8, 14, 9, 10, 13, 8, 13, 11, 16, 13, 13, 10, 12, 14, 12, 9, 12}
```

In[255]:=
```
sparseadjacencymatrix2 = SparseArray[adjacencymatrix2, {30, 30}]
```

Out[255]=
```
SparseArray[<332>, {30, 30}]
```

In[256]:=
```
Degrees[FromAdjacencyMatrix[sparseadjacencymatrix2]]
```

Out[256]=
```
Degrees[FromAdjacencyMatrix[SparseArray[<332>, {30, 30}]]]
```

In[257]:=
```
Plus @@ # & /@ adjacencymatrix2
Length[#] & /@
 (Split[#[[1]] & /@ Most[ArrayRules[sparseadjacencymatrix2]], #1[[1]] == #2[[1]] &])
```

Out[257]=
```
{17, 11, 6, 9, 9, 13, 10, 7, 9, 12, 10, 11, 11,
 8, 14, 9, 10, 13, 8, 13, 11, 16, 13, 13, 10, 12, 14, 12, 9, 12}
```

Out[258]=
```
{17, 11, 6, 9, 9, 13, 10, 7, 9, 12, 10, 11, 11,
 8, 14, 9, 10, 13, 8, 13, 11, 16, 13, 13, 10, 12, 14, 12, 9, 12}
```

In[259]:=
```
? ConnectedQ
ConnectedQ[FromAdjacencyMatrix[adjacencymatrix2]]
sparseadjacencymatrix2 = SparseArray[adjacencymatrix2]
ConnectedQ[FromAdjacencyMatrix[sparseadjacencymatrix2]]
```

ConnectedQ[*g*] yields True if undirected graph g is connected. If g is directed, the function returns True if the underlying undirected graph is connected.
ConnectedQ[*g*, Strong] yields True if the directed graph g is strongly connected.
ConnectedQ[*g*, Weak] yields True if the directed graph g is weakly connected. ≫

Out[260]=
```
True
```

Out[261]=
```
SparseArray[<332>, {30, 30}]
```

Out[262]=
```
ConnectedQ[FromAdjacencyMatrix[SparseArray[<332>, {30, 30}]]]
```

In[263]:=
```
SparseConnectedQ[sparsearray_] :=
  ConnectedQ[FromUnorderedPairs[Most[First[#] & /@ ArrayRules[sparsearray]]]];
SparseConnectedQ[sparseadjacencymatrix2]
```

Out[264]=
```
True
```

In[265]:=
```
? ConnectedComponents
```

ConnectedComponents[*g*] gives the vertices of graph g partitioned into connected components. ≫

In[268]:=
```
tmp2 = Table[Table[RandomInteger[RandomInteger[10]], {i, 1, 30}], {i, 1, 30}];
weightedadj = SparseArray[tmp2 + Transpose[tmp2]]
```

Out[269]=
```
SparseArray[<821>, {30, 30}]
```

In[270]:=
```
Uniqrule[list_, th_] := # /. (x_ /; x ≥ th → 1) & /@ (# /. (x_ /; x < th → 0) & /@ list)
```

In[271]:=
```
ruleofweightedadj = Most[ArrayRules[weightedadj]];
newrule = MapThread[#1 → #2 &,
   {#[[1]] & /@ ruleofweightedadj, Uniqrule[#[[2]] & /@ ruleofweightedadj, 9]}];
SparseArray[newrule, {30, 30}]
```

Out[273]=
```
SparseArray[<141>, {30, 30}]
```

In[274]:=
```
SparseArrayThreshold[sparsearray_, threshold_] := Module[{}, ars = ArrayRules[sparsearray];
   MapThread[#1 → #2 &, {#[[1]] & /@ ars, Uniqrule[#[[2]] & /@ ars, threshold]}]] // SparseArray
```

In[275]:=
```
SparseArrayRemoveDiagonal[sparsearray_] := Module[{}, ars = ArrayRules[sparsearray];
    # /. (x_ /; #[[1]][[1]] == #[[1]][[2]] → (#[[1]] → 0)) & /@ ars] // SparseArray;
SparseArrayRemoveDiagonal[SparseArray[newrule, {30, 30}]]
```

Out[276]=
```
SparseArray[<136>, {30, 30}]
```

In[277]:=
```
Eachcurv[sparseadjacencymatrix_, nodenum_] :=
 Module[{ar, pairs, subset, lengthsubset}, ar = ArrayRules[sparseadjacencymatrix];
  pairs = #[[1, 2]] & /@ Select[ar, #[[1]][[1]] == nodenum &];
  subset = KSubsets[pairs, 2];
  lengthsubset = Length[subset];
  If[subset == {}, 0 // N, (Length[Intersection[#[[1]] & /@ ar, subset]] / lengthsubset) // N]]
```

