EX1: I'm choosing the first paper - "Autoscaling Bloom Filters - Controlling trade-off between true and false positives". The reason why I chose this topic, it's because when I was reading about the counting Bloom Filters for the first time I was really interested in the way how it's possible to improve it. The article propose Autoscaling Bloom Filter which is derived from a CBF and allows minimization of the false positive rate without requiring rebuilding of the entire filter. Also, there was shown the mathematical analysis and experimental evaluation of Autoscaling Bloom Filters. I will try to briefly talk in the essay what is written in the paper.

EX2: I've implemented a BFS algorithm for facebook and twitter data. Measured time for 1000 BFS searches and estimated the diameter of the graphs, also I showed the distribution of lengths of longest shortest paths. Code was written in Python 2 as snap did not publish the version for the Python 3 on Windows. To run the code I've included two folders with .edges files.

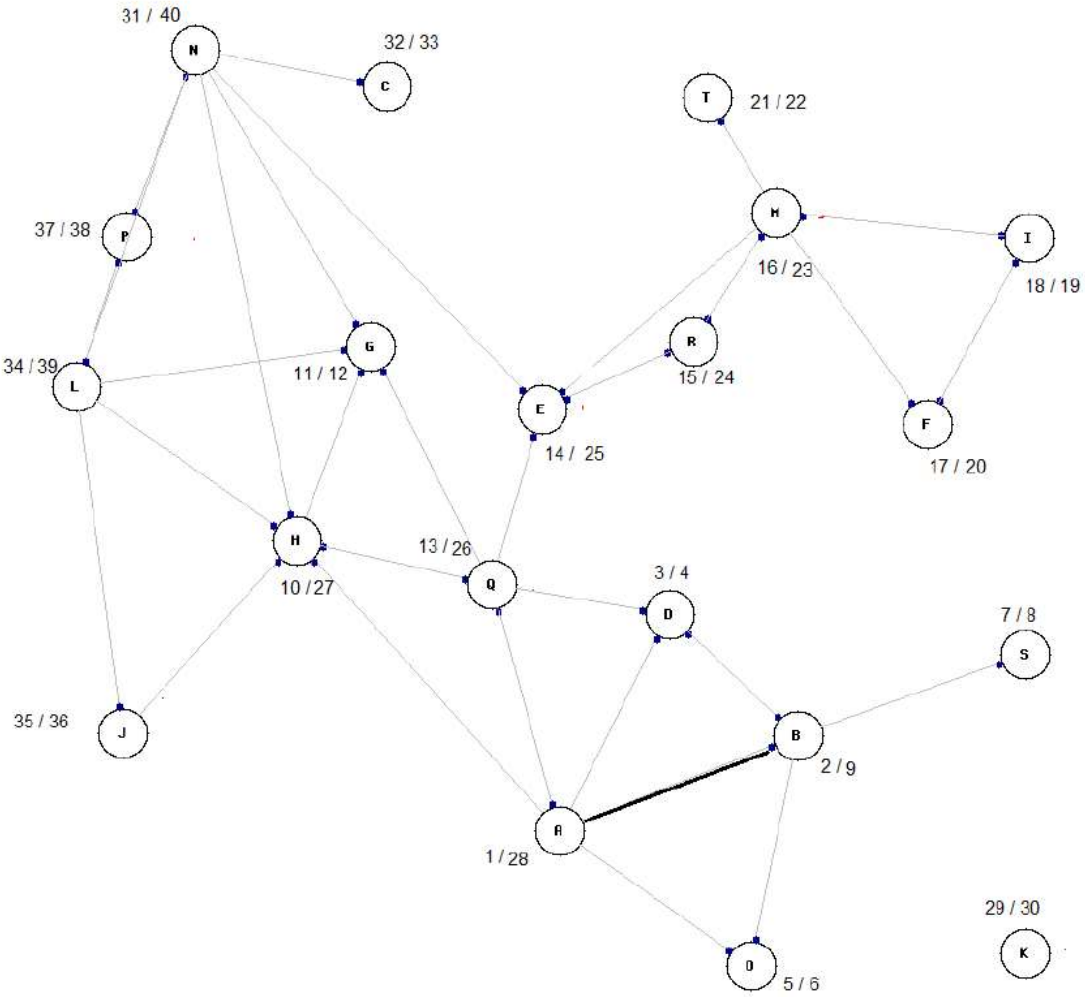
EX4: For identification of Strongly connected components there several steps.

1. We should find DFS for directed graph as in Graph 5.

In [1]:
```
from IPython.display import Image

Image("ex4_1.png")
```

Out[1]:



