Use of CostOfPath and ShortestPath in a Digraph Defined by a Matrix of Weights (Costs)

By José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/
Mexico

Load the package. Press [SHIFT]-[ENTER] at the same time with the cursor on next command.

```
Needs["Combinatorica`"]
```

This is matrix represents the weights (costs) for going from one vertex to another in a graph with 5 vertices. For example, for going from vertex 5 to vertex 2 the cost is 0.20 (5th row, 2nd column). Notice that the cost for going in the opposite direction, from vertex 2 to vertex 5, is different (2nd row, 5th column gives 0.05).
A cost of 0 is assigned to pairs of vertices that are actually **disconnected**. Notice that vertices can be disconnected in one direction but connected in the other:

$$\text{mymatrix} = \begin{pmatrix} 1 & 0.10 & 0.15 & 0.05 & 0 \\ 0.05 & 1 & 0 & 0 & 0.05 \\ 0 & 0.10 & 1 & 0.15 & 0 \\ 0.15 & 0.05 & 0 & 1 & 0.20 \\ 0 & 0.20 & 0.05 & 0.15 & 1 \end{pmatrix}$$

```
{{1, 0.1, 0.15, 0.05, 0}, {0.05, 1, 0, 0, 0.05},
 {0, 0.1, 1, 0.15, 0}, {0.15, 0.05, 0, 1, 0.2}, {0, 0.2, 0.05, 0.15, 1}}
```

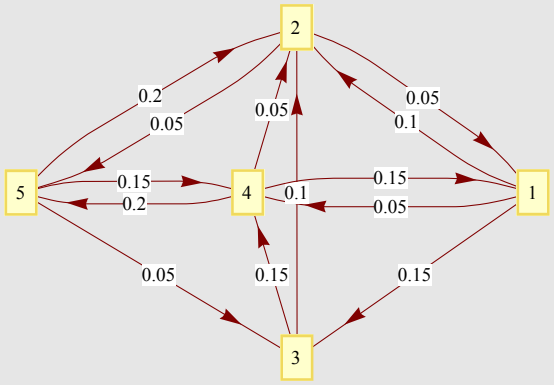
Next commands transform the matrix of weights (mymatrix) into a list of edges and weights (myconnections) with the proper syntax for the command GraphPlot.
Notice that the last command (DeleteCases) **erases those connections with a weight of 0 or 0.0**, since we are using a weight of 0 or 0.0 to indicate **disconnected** vertices:

```
myperm = Permutations[Range[Length[mymatrix]], {2}];
allconnections = Map[Function[edge, {edge[[1]] → edge[[2]],
      mymatrix[[edge[[1]], edge[[2]]]]}], myperm];
myconnections = DeleteCases[allconnections, {_, 0 | 0.0}]
```

```
{{1 → 2, 0.1}, {1 → 3, 0.15}, {1 → 4, 0.05}, {2 → 1, 0.05}, {2 → 5, 0.05}, {3 → 2, 0.1}, {3 → 4, 0.15},
 {4 → 1, 0.15}, {4 → 2, 0.05}, {4 → 5, 0.2}, {5 → 2, 0.2}, {5 → 3, 0.05}, {5 → 4, 0.15}}
```

Plot the graph with edges and weights that were stored above in the variable myconnections

```
GraphPlot[myconnections, DirectedEdges → True, VertexLabeling → True]
```



Next commands transform the matrix of weights (mymatrix) into a list of weights (myweights) with the proper syntax for the command SetGraphOptions:
NOTICE THAT IS NECESSARY TO SPECIFY A LARGE WEIGHT (1000) FOR VERTICES THAT ARE ACTUALLY DISCONNECTED. If we do not specify a weight, then other commandswill assign a weight of 1 (one) that connection that should not exist.

```
myperm = Permutations[Range[Length[mymatrix]], {2}];
allweights = Map[Function[edge, {{edge[[1]], edge[[2]]},
     EdgeWeight → mymatrix[[edge[[1]], edge[[2]]]]}], myperm];
myweights = ReplaceAll[allweights,
  ({v_, EdgeWeight → 0}) :→ {v, EdgeWeight → 1000}]
```

```
{{{1, 2}, EdgeWeight → 0.1}, {{1, 3}, EdgeWeight → 0.15},
 {{1, 4}, EdgeWeight → 0.05}, {{1, 5}, EdgeWeight → 1000}, {{2, 1}, EdgeWeight → 0.05},
 {{2, 3}, EdgeWeight → 1000}, {{2, 4}, EdgeWeight → 1000}, {{2, 5}, EdgeWeight → 0.05},
 {{3, 1}, EdgeWeight → 1000}, {{3, 2}, EdgeWeight → 0.1}, {{3, 4}, EdgeWeight → 0.15},
 {{3, 5}, EdgeWeight → 1000}, {{4, 1}, EdgeWeight → 0.15}, {{4, 2}, EdgeWeight → 0.05},
 {{4, 3}, EdgeWeight → 1000}, {{4, 5}, EdgeWeight → 0.2}, {{5, 1}, EdgeWeight → 1000},
 {{5, 2}, EdgeWeight → 0.2}, {{5, 3}, EdgeWeight → 0.05}, {{5, 4}, EdgeWeight → 0.15}}
```