General::spell1 :
New symbol name "subset" is similar to existing symbol "Subset" and may be misspelled. ≫

In[278]:=
```
Eachcurv[SparseArray[ToAdjacencyMatrix[Star[20]]], #] & /@ Range[20]
```

Out[278]=
```
{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}
```

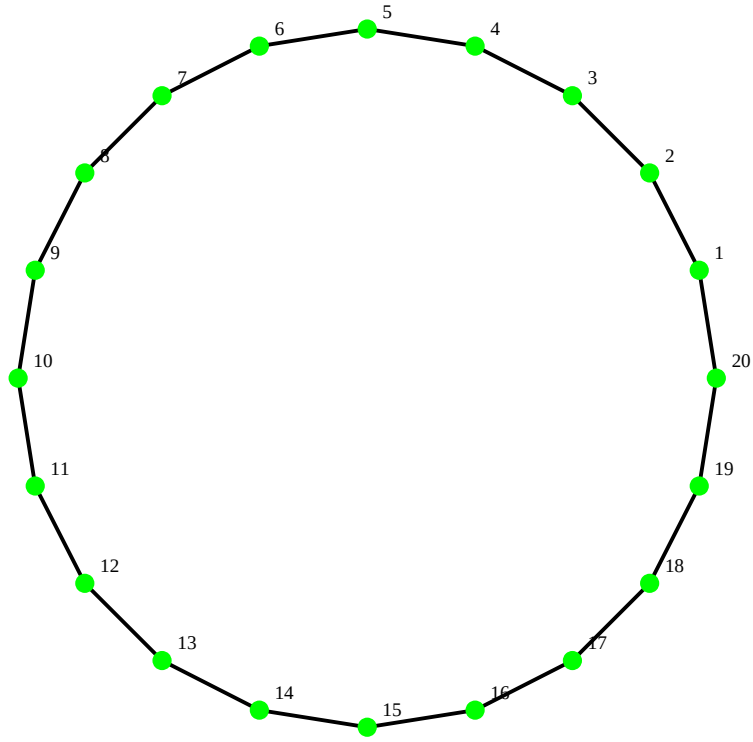
In[279]:=
```
Eachcurv[SparseArray[ToAdjacencyMatrix[CompleteGraph[20]]], #] & /@ Range[20]
```

Out[279]=
```
{1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.}
```

In[280]:=
```
? Cycle
ShowGraph[Cycle[20], VertexLabel → True, VertexColor → Green]
Eachcurv[SparseArray[ToAdjacencyMatrix[Cycle[20]]], #] & /@ Range[20]
```

Cycle[*n*] constructs the cycle on n vertices, the 2−regular connected graph. An option Type that takes on values Directed or Undirected is allowed. The default setting is Type −> Undirected. ≫

Out[281]=

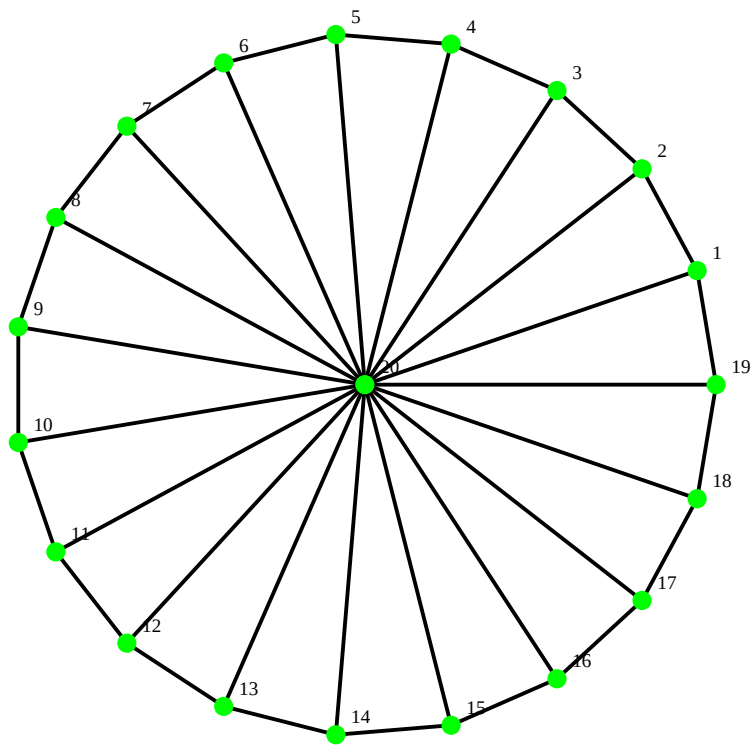


Out[282]= {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}

In[283]:=
```
? Wheel
ShowGraph[Wheel[20], VertexLabel → True, VertexColor → Green]
Eachcurv[SparseArray[ToAdjacencyMatrix[Wheel[20]]], #] & /@ Range[20]
% // Mean
```

Wheel[n] constructs a wheel on n vertices, which is the join of CompleteGraph[1] and Cycle[$n-1$]. ≫

Out[284]=



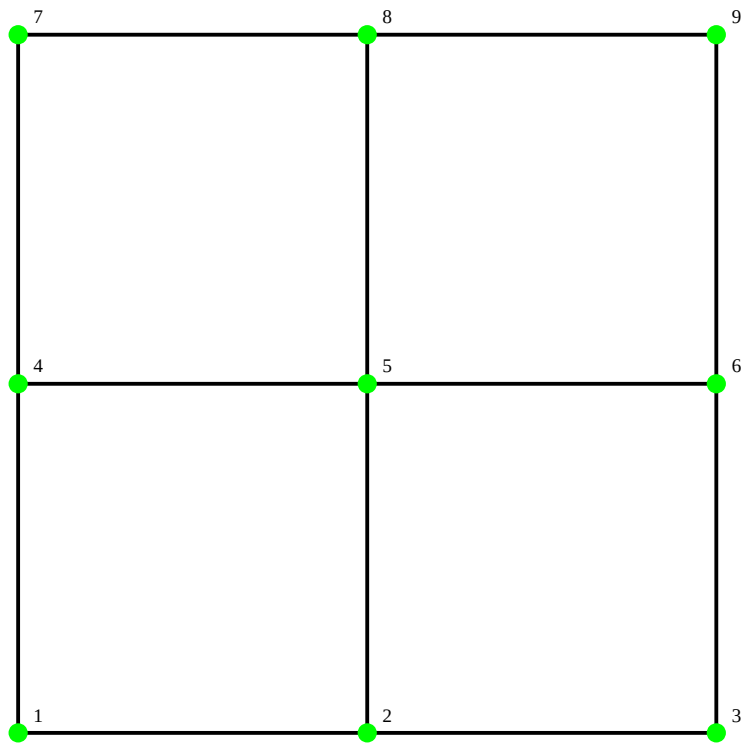
Out[285]= {0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.666667, 0.111111}

Out[286]= 0.638889

In[287]:=
```
? GridGraph
ShowGraph[GridGraph[3, 3], VertexLabel → True, VertexColor → Green]
Eachcurv[SparseArray[ToAdjacencyMatrix[GridGraph[3, 3]]], #] & /@ Range[9]
% // Mean
```

GridGraph[*n*, *m*] constructs an n∗m grid graph, the product of paths on n and m vertices.
GridGraph[*p*, *q*, *r*] constructs a p∗q∗r grid graph, the product of GridGraph[*p*, *q*] and a path of length r. ≫

Out[288]=



Out[289]= {0., 0., 0., 0., 0., 0., 0., 0., 0.}

Out[290]= 0.

In[291]:=
```
ClusteringCoefficient[sparseadjacencymatrix_] :=
  Eachcurv[sparseadjacencymatrix, #] & /@ Range[Length[sparseadjacencymatrix]] // Mean;
```

In[292]:=
```
ClusteringCoefficient[sparseadjacencymatrix2]
```

Out[292]= 0.394589

In[293]:=
```
adjacencymatrixsample1 = {{0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1}, {1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
   {1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0},
   {0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0}, {0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0},
   {0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0}, {0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},
   {1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0}};
```

In[294]:=
```
? ShortestPath
ShortestPath[FromAdjacencyMatrix[adjacencymatrixsample1], 1, 9]
```

ShortestPath[*g*, *start*, *end*] finds a shortest path between vertices *start* and *end* in graph g. ≫

Out[295]= {1, 11, 9}

In[296]:=
```
sparseadjacencymatrixsample1 = SparseArray[adjacencymatrixsample1]
```

Out[296]=
```
SparseArray[<38>, {11, 11}]
```

In[297]:=
```
? CompleteGraph
```

CompleteGraph[*n*] creates a complete graph on n vertices. An option Type that takes on the values Directed or Undirected is allowed. The default setting for this option is Type –> Undirected.
