```
(*2. Считать данные из полученного входного файла в
 системе Wolfram Mathematica (учесть, что размерность графа может
    быть любой и параметры bi могут идти в произвольном порядке).*)
```

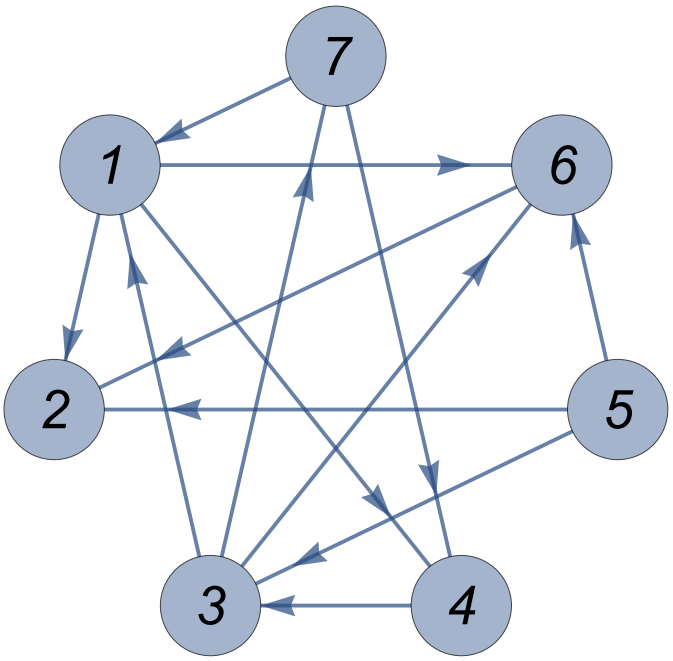
In[84]:=
```
inFileName = StringJoin[NotebookDirectory[], "input.txt"];
             соединить с… директория файла блокнота
fileStream = OpenRead[inFileName];
             открыть для считывания
Is = Read[fileStream, {Word, Number}][[2]];
     считать          слово число
Us = Read[fileStream, {Word, Number}][[2]];
     считать          слово число
U = ReadList[fileStream, Expression, Us];
    считать в список  выражение
weight = ReadList[fileStream, String, Is];
         считать в список  строка
weight =
  Sort[Table[StringSplit[weight[[i]], {"b", "_", "/*", "*/ ", " "}], {i, Is}]];
  сор… табл… разбить строку
UDirect = Table[U[[i, 1]] ↦ U[[i, 2]], {i, Us}]
          таблица значений
Close[fileStream];
закрыть
```

Out[91]=
$$\{1 \mapsto 2,\ 3 \mapsto 1,\ 1 \mapsto 4,\ 5 \mapsto 2,\ 1 \mapsto 6,\ 3 \mapsto 6,\ 3 \mapsto 7,\ 4 \mapsto 3,\ 5 \mapsto 3,\ 5 \mapsto 6,\ 6 \mapsto 2,\ 7 \mapsto 1,\ 7 \mapsto 4\}$$

