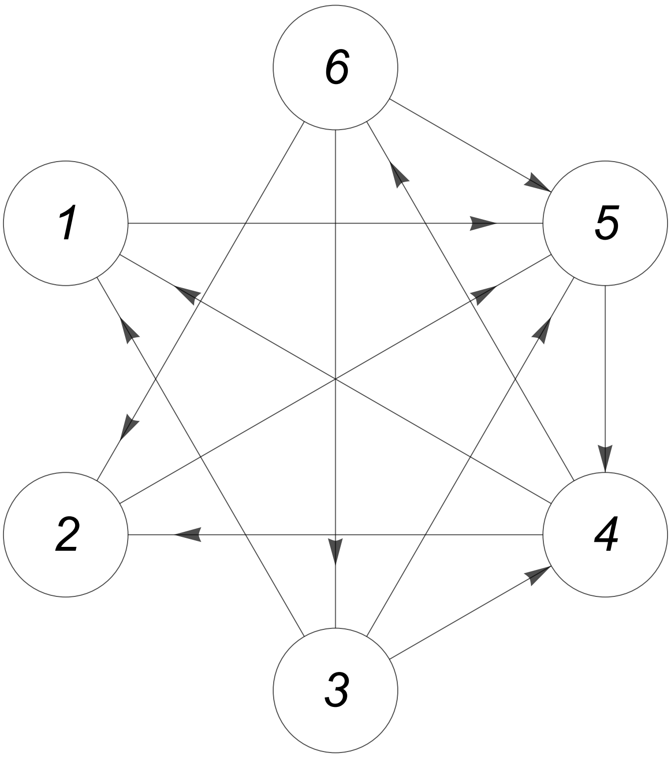
Dunaev Viktor, 3 kurs, 6 group, 23 variant

Pre - Task

```
In[299]:=  fname = NotebookDirectory[] <> "input.txt"

Out[299]=  C:\6_Cemestr\DS_Laguto\input.txt

In[300]:=  stream = OpenRead[fname];

In[301]:=  vertexNum = Read[stream, {Word, Number}][[2]]

Out[301]=  6

In[302]:=  edgesNum = Read[stream, {Word, Number}][[2]]

Out[302]=  12

In[303]:=  edges = Table[#[[1]] → #[[2]] &[ToExpression[Read[stream]]], edgesNum]

Out[303]=  {1 → 5, 2 → 5, 3 → 1, 3 → 4, 3 → 5, 4 → 1, 4 → 2, 4 → 6, 5 → 4, 6 → 2, 6 → 3, 6 → 5}

In[304]:=  vertex = Table[i, {i, 1, vertexNum}]

Out[304]=  {1, 2, 3, 4, 5, 6}

In[305]:=  weight = Range[vertexNum]

Out[305]=  {1, 2, 3, 4, 5, 6}

In[306]:=  Table[(pos = ToExpression[Characters[#[[1]]][[5]]];
               val = #[[2]];
               weight[[pos]] = val) &[Read[stream, {Word, Number}]], vertexNum]

Out[306]=  {7, 4, -1, -7, -2, -1}
```

```
In[307]:= g = Graph[vertex, edges, VertexSize -> Large,
            VertexStyle -> White, VertexLabels -> Placed["Name", Center],
            VertexLabelStyle -> Directive[Black, Italic, 25], GraphLayout ->
             {"CircularEmbedding"}, EdgeShapeFunction -> "Arrow", EdgeStyle -> Black]
```