CompleteGraph[*a*, *b*, *c*, …] creates a complete k–partite graph of the prescribed shape. The use of CompleteGraph to create a complete k–partite graph is obsolete; use CompleteKPartiteGraph instead. ≫

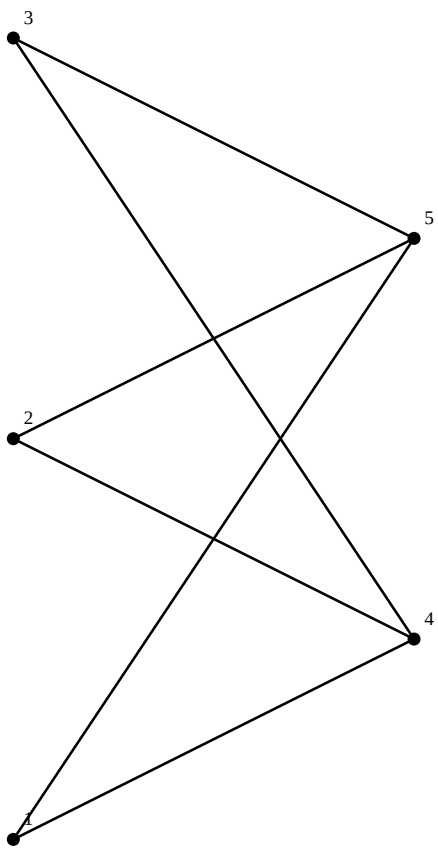
In[298]:=
```
SparseCompleteGraph[n_, opt_] :=
  Module[{bipartiterule = {i_, j_} /; ((i ≤ n && j ≥ n + 1) || (i ≥ n + 1 && j ≤ n)) → 1},
   SparseArray[bipartiterule, {n + opt, n + opt}]];
```

In[299]:=
```
CompleteGraph[3, 2] // ToAdjacencyMatrix // MatrixForm
ShowGraph[CompleteGraph[3, 2], VertexLabel → True]
```

Out[299]//MatrixForm=

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

Out[300]=



In[301]:=
```
Normal[SparseCompleteGraph[3, 2]] == ToAdjacencyMatrix[CompleteGraph[3, 2]]
```

Out[301]=
```
True
```

In[302]:=
```
BipartiteGraph[i_, j_, prob_] := Module[{randseries, lowerleft, upperright, bipartiterule},
  randseries = Table[RandomInteger[BinomialDistribution[Binomial[2, 2], prob]], {i * j}];
  lowerleft = Partition[randseries, i]; upperright = Transpose[lowerleft]; bipartiterule =
   Join[{#[[1, 1]] + i, #[[1, 2]]} → 1 & /@ Most[ArrayRules[SparseArray[lowerleft]]],
    {#[[1, 1]], #[[1, 2]] + i} → 1 & /@ Most[ArrayRules[SparseArray[upperright]]]];
  SparseArray[bipartiterule, {i + j, i + j}]]
```

In[303]:=
```
BipartiteGraph[12, 6, 0.4] // Normal // MatrixForm
```

Out[303]//MatrixForm=

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

In[304]:=
```
i = 10; j = 5; prob = 0.8;
randseries = Table[RandomInteger[BinomialDistribution[Binomial[2, 2], prob]], {i * j}]
lowerleft = Partition[randseries, i]
upperright = Transpose[lowerleft]
bipartiterule =
 Join[{#[[1, 1]] + i, #[[1, 2]]} → 1 & /@ Most[ArrayRules[SparseArray[lowerleft]]],
  {#[[1, 1]], #[[1, 2]] + i} → 1 & /@ Most[ArrayRules[SparseArray[upperright]]]]
SparseArray[bipartiterule, {i + j, i + j}]
% // Normal // MatrixForm
ShowGraph[
 FromAdjacencyMatrix[Normal[SparseArray[bipartiterule, {i + j, i + j}]]], VertexLabel → True]
```

Out[305]=
```
{0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0}
```

Out[306]=
```
{{0, 1, 1, 1, 1, 0, 1, 1, 1, 1}, {1, 0, 1, 1, 1, 1, 1, 1, 1, 1},
 {0, 1, 1, 1, 1, 0, 1, 1, 1, 1}, {0, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 1, 1, 1, 1, 1, 1, 1, 1, 0}}
```

Out[307]=
```
{{0, 1, 0, 0, 0}, {1, 0, 1, 1, 1}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 1},
 {0, 1, 0, 1, 1}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 0}}
```

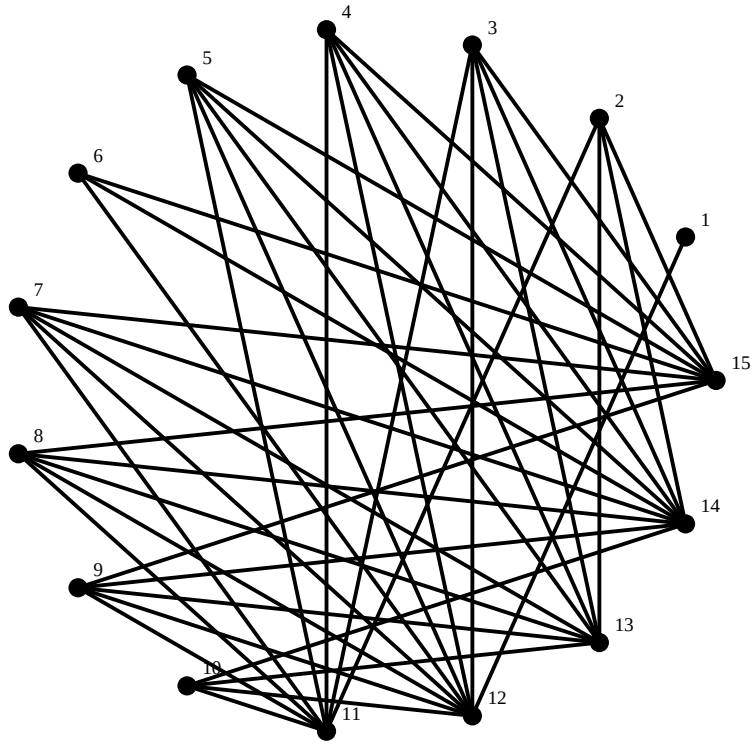
Out[308]=
```
{{11, 2} → 1, {11, 3} → 1, {11, 4} → 1, {11, 5} → 1, {11, 7} → 1, {11, 8} → 1, {11, 9} → 1,
 {11, 10} → 1, {12, 1} → 1, {12, 3} → 1, {12, 4} → 1, {12, 5} → 1, {12, 6} → 1, {12, 7} → 1,
