This assignment is **due on April 26** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go to the last page and read the section: "Advice on how to do the assignment".

Important: *This assignment is for all COMP3x27 students. COMP3027 students should do Problems 1 and 2 while COMP3927 students should do Problems 1 and 3.*

"Historical" background. It is a well-known and historically established fact that Viking engineers had ingenious networks to distribute their (non-alcoholic) ale to the Vikings.[1] Unfortunately, a new chief has come to power and he's rather set on making sure he gets all the ale and as fast as possible using the existing network. Even more unfortunately, you're the second in command, so it's your job to handle that. So, you grudgingly closed off everyone else's access to this delicious drink, leaving only the chief's tap functioning. For a time this was good, but the chief thinks he isn't getting his ale fast enough, so you're instructed to investigate the current network and its associated flow of ale and determine whether this can be improved, possibly by modifying the network. You'd better do this fast, he seems thirsty!

Throughout this assignment, you can assume that the network you're given is connected, that the capacities are all positive integers, and that the flow you're given is feasible and integtal. Additionally, you can assume that you're given the network and the flow's residual graph in adjacency list form.

Problem 1. (20 points)

We start by investigating whether the current amount of ale reaching the chief is really the best we can do given our current network. To allow for future use of our methods, instead of checking only the current network and flow of ale, your aim is to design an algorithm that can determine this for any network and flow. In other words, given a directed graph G and two vertices s (the brewery: the source!) and t (the chief's house, unfortunately the terminus) along with capacities of all the pipes in our network, as well as the current flow through these pipes, design an algorithm to determine whether the current flow is the maximum possible in the network.

Your task is to design the above algorithm. Your algorithm should run in $O(n + m)$ time, where n is the number of vertices and m is the number of edges of the network.

a) Describe the algorithm in plain English.

b) Prove its correctness.

c) Establish its time complexity.

[1] Unfortunately, none of this is true, but it would be cool if it was.

Problem 2. (80 points) **(COMP3027 only)**

Time to explore the capabilities of our network. We first want to determine whether there are any pipes that are crucial to the flow of ale. We call a pipe *expandable* if increasing its capacity by 1 increases the value of the maximum flow by 1 as well. Analogously, we call a pipe *limiting* if reducing its capacity by 1 decreases the value of the maximum flow by 1 as well. We start by figuring out whether every network actually has these types of pipes.

a) Does every network have at least one expandable pipe? Prove your answer is correct, i.e., prove that every network has such an edge, or provide a network that doesn't and explain why.

b) Does every network have at least one limiting pipe? Prove your answer is correct, i.e., prove that every network has such an edge, or provide a network that doesn't and explain why.

Next, it's time to design an algorithm to actually find the expandable pipes and make our chief happy. Design an algorithm that given a directed graph G and two vertices s (the brewery) and t (the chief's house) along with capacities of all the pipes in our network, as well as a maximum flow through these pipes, returns all expandable pipes.

Your task is to design the above algorithm. Your algorithm should run in $O(m)$ time, where m is the number of edges of the network.

c) Describe the algorithm in plain English.

d) Prove its correctness.

e) Establish its time complexity.

Expanding these pipes doesn't seem what the chief had in mind. He wanted a complete overhaul of the entire network to maximize the throughput to his house by redistributing the capacity of the pipes in the network (you can move capacity from any pipe to any other pipe, no restrictions as long as the capacities stay integer). Your goal in doing so is to see if there's any way at all to facilitate the chief's demand for a total throughput (maximum flow) of at least k.

Your task is to design an algorithm that given a directed graph G and two vertices s (the brewery) and t (the chief's house) along with capacities of all the pipes in our network, as well as a maximum flow through these pipes, returns whether the capacities can be redistributed in such a way to allow for a maximum flow of k. Your algorithm should run in $O(m)$ time (i.e., not depend on k and k is not a constant), where m is the number of edges of the network.

f) Describe the algorithm in plain English.

g) Prove its correctness.

h) Establish its time complexity.

Problem 3. (80 points) **(COMP3927 only)**

Time to explore the capabilities of our network. We first want to determine whether there are any pipes that are crucial to the flow of ale. We call a pipe *expandable* if increasing its capacity by 1 increases the value of the maximum flow by 1 as well. Analogously, we call a pipe *limiting* if reducing its capacity by 1 decreases the value of the maximum flow by 1 as well. We start by figuring out whether every network actually has these types of pipes.

a) Does every network have at least one expandable pipe? Prove your answer is correct, i.e., prove that every network has such an edge, or provide a network that doesn't and explain why.

b) Does every network have at least one limiting pipe? Prove your answer is correct, i.e., prove that every network has such an edge, or provide a network that doesn't and explain why.

Next, it's time to design an algorithm to actually find the expandable edges and make our chief happy. Design an algorithm that given a directed graph G and two vertices s (the brewery) and t (the chief's house) along with capacities of all the pipes in our network, as well as a maximum flow through these pipes, returns all expandable pipes.

Your task is to design the above algorithm. Your algorithm should run in $O(m)$ time, where m is the number of edges of the network.

c) Describe the algorithm in plain English.

d) Prove its correctness.

e) Establish its time complexity.

A rebellion is brewing[2], as the other Vikings are rather unhappy about their lack of ale. The rebels have also come to you with a request: design an algorithm to find all limiting pipes in the network, so they can leverage this when going to the chief.

Your task is to design an algorithm that given a directed graph G and two vertices s (the brewery) and t (the chief's house) along with capacities of all the pipes in our network, as well as a maximum flow through these pipes, returns all limiting pipes. Your algorithm should run in $O(n \cdot m)$ time, where n is the number of vertices and m is the number of edges of the network.

f) Describe the algorithm in plain English.

g) Prove its correctness.

h) Establish its time complexity.

Epilogue (or more flavour text). Hearing about the chief's demands on the network and being rather unhappy to no longer be getting their ale, your fellow Vikings decide to mount a peaceful revolt, giving the chief the choice of giving them access to the ale again, or being put on the first ship out of the village. The chief sees reason and reverts his decision to have sole access to the ale. Everyone's taps are reactivated and a celebratory feast is organised. Good thing you hadn't started modifying the network yet, but better keep an eye on that chief in the future...

[2]Pun fully intended.

Advice on how to do the assignment

- Assignments should be typed and submitted as pdf (no pdf containing text as images, no handwriting).
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- When asked to give a plain English description, describe your algorithm as you would to a friend over the phone, such that you completely and unambiguously describe your algorithm, including all the important (i.e., non-trivial) details. It often helps to give a very short (1-2 sentence) description of the overall idea, then to describe each step in detail. At the end you can also include pseudocode, but this is optional.
- In particular, when designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you marks for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the slides or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times, etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources and explain it in your own words. Only citing a source doesn't show your understanding and will thus get you very few (if any) marks. Copying from any source without reference is considered plagiarism.