COMP 221-01 Homework 2

Your Name

Due: Friday, February 11, by 11:59 pm

1 Guidelines

Please **type up** and submit a PDF of your solutions. I recommend LaTex (Overleaf is really nice to use online), but you are certainly welcome to use Google Docs or some other word processing program. Just make sure to save it as a PDF before submission.

Programming language: You are welcome to use Python or Java.

Do your own work: Homework should be individual work. You may discuss problems with other people, but you must write up your solutions yourself. I will not say that the web is off limits, but handing in solutions you find online as if they are your own is also not acceptable.

Make sure you have your name on your homework!

- **Question 1**

  For the following questions, find positive constants c and n_0 such that the relationship is true.

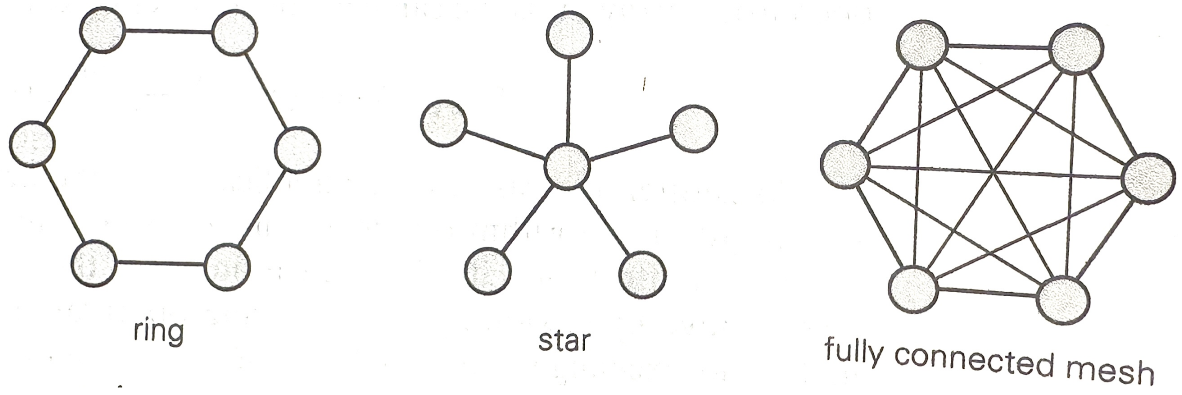
  (a) $f(n) = n^2 + 3n + 2, f(n) = O(n^2)$

  (b) $f(n) = 4n^3 + n^2 + nlogn + 5, f(n) = \Theta(n^3)$

  (c) $f(n) = n^2 - 8n + 1, f(n) = \Omega(n)$

- **Question 2**

  From Levitin: A network topology specifies how computers, printers, and other devices are connected over a network. The figure below illustrates three common topologies of networks: the ring, the star, and the fully connected mesh:



  You are given a boolean matrix $A[0..n-1, 0...n-1]$, where $n > 3$, which is supposed to be the adjacency matrix of a graph modeling a network with one of these topologies. Your task is to determine which of these three topologies, if any, the matrix represents. Design a brute-force algorithm for this task and indicate it's big-Θ run-time.

- **Question 3 - Programming**

  You should develop a program that analyzes the performance of add and delete operations on lists. I would like you to compare the performance of adding and removing to the front, back, and middle of a list, by filling in the table below.

  You will need to submit your code, but the preceptors will not be grading it. They will only be grading you based on your table (unless there is reason for them to look at your code).

n	1	10	100	1k	10k	100k	1M	10M	100M
add to front of list									
add to middle of list									
add to end of list									
del from front of list									
del from middle of list									
del from end of list									

  For each cell of the table, you should perform the following steps:

  Repeat this process 10 times, and take the overall average time:

  – If you are timing deletes, fill the list up to the specified size (you are free to use random numbers, the same number, etc).
  – Start the timer (you can look up the appropriate timer class for your language)
  – Perform the requested operation n times. If the operation is an add, your data structure will grow from size 0 to size n. If your operation is a delete, your data structure will shrink from size n to size 0.
  – Stop the timer.
  – Calculate the time per operation by dividing the elapsed time by n.

  When you are done filling out the table, write a few brief statements analyzing what you see. Are the values what you expected? Did anything surprise you?

  Notes:

  – **Automate things.** This process may look laborious, but it should not be! You can (and should) do all of it within a single program. Try to automate things so that you don't to run your program for each cell over and over by hand.
  – **Running out of space / time with large N.** Note that due to memory and CPU constraints, you may not be able to complete the full table. You might run out of memory or your program may take too long. That's okay. Just write ”too big” in the table!
  – **Timing functions.** In Java you can get the current time in milliseconds (one-thousandth of a second) by `System.currentTimeMillis()`. In Python you can get the time in seconds using `time.time()`.
  – **Java ArrayList:** If using Java, you should use `ArrayList` rather than LinkedList.