Мухамадиев Владимир

Задание 11

Загрузка и предварительная обработка

In[1]:=
```
(*Get["https://raw.githubusercontent.com/szhorvat/IGraphM/master/IGInstaller.m"]*)
```
взять

In[2]:=
```
Needs["IGraphM`"]
```
необходимо

Out[2]=
```
IGraph/M 0.4 (April 2, 2020)
Evaluate IGDocumentation[] to get started.
```

In[3]:=
```
datatest = Import[NotebookDirectory[] <> "\\k-core-exp.txt", "Data"];
```
импорт директория файла блокнота

In[4]:=
```
eltest = Table[datatest[[i, 1]] ⟷ datatest[[i, 2]], {i, 1, Length[datatest]}];
```
таблица значений длина

In[5]:=
```
gtest = Graph[eltest];
```
граф

In[6]:=
```
airports = Import[NotebookDirectory[] <> "\\airports.graphml"];
```
импорт директория файла блокнота

In[7]:=
```
data1 = Import[NotebookDirectory[] <> "\\asoiaf-book1-edges.csv", "CSV"];
```
импорт директория файла блокнота

In[8]:=
```
el1 = Table[data1[[i, 1]] ⟷ data1[[i, 2]], {i, 2, Length[data1]}];
```
таблица значений длина

In[9]:=
```
ew1 = data1[[2 ;; All, 4]];
```
всё

In[10]:=
```
g1 = Graph[el1, EdgeWeight → ew1];
```
граф вес ребра

In[11]:=
```
data2 = Import[NotebookDirectory[] <> "\\asoiaf-book2-edges.csv", "CSV"];
```
импорт директория файла блокнота

In[12]:=
```
el2 = Table[data2[[i, 1]] ⟷ data2[[i, 2]], {i, 2, Length[data2]}];
```
таблица значений длина

In[13]:=
```
ew2 = data2[[2 ;; All, 4]];
```
всё

In[14]:=
```
g2 = Graph[el2, EdgeWeight → ew2];
```
граф вес ребра

In[15]:=
```
data3 = Import[NotebookDirectory[] <> "\\asoiaf-book3-edges.csv", "CSV"];
```
импорт директория файла блокнота

In[16]:=
```
el3 = Table[data3[[i, 1]] ⟷ data3[[i, 2]], {i, 2, Length[data3]}];
```
таблица значений длина

In[17]:=
```
ew3 = data3[[2 ;; All, 4]];
```
всё

In[18]:=
```
g3 = Graph[el3, EdgeWeight → ew3];
```
граф вес ребра

In[19]:=
```
data4 = Import[NotebookDirectory[] <> "\\asoiaf-book4-edges.csv", "CSV"];
```
импорт директория файла блокнота

In[20]:=
```
el4 = Table[data4[[i, 1]] ⟷ data4[[i, 2]], {i, 2, Length[data4]}];
```
таблица значений длина

In[21]:=
```
ew4 = data4[[2 ;; All, 4]];
```
всё

In[22]:=
```
g4 = Graph[el4, EdgeWeight → ew4];
```
граф вес ребра

In[23]:=
```
data5 = Import[NotebookDirectory[] <> "\\asoiaf-book5-edges.csv", "CSV"];
```
импорт директория файла блокнота

In[24]:=
```
el5 = Table[data5[[i, 1]] ⟷ data5[[i, 2]], {i, 2, Length[data5]}];
```
таблица значений длина

In[25]:=
```
ew5 = data5[[2 ;; All, 4]];
```
всё

In[26]:=
```
g5 = Graph[el5, EdgeWeight → ew5];
```
граф вес ребра

1. Разложение по k-core и визуализация сложной сети

Функция которая приводит список ребер к виду, где в каждом ребре $v_1 \longleftrightarrow v_2, v_1 \leq v_2$

In[27]:=
```
EdgeSort[edgelist_] := Module[{out = edgelist},
```
программный модуль
```
  Do[If[edgelist[[i, 1]] > edgelist[[i, 2]], out[[i]] = edgelist[[i, 2]] ⟷ edgelist[[i, 1]]],
```
… условный оператор
```
   {i, 1, Length[edgelist]}];
```
длина
```
  out]
```

