# CPSC 535 Advanced Algorithms
# Project 2: Cumulative frequencies

Prof. Doina Bein, CSU Fullerton

dbein@fullerton.edu

## Introduction

In this project you will design and implement one algorithm related to strings. You will design the algorithm, describe the algorithm using clear pseudocode, and implement it using C/C++/C#/Java/Python, compile, test it, and submit BOTH the report (as a PDF file) and the files. The execution should take less than one hour **for each input example**.

## Calculating frequency of similar words

Given a text S of length *n*, one can easily calculate the frequency of each word in S using a linear time O(n) algorithm. But words can be similar, for example "foot" and "feet", "day" and "days", "fear"and "scared", "long" and "big", "big" and "large". So one can extract the *cumulative frequency* of similar words instead of individual words. This will be extremely useful for sentiment analysis of reviews (movies, products, services,etc.) and text summarization (blogs, text files, web articles, etc.), because it will output the essence of the entire text in a much smaller set of data. (This problem is similar but not the same as Exercise 17.7 "Baby Names" in "Cracking the Code Interview" book.)

Given two lists, one of words and their frequencies and the other of pairs of similar words, write an algorithm to print a new list of cumulative frequency of each set of similar words. Note that if "long" and "big" are similar, and "big" and "large" are similar, then "long" and "large" are similar. (It is both transitive and symmetric.) In the final list, any word can be used as representative of its set but for testing purposes, choosing the words that are the earliest in the alphabet is recommended.

Example 1:

**Input:**

Words_Frequencies: WF[] = { ("foot", 5), ("feet", 12), ("day", 3), ("days", 8), ("fear", 2), ("scared", 1), ("long", 12), ("large", 5), ("big",5), ("was", 4), ("is", 4), ("are", 15)} of size 12

Synonyms: SYN[] = { ("foot", "feet"), ("day","days"), ("fear", "scared"), ("long" ,"big"), ("big" , "large"), ("is", "are"), ("is", "was") } of size 7

**Output:** CF[] = { ("**feet**", 17), ("day", 11), ("fear",3), ("big", 22), ("are", 23) } of size 5

~~More examples will be added.~~

**Example 2:**

**Input:**

**Words_Frequencies: WF[] = { (“tons of”, 2), (“large number of ”, 12), (“mystical”, 13), (“magical”, 28), (“magic”, 5), (“unexplained”, 11), (“huge”, 2), (“large”, 51), (“horses”, 25), (“horse, 24), (“large mammal”, 24), (“herbivore”, 5)} of size 12**

**Synonyms: SYN[] = { (“herbivore”, “horses”), (“horse”,“large mammal”), (“horses”, “large mammal”), (“large number of” ,“huge”), (“tons of” , “large”), (“huge”, “large”), (“mystical”, “magical”) , (“magical”,”unexplained”), (“magic”, “magical”)} of size 9**

**Output: CF[] = { (“herbivore”, 78), (“huge”, 67), (“magic”,57)} of size 3**

**Example 3:**

**Words_Frequencies: WF[] = { (“tons of”, 2), (“large number of ”, 12), (“mystical”, 13), (“magical”, 28), (“magic”, 5), (“unexplained”, 11), (“huge”, 2), (“large”, 51), (“horses”, 25), (“horse, 24), (“large mammal”, 24), (“herbivore”, 5), (“large number of”,12)} of size 13**

**Synonyms: SYN[] = { (“herbivore”, “horses”), (“horse”,“large mammal”), (“horses”, “large mammal”), (“large number of” ,“huge”), (“tons of” , “large”), (“huge”, “large”), (“mystical”, “magical”) , (“magical”,”unexplained”), (“magic”, “magical”), (“horse”,”large mammal”)} of size 10**

**Output: CF[] = { (“herbivore”, 78), (“huge”, 67), (“magic”,57)} of size 3**

The variables WF[] and SYN[] must be read from the user, most preferred from a file.

The output should display the resulting variable CF[]. Please check that your program executes correctly on the given examples. Please create more examples on your own and test them.

## Grading rubric

The total grade is 35 points. Your grade will be comprised of three parts, Form, Function, and Report:

- Function refers to whether your code works properly (18 points).
- Form refers to the design, organization, and presentation of your code. The instructor will read your code and evaluate these aspects of your submission (6 points):
    - README.md completed clearly = 2 points
    - Style (whitespace, variable names, comments, helper functions, etc.) = 2 points
    - Craftsmanship (appropriate handling of encapsulation, memory management, avoids gross inefficiency and taboo coding practices, etc.) = 2 points
- Report (11 points) divided as follows:
    - Summary of report document (2 points)
    - Pseudocode of the algorithm (6 points)
    - Four screenshots: one for the group members and three for the three sample input **examples** ~~files~~ (1 point each, total 3 points)

## Obtaining and Submitting Code

This document explains how to obtain and submit your work:

GitHub Education / Tuffix Instructions

Here is the invitation link for this project:
https://classroom.github.com/g/el1VJCYB

## Implementation

You are provided with the following files.

1. README.md contains a brief description of the project, and a place to write the names and CSUF email addresses of the group members. You need to modify this file to identify your group members.
2. LICENSE contains a description of the MIT license.

## What to Do

First, add your group members' names to README.md. Then write clear pseudocode for the rewriting algorithm, describe what parameters are needed to execute your program, and submit it as a PDF report.Your report should include the following:

1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 1.
2. A full-screen screenshot with your group member names shown clearly. One way to make your names appear in Atom is to simply open your README.md.
3. The pseudocode for the algorithm
4. A brief description on how to run the code.
5. Three snapshots of code executing for the three given examples ~~(right now only one example, but two more examples will be added)~~

Then implement your algorithm in C/C++/C#/Java/Python. Submit your PDF by committing it to your GitHub repository along with your code.