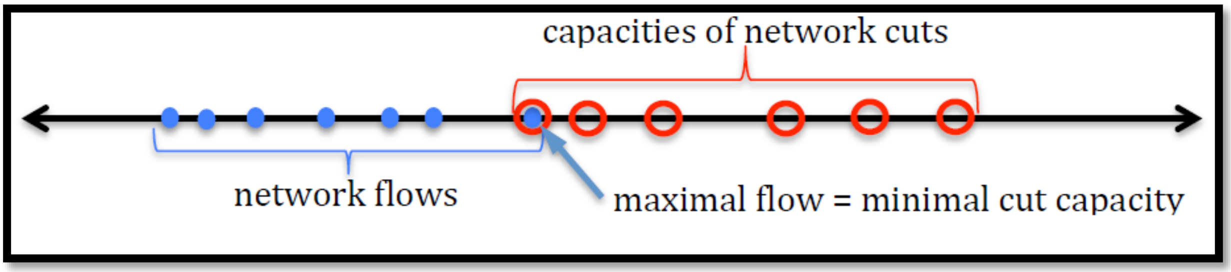
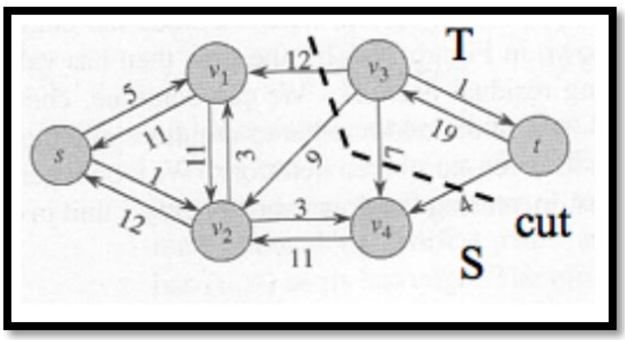
Dheeraj Satya Sushant Viswanadham
(z5204820)

COMP3121 Assignment 5 – Question 3

3) We are given N computers in a network which are labelled $\{1, 2, 3, \ldots, N\}$ respectively. We are also told that there are M unidirectional links which connects pairs of computers. If computer 1 is trying to send a virus to computer N (which can happen as long as a path exists between the two computers), then we are required to prevent this from happening by removing some of the links such that the two computers are no longer connected. In order to remove the links, we need to prioritise the ones which result in the minimum total cost to disconnect the computers and find the edges which should be removed to achieve this.

See below for an overall visual diagram (taken from lecture notes) for what we are trying to achieve:





We can tackle this problem as we would a typical Max Flow – Min Cut (or Min-**cost**) problem. Firstly, we need to construct a corresponding flow network with the computers as vertices and with computer 1 as the source and computer N as the sink. Since the links are one-directional, we can try to construct a graph with the computers from 1 to N as our vertices and the unidirectional links as our edges. Additionally, the question states that we have already calculated the cost for removing a link. Hence, the capacity of the edges will be the **cost** of removing that edge from our network. We then apply the Ford-Fulkerson algorithm to find the minimum cost and the corresponding edges that we should remove. Simply put: (adapted from proof in lecture slides)

We need to find a cut that will produce two subsets S and T in our computer network diagram such that:

1) $S \cup T = V$
2) $S \cap T = \phi$
3) $s \in S \text{ and } t \in T.$

The capacity $c(S, T)$ of the cut will be equal to the sum of capacities of all edges leaving S (which contains computer 1) and entering T (which contains computer N) i.e.

$$c(S, T) = \sum_{(u,v) \in E} \{c(u, v) : u \in S \;\&\; v \in T\}$$

We then take the residual graph of our flow network and determine which edges are most suitable to be cut and would result in the minimum cost. This is because when the Ford-Fulkerson algorithm terminates, it produces a flow equal to the capacity of the defined cut (using the proof in the lecture slides) where if an edge (u, v) still had capacity left, then in the residual flow network the path from *s* to *u* could be extended to a path from *s* to *v* which would otherwise contradict our assumption that $v \in T$ defined above. Hence, we now know the edges (or links) that we should remove from the network and their corresponding costs.

In conclusion, this would take an overall time-complexity of $\boldsymbol{O}(\boldsymbol{E} * |\boldsymbol{f}^*|)$, where *E* is the set of edges (or links) in the residual graph and f^* is the maximum flow.

End of Solution