In[10]:=

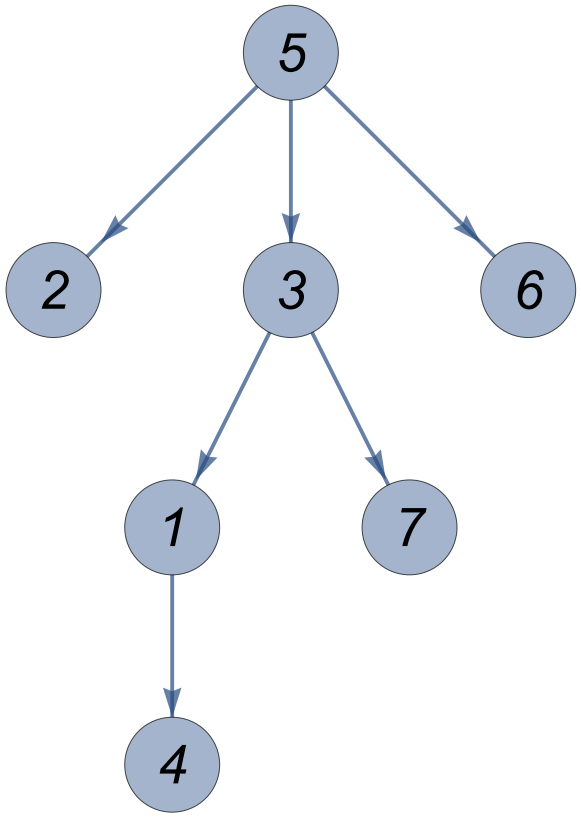
```
(*3. По полученным данным создать ориентированный граф с заданием стилей*)
g = Graph[UDirect, GraphLayout → "CircularEmbedding",
    граф              укладка графа
  VertexLabels → Placed["Name", Center], VertexSize → 0.4,
  метки для вершин  расположен      центр      размер вершины
  VertexLabelStyle → Directive[Italic, 28], EdgeShapeFunction →
  стиль меток вершин      директива   курсив      функция формы ребра
   GraphElementData["FilledArrow", "ArrowSize" → 0.05], EdgeStyle → Thick]
                                                    стиль ребра   жирный
```

Out[10]=



In[ ]:=
```
e = {5 ↔ 2, 5 ↔ 3, 5 ↔ 6, 3 ↔ 1, 3 ↔ 7, 1 ↔ 4};
g = Graph[e, GraphLayout → {"LayeredEmbedding", "RootVertex" → 5},
    граф  укладка графа
  VertexLabels → Placed["Name", Center], VertexSize → 0.4,
  метки для вершин  расположен  центр  размер вершины
  VertexLabelStyle → Directive[Italic, 28], EdgeShapeFunction →
  стиль меток вершин  директива  курсив  функция формы ребра
   GraphElementData["FilledArrow", "ArrowSize" → 0.05], EdgeStyle → Thick]
                                                        стиль ребра  жирный
p = {3, 5, 5, 1, 0, 5, 3};
depth = {2, 1, 1, 3, 0, 1, 2};

NestList[p[[#]] &, 4, depth[[4]]];
список итераций
NestWhileList[p[[#]] &, 4, depth[[#]] > 0 &];
список итераций до
h[x_] := Module[{u},
         программный модуль
   u = NestWhileList[p[[#]] &, x, depth[[#]] > 0 &];
       список итераций до
   Return[u]];
   вернуть управление
h[4]
```

Out[ ]=



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

Лабораторная 4.

In[11]:=
```
(*1. Построить покрывающее дерево для
 индивидуального графа (от произвольно заданного узла).*)
ug = UndirectedGraph[g];
     ненаправленный граф
Graph[ug, GraphLayout → "CircularEmbedding", VertexLabels → Placed["Name", Center],
граф      укладка графа                      метки для вершин  расположен      центр
  VertexSize → 0.4, VertexLabelStyle → Directive[Italic, 28], EdgeStyle → Thick];
   размер вершины     стиль меток вершин     директива  курсив      стиль ребра  жирный
DepthFirstScan[ug, 1];
проход в глубину
Ut = TreeGraph[VertexList[ug], %];
     граф дерево  список вершин графа
EdgeList[Ut]
список рёбер
(*HighlightGraph[ug,TreeGraph[VertexList[ug],%],GraphLayout→"CircularEmbedding",
  граф с подкраской     граф дерево  список вершин графа  укладка графа
  EdgeStyle→Thick, VertexLabels→Placed["Name",Center], VertexSize→0.4,
  стиль ребра  жирный  метки для вер···  расположен  центр  размер вершины
  VertexLabelStyle→ Directive[Italic, 28]];*)(*//HighlightGraph[%,#]&*)
   стиль меток вершин  директива  курсив  граф с подкраской
```

Out[15]= $\{5 \leftrightarrow 2, 1 \leftrightarrow 3, 3 \leftrightarrow 4, 3 \leftrightarrow 5, 2 \leftrightarrow 6, 4 \leftrightarrow 7\}$