 {12, 8} → 1, {12, 9} → 1, {12, 10} → 1, {13, 2} → 1, {13, 3} → 1, {13, 4} → 1, {13, 5} → 1,
 {13, 7} → 1, {13, 8} → 1, {13, 9} → 1, {13, 10} → 1, {14, 2} → 1, {14, 3} → 1, {14, 4} → 1,
 {14, 5} → 1, {14, 6} → 1, {14, 7} → 1, {14, 8} → 1, {14, 9} → 1, {14, 10} → 1, {15, 2} → 1,
 {15, 3} → 1, {15, 4} → 1, {15, 5} → 1, {15, 6} → 1, {15, 7} → 1, {15, 8} → 1, {15, 9} → 1,
 {1, 12} → 1, {2, 11} → 1, {2, 13} → 1, {2, 14} → 1, {2, 15} → 1, {3, 11} → 1, {3, 12} → 1,
 {3, 13} → 1, {3, 14} → 1, {3, 15} → 1, {4, 11} → 1, {4, 12} → 1, {4, 13} → 1, {4, 14} → 1,
 {4, 15} → 1, {5, 11} → 1, {5, 12} → 1, {5, 13} → 1, {5, 14} → 1, {5, 15} → 1, {6, 12} → 1,
 {6, 14} → 1, {6, 15} → 1, {7, 11} → 1, {7, 12} → 1, {7, 13} → 1, {7, 14} → 1, {7, 15} → 1,
 {8, 11} → 1, {8, 12} → 1, {8, 13} → 1, {8, 14} → 1, {8, 15} → 1, {9, 11} → 1, {9, 12} → 1,
 {9, 13} → 1, {9, 14} → 1, {9, 15} → 1, {10, 11} → 1, {10, 12} → 1, {10, 13} → 1, {10, 14} → 1}
```

Out[309]=
```
SparseArray[<84>, {15, 15}]
```

Out[310]//MatrixForm=

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

Out[311]=



In[312]:=
```
HadamardProduct[list1_, list2_] := MapThread[Times[#1, #2] &, {list1, list2}];
MatrixHadamardProduct[m1_, m2_] := MapThread[HadamardProduct[#1, #2] &, {m1, m2}];
SparseHadamardProduct[list1_, list2_] :=
  SparseArray[MapThread[Times[#1, #2] &, {list1, list2}]];
```

In[315]:=
```
bg = BipartiteGraph[28, 32, 0.3]
SparseHadamardProduct[bg, SparseCompleteGraph[28, 32]] == bg
```

Out[315]= SparseArray[<540>, {60, 60}]

Out[316]= True

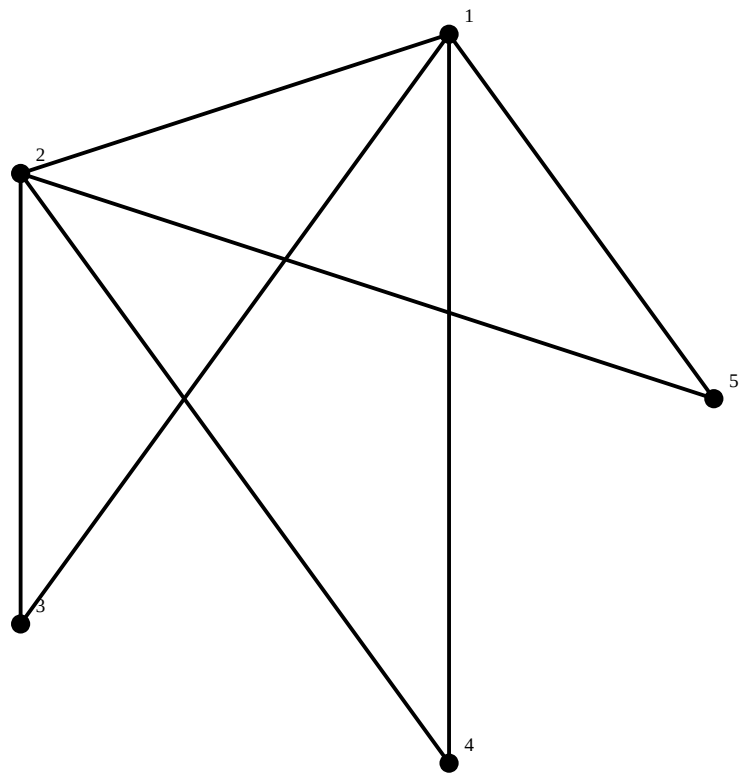
In[317]:=
```
pairssample = {{1, 2}, {1, 3}, {1, 4}, {1, 5}, {2, 3}, {2, 4}, {2, 5}};
adjsample = ToAdjacencyMatrix[FromUnorderedPairs[pairssample]];
adjsample // MatrixForm
```

Out[319]//MatrixForm=

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

In[320]:=
```
ShowGraph[FromAdjacencyMatrix[adjsample], VertexLabel → {1, 2, 3, 4, 5}]
```

Out[320]=



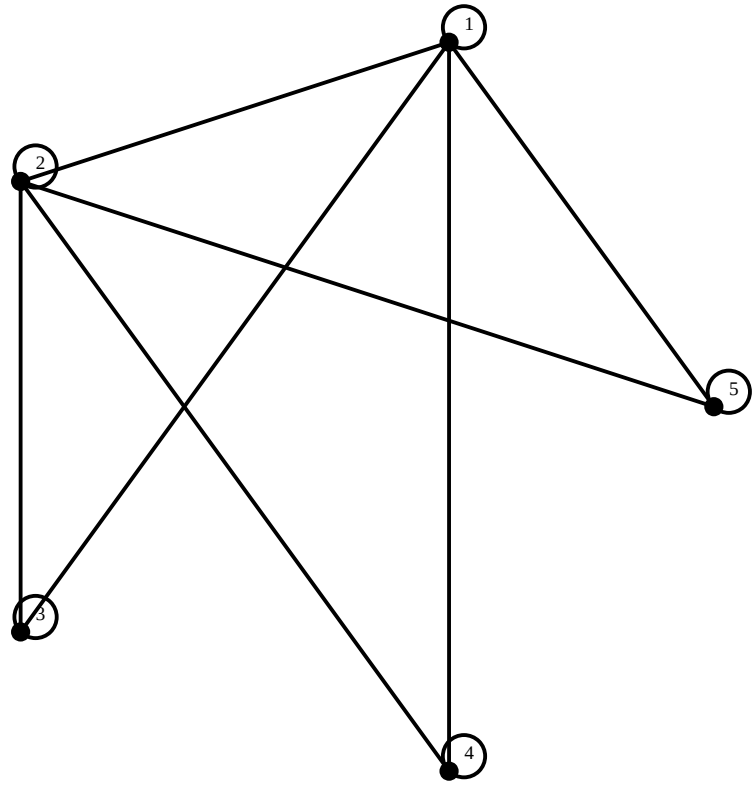
In[321]:=
```
associatedsample = adjsample + IdentityMatrix[Length[adjsample]];
associatedsample // MatrixForm
```

Out[322]//MatrixForm=

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

In[323]:=
```
ShowGraph[FromAdjacencyMatrix[associatedsample], VertexLabel → {1, 2, 3, 4, 5}]
```

Out[323]=



In[326]:=
```
numoflength2 = MatrixPower[associatedsample, 2];
numoflength2 // MatrixForm
? MatrixPower
```

Out[327]//MatrixForm=

$$\begin{pmatrix} 5 & 5 & 3 & 3 & 3 \\ 5 & 5 & 3 & 3 & 3 \\ 3 & 3 & 3 & 2 & 2 \\ 3 & 3 & 2 & 3 & 2 \\ 3 & 3 & 2 & 2 & 3 \end{pmatrix}$$

MatrixPower[*m*, *n*] gives the nth matrix power of the matrix *m*.