Рандомизация графа которая не создает новых петель и мультиребер

In[28]:=
```
GraphRandomization[g_, n_] :=
 Module[{el = EdgeSort[EdgeList[g]], eltemp = {}, vltemp = {}, temp = {{}, {}}, k = 0},
```
программный модуль список рёбер
```
  Do[temp[[1]] = RandomChoice[el] (*Выбираем первое ребро/петлю*);
```
оператор цикла случайный выбор
```
   eltemp = DeleteCases[el, temp[[1]]] (*Создаем временную выборку из ребер/петель
```
удалить случаи по образцу
```
    и удалаем из нее все мультиребра/мультипетли для выбранного ребра/петли*);
   If[temp[[1, 1]] == temp[[1, 2]], (*Алгоритм выбрал петлю*)
```
условный оператор
```
    eltemp = DeleteCases[eltemp, v_ ⟷ v_] (*удаляем из временной выбрки все петли*);
```
удалить случаи по образцу
```
    vltemp = Flatten[{Cases[eltemp, temp[[1, 1]] ⟷ _], Cases[eltemp, _ ⟷ temp[[1, 1]]]}];
```
уплостить случаи по образцу случаи по образцу

```
уплостить  случаи по образцу                    случаи по образцу
vltemp = DeleteDuplicates[Flatten[{vltemp[[All, 1]], vltemp[[All, 2]]}]]
         удалить дубликаты  уплостить  всё  всё
(*Список вершин с которыми граничит выбранная петля*);
Do[eltemp = DeleteCases[eltemp, vltemp[[j]] :> _];
оператор ци…  удалить случаи по образцу
 eltemp = DeleteCases[eltemp, _ :> vltemp[[j]]], {j, 1, Length[vltemp]}](*Удаляем из
          удалить случаи по образцу          длина
 временной выборки все ребра/петли, вершины которых связаны с данной петлей*);
If[eltemp == {}, i++;
условный оператор
 Goto[end](*Алгоритм выбрал петлю которую нельзя изменить,
 перейти
 так как до любой вершины из нее можно добраться в два шага. Переходим в
  конец цикла и считаем что на этом шаге ничего не поменялось*), temp[[2]] =
  RandomChoice[eltemp](*Выбираем второе ребро*)], (*Алгоритм выбрал ребро*)
  случайный выбор
vltemp = Flatten[{Cases[eltemp, temp[[1, 1]] :> _], Cases[eltemp, _ :> temp[[1, 1]],
         уплостить  случаи по образцу                  случаи по образцу
   Cases[eltemp, temp[[1, 2]] :> _], Cases[eltemp, _ :> temp[[1, 2]]}];
   случаи по образцу                  случаи по образцу
vltemp = DeleteDuplicates[Flatten[{vltemp[[All, 1]], vltemp[[All, 2]]}]]
         удалить дубликаты  уплостить  всё  всё
(*Список вершин с которыми граничит выбранное ребро*);
Do[eltemp = DeleteCases[eltemp, vltemp[[j]] :> _];
оператор ци…  удалить случаи по образцу
 eltemp = DeleteCases[eltemp, _ :> vltemp[[j]]], {j, 1, Length[vltemp]}](*Удаляем из
          удалить случаи по образцу          длина
 временной выборки все ребра/петли, вершины которых связаны с данным ребром*);
If[eltemp == {}, i++;
условный оператор
 Goto[end](*Алгоритм выбрал ребро которое нельзя изменить,
 перейти
 так как до любой вершины из него можно добраться в два шага. Переходим
  в конец цикла и считаем что на этом шаге ничего не поменялось*),
 temp[[2]] = RandomChoice[eltemp](*Выбираем второе ребро/петлю*)]];
             случайный выбор
k = 1;
While[(el[[k]] === temp[[1]]) == False, k++];
цикл–пока                  ложь
el = Delete[el, k];
     удалить элемент
k = 1;
While[(el[[k]] === temp[[2]]) == False, k++];
цикл–пока                  ложь
el = Delete[el, k](*Удаляем из списка ребер выбранные*);
     удалить элемент
If[RandomInteger[{1, 2}] == 1(*Случайно выбираем как переключить ребра и добавляем
…  случайное целое число
  новые к списку ребер*), AppendTo[el, EdgeSort[{temp[[1, 1]] :> temp[[2, 1]]}][[1]]];
                          добавить в конец к
 AppendTo[el, EdgeSort[{temp[[1, 2]] :> temp[[2, 2]]}][[1]]],
 добавить в конец к
 AppendTo[el, EdgeSort[{temp[[1, 1]] :> temp[[2, 2]]}][[1]]];
 добавить в конец к
```

```
    добавить в конец к
    AppendTo[el, EdgeSort[{temp[[1, 2]] ⟷ temp[[2, 1]]}][[1]]];
    добавить в конец к
  Label[end], {i, 1, n}];
  отметка
 Graph[el]]
 граф
```

k-core распределение

In[29]:=
```
KCoreDistribution[graph_] :=
 Table[Length[Flatten[KCoreComponents[graph, i]]]/VertexCount[graph], {i, 1, Max[VertexDegree[graph]]}]
 таблица значений                                                    ма··  степень вершины
```

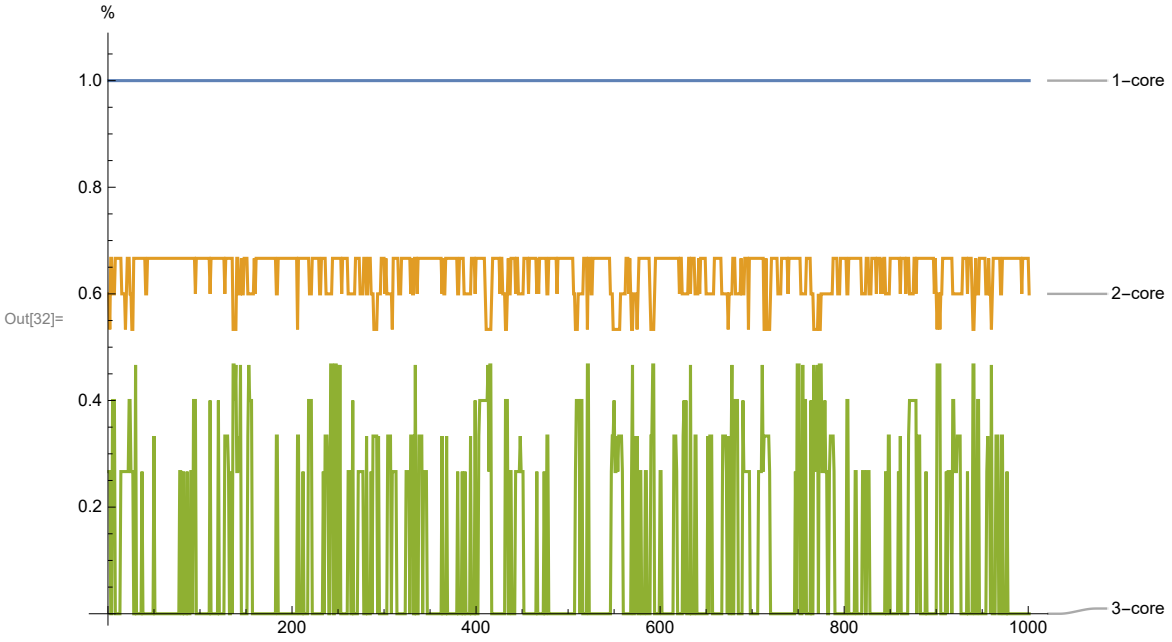
In[30]:=
```
GraphRandomizationKCoreDistributionEvolution[graph_, steps_] := Module[
                                                                 программный модуль
 {temp = graph, out = ConstantArray[{}, steps + 1]}, out[[1]] = KCoreDistribution[temp];
                     постоянный массив
 Do[temp = GraphRandomization[temp, 1];
 оператор цикла
  out[[l]] = KCoreDistribution[temp], {l, 2, steps + 1}];
 out = out^T;
 Table[{Range[steps + 1], out[[i]]}^T, {i, 1, Max[VertexDegree[graph]]}];
 таблиц··  диапазон                    ма··  степень вершины
 {temp, out}]
```

