```
n = 6; (*number of nodos*) k = 3; (*in-degree*) numexp = 100000; (*number of random digraphs to
generate*) numperms = 1000; (*number of random permutations*)
complexity = Table[0, numexp]; (*list for K-complexity (1-D representation)*)
complexitymat = Table[0, numexp]; (*list for K-complexity (adjacency matrix)*)
entropy = Table[0, numexp]; (*list for entropy values*)

    (*create the random digraph*)
Do[Mnodes = {};(*adjacency matrix*)
 For[i = 1, i <= n, i++
    AppendTo[Mnodes, Table[0, n]]]
  For[i = 1, i <= n, i++, flag = 0;
   While[flag < k, position = RandomInteger[{1, n}];
    If[Mnodes[[i, position]] == 0, Mnodes[[i, position]] = 1; flag++,]]];
matadj = Transpose[Mnodes]; (*random digraph*)
Export[NotebookDirectory[]<>ToString[l]<>"file_name.txt",matadj];

    (*generate the isomorphisms*)
perms = Cycles[{#}] & /@ RandomChoice[Permutations[Table[i, {i, n}]], numperms];(*generators of
permutations*)
matpermuted = DeleteDuplicates[Permute[Table[Permute[matadj[[i]], #], {i, n}], #] & /@ perms]
;(*permuted matrices*)
complexitypermut = Table[Null, Length[matpermuted] + 1]; (*list for K-complexity (1-D
representation) permuted*)
complexitymatpermut = Table[Null, Length[matpermuted] + 1]; (*list for K-complexity (adjacency
matrix) permuted*)
entropypermut = Table[Null, Length[matpermuted] + 1]; (*list for entropy values permuted*)

    (*complexity of the random graph*)
matadjflatten = Flatten[matadj]; (*1-D representation of the random graph*)
If[First[matadjflatten] == 0, output = "0" <> ToString[FromDigits[matadjflatten]], output =
ToString[FromDigits[matadjflatten]]];
complexitypermut[[1]] = StringBDM[output] // N;
complexitymatpermut[[1]] = BDM[matadj, 4] // N;
entropypermut[[1]] = Entropy[output] // N;

    (*measure the complexity of the isomorphisms*)
Do[rg = matpermuted[[l]];
 mat = Flatten[rg];
 If[First[mat] == 0, output = "0" <> ToString[FromDigits[mat]], output =
ToString[FromDigits[mat]]];
 complexity[[l + 1]] = StringBDM[output] // N;
 complexitymat[[l + 1]] = BDM[rg, 4] // N;
 entropy[[l + 1]] = Entropy[output] // N;
  If[ IsomorphicGraphQ[AdjacencyGraph[matrix], AdjacencyGraph[matpermuted[[l]]]] == False,
Abort[]] (*check isomorphism*)
 , {l, 1, Length[matpermuted]}]
    (*the complexity is the minimun vale*)
   complexity[[l]] = {l, Min[complexitypermut]};
   complexitymat[[l]] = {l, Min[complexitymatpermut]};
   entropy[[l]] = {l, Min[entropypermut]};
   Export[NotebookDirectory[] <> ToString[l] <> "Vecinos_red.txt", matadj];
   , {l, 1, numexp}]

    (*order the complxities by increasing value*)
sorted = Sort[complexity, #1[[2]] < #2[[2]] &] ;
list = Flatten[First[sorted[[#]]] & /@ Table[i, {i, Length[sorted]}]] ;
data = Flatten[Take[sorted[[#]], {2, 2}] & /@ Table[i, {i, Length[sorted]}]];

sortedmat = Sort[complexitymat, #1[[2]] < #2[[2]] &];
listmat = Flatten[First[sortedmat[[#]]] & /@ Table[i, {i, Length[sortedmat]}]];
datamat = Flatten[Take[sortedmat[[#]],{2, 2}]&/@Table[i,{i, Length[sortedmat]}]];

sortedent = Sort[entropy, #1[[2]] < #2[[2]] &] // N;
listent = Flatten[First[sortedent[[#]]] & /@ Table[i, {i, Length[sortedent]}]];
dataent = Flatten[Take[sortedent[[#]],{2,2}]&/@Table[i, {i, Length[sortedent]}]];

    (*plot the results*)
ListLinePlot[{Rescale[data, {0, Max[data]}], Rescale[datamat, {0, Max[datamat]}],
Rescale[dataent, {0, Max[dataent]}]}, TargetUnits -> {"experimento", "C(red)"}, AxesLabel ->
Automatic, PlotRange -> All, PlotLegends -> {"K-Complexity (1-D representation)", "K-Complexity
(adjacency matrix)", "Entropy"}, Frame -> True, GridLines -> Automatic, FrameLabel -> {"Ordered
Digraphs", "C(D)"}]

    (*draw some of the digraphs by increasing order of complexity*)
numdig = 27; (*number of digraphs to plot*) r = 1; (*flag*)
Table[If[net == Round[(numexp*r/numdig)] || net == 1,
   state = Import[NotebookDirectory[] <> ToString[list[[net]]] <> "file_name.txt", "Lines"]; r++;
   AdjacencyGraph[ToExpression[state], PlotLabel -> "Digraph " <> ToString[lista[[net]]]],
VertexStyle -> RGBColor[1, .78, .72], EdgeStyle -> Black]
  ,{net, 1, numexp}] /. Null -> Sequence[]
```