MatrixPower[*m*, *n*, *v*] gives the nth matrix power of the matrix *m* applied to the vector *v*. ≫

In[329]:=
```
numoflength5 = MatrixPower[associatedsample, 5];
numoflength5 // MatrixForm
```

Out[330]//MatrixForm=

$$\begin{pmatrix} 307 & 307 & 205 & 205 & 205 \\ 307 & 307 & 205 & 205 & 205 \\ 205 & 205 & 137 & 136 & 136 \\ 205 & 205 & 136 & 137 & 136 \\ 205 & 205 & 136 & 136 & 137 \end{pmatrix}$$

In[331]:=
```
MakeTransititionMatrix[matrix_] := (# / Plus @@ # &) /@ matrix;
transitionsample = MakeTransititionMatrix[associatedsample];
transitionsample // N // MatrixForm
```

Out[333]//MatrixForm=

$$\begin{pmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.333333 & 0.333333 & 0.333333 & 0. & 0. \\ 0.333333 & 0.333333 & 0. & 0.333333 & 0. \\ 0.333333 & 0.333333 & 0. & 0. & 0.333333 \end{pmatrix}$$

In[334]:=
```
transitionsample2 = MatrixPower[transitionsample, 2];
transitionsample2 // N // MatrixForm
```

Out[335]//MatrixForm=

$$\begin{pmatrix} 0.28 & 0.28 & 0.146667 & 0.146667 & 0.146667 \\ 0.28 & 0.28 & 0.146667 & 0.146667 & 0.146667 \\ 0.244444 & 0.244444 & 0.244444 & 0.133333 & 0.133333 \\ 0.244444 & 0.244444 & 0.133333 & 0.244444 & 0.133333 \\ 0.244444 & 0.244444 & 0.133333 & 0.133333 & 0.244444 \end{pmatrix}$$

In[336]:=
```
Plus @@ # & /@ transitionsample2
```

Out[336]= {1, 1, 1, 1, 1}

In[337]:=
```
init1 = {1, 0, 0, 0, 0};
init1.transitionsample2 // N
init3 = {0, 0, 1, 0, 0};
init3.transitionsample2 // N
```

Out[338]= {0.28, 0.28, 0.146667, 0.146667, 0.146667}

Out[340]= {0.244444, 0.244444, 0.244444, 0.133333, 0.133333}

In[341]:=
```
transitionsample15 = MatrixPower[transitionsample, 15];
transitionsample15 // N // MatrixForm
```

Out[342]//MatrixForm=

$$\begin{pmatrix} 0.263158 & 0.263158 & 0.157895 & 0.157895 & 0.157895 \\ 0.263158 & 0.263158 & 0.157895 & 0.157895 & 0.157895 \\ 0.263158 & 0.263158 & 0.157895 & 0.157895 & 0.157895 \\ 0.263158 & 0.263158 & 0.157895 & 0.157895 & 0.157895 \\ 0.263158 & 0.263158 & 0.157895 & 0.157895 & 0.157895 \end{pmatrix}$$

In[343]:=
```
TauGFunction[associatedmatrix_] := Transpose[MakeTransititionMatrix[associatedmatrix]];
taugsample = TauGFunction[associatedsample];
taugsample // N // MatrixForm
Transpose[transitionsample] // N // MatrixForm
```

Out[345]//MatrixForm=

$$\begin{pmatrix} 0.2 & 0.2 & 0.333333 & 0.333333 & 0.333333 \\ 0.2 & 0.2 & 0.333333 & 0.333333 & 0.333333 \\ 0.2 & 0.2 & 0.333333 & 0. & 0. \\ 0.2 & 0.2 & 0. & 0.333333 & 0. \\ 0.2 & 0.2 & 0. & 0. & 0.333333 \end{pmatrix}$$

Out[346]//MatrixForm=

$$\begin{pmatrix} 0.2 & 0.2 & 0.333333 & 0.333333 & 0.333333 \\ 0.2 & 0.2 & 0.333333 & 0.333333 & 0.333333 \\ 0.2 & 0.2 & 0.333333 & 0. & 0. \\ 0.2 & 0.2 & 0. & 0.333333 & 0. \\ 0.2 & 0.2 & 0. & 0. & 0.333333 \end{pmatrix}$$

In[347]:=
```
MatrixPower[taugsample, 15] // N // MatrixForm
```

Out[347]//MatrixForm=

$$\begin{pmatrix} 0.263158 & 0.263158 & 0.263158 & 0.263158 & 0.263158 \\ 0.263158 & 0.263158 & 0.263158 & 0.263158 & 0.263158 \\ 0.157895 & 0.157895 & 0.157895 & 0.157895 & 0.157895 \\ 0.157895 & 0.157895 & 0.157895 & 0.157895 & 0.157895 \\ 0.157895 & 0.157895 & 0.157895 & 0.157895 & 0.157895 \end{pmatrix}$$

In[348]:=
```
Squaring2[list_] := (1 / Plus @@ (list^2)) list^2;
```

In[349]:=
```
listsample1 = {0, 3, 0, 1, 2};
listsample2 = {0, 1 / 2, 0, 1 / 6, 1 / 3}; listsample3 = {1 / 4, 1 / 4, 1 / 4, 1 / 4, 0};
listsample4 = {0.151, 0.159, 0.218, 0.225, 0.247};
listsample5 = {0.080, 0.000, 0.113, 0.801, 0.000};
Squaring2[listsample1] // N
Squaring2[listsample2] // N
Squaring2[listsample3] // N
