ESD5 – Fall 2024
Problem Set 4

Department of Electronic Systems
Aalborg University

September 30, 2024

Problem 1 – Bellman-Ford Algorithm

The *Bellman-Ford algorithm* is an algorithm for finding and computing the shortest paths in a graph.[1] In this exercise, we are going to see an example of how this algorithm can be used when considering a weighted graph. But first, why are we interested in graphs and algorithms over them? A weighted graph can be used as a mathematical structure to represent a communication network, where nodes are devices such as computers and edges are links (cables). The weights of each edge can correspond to the communication cost/efficiency of the corresponding link. Therefore, to improve a practical network's performance, we can use graph algorithms to perform routing or other functionalities we would like our network to perform.

Consider the graph and weights illustrated in Fig. 1. Consider that the weight of each edge corresponds to the distance between the nodes. We want to find the shortest distance paths between the source node S and all the other nodes. To do this, we will apply the Bellman-Ford algorithm to find the shortest network paths. Answer the following:

(a) What is the maximum number of iterations required to complete the Bellman algorithm?

(b) Describe the Bellman algorithm through each iteration and construct the shortest path tree.

Challenge: You could also try to run the algorithm when changing the weight between A-D to -4. However, we should not interpret the weights as distances anymore because it does not make physical sense to have negative distances. In this case, instead of finding the

[1] `https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm`

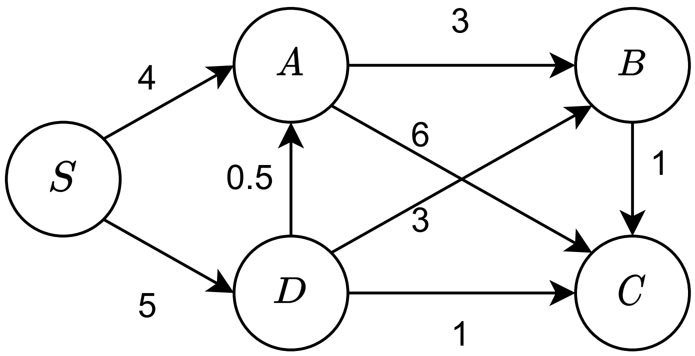


Figure 1: Weighted graph with weights representing distance.

shortest path, let's assume the **objective** is that we would like to find the path with the lowest weight. We also assume that the end path cannot have a negative weight!

Note: We could also have set the **objective** to find the path with the largest weight! Everything depends on what we want in a given context.

Problem 2 – Routing

The purpose of *routing* in a network is to find the best path to transmit a packet from a node X to another node Y. In Problem 1, we have learned how graphs can represent networks and how to use the Bellman-Ford algorithm. In this part, we will apply what we have learned to perform routing!

The *distance-vector routing protocol* is a routing protocol based on finding the shortest distances. Consider the network of Fig. 2 and that distance-vector routing protocol is used. To be familiarized with the nomenclature used by the protocol, please read `https://en.wikipedia.org/wiki/Distance-vector_routing_protocol`. Now, consider that a message is sent through the network in the figure and nodes B, D, and E have the following routing tables: B: (5, 0, 8, 12, 6, 2); D: (16, 12, 6, 0, 9, 10); E: (7, 6, 3, 9, 0, 4). Note that each routing table entry (A, B, C, D, E, F) corresponds to the *distance between the current node and all other nodes.* For example, B: (5, 0, 8, 12, 6, 2) means that the distance to transmit from B to A is 5, B to B is 0, B to C is 8, and so on. Consider that the distance between the C to B, D, and E links are 6, 3, and 5, respectively. Based on these three routing tables, *what is C's new routing table? Give the routing table and the corresponding best nodes.*

Hint: For example, to obtain the value for the first element of C's routing table, you have to determine the path that will cost the less from C to A based on the possible ways to reach A from C: through B, D, and E.

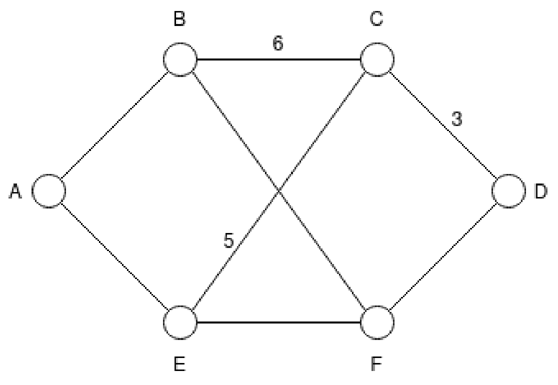


Figure 2: Example of a network.