Out[307]=



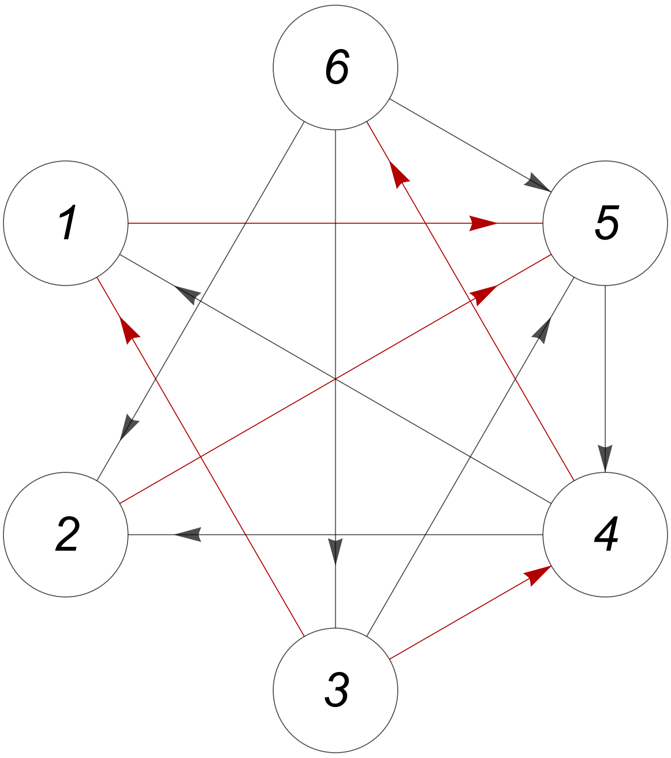
Task I

```
In[308]:= (*    1) Реализовать функцию пользователя построения характеристических векторов.
                 2) Вывести покрывающее дерево графа с циклами,
            порожденными дугами множества Un (см.пример).
                 3) Вычислить характеристические вектора,порожденные дугами
          множества Un.Компоненты веторов вывести в виде таблицы (см.пример) *)
```

```
In[309]:= startNode = RandomChoice[vertex]
Out[309]= 6

In[310]:= dirs = Table[0, Length[vertex]]
Out[310]= {0, 0, 0, 0, 0, 0}

In[311]:= pred = Table[0, Length[vertex]]
Out[311]= {0, 0, 0, 0, 0, 0}

In[312]:= depth = Table[0 , Length[vertex]]
Out[312]= {0, 0, 0, 0, 0, 0}
```

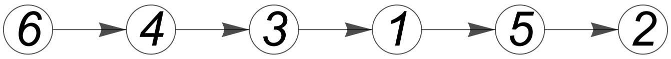
```
In[313]:= dinast = {}

Out[313]= {}

In[314]:= tree = {}

Out[314]= {}

In[315]:= graphTree = {}

Out[315]= {}

In[316]:= DepthFirstScan[UndirectedGraph@g, startNode,
            {"FrontierEdge" → ((edge = # /. (x_ •—• y_) → (x •⟶ y); AppendTo[tree, edge];
                          If[MemberQ[edges, edge], dirs⟦#⟦2⟧⟧ = 1;
                            AppendTo[graphTree, edge], dirs⟦#⟦2⟧⟧ = −1;
                            AppendTo[graphTree, #⟦2⟧ •⟶ #⟦1⟧]];
                          pred⟦#⟦2⟧⟧ = #⟦1⟧;
                          depth⟦#⟦2⟧⟧ = depth⟦#⟦1⟧⟧ + 1) &), "PrevisitVertex" → ((AppendTo[dinast, #]) &)}]

Out[316]= {3, 5, 4, 6, 1, 6}

In[317]:= tree

Out[317]= {6 •⟶ 4, 4 •⟶ 3, 3 •⟶ 1, 1 •⟶ 5, 5 •⟶ 2}

In[318]:= HighlightGraph[g, graphTree]
```

Out[318]=



```
In[319]:= dirs

Out[319]= {1, −1, −1, −1, 1, 0}
```

```
In[320]:= pred

Out[320]= {3, 5, 4, 6, 1, 0}

In[321]:= dinast

Out[321]= {6, 4, 3, 1, 5, 2}

In[322]:= depth

Out[322]= {3, 5, 2, 1, 4, 0}

In[323]:= Graph[tree,  VertexSize −> Large,  VertexStyle −> White,
            VertexLabels −> Placed["Name", Center], VertexLabelStyle −> Directive[Black, Italic, 25],
            GraphLayout −> {"RadialEmbedding"}, EdgeShapeFunction −> "Arrow",  EdgeStyle −> Black]

Out[323]=
```

