# Project 2: Implementing Algorithm

Fall 2024  CPSC 335 - Algorithm Engineering

## Abstract

Develop a pseudocode for an algorithm; analyze your pseudocode mathematically; implement the code for the algorithm of your choice; test your implementation; and describe your results.

## Problem 1:  Top k Most frequent elements

Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.

**Example 1:**

Input: nums = [1, 1, 1, 2, 2, 3], k = 2
Output: [1, 2]

**Example 2:**

Input: nums = [1], k = 1
Output: [1]

**Constraints:**

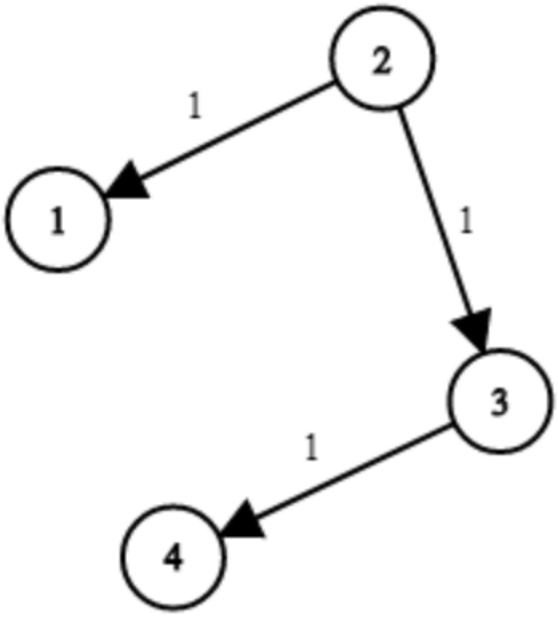
- 1 <= nums.length <= $10^5$
- $-10^4$ <= nums[i] <= $10^4$
- k is in the range [1, the number of unique elements in the array].
- It is **guaranteed** that the answer is **unique**.

Your code should work for any input array. Try to solve it using priority queue as it gives better time complexity in problems like this. But feel free to use any approach which is easy for you.

## Problem 2: Network delay time

You are given a network of n nodes, labeled from 1 to n. You are also given times, a list of travel times as directed edges times[i] = ($u_i$, $v_i$, $w_i$), where $u_i$ is the source node, $v_i$ is the target node, and $w_i$ is the time it takes for a signal to travel from source to target.

We will send a signal from a given node k. Return *the minimum time it takes for all the* n *nodes to receive the signal*. If it is impossible for all the n nodes to receive the signal, return -1.




**Example 1(Above graph):**
**Input: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2**
**Output: 2**

**Example 2:**
**Input: times = [[1,2,1]], n = 2, k = 1**
**Output: 1**

**Example 3:**
**Input: times = [[1,2,1]], n = 2, k = 2**
**Output: -1**

## To Do

1. Produce a written project report ***in PDF format***. Your report should include:
    a. Your name(s), CSUF-supplied email address(es), and an indication that the submission is for project 2.
2. Develop the pseudocode for Problem 1 and 2, and implement the **two** algorithms in Python or C++ or Java or C.
3. Your codes should be saved and submitted in the corresponding executable file.
4. Mathematically analyze each algorithm and state the big *O* efficiency class.

## Grading Rubric

The suggested grading rubric is given below:

Algorithm (55 points each)

a. Clear and complete Pseudocode = 10 points
b. Mathematical analysis and correct Big O efficiency class = 5 points
c. Inclusion of a Readme file = 5 points
d. Well commented codes = 5 points
e. Successful compilation of codes= 20 points
f. Produces accurate results = 10 points

Algorithm 2 (45 points)

a. Clear and complete Pseudocode = 10 points
b. Mathematical analysis and correct Big O efficiency class = 5 points
c. Inclusion of a Readme file = 5 points
d. Well commented codes = 5 points
e. Successful compilation of codes= 15 points
f. Produces accurate results = 5 points

## Submitting your code

Submit your files to the Project 2 Assignment on Canvas. It allows for multiple submissions. You may submit your files **separately**. You can also create a zip file and submit.

Ensure your submissions are your own works. Be advised that your submissions may be checked for plagiarism using automated tools. Do not plagiarize. As stated in the syllabus, a submission that involves academic dishonesty will receive a 0% score on that assignment. A repeat offense will result in an "F" in the class and the incident will be reported to the Office of Student Conduct.

## Deadline

The project deadline is **Thursday, Nov 21, by 11:59 pm** on Canvas.

Penalty for late submission (within 48 hours) is as stated in the syllabus. Projects submitted more than 48 hours after the deadline will not be accepted.