На малом графе

In[31]:=
```
dg = GraphRandomizationKCoreDistributionEvolution[gtest, 1000];
```

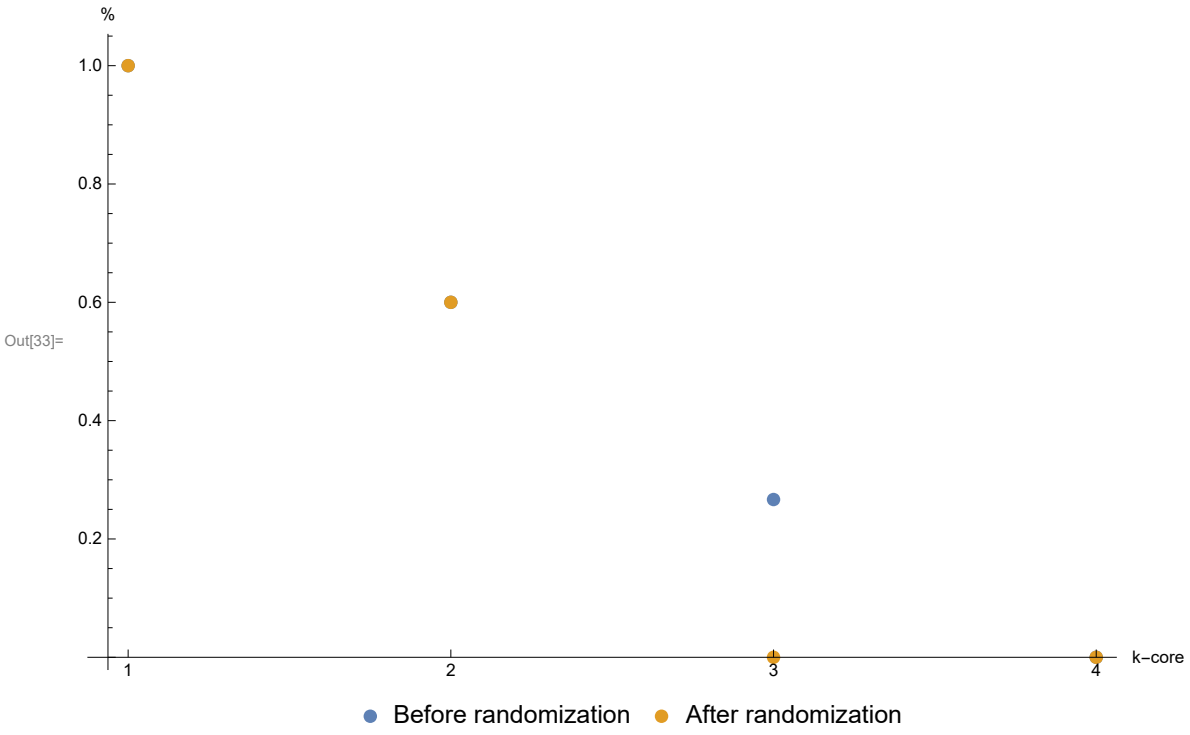
In[32]:=
```
ListLinePlot[dg[[2, 1 ;; 3]], ImageSize → Large,
 PlotLabels → Table[ToString[i] <> "-core", {i, 1, 3}],
 AxesLabel → {None, "%"}, PlotRange → All]
```

Out[32]=



In[33]:=
```
ListPlot[{{Range[4], KCoreDistribution[gtest][[1 ;; 4]]}^T,
  {Range[4], KCoreDistribution[dg[[1]]][[1 ;; 4]]}^T}, PlotRange → All,
 PlotStyle → PointSize[Large], Ticks → {Range[4], Automatic},
 PlotLegends → Placed[{"Before randomization", "After randomization"}, Below],
 AxesLabel → {"k-core", "%"}, ImageSize → Large]
```

Out[33]=

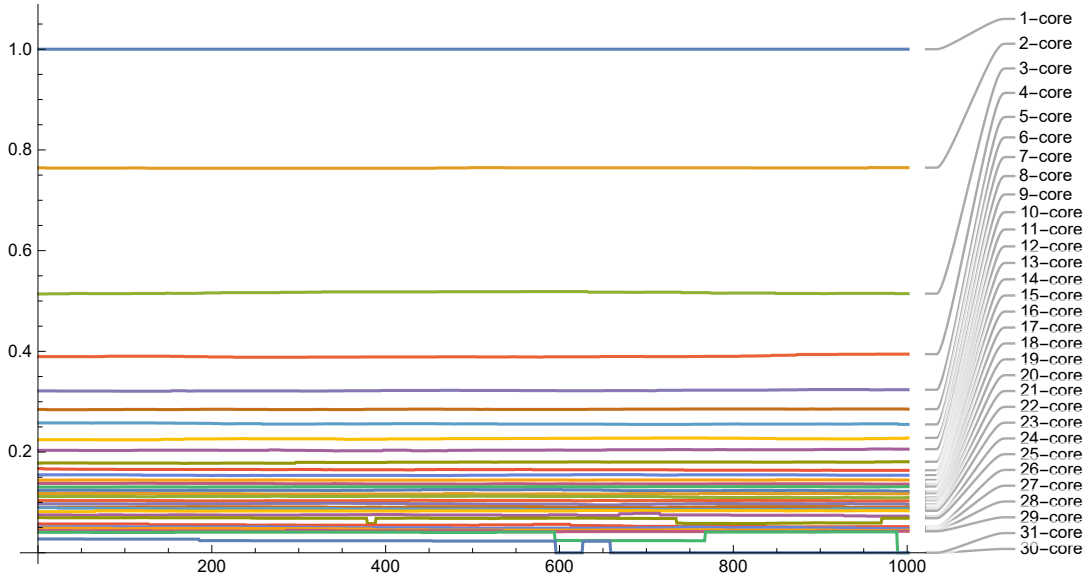


На большом графе

In[34]:=
```
da = Import[NotebookDirectory[] <> "da.m"];
(*da=GraphRandomizationKCoreDistributionEvolution[airports,1000];*)
```