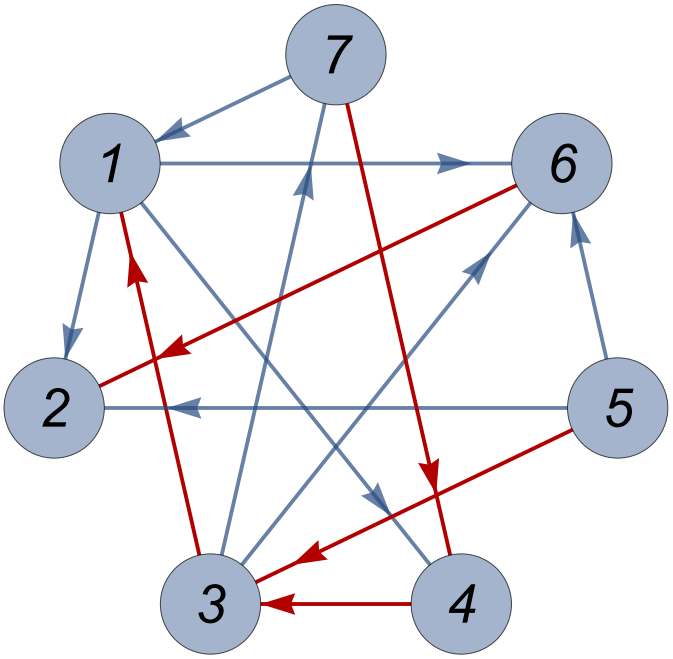
In[372]:=
```
(*3. Построить списковые структуры хранения корневого
 дерева (список узлов, список предков, список направлений,
  список глубин узлов, список связи или династического обхода;
  корень дерева-узел, от которого строили покрывающее дерево,
  либо любой произвольный узел.*)
pred = ConstantArray[0, Is];(*список предков*)
dir = ConstantArray[0, Is];(*список направлений*)
depth = ConstantArray[0, Is]; (*список глубин узлов*)
dinast = ConstantArray[0, Is];(*династичесий обход*)
posl = {};                    (*последовательность обхода*)
RootTree = {};                (*корневое дерево*)
Ut = {};                      (*покрывающее дерево*)
root = 1;
DepthFirstScan[ug, root, {"FrontierEdge" → Function[edge, {
       pred[[edge[[2]]]] = edge[[1]],
       depth[[edge[[2]]]] = depth[[edge[[1]]]] + 1,
       {i = edge[[1]] ↔ edge[[2]], AppendTo[RootTree, i],
        If[MemberQ[UDirect, i], {AppendTo[Ut, i], dir[[edge[[2]]]] = 1},
          {AppendTo[Ut, edge[[2]] ↔ edge[[1]]], dir[[edge[[2]]]] = -1}]}
      }],
   "PrevisitVertex" → Function[vertex,
     {If[posl ≠ {}, dinast[[posl[[-1]]]] = vertex], AppendTo[posl, vertex]}]
  }];
dinast[[posl[[-1]]]] = root;
(*Print[Range[Is]];
Print[pred];
Print[depth];
Print[dir];
Print[dinast];
Print[posl];*)
```

In[ ]:=
```
(*2. Вывести полученное множество дуг покрывающего дерева Ut и множество дуг U\Ut.*)
Print[UDirect]
Print[Ut]
Print[Complement[UDirect, Ut]]
```

```
{1 ↔ 2, 3 ↔ 1, 1 ↔ 4, 5 ↔ 2, 1 ↔ 6, 3 ↔ 6, 3 ↔ 7, 4 ↔ 3, 5 ↔ 3, 5 ↔ 6, 6 ↔ 2, 7 ↔ 1, 7 ↔ 4}

{3 ↔ 1, 4 ↔ 3, 7 ↔ 4, 5 ↔ 3, 5 ↔ 2, 6 ↔ 2}

{1 ↔ 2, 1 ↔ 4, 1 ↔ 6, 3 ↔ 6, 3 ↔ 7, 5 ↔ 6, 7 ↔ 1}
```

