

# Network Analytics

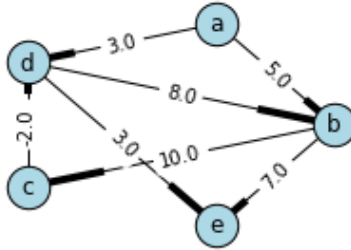
## Homework 1 - Group Part

### Group 1

November 22, 2016

#### Exercise 1

In this exercise we will use the directed graph specified in the HW1\_problem1.txt file. The directed graph is shown in the following figure.



(a) First print the node-arc incidence matrix, where each row represents a node and each column represents an edge. If an edge goes from node  $i$  to node  $j$  then  $\alpha_{ik} = -1$  and  $\alpha_{jk} = 1$ . The rest of the elements are equal to zero. The resulting matrix is as follows:

Table 1: Adjacency matrix of directed graph

	0	1	2	3	4	5	6
a	0	0	-1	-1	0	0	0
b	-1	-1	1	0	1	0	0
c	0	1	0	0	0	0	-1
d	0	0	0	1	-1	-1	1
e	1	0	0	0	0	1	0

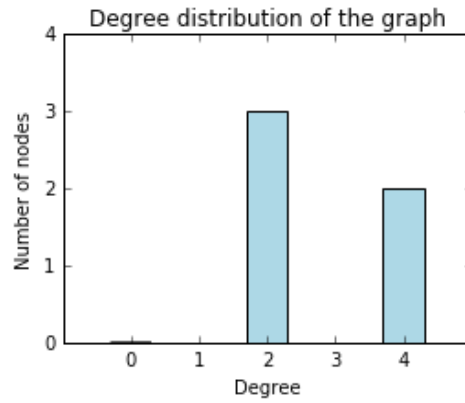
(b) Next, we will calculate and print the shortest-path matrix which presents the lengths of the shortest paths between all the nodes. Since this graph includes negative weight, we use Floyd-Warshall algorithm to compute all-pairs shortest path lengths. Note that in case there is no possible path from node  $i$  to node  $j$ , then the shortest path length,  $\alpha_{ij} = \infty$ . The shortest-path matrix is the following:

Table 2: Shortest paths matrix of directed graph

	a	b	c	d	e
a	0	5	15	3	6
b	$\infty$	0	10	8	7
c	$\infty$	6	0	-2	1
d	$\infty$	8	18	0	3
e	$\infty$	$\infty$	$\infty$	$\infty$	0

(c) The diameter  $d$  of the graph, which is the longest shortest path amongst all the paths, is the maximum finite element in the second table, and is in this case 18.

(d) The total degree distribution of the graph represents the total number of edges (including both incoming and outgoing edges) each node is connected to and is shown in the following bar chart.



(e) Finally, we will check whether the graph is connected. As it is a directed graph it can be either *strongly connected*, which is the case when there is a path from each node to every other node in the graph, or *weakly connected*, which is the case when the equivalent undirected graph is connected. Our graph is weakly connected but not strongly connected.

---

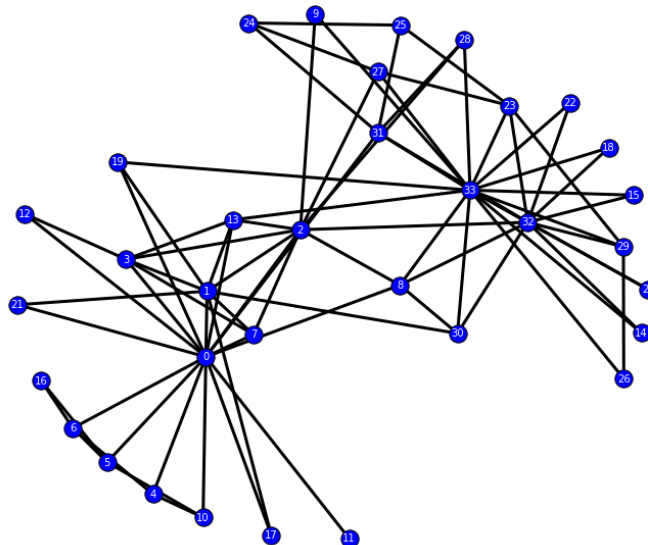
## Exercise 2

In this exercise we will use the undirected graph specified in the `HW2_problem2.txt` file.