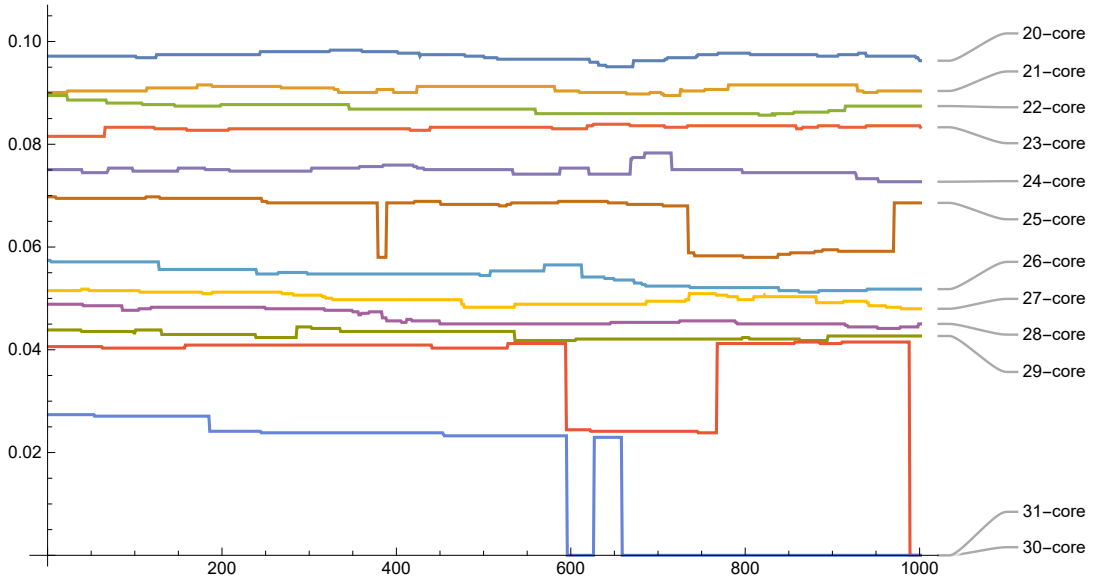
In[35]:=
```
ListLinePlot[da[[2, 1 ;; 31]], ImageSize → Large,
 PlotLabels → Table[ToString[i] <> "-core", {i, 1, 31}], PlotRange → All]
```

Out[35]=



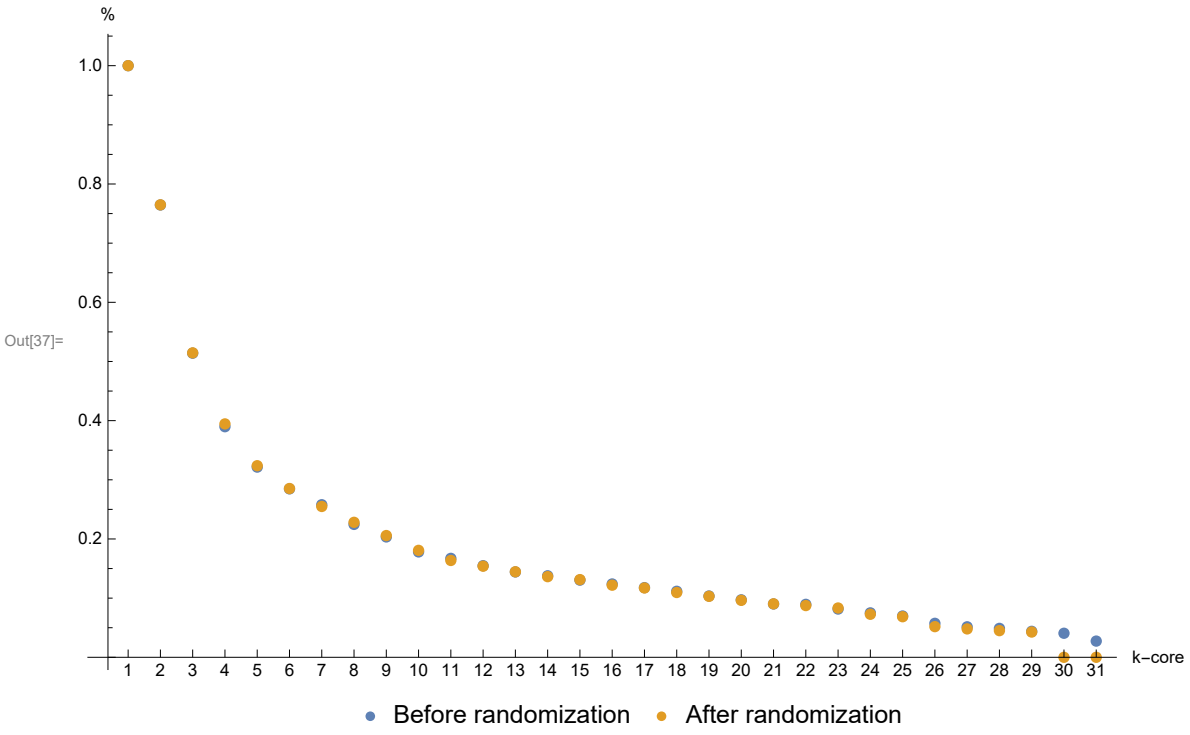
In[36]:=
```
ListLinePlot[da[[2, 20 ;; 31]], ImageSize → Large,
 PlotLabels → Table[ToString[i] <> "-core", {i, 20, 31}], PlotRange → All]
```

Out[36]=



In[37]:=
```
ListPlot[{{Range[31], KCoreDistribution[airports][[1 ;; 31]]}ᵀ,
 {Range[31], KCoreDistribution[da[[1]]][[1 ;; 31]]}ᵀ},
 PlotRange → All, Ticks → {Range[31], Automatic},
 PlotLegends → Placed[{"Before randomization", "After randomization"}, Below],
 AxesLabel → {"k-core", "%"}, ImageSize → Large]
```

Out[37]=



Укладка графа по k-shell

In[38]:=
```
CircleLayout[n_] := Table[{Cos[2 π i / n], Sin[2 π i / n]}, {i, n}]
```