Squaring2[listsample4] // N
Squaring2[listsample5] // N
```

Out[351]= {0., 0.642857, 0., 0.0714286, 0.285714}

Out[352]= {0., 0.642857, 0., 0.0714286, 0.285714}

Out[353]= {0.25, 0.25, 0.25, 0.25, 0.}

Out[354]= {0.110022, 0.121989, 0.229319, 0.244282, 0.294388}

Out[355]= {0.00968567, 0., 0.0193244, 0.97099, 0.}

In[356]:=
```
Gamma2[matrix_] := Transpose[Squaring2[#] & /@ Transpose[matrix]];
Gamma2[Gamma2[Gamma2[taugsample]]] // N // MatrixForm
```

Out[357]//MatrixForm=

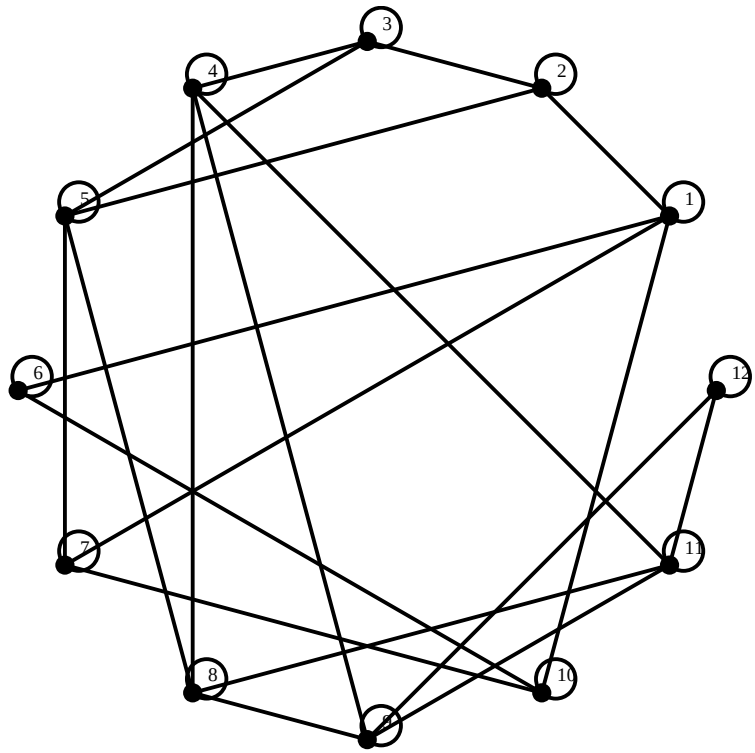
$$\begin{pmatrix} 0.2 & 0.2 & 0.333333 & 0.333333 & 0.333333 \\ 0.2 & 0.2 & 0.333333 & 0.333333 & 0.333333 \\ 0.2 & 0.2 & 0.333333 & 0. & 0. \\ 0.2 & 0.2 & 0. & 0.333333 & 0. \\ 0.2 & 0.2 & 0. & 0. & 0.333333 \end{pmatrix}$$

In[358]:=
```
MGadjacency = {{0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0}, {1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0},
   {0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0},
   {0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0}, {1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
   {1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0}, {0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0},
   {0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1}, {1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0}};
```

In[359]:=
```
ToAssociatedMatrix[adjacencyMatrix_] :=
  adjacencyMatrix + IdentityMatrix[Length[adjacencyMatrix]];
MG = ToAssociatedMatrix[MGadjacency]
ShowGraph[FromAdjacencyMatrix[MG], VertexLabel → {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}]
```

Out[360]=
```
{{1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0}, {1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0},
 {0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0},
 {0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0}, {1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0},
 {1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0}, {0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0},
 {0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1}, {1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0},
 {0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1}}
```

Out[361]=



In[362]:=
```
TauG = TauGFunction[MG];
TauG // N // MatrixForm
```

Out[363]//MatrixForm=

$$\begin{pmatrix} 0.2 & 0.25 & 0. & 0. & 0. & 0.333333 & 0.25 & 0. & 0. & 0.25 & 0. & 0. \\ 0.2 & 0.25 & 0.25 & 0. & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0.25 & 0.25 & 0.2 & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0.25 & 0.2 & 0. & 0. & 0. & 0.2 & 0.2 & 0. & 0.2 & 0. \\ 0. & 0.25 & 0.25 & 0. & 0.2 & 0. & 0.25 & 0.2 & 0. & 0. & 0. & 0. \\ 0.2 & 0. & 0. & 0. & 0. & 0.333333 & 0. & 0. & 0. & 0.25 & 0. & 0. \\ 0.2 & 0. & 0. & 0. & 0.2 & 0. & 0.25 & 0. & 0. & 0.25 & 0. & 0. \\ 0. & 0. & 0. & 0.2 & 0.2 & 0. & 0. & 0.2 & 0.2 & 0. & 0.2 & 0. \\ 0. & 0. & 0. & 0.2 & 0. & 0. & 0. & 0.2 & 0.2 & 0. & 0.2 & 0.333333 \\ 0.2 & 0. & 0. & 0. & 0. & 0.333333 & 0.25 & 0. & 0. & 0.25 & 0. & 0. \\ 0. & 0. & 0. & 0.2 & 0. & 0. & 0. & 0.2 & 0.2 & 0. & 0.2 & 0.333333 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.2 & 0. & 0.2 & 0.333333 \end{pmatrix}$$

In[382]:=
```
Clear[tmp]
tmp[1] = TauG;
tmp[2] = MatrixPower[tmp[1], 2];
% // N // MatrixForm
tmp[3] = Gamma2[tmp[2]];
% // N // MatrixForm
tmp1 = Transpose[tmp[2]][[1]] // N
tmp2 = Squaring2[tmp1]
```

Out[385]//MatrixForm=

0.256667	0.1125	0.0625	0.	0.1	0.261111	0.175	0.	0.	0.258333	0.	0.
0.09	0.225	0.175	0.05	0.14	0.0666667	0.1	0.04	0.	0.05	0.	0.