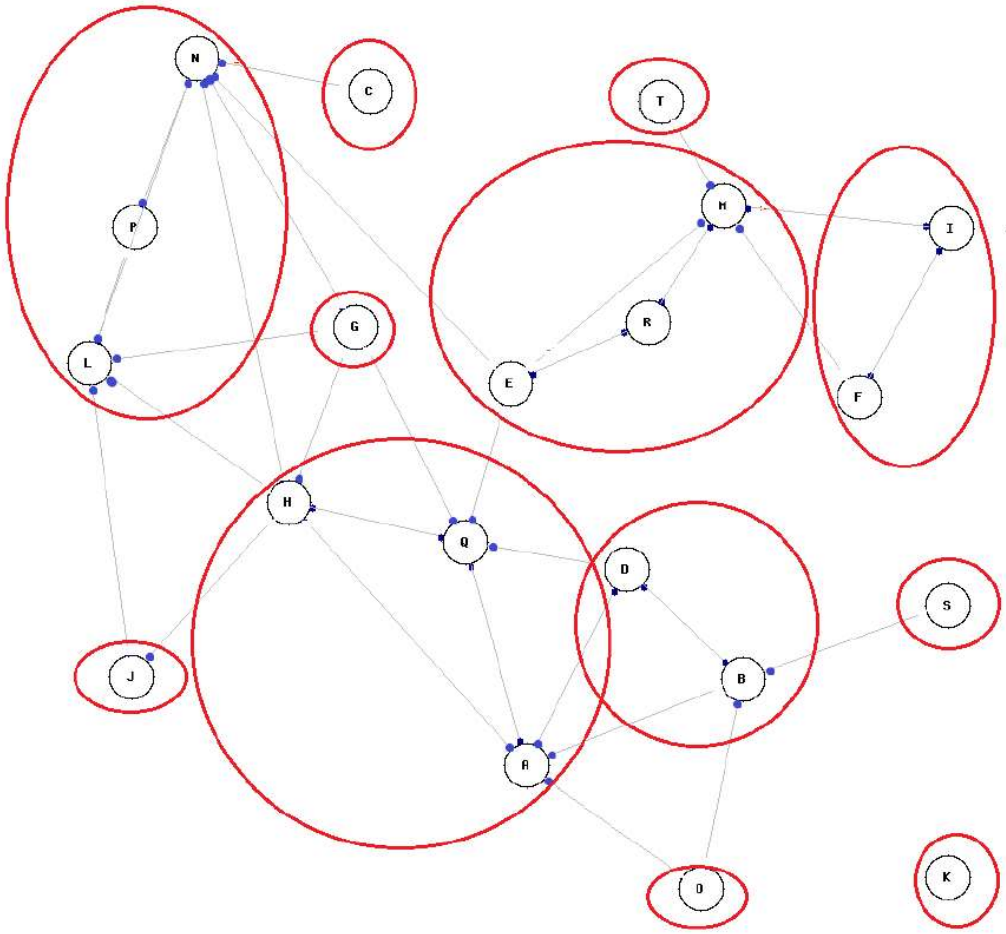
In Graph 5. Given discovery and finish time for each node.

1. Then numeration/denomination numbers. Topological Sorting using Finishing Decreasing Time: N, L , P, J, C, K, A, H, Q, E, R, M, T ,F, I, G, B, S, O, D
2. We should reverse the graph (change the direction of arrows)
3. And finally choose Strongly connected components.

In [2]:
```
from IPython.display import Image

Image("ex4_2.png")
```

Out[2]:



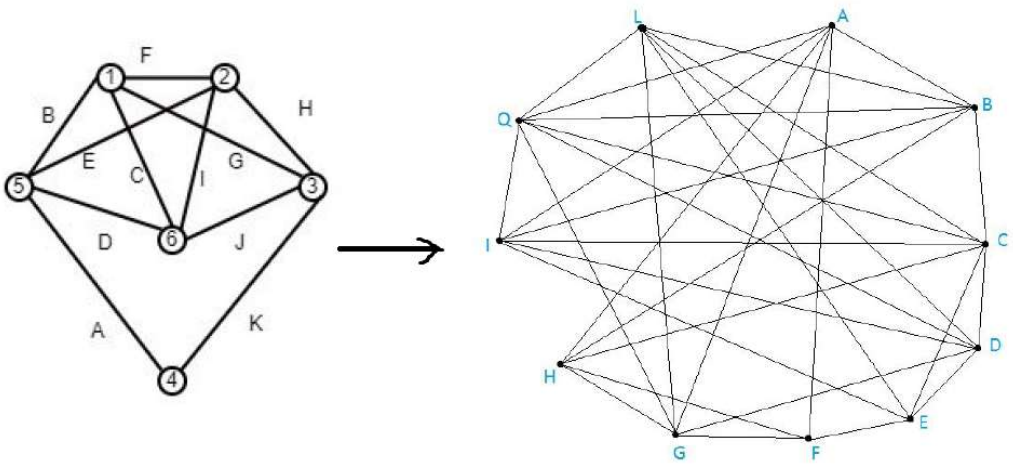
(N, P, L) (J) (C) (K) (A, H, Q) (E, R, M) (T) (F, I) (G) (B, D) (S) (O)

EX5: An Eulerian trail (or Eulerian path) is a trail in a finite graph which visits every edge exactly once and all vertices in the graph have an even degree. A Hamiltonian path (or traceable path) is a path in an undirected or directed graph that visits each vertex exactly once. Below you can observe the trasformation to the line graph.

In [3]:
```
from IPython.display import Image

Image("ex5_1.png")
```

Out[3]:



The Euler Circuit is constructed by following the nodes alphabetically. The Euler circuit can be done only because each node degree is even in this graph. When transforming to the line graph, the properties of the whole graph do not change.As the property does not change but we trade edges for nodes, we can construct the same Euler circuit on the line graph by following the alphabetical order. As for the Hamilton circuit, there is no algorithm which gives a good polynomial time solution for all types of circuit. However, in this particular case, the Euler circuit of the line graph is actually the Hamilton circuit of the other graph, and vice-versa. The Hamilton circuit is defined as the circuit going through each node once, Eulerian circuit is defined as the circuit going through each edge once.