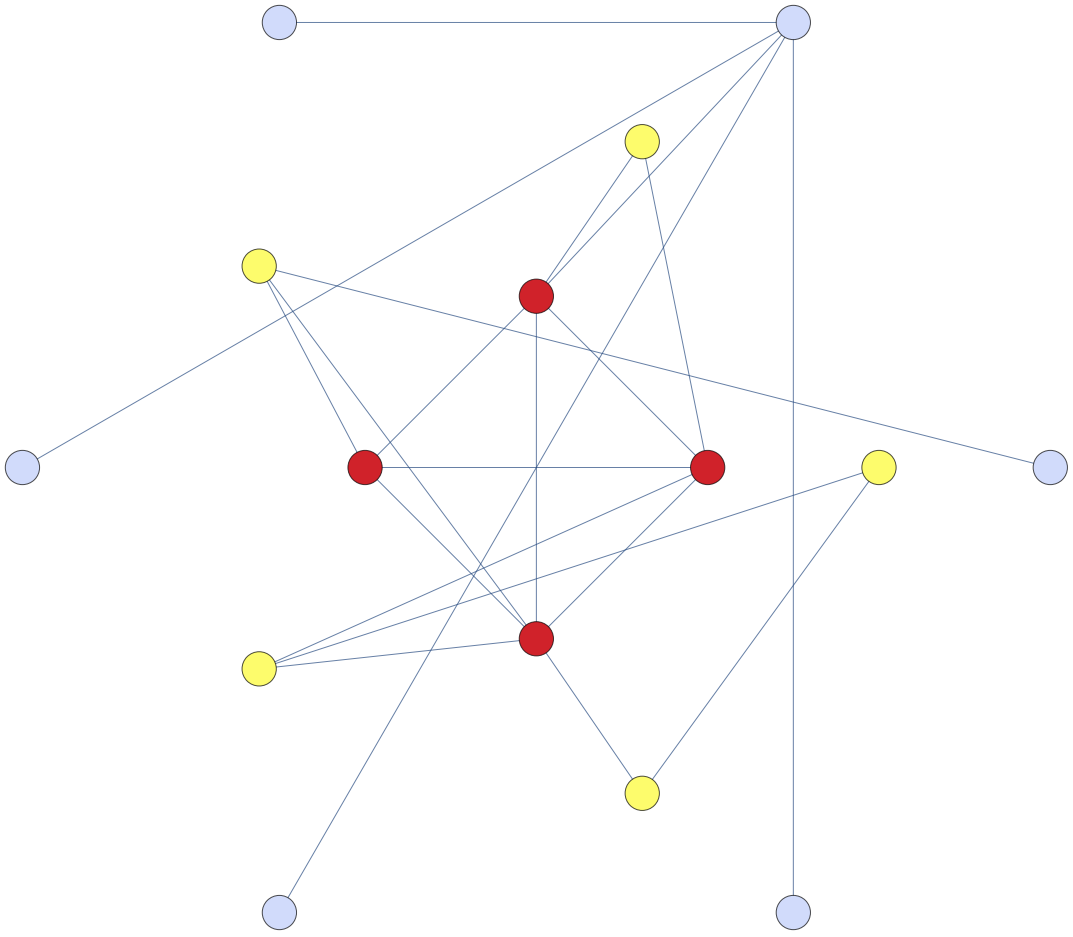
In[39]:=
```
Degeneracy[g_] := Module[{i = 1}, While[KCoreComponents[g, i] ≠ {}, ++i];
 i - 1]
```

In[40]:=
```
KShellComponents[graph_] := Table[Complement[Flatten[KCoreComponents[graph, i]],
  Flatten[KCoreComponents[graph, i + 1]]], {i, 1, Degeneracy[graph]}]
```

In[41]:=
```
KShellRadialEmbedding[graph_, imagesize_] :=
 Module[{ksc = KShellComponents[graph]}, HighlightGraph[
  Graph[graph, VertexCoordinates → Flatten[Table[Map[{#[[1]] → #[[2]]} &, {ksc[[i]],
      (Length[ksc] + 1 - i) CircleLayout[Length[ksc[[i]]]]}^T], {i, 1, Length[ksc]}]]],
  Table[Style[ksc[[i]], ColorData["TemperatureMap"][i / Length[ksc]]],
   {i, Length[ksc]}], VertexSize → Medium, ImageSize → imagesize]]
```

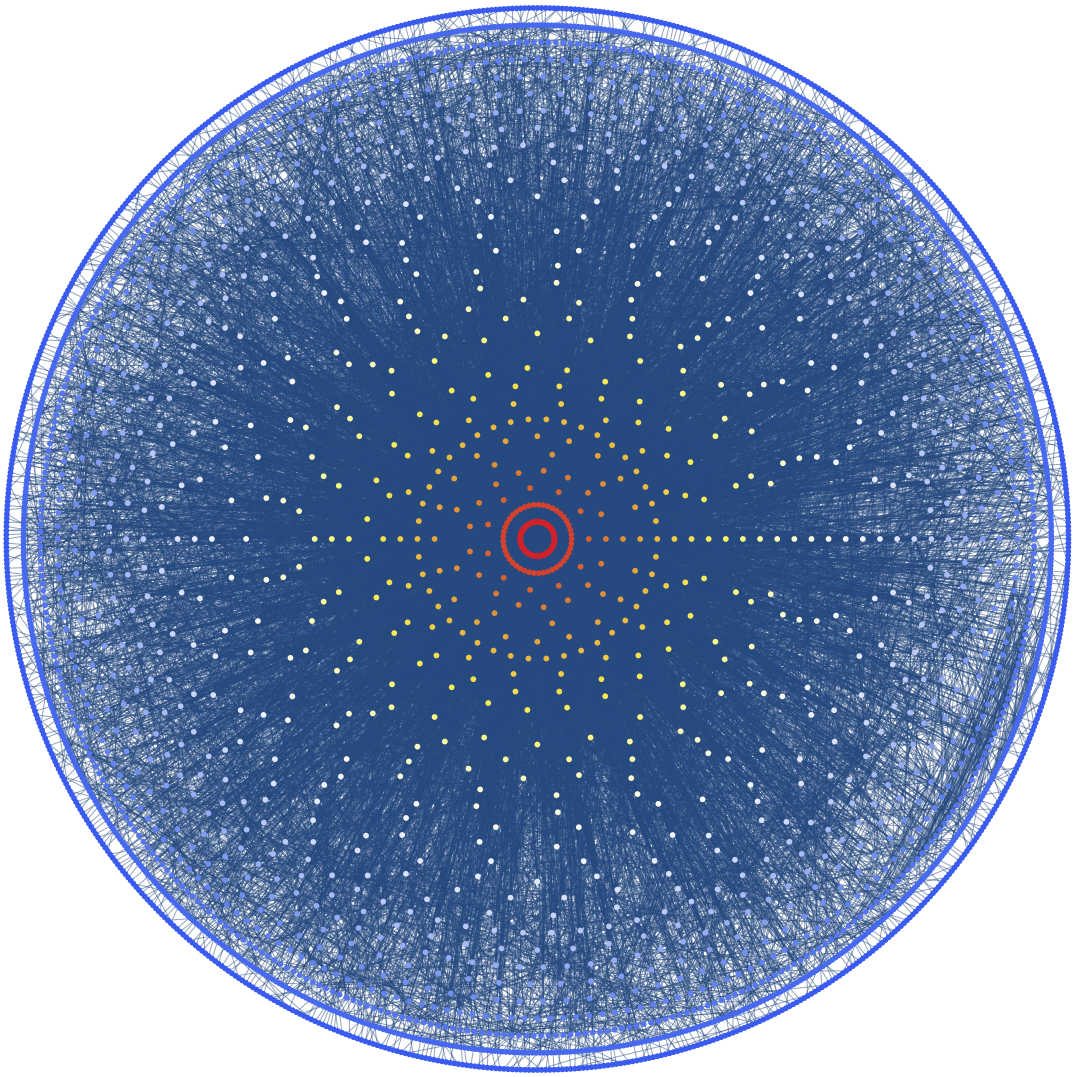
In[42]:=
```
KShellRadialEmbedding[gtest, Large]
```

