# CSCC73 - Algorithm Design and Analysis
# Assignment #1: Greedy Algorithms

Due: Jan 21, 2023 at 11.59pm This exercise is worth 5% of your final grade.

**Warning:** Your electronic submission affirms that this exercise is your and your partners own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCC73. Please see syllabus for details. Late assignments will not be accepted. If you are working with a partner your partners' name must be listed on your assignment and you must sign up as a "group" on Gradescope. Recall you must not consult **any outside sources except your partner, textbook, TAs and instructor**.

1. In this question we consider a variation of the scheduling problem in which we are given $n$ distinct jobs $j_i$ each of which can be completed independently. Each job has two stages to be completed. The first stage of each job is the bottleneck in the sense that there is only one resource that can handle it and for job $j_i$ takes $f_i$ time. The second stage is less complicated and can be completed in parallel with other jobs as there are many resources available to complete this portion of the job. We will denote the time required to complete stage 2 of job $j_i$ as $s_i$. [10]

   For example, consider a factory where there is a unique machine that must be used in the production of each product before it can go through the final stages with an employee. Assume that there are enough employees (i.e., $n$ employees) to handle the second stage of all the jobs at the same time. Give a polynomial-time algorithm to return a *schedule* that orders the jobs for stage 1 so that the *completion time* of stage 2 for all the jobs is minimized (ie, the total time for all jobs to be completed is minimized). Prove your algorithm is correct and explain it's complexity.

2. Timing circuits are an integral component of VLSI chips. Here's a simple model of such a timing circuit. Consider a complete balanced binary tree with $n$ leaves, where $n$ is a power of 2. Each edge $e$ of the tree has an associated length $\ell_e$, which is a positive number. The *distance* from the root to a given leaf is the sum of the lengths of all the edges on the path from the root to the leaf. [10]

   The root generates a *clock signal* which is propagated along the edges to the leaves. We'll assume that the time it takes for the signal to reach a given leaf is proportional to the distance from the root to the leaf.

   Now, if all leaves do not have the same distance from the root, then the signal will not reach the leaves at the same time, and this is a big problem. We want the leaves to be completely synchronized, and all to received the signal at the same time. To make this happen, we will have to *increase* the lengths of certain edges, so that all root-to-leaf paths have the same length (we're not able to shrink edge lengths). If we achieve this, then the tree (with its new edge lengths) will be said to have *zero skew*. Our goal is to achieve zero skew in a way that keeps the sum of all the edge lengths as small as possible.

   Give an algorithm that increases the lengths of certain edges so that the resulting tree has zero skew and the total edge length is as small as possible.

   **Example.** Consider the tree in Figure 1, in which letters name the nodes and numbers indicate the edge lengths. The unique optimal solution for this instance would be to take the three length 1 edges and increase each of their lengths to 2. The resulting tree has zero skew, and the total edge length is 12, the smallest possible.

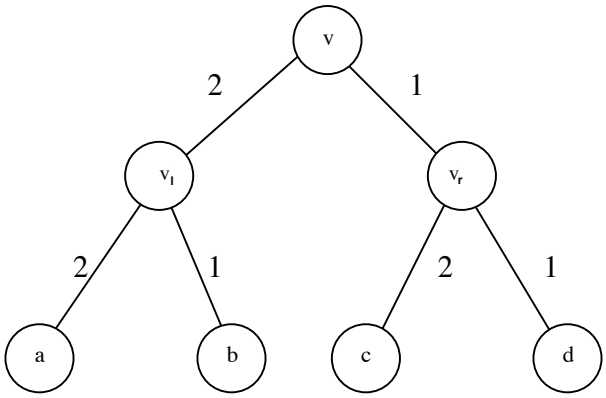


Figure 1: An instance of the zero-skew problem.