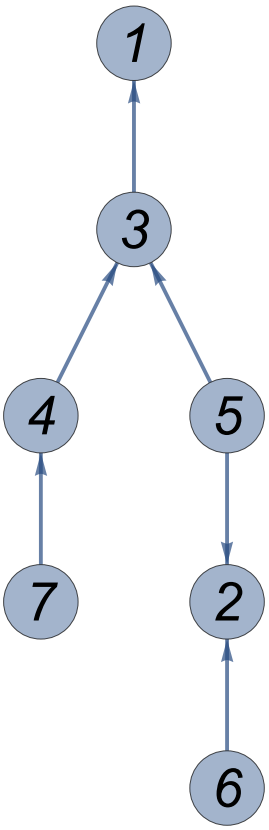
In[ ]:=
```
(*4. Вывести картинки:
  -исходный граф с подсвеченными дугами покрывающего дерева.*)

Graph[g, GraphHighlight → EdgeList[Ut], GraphLayout → "CircularEmbedding",
 EdgeStyle → Thick, VertexLabels → Placed["Name", Center],
 VertexSize → 0.4, VertexLabelStyle → Directive[Italic, 28]]
```
граф | выделить в графе | список рёбер | укладка графа
стиль ребра | жирный | метки для вершин | расположен | центр
размер вершины | стиль меток вершин | директива | курсив

Out[ ]=



In[382]:=
```
(*4. Вывести картинки:
  -покрывающее дерево с использованием layout для деревьев.*)
Graph[Ut, GraphLayout → {"LayeredEmbedding", "RootVertex" → root},
 EdgeStyle → Thick, VertexLabels → Placed["Name", Center],
 VertexSize → 0.4, VertexLabelStyle → Directive[Italic, 28]]
```
граф укладка графа стиль ребра жирный метки для вершин расположен центр размер вершины стиль меток вершин директива курсив

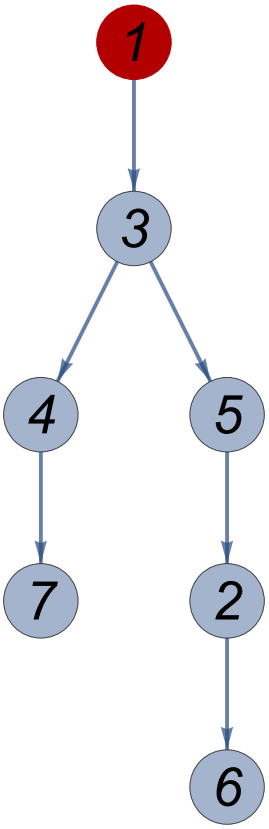
Out[382]=




```
(*4. Вывести картинки:
-корневое дерево с подсвеченным корнем.*)
```

In[384]:=
```
TreeGraph[RootTree, GraphHighlight → root,
 GraphLayout → {"LayeredEmbedding", "RootVertex" → root},
 EdgeStyle → Thick, VertexLabels → Placed["Name", Center],
 VertexSize → 0.4, VertexLabelStyle → Directive[Italic, 28]]
```

Out[384]=



In[385]:=
```
(*5. Списковые структуры вывести в виде таблицы
  (список связи или династического обхода можно не встраивать в таблицу,
    а вывести отдельным списком).*)
Grid[{Prepend[Range[Is], "i"], Prepend[pred, "pred[i]"], Prepend[depth, "depth[i]"],
  Prepend[dir, "dir[i]"], Prepend[dinast, "dinast[i]"]},
 Frame → All, ItemStyle → {{Bold}, {Bold}}]
Print[posl]
```

Out[385]=

i	1	2	3	4	5	6	7
pred[i]	0	5	1	3	3	2	4
depth[i]	0	3	1	2	2	4	3
dir[i]	0	1	−1	−1	−1	−1	−1
dinast[i]	3	6	4	7	2	1	5

```
{1, 3, 4, 7, 5, 2, 6}
```