Out[42]=



In[43]:=
```
KShellRadialEmbedding[airports, Large]
                      крупный
```

Out[43]=



2. Предсказание связей в сложных сетях

In[44]:=
```
TopLinksPreditions[graph_, metrics_] :=
 Module[{vnm = {VertexList[graph]}^T.{VertexList[graph]},
  программный мо··· список вершин графа  список вершин графа
   nl = DeleteCases[Map[If[#[[1]] ≥ #[[2]], {}, #] &,
     удалить случ··· п···  условный оператор
      Position[Normal[AdjacencyMatrix[graph]], 0]], {}], sim = metrics[graph]},
     позиция п·· норма···  матрица смежности
  ReverseSortBy[Table[{vnm[[nl[[i, 1]], nl[[i, 2]]]][[1]] ⟷ vnm[[nl[[i, 1]], nl[[i, 2]]]][[2]],
  сортировка в об···  таблица значений
     sim[[nl[[i, 1]], nl[[i, 2]]]}, {i, 1, Length[nl]}], Last]]
                                    длина  последний
```

Число общих соседей

In[45]:=
```
(c1 = TopLinksPreditions[g1, IGCocitationCoupling][[1 ;; 10]]) // MatrixForm
```
матричная форма

Out[45]//MatrixForm=

Arya-Stark ⟷ Tyrion-Lannister	16
Cersei-Lannister ⟷ Robb-Stark	14
Bran-Stark ⟷ Jaime-Lannister	14
Sansa-Stark ⟷ Tywin-Lannister	13
Petyr-Baelish ⟷ Robb-Stark	13
Joffrey-Baratheon ⟷ Jory-Cassel	13
Rodrik-Cassel ⟷ Sansa-Stark	12
Robert-Baratheon ⟷ Rodrik-Cassel	12
Jon-Snow ⟷ Petyr-Baelish	12
Jon-Arryn ⟷ Sansa-Stark	12

Коэффициент Салтона

In[46]:=
```
(c2 = TopLinksPreditions[g1, N[VertexCosineSimilarity[#]] &][[1 ;; 10]]) // MatrixForm
```
косинусный коэффициент схожести вершин
матричная форма

Out[46]//MatrixForm=

Varly ⟷ Wylla	1.
Tregar ⟷ Wylla	1.
Tregar ⟷ Varly	1.
Tobho-Mott ⟷ Wylla	1.
Tobho-Mott ⟷ Varly	1.
Tobho-Mott ⟷ Tregar	1.
Porther ⟷ Wylla	1.
Porther ⟷ Varly	1.
Porther ⟷ Tregar	1.
Porther ⟷ Tobho-Mott	1.

Коэффициент Жаккара

In[47]:=
```
(c3 = TopLinksPreditions[g1, IGJaccardSimilarity][[1 ;; 10]]) // MatrixForm
```
матричная форма

Out[47]//MatrixForm=

Varly ⟷ Wylla	1.
Tregar ⟷ Wylla	1.
Tregar ⟷ Varly	1.
Tobho-Mott ⟷ Wylla	1.
Tobho-Mott ⟷ Varly	1.
Tobho-Mott ⟷ Tregar	1.
Porther ⟷ Wylla	1.
Porther ⟷ Varly	1.
Porther ⟷ Tregar	1.
Porther ⟷ Tobho-Mott	1.

Коэффициент Серенсена

In[48]:=
```
(c4 = TopLinksPreditions[g1, IGDiceSimilarity][[1 ;; 10]]) // MatrixForm
```
матричная форма

Out[48]//MatrixForm=

Varly ⟷ Wylla	1.
Tregar ⟷ Wylla	1.
Tregar ⟷ Varly	1.
Tobho-Mott ⟷ Wylla	1.
Tobho-Mott ⟷ Varly	1.
Tobho-Mott ⟷ Tregar	1.
Porther ⟷ Wylla	1.
Porther ⟷ Varly	1.
Porther ⟷ Tregar	1.
Porther ⟷ Tobho-Mott	1.

Коэффициент предпочтительного присоединения

In[49]:=
```
(c5 = TopLinksPreditions[g1, {VertexDegree[#]}ᵀ.{VertexDegree[#]} &][[1 ;; 10]]) //
```
степень вершины степень вершины

```
 MatrixForm
```
матричная форма

Out[49]//MatrixForm=

Drogo ⟷ Eddard-Stark	1254
Arya-Stark ⟷ Tyrion-Lannister	1242
Jaime-Lannister ⟷ Jon-Snow	1073
Cersei-Lannister ⟷ Robb-Stark	1050
Jory-Cassel ⟷ Tyrion-Lannister	966
Daenerys-Targaryen ⟷ Tyrion-Lannister	966
Jon-Snow ⟷ Petyr-Baelish	962
Bran-Stark ⟷ Jaime-Lannister	928
Benjen-Stark ⟷ Eddard-Stark	924
Petyr-Baelish ⟷ Robb-Stark	910

Коэффициент Адамика-Адара

In[50]:=
```
(c6 = TopLinksPreditions[g1, IGInverseLogWeightedSimilarity][[1 ;; 10]]) // MatrixForm
```
матричная форма

Out[50]//MatrixForm=

Arya-Stark ⟷ Tyrion-Lannister	5.31863
Bran-Stark ⟷ Jaime-Lannister	4.30283
Sansa-Stark ⟷ Tywin-Lannister	4.23827
Cersei-Lannister ⟷ Robb-Stark	4.13939
Joffrey-Baratheon ⟷ Jory-Cassel	3.98908
Petyr-Baelish ⟷ Robb-Stark	3.92302
Jaime-Lannister ⟷ Pycelle	3.8436
Jon-Arryn ⟷ Sansa-Stark	3.72105
Joffrey-Baratheon ⟷ Jon-Arryn	3.72105
Jon-Snow ⟷ Petyr-Baelish	3.57408

Общее по числу общих соседей, коэффициенту предпочтительного присоединения и коэффициенту Адамика-Адара