0.05	0.175	0.225	0.09	0.14	0.	0.05	0.08	0.04	0.	0.04	0.
0.	0.0625	0.1125	0.21	0.09	0.	0.	0.16	0.16	0.	0.16	0.13333
0.1	0.175	0.175	0.09	0.23	0.	0.1125	0.08	0.04	0.0625	0.04	0.
0.156667	0.05	0.	0.	0.	0.261111	0.1125	0.	0.	0.195833	0.	0.
0.14	0.1	0.05	0.	0.09	0.15	0.225	0.04	0.	0.175	0.	0.
0.	0.05	0.1	0.16	0.08	0.	0.05	0.2	0.16	0.	0.16	0.13333
0.	0.	0.05	0.16	0.04	0.	0.	0.16	0.226667	0.	0.226667	0.24444
0.206667	0.05	0.	0.	0.05	0.261111	0.175	0.	0.	0.258333	0.	0.
0.	0.	0.05	0.16	0.04	0.	0.	0.16	0.226667	0.	0.226667	0.24444
0.	0.	0.	0.08	0.	0.	0.	0.08	0.146667	0.	0.146667	0.24444

Out[387]//MatrixForm=

0.380064	0.0867238	0.0267666	0.	0.0766871	0.294533	0.201232	0.	0.
0.0467308	0.346895	0.20985	0.0171233	0.150307	0.0192	0.0657084	0.0114943	0.
0.0144231	0.20985	0.346895	0.0554795	0.150307	0.	0.0164271	0.045977	0.00895522
0.	0.0267666	0.0867238	0.302055	0.0621166	0.	0.	0.183908	0.143284
0.0576923	0.20985	0.20985	0.0554795	0.405675	0.	0.0831622	0.045977	0.00895522
0.141603	0.0171306	0.	0.	0.	0.294533	0.0831622	0.	0.
0.113077	0.0685225	0.0171306	0.	0.0621166	0.0972	0.332649	0.0114943	0.
0.	0.0171306	0.0685225	0.175342	0.0490798	0.	0.0164271	0.287356	0.143284
0.	0.	0.0171306	0.175342	0.0122699	0.	0.	0.183908	0.287562
0.24641	0.0171306	0.	0.	0.0191718	0.294533	0.201232	0.	0.
0.	0.	0.0171306	0.175342	0.0122699	0.	0.	0.183908	0.287562
0.	0.	0.	0.0438356	0.	0.	0.	0.045977	0.120398

Out[388]= {0.256667, 0.09, 0.05, 0., 0.1, 0.156667, 0.14, 0., 0., 0.206667, 0., 0.}

Out[389]= {0.380064, 0.0467308, 0.0144231, 0., 0.0576923, 0.141603, 0.113077, 0., 0., 0.24641, 0., 0.}

In[390]:=
```
tmp[4] = MatrixPower[tmp[3], 2];
% // N // MatrixForm
tmp[5] = Gamma2[tmp[4]];
% // N // MatrixForm
```

Out[391]//MatrixForm=

$$\begin{pmatrix} 0.29651 & 0.109063 & 0.0571972 & 0.00722455 & 0.0959403 & 0.31403 & 0.244731 & 0.00806633 \\ 0.058768 & 0.20566 & 0.181783 & 0.0331089 & 0.154181 & 0.0359916 & 0.0741969 & 0.0277536 \\ 0.0308205 & 0.181783 & 0.204851 & 0.0591378 & 0.152707 & 0.00987391 & 0.0411094 & 0.0521695 \\ 0.0060853 & 0.0517548 & 0.0924424 & 0.186075 & 0.0735624 & 0.000513919 & 0.0113702 & 0.172055 \\ 0.0721766 & 0.215259 & 0.213201 & 0.0657023 & 0.243527 & 0.0346135 & 0.0947653 & 0.0583767 \\ 0.150976 & 0.0321125 & 0.0088097 & 0.000293333 & 0.0221201 & 0.190952 & 0.11873 & 0.00115279 \\ 0.13752 & 0.0773744 & 0.0428699 & 0.00758535 & 0.0707833 & 0.138771 & 0.181139 & 0.0115577 \\ 0.00647788 & 0.0413629 & 0.0777516 & 0.164042 & 0.0623163 & 0.00192562 & 0.0165177 & 0.17712 \\ 0.000954957 & 0.0140135 & 0.0461778 & 0.199878 & 0.034527 & 0. & 0.00432288 & 0.205005 \\ 0.238756 & 0.0556436 & 0.0176609 & 0.00135697 & 0.0478747 & 0.273328 & 0.20804 & 0.00339138 \\ 0.000954957 & 0.0140135 & 0.0461778 & 0.199878 & 0.034527 & 0. & 0.00432288 & 0.205005 \\ 0. & 0.00196094 & 0.011077 & 0.0757176 & 0.00793401 & 0. & 0.000755269 & 0.0783468 \end{pmatrix}$$

Out[393]//MatrixForm=

$$\begin{pmatrix} 0.447826 & 0.0801138 & 0.0225714 & 0.000334168 & 0.0681139 & 0.425728 & 0.359293 \\ 0.0175919 & 0.284874 & 0.227989 & 0.00701831 & 0.175911 & 0.00559233 & 0.0330249 \\ 0.0048385 & 0.222567 & 0.289525 & 0.022391 & 0.172565 & 0.00042089 & 0.010138 \\ 0.000188623 & 0.0180408 & 0.0589591 & 0.221676 & 0.0400448 & 1.14019\times10^{-6} & 0.00077555 \\ 0.0265353 & 0.312088 & 0.313609 & 0.0276379 & 0.438862 & 0.00517227 & 0.0538728 \\ 0.116104 & 0.00694548 & 0.000535464 & 5.50889\times10^{-7} & 0.00362082 & 0.157413 & 0.0845645 \\ 0.0963303 & 0.0403228 & 0.0126798 & 0.000368379 & 0.0370762 & 0.0831358 & 0.196832 \\ 0.000213746 & 0.0115233 & 0.0417087 & 0.172287 & 0.0287367 & 0.0000160078 & 0.00163671 \\ 4.64515\times10^{-6} & 0.00132266 & 0.0147121 & 0.255784 & 0.00882169 & 0. & 0.000112103 \\ 0.290363 & 0.0208538 & 0.00215195 & 0.0000117892 & 0.0169608 & 0.322521 & 0.259635 \\ 4.64515\times10^{-6} & 0.00132266 & 0.0147121 & 0.255784 & 0.00882169 & 0. & 0.000112103 \\ 0. & 0.0000258992 & 0.000846555 & 0.036706 & 0.000465821 & 0. & 3.42195\times10^{-6} \end{pmatrix}$$

In[394]:=
```
tmp[6] = MatrixPower[tmp[5], 2];
% // N // MatrixForm
tmp[7] = Gamma2[tmp[6]];
% // N // MatrixForm
```

Out[395]//MatrixForm=

0.413314	0.111442	0.0620193	0.0033847	0.100597	0.427635	0.386297	0.00291