Create a graph with the weights (costs) that were stored above in the variable myweights. SetGraphOptions, CompleteGraph and Directed can only be used if you first evaluated Needs["Combinatorica`"], see the top of this document.

```
mygraph = SetGraphOptions[
   CompleteGraph[Length[mymatrix], Type → Directed],
   myweights]
```

```
- Graph:< 20,5,Directed >-
```

Now we can use ShortestPath and CostOfPath in our graph. ShortestPath and CostOfPath can only be used if you first evaluated Needs["Combinatorica`"], see the top of this document.

```
ShortestPath[mygraph, 1, 5]
```

```
{1, 2, 5}
```

This is the cost of the path:

```
CostOfPath[mygraph, {1, 2, 5}]
```

```
0.15
```

The shortest path in the opposite direction is different:

```
ShortestPath[mygraph, 5, 1]
```

```
{5, 3, 2, 1}
```

This is the cost of the path:

```
CostOfPath[mygraph, {5, 3, 2, 1}]
```

```
0.2
```

This is a list of all shortest paths in our graph

```
myshortestpaths =
 DeleteCases[Flatten[Table[ShortestPath[mygraph, j, k],
    {j, 1, 5}, {k, 1, 5}], 1], {onevertex_}]
```

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

These are the costs of all our shortest paths:

```
myshortestcosts =
 Map[Function[{path}, CostOfPath[mygraph, path]], myshortestpaths]
```

```
{0.1, 0.15, 0.05, 0.15, 0.05, 0.1, 0.1, 0.05, 0.15,
 0.1, 0.15, 0.15, 0.1, 0.05, 0.15, 0.1, 0.2, 0.15, 0.05, 0.15}
```

Here we show each shortest path with its corresponding cost:

```
Grid[
 Join[{{"Shortest Paths", SpanFromLeft}},
  Transpose[{myshortestpaths, myshortestcosts}]], Dividers → All]
```

Shortest Paths	
{1, 2}	0.1
{1, 3}	0.15
{1, 4}	0.05
{1, 2, 5}	0.15
{2, 1}	0.05
{2, 5, 3}	0.1
{2, 1, 4}	0.1
{2, 5}	0.05
{3, 2, 1}	0.15
{3, 2}	0.1
{3, 4}	0.15
{3, 2, 5}	0.15
{4, 2, 1}	0.1
{4, 2}	0.05
{4, 2, 5, 3}	0.15
{4, 2, 5}	0.1
{5, 3, 2, 1}	0.2
{5, 3, 2}	0.15
{5, 3}	0.05
{5, 4}	0.15

This is a matrix where the (i, j)^th entry is the length of a shortest path in g between vertices i and j.

```
shortestpathmatrix = AllPairsShortestPath[mygraph]
```

```
{{0, 0.1, 0.15, 0.05, 0.15}, {0.05, 0, 0.1, 0.1, 0.05},
 {0.15, 0.1, 0, 0.15, 0.15}, {0.1, 0.05, 0.15, 0, 0.1}, {0.2, 0.15, 0.05, 0.15, 0}}
```

In matrix form:

```
MatrixForm[shortestpathmatrix]
```

$$\begin{pmatrix} 0 & 0.1 & 0.15 & 0.05 & 0.15 \\ 0.05 & 0 & 0.1 & 0.1 & 0.05 \\ 0.15 & 0.1 & 0 & 0.15 & 0.15 \\ 0.1 & 0.05 & 0.15 & 0 & 0.1 \\ 0.2 & 0.15 & 0.05 & 0.15 & 0 \end{pmatrix}$$