In[51]:=
```
ci = c1[[All, 1]] ⋃ c5[[All, 1]] ⋃ c6[[All, 1]]
```
всё всё всё

Out[51]=
```
{Arya-Stark ⟷ Tyrion-Lannister, Benjen-Stark ⟷ Eddard-Stark,
 Bran-Stark ⟷ Jaime-Lannister, Cersei-Lannister ⟷ Robb-Stark,
 Daenerys-Targaryen ⟷ Tyrion-Lannister, Drogo ⟷ Eddard-Stark,
 Jaime-Lannister ⟷ Jon-Snow, Jaime-Lannister ⟷ Pycelle,
 Joffrey-Baratheon ⟷ Jon-Arryn, Joffrey-Baratheon ⟷ Jory-Cassel,
 Jon-Arryn ⟷ Sansa-Stark, Jon-Snow ⟷ Petyr-Baelish, Jory-Cassel ⟷ Tyrion-Lannister,
 Petyr-Baelish ⟷ Robb-Stark, Robert-Baratheon ⟷ Rodrik-Cassel,
 Rodrik-Cassel ⟷ Sansa-Stark, Sansa-Stark ⟷ Tywin-Lannister}
```

2 книга

In[52]:=
```
{ci, Table[ContainsAll[EdgeList[g2], {ci[[i]]}] ||
```
табл··· содержит всё список рёбер
```
   ContainsAll[EdgeList[g2], {Reverse[ci[[i]]]}], {i, 1, Length[ci]}]}ᵀ // MatrixForm
```
содержит всё список рёбер расположить в обратном поря··· длина матричная форма

Out[52]//MatrixForm=

Arya-Stark ⟷ Tyrion-Lannister	True
Benjen-Stark ⟷ Eddard-Stark	False
Bran-Stark ⟷ Jaime-Lannister	False
Cersei-Lannister ⟷ Robb-Stark	True
Daenerys-Targaryen ⟷ Tyrion-Lannister	False
Drogo ⟷ Eddard-Stark	False
Jaime-Lannister ⟷ Jon-Snow	True
Jaime-Lannister ⟷ Pycelle	False
Joffrey-Baratheon ⟷ Jon-Arryn	False
Joffrey-Baratheon ⟷ Jory-Cassel	False
Jon-Arryn ⟷ Sansa-Stark	False
Jon-Snow ⟷ Petyr-Baelish	False
Jory-Cassel ⟷ Tyrion-Lannister	False
Petyr-Baelish ⟷ Robb-Stark	False
Robert-Baratheon ⟷ Rodrik-Cassel	False
Rodrik-Cassel ⟷ Sansa-Stark	False
Sansa-Stark ⟷ Tywin-Lannister	False

3 книга

In[53]:=
```
{ci, Table[ContainsAll[EdgeList[g3], {ci[[i]]}] ||
 ContainsAll[EdgeList[g3], {Reverse[ci[[i]]]}], {i, 1, Length[ci]}]}^T // MatrixForm
```

Out[53]//MatrixForm=

Arya-Stark ⟷ Tyrion-Lannister	True
Benjen-Stark ⟷ Eddard-Stark	False
Bran-Stark ⟷ Jaime-Lannister	False
Cersei-Lannister ⟷ Robb-Stark	False
Daenerys-Targaryen ⟷ Tyrion-Lannister	False
Drogo ⟷ Eddard-Stark	False
Jaime-Lannister ⟷ Jon-Snow	False
Jaime-Lannister ⟷ Pycelle	False
Joffrey-Baratheon ⟷ Jon-Arryn	False
Joffrey-Baratheon ⟷ Jory-Cassel	False
Jon-Arryn ⟷ Sansa-Stark	False
Jon-Snow ⟷ Petyr-Baelish	False
Jory-Cassel ⟷ Tyrion-Lannister	False
Petyr-Baelish ⟷ Robb-Stark	True
Robert-Baratheon ⟷ Rodrik-Cassel	False
Rodrik-Cassel ⟷ Sansa-Stark	False
Sansa-Stark ⟷ Tywin-Lannister	True

4 книга

In[54]:=
```
{ci, Table[ContainsAll[EdgeList[g4], {ci[[i]]}] ||
 ContainsAll[EdgeList[g4], {Reverse[ci[[i]]]}], {i, 1, Length[ci]}]}^T // MatrixForm
```

Out[54]//MatrixForm=

Arya-Stark ⟷ Tyrion-Lannister	False
Benjen-Stark ⟷ Eddard-Stark	False
Bran-Stark ⟷ Jaime-Lannister	False
Cersei-Lannister ⟷ Robb-Stark	False
Daenerys-Targaryen ⟷ Tyrion-Lannister	False
Drogo ⟷ Eddard-Stark	False
Jaime-Lannister ⟷ Jon-Snow	False
Jaime-Lannister ⟷ Pycelle	True
Joffrey-Baratheon ⟷ Jon-Arryn	False
Joffrey-Baratheon ⟷ Jory-Cassel	False
Jon-Arryn ⟷ Sansa-Stark	False
Jon-Snow ⟷ Petyr-Baelish	False
Jory-Cassel ⟷ Tyrion-Lannister	False
Petyr-Baelish ⟷ Robb-Stark	False
Robert-Baratheon ⟷ Rodrik-Cassel	False
Rodrik-Cassel ⟷ Sansa-Stark	False
Sansa-Stark ⟷ Tywin-Lannister	True

5 книга

In[55]:=
```
{ci, Table[ContainsAll[EdgeList[g5], {ci[[i]]}] ||
 табл··· содержит всё список рёбер
  ContainsAll[EdgeList[g5], {Reverse[ci[[i]]]}], {i, 1, Length[ci]}]}^T // MatrixForm
  содержит всё список рёбер расположить в обратном поря··· длина матричная форма
```

Out[55]//MatrixForm=

Arya-Stark ⟷ Tyrion-Lannister	False
Benjen-Stark ⟷ Eddard-Stark	False
Bran-Stark ⟷ Jaime-Lannister	False
Cersei-Lannister ⟷ Robb-Stark	False
Daenerys-Targaryen ⟷ Tyrion-Lannister	True
Drogo ⟷ Eddard-Stark	False
Jaime-Lannister ⟷ Jon-Snow	False
Jaime-Lannister ⟷ Pycelle	False
Joffrey-Baratheon ⟷ Jon-Arryn	False
Joffrey-Baratheon ⟷ Jory-Cassel	False
Jon-Arryn ⟷ Sansa-Stark	False
Jon-Snow ⟷ Petyr-Baelish	False
Jory-Cassel ⟷ Tyrion-Lannister	False
Petyr-Baelish ⟷ Robb-Stark	False
Robert-Baratheon ⟷ Rodrik-Cassel	False
Rodrik-Cassel ⟷ Sansa-Stark	False
Sansa-Stark ⟷ Tywin-Lannister	False