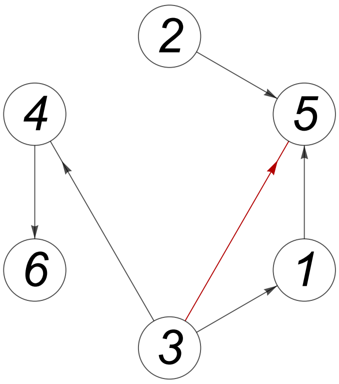


```
In[324]:= TableForm[{vertex, pred, depth,  dirs},
            TableHeadings → {{"Вершины", "Список предков", "Список глубин",  "Список направлений"}}]

Out[324]//TableForm=
          Вершины              | 1    2    3    4    5    6
          Список предков       | 3    5    4    6    1    0
          Список глубин        | 3    5    2    1    4    0
          Список направлений   | 1   −1   −1   −1    1    0

In[325]:= xp  =  Table[0, Length[vertex] ];

In[326]:= For[n  =  Length[vertex], n > 1, n−−,
            i  =  dinast[[n]];
            xp[[i]] +=  dirs[[i]] * weight[[i]];
            xp[[pred[[i]]]] +=  dirs[[i]] * dirs[[pred[[i]]]] * xp[[i]];
          ]

In[327]:= xp

Out[327]= {9, −4, −8, −1, 2, 0}

In[328]:= Un = Select[edges, Not[MemberQ[graphTree, #]] &]

Out[328]= {3 ↦ 5, 4 ↦ 1, 4 ↦ 2, 5 ↦ 4, 6 ↦ 2, 6 ↦ 3, 6 ↦ 5}

In[329]:= Ut = graphTree

Out[329]= {4 ↦ 6, 3 ↦ 4, 3 ↦ 1, 1 ↦ 5, 2 ↦ 5}

In[330]:= table  =  Table[0, Length[Un], edgesNum];

In[331]:=  edgesSet =  Join[Un, Ut]

Out[331]= {3 ↦ 5, 4 ↦ 1, 4 ↦ 2, 5 ↦ 4, 6 ↦ 2, 6 ↦ 3, 6 ↦ 5, 4 ↦ 6, 3 ↦ 4, 3 ↦ 1, 1 ↦ 5, 2 ↦ 5}

In[332]:= graphs  =  {};
```

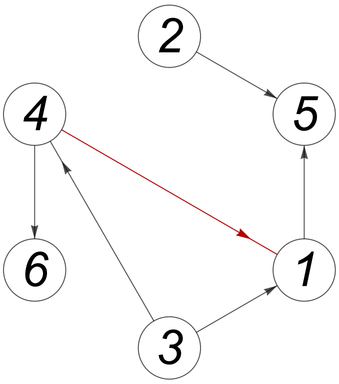
```
In[333]:= For[i = 1, i ≤ Length[Un], i++,
            τ = Un⟦i⟧⟦1⟧;
            ρ = Un⟦i⟧⟦2⟧;
            δ = 0;
            AppendTo[graphs, Graph[Join[Ut, {τ •→ ρ}], GraphHighlight → {τ •→ ρ},
                GraphHighlight → {τ •→ ρ}, VertexSize -> Large, VertexStyle -> White,
                VertexLabels -> Placed["Name", Center], VertexLabelStyle -> Directive[Black, Italic, 25],
                GraphLayout -> {"CircularEmbedding"}, EdgeShapeFunction -> "Arrow", EdgeStyle -> Black]];
            table⟦i⟧⟦i⟧ = 1;
            If[depth⟦τ⟧ > depth⟦ρ⟧, δ = 1, τ = Un⟦i⟧⟦2⟧; ρ = Un⟦i⟧⟦1⟧; δ = −1];
            depthDelta = depth⟦τ⟧ − depth⟦ρ⟧;
            For[j = 0, j < depthDelta, j++,
              ver = pred⟦τ⟧;
              If[dirs⟦τ⟧ > 0, rib = ver •→ τ, rib = τ •→ ver];
              pos = Position[edgesSet, rib]⟦1⟧;
              table⟦i⟧⟦pos⟧ = dirs⟦τ⟧ * δ;
              τ = ver;
            ];
            If[τ ≠ ρ,
              While[True,
                predT = pred⟦τ⟧;
                predRho = pred⟦ρ⟧;
                If[dirs⟦τ⟧ > 0, ribT = predT •→ τ, ribT = τ •→ predT];
                If[dirs⟦ρ⟧ > 0, ribRho = predRho •→ ρ, ribRho = ρ •→ predRho];
                posT = Position[edgesSet, ribT]⟦1⟧;
                table⟦i⟧⟦posT⟧ = dirs⟦τ⟧ * δ;
                posRho = Position[edgesSet, ribRho]⟦1⟧;
                table⟦i⟧⟦posRho⟧ = dirs⟦ρ⟧ * δ * −1;
                τ = predT;
                ρ = predRho;
                If[τ == ρ, Break[]]
              ]
            ]
          ]

In[334]:= TableForm[table, TableHeadings → {Un, edgesSet}]
Out[334]//TableForm=
              | 3 •→ 5   4 •→ 1   4 •→ 2   5 •→ 4   6 •→ 2   6 •→ 3   6 •→ 5   4 •→ 6   3 •→ 4   3 •–
        ------+----------------------------------------------------------------------------------------
        3 •→ 5| 1        0        0        0        0        0        0        0        0        −1
        4 •→ 1| 0        1        0        0        0        0        0        0        1        −1
        4 •→ 2| 0        0        1        0        0        0        0        0        1        −1
        5 •→ 4| 0        0        0        1        0        0        0        0        −1       1
        6 •→ 2| 0        0        0        0        1        0        0        1        1        −1
        6 •→ 3| 0        0        0        0        0        1        0        1        1        0
        6 •→ 5| 0        0        0        0        0        0        1        1        1        −1
```