0.0244712	0.189901	0.187582	0.0145221	0.16964	0.0159109	0.0362719	0.01155
0.0132947	0.183121	0.191052	0.0223265	0.167009	0.00544964	0.0235758	0.01902
0.00191051	0.0373295	0.0587892	0.158469	0.0477074	0.000484224	0.00408339	0.1579
0.0391897	0.300962	0.303512	0.0329264	0.307391	0.023907	0.060313	0.02764
0.116624	0.0197608	0.00693314	0.000233955	0.016736	0.123489	0.106068	0.000252
0.10357	0.0442872	0.0294869	0.00193957	0.0415084	0.104276	0.110725	0.00167
0.00151472	0.0273638	0.0461475	0.145501	0.0368909	0.000481711	0.00325449	0.1470
0.000449521	0.0149558	0.0433803	0.282674	0.0304398	0.0000751331	0.00139919	0.2876
0.285179	0.0542829	0.0213998	0.000875096	0.0474235	0.298212	0.26647	0.000884
0.000449521	0.0149558	0.0433803	0.282674	0.0304398	0.0000751331	0.00139919	0.2876
0.0000331001	0.001637	0.00631663	0.054473	0.00421675	3.85764×10^{-6}	0.000143099	0.05578

Out[397]//MatrixForm=

0.612724	0.0688112	0.0215582	0.0000543283	0.0591659	0.612062	0.59862
0.00214791	0.199809	0.197216	0.0010001	0.168253	0.000847303	0.00527775
0.000633965	0.185797	0.204579	0.0023639	0.163074	0.0000993994	0.00222968
0.000013092	0.00772085	0.0193711	0.11909	0.0133069	7.84769×10^{-7}	0.0000668884
0.00550871	0.501863	0.516311	0.00514133	0.552444	0.00191292	0.0145925
0.0487846	0.00216356	0.000269413	2.59567×10^{-7}	0.00163759	0.0510395	0.0451316
0.0384749	0.0108672	0.00487323	0.0000178401	0.0100734	0.0363931	0.0491813
8.22942×10^{-6}	0.00414871	0.0119359	0.100397	0.00795688	7.76644×10^{-7}	0.0000424889
7.2478×10^{-7}	0.00123932	0.0105474	0.37893	0.00541735	1.88934×10^{-8}	7.85353×10^{-6}
0.291703	0.0163263	0.00256672	3.6316×10^{-6}	0.013149	0.297644	0.284842
7.2478×10^{-7}	0.00123932	0.0105474	0.37893	0.00541735	1.88934×10^{-8}	7.85353×10^{-6}
3.92975×10^{-9}	0.0000148476	0.00022363	0.0140718	0.000103959	4.98072×10^{-11}	8.21451×10^{-8}

In[398]:=
```
$MaxPrecision = 7; k = 1; tmat[1] = TauG; reiterationtimes = 10;
While[k ≤ reiterationtimes, tmat[2 k] = SetPrecision[MatrixPower[tmat[2 k - 1], 2], 7];
  tmat[2 k + 1] = SetPrecision[Gamma2[tmat[2 k]], 7]; k++];
convergence = tmat[2 * reiterationtimes + 1] // N;
convergence // MatrixForm
pos1 = Position[convergence, 1.]
PartitionByLengthSequence[list_, lengthseq_] := Module[{mylist}, mylist = list;
   Map[Block[{a}, a = Take[mylist, #]; mylist = Drop[mylist, #]; a] &, lengthseq]];
Clusteringresult =
 PartitionByLengthSequence[#[[2]] & /@ pos1, Length[#] & /@ Split[pos1, #1[[1]] == #2[[1]] &]]
```

Out[401]//MatrixForm=

$$\begin{pmatrix} 1. & 6.26925\times10^{-51} & 3.56317\times10^{-51} & 2.08462\times10^{-281} & 5.07224\times10^{-51} & 1. \\ 1.47192\times10^{-258} & 2.29499\times10^{-61} & 2.29499\times10^{-61} & 1.49779\times10^{-291} & 2.29499\times10^{-61} & 1.09026\times10^{-} \\ 2.98003\times10^{-260} & 8.17513\times10^{-63} & 8.17513\times10^{-63} & 5.33538\times10^{-293} & 8.17513\times10^{-63} & 1.98013\times10^{-} \\ 1.229000\times10^{-384} & 3.76091\times10^{-187} & 3.76091\times10^{-187} & 2.74317\times10^{-76} & 3.76091\times10^{-187} & 7.959000\times10 \\ 5.18906\times10^{-198} & 1. & 1. & 6.52635\times10^{-231} & 1. & 3.70571\times10^{-} \\ 4.70595\times10^{-141} & 2.18528\times10^{-191} & 1.11418\times10^{-191} & 6.353000\times10^{-422} & 1.70463\times10^{-191} & 4.70595\times10^{-} \\ 4.94316\times10^{-154} & 1.88956\times10^{-197} & 1.88918\times10^{-197} & 1.232900\times10^{-427} & 1.8894\times10^{-197} & 4.94316\times10^{-} \\ 1.590700\times10^{-404} & 5.09997\times10^{-207} & 5.09997\times10^{-207} & 8.16447\times10^{-84} & 5.09997\times10^{-207} & 1.018500\times10 \\ 1.928800\times10^{-337} & 3.05755\times10^{-139} & 5.10295\times10^{-138} & 0.5 & 4.33827\times10^{-139} & 9.313000\times10 \\ 1.06306\times10^{-41} & 5.07444\times10^{-92} & 2.61569\times10^{-92} & 1.495500\times10^{-322} & 3.97267\times10^{-92} & 1.06306\times10^{-} \\ 1.928800\times10^{-337} & 3.05755\times10^{-139} & 5.10295\times10^{-138} & 0.5 & 4.33827\times10^{-139} & 9.313000\times10 \\ 1.593700\times10^{-514} & 3.031800\times10^{-316} & 6.113000\times10^{-315} & 6.34517\times10^{-178} & 4.485000\times10^{-316} & 7.495000\times10 \end{pmatrix}$$

Out[402]= {{1, 1}, {1, 6}, {1, 7}, {1, 10}, {5, 2}, {5, 3}, {5, 5}}

Out[404]= {{1, 6, 7, 10}, {2, 3, 5}}