In[56]:=
```
Module[{vnm = {VertexList[g1]}^T.{VertexList[g1]}, nl = DeleteCases[
 программный мо··· список вершин графа список вершин графа удалить случаи по образцу
  Map[If[#[[1]] ≥ #[[2]], {}, #] &, Position[Normal[AdjacencyMatrix[g1]], 0]], {}],
   п··· условный оператор позиция п·· норма··· матрица смежности
 sim = Module[{vw = Table[Total[{Cases[{el1, ew1}^T, {VertexList[g1][[i]] ⟷ _, _}][[All,
  программный··· табл··· сумми··· случаи по образцу список вершин графа всё
   2]], Cases[{el1, ew1}^T, {_ ⟷ VertexList[g1][[i]], _}][[All, 2]]}, 2], {i, 1,
    случаи по образцу список вершин графа всё
  VertexCount[g1]}]}, {vw}^T.{vw} * {VertexDegree[g1]}^T.{VertexDegree[g1]}]},
   число вершин степень вершины степень вершины
 ReverseSortBy[Table[{vnm[[nl[[i, 1]], nl[[i, 2]]]][[1]] ⟷ vnm[[nl[[i, 1]], nl[[i, 2]]]][[2]],
 сортировка в об··· таблица значений
  sim[[nl[[i, 1]], nl[[i, 2]]]]}, {i, 1, Length[nl]}], Last]][[1 ;; 10]] // MatrixForm
                                  длина последний матричная форма
```

Out[56]//MatrixForm=

Drogo ⟷ Eddard-Stark	412 194 816
Arya-Stark ⟷ Tyrion-Lannister	347 139 000
Daenerys-Targaryen ⟷ Tyrion-Lannister	278 159 700
Daenerys-Targaryen ⟷ Jon-Snow	269 861 424
Jon-Snow ⟷ Petyr-Baelish	236 067 104
Cersei-Lannister ⟷ Robb-Stark	229 723 200
Catelyn-Stark ⟷ Daenerys-Targaryen	208 015 080
Jaime-Lannister ⟷ Jon-Snow	202 736 912
Eddard-Stark ⟷ Jeor-Mormont	199 402 632
Luwin ⟷ Robert-Baratheon	177 849 000

Взвешенное предпочтительное присоединение $S_{xy} = k(x)\, w(x)\, k(y)\, w(y)$, вес вершины - сумма весов ребер

In[57]:=
```
(cs = Module[{vnm = {VertexList[g1]}ᵀ.{VertexList[g1]}, nl = DeleteCases[
                                                                 Map[If[#[[1]] ≥ #[[2]], {}, #] &, Position[Normal[AdjacencyMatrix[g1]], 0]], {}],
  sim = Module[{vw = Table[Total[{Cases[{el1, ew1}ᵀ, {VertexList[g1][[i]] ↦ _, _}][[
                                   All, 2]], Cases[{el1, ew1}ᵀ, {_ ↦ VertexList[g1][[i]], _}][[
                                   All, 2]]}, 2], {i, 1, VertexCount[g1]}]},
    {vw}ᵀ.{vw} * {VertexDegree[g1]}ᵀ.{VertexDegree[g1]}]},
  ReverseSortBy[Table[{vnm[[nl[[i, 1]], nl[[i, 2]]]][[1]] ↦ vnm[[nl[[i, 1]], nl[[i, 2]]]][[2]],
    sim[[nl[[i, 1]], nl[[i, 2]]]}, {i, 1, Length[nl]}], Last]][[1 ;; 10]]) // MatrixForm
```

Out[57]//MatrixForm=

Drogo ↦ Eddard-Stark	412 194 816
Arya-Stark ↦ Tyrion-Lannister	347 139 000
Daenerys-Targaryen ↦ Tyrion-Lannister	278 159 700
Daenerys-Targaryen ↦ Jon-Snow	269 861 424
Jon-Snow ↦ Petyr-Baelish	236 067 104
Cersei-Lannister ↦ Robb-Stark	229 723 200
Catelyn-Stark ↦ Daenerys-Targaryen	208 015 080
Jaime-Lannister ↦ Jon-Snow	202 736 912
Eddard-Stark ↦ Jeor-Mormont	199 402 632
Luwin ↦ Robert-Baratheon	177 849 000

Сравнение

In[58]:=
```
Grid[
 таблица
  {Prepend[Range[10], "№"], Prepend[cs^T[[1]], "Без весов"], Prepend[cs^T[[1]], "С весами"]}^T,
   добавит… диапазон  добавить в начало  добавить в начало
  Background → {None, {{{Pink, Lighter[Blue, 0.7]}}, {1 → Gray}}},
   фон  ни одног… роз… более с… синий  серый
  Dividers → All, Spacings → {1, 1}]
   разделители  всё  размер зазора
```

Out[58]=

№	Без весов	С весами
1	Drogo ⊷ Eddard–Stark	Drogo ⊷ Eddard–Stark
2	Arya–Stark ⊷ Tyrion–Lannister	Arya–Stark ⊷ Tyrion–Lannister
3	Jaime–Lannister ⊷ Jon–Snow	Daenerys–Targaryen ⊷ Tyrion–Lannister
4	Cersei–Lannister ⊷ Robb–Stark	Daenerys–Targaryen ⊷ Jon–Snow
5	Jory–Cassel ⊷ Tyrion–Lannister	Jon–Snow ⊷ Petyr–Baelish
6	Daenerys–Targaryen ⊷ Tyrion–Lannister	Cersei–Lannister ⊷ Robb–Stark
7	Jon–Snow ⊷ Petyr–Baelish	Catelyn–Stark ⊷ Daenerys–Targaryen
8	Bran–Stark ⊷ Jaime–Lannister	Jaime–Lannister ⊷ Jon–Snow
9	Benjen–Stark ⊷ Eddard–Stark	Eddard–Stark ⊷ Jeor–Mormont
10	Petyr–Baelish ⊷ Robb–Stark	Luwin ⊷ Robert–Baratheon