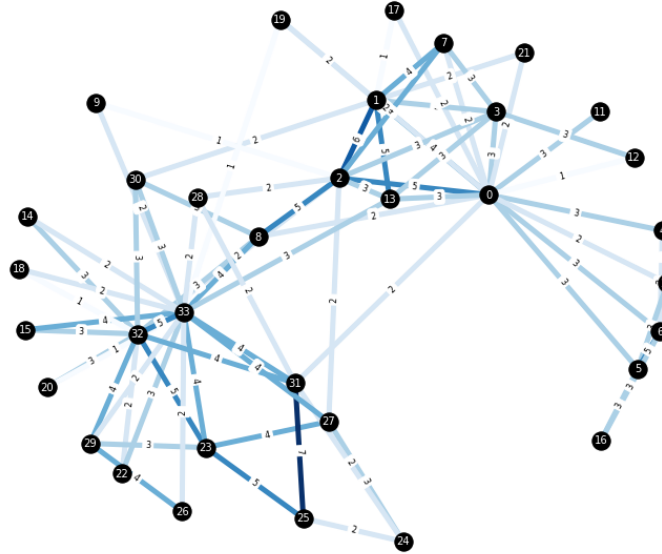
The file contains information about an undirected graph of 34 nodes and is consisted of a matrix of 68 rows and 34 columns. The first 34 rows represent the adjacency matrix of the undirected graph and the rest of the rows contain the weights associated with each edge.

Before drawing the graph, we create and apply the function `graphAdjMatrix()` to the two matrices that represent the adjacency matrix and the corresponding weights to check if the adjacency matrix contains more edges than the weighted adjacency matrix (i.e. edges with 0 weight). In case it does, the function adds 0-weight edges to the final graph for the edges that are actually represented in the adjacency matrix but cannot be recognised in the weight-related matrix due to the value of their weight which is equal to zero. In this particular graph though, we do not have such 0-weight edges.

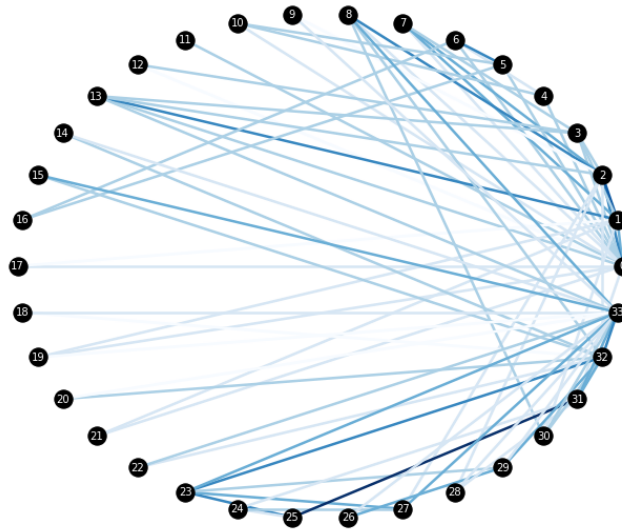
We first draw the undirected network without the weights. The resulting graph is as follows:



Then we represent the weights of the edges using both labels (number shown on each edge) and colours (darker colour indicates heavier weight) of the edges. We present the graph in two different layouts (spring layout and circular layout).



The above two graphs were plotted using spring layout. This layout attempts to minimise the number of crossing edges and allow us to inspect the structure of the graph.



The last circular format (shown above) of the graph, which represents edge-weights using colour (with darker colour indicating heavier weight), gives a clearer image of the structure of the graph, as we can easily see which nodes are the ones with more edges and which ones are not. At the same time, we can easily compare the weights of different edges that belong to the same nodes.

---

## Notes

The code used in the assignment is included in the `Group1_HW1.py` file.