In[335]:= **graphs**

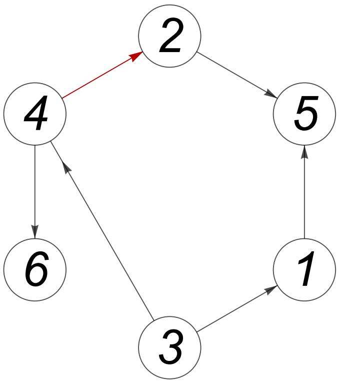
Out[335]= {



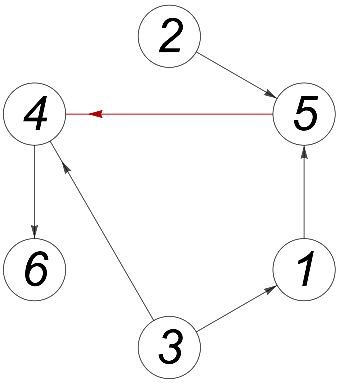
,



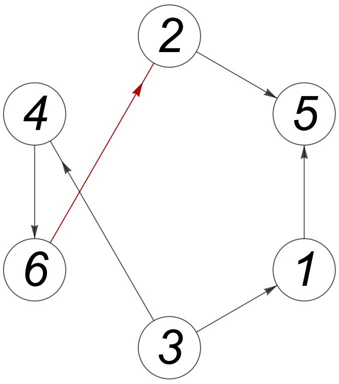
,



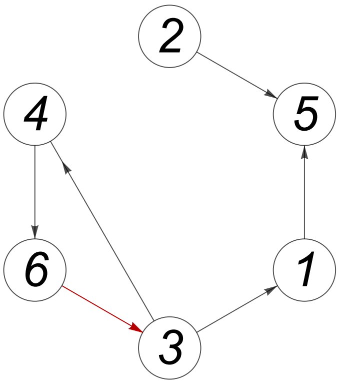
,



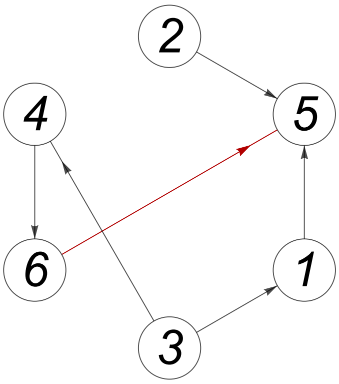
,



,



,



}