Problem 3 – Flow control

A CPU executes instructions at the rate of 1000 MIPS (million instructions per second). A transmitter wants to send a packet of 64 bits to a receiver that uses this CPU. Consider that the processing of each packet costs 10 instructions. Consider that the CPU must `copy` the packet 4 times, where the `copy` operation over a packet can occur simultaneously. Can the receiver's CPU handle it if the transmitter and receiver are connected using a link to transmit data with 1 Gbps? For simplicity, assume that all instructions, even those that read or write memory, run at the total 1000-MIPS rate.

Motivation: This exercise represents an example of flow control, where the data rate of the link between the transmitter and the receiver should adapt to the processing capability of the receiver! There is no point in transmitting faster than the receiver can handle it.

Problem 4 – TCP/IP

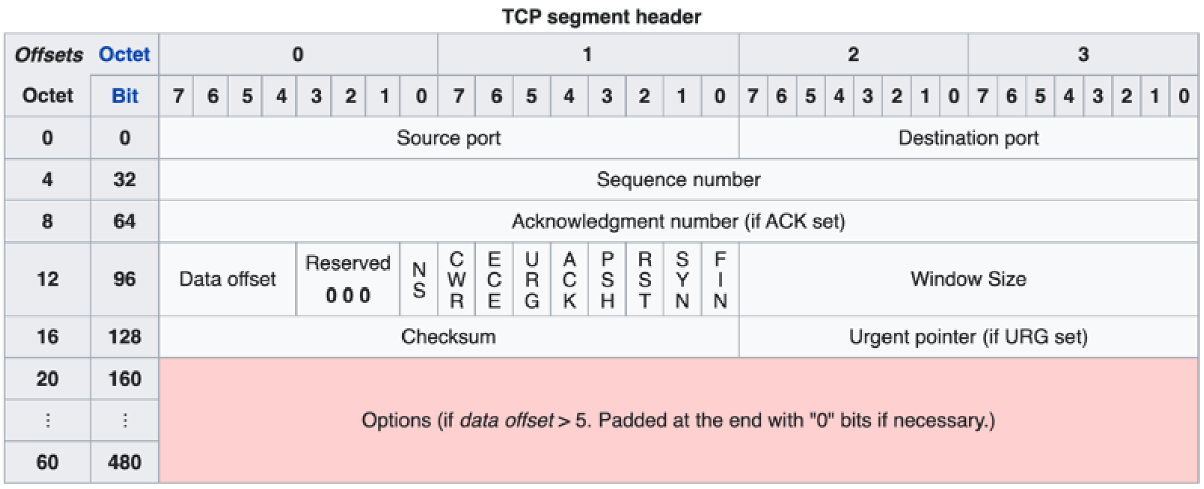
This exercise suggests that you read and familiarize yourself with the TCP-/IP, the protocol that enabled the internet boom.[2] Without this protocol, you would not be able to have Netflix, YouTube, and social media.

(a) To address the limitations of IP version 4, a major effort had to be undertaken via IETF that resulted in the design of IP version 6. There is still a significant reluctance to adopt this new version. However, no such major effort is needed to address the limitations of TCP. Explain why this is the case.

(b) In the figure of the TCP segment header as in Fig. 3, we saw that in addition to the 32-bit acknowledgment field, there is an ACK bit in the fourth word. Does this add

[2]For a quick intro, you can find YouTube videos, such as `https://www.youtube.com/watch?v=PpsEaqJV_A0&ab_channel=Techquickie`.

anything? Why or why not?



TCP segment header

Offsets	Octet	0								1								2								3							
Octet	**Bit**	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	**0**	Source port																Destination port															
4	**32**	Sequence number																															
8	**64**	Acknowledgment number (if ACK set)																															
12	**96**	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	**128**	Checksum																Urgent pointer (if URG set)															
20	**160**	Options (if *data offset* > 5. Padded at the end with "0" bits if necessary.)																															
⋮	⋮																																
60	**480**																																

Figure 3: TCP segment header.