CS 630, Fall 2024, Homework 3
Due Wednesday, October 2, 2024, 11:59 pm EST, via Gradescope

Homework Guidelines

Collaboration policy Collaboration on homework problems, with the exception of programming assignments and reading quizzes, is permitted, but not encouraged. If you choose to collaborate on some problems, you are allowed to discuss each problem with at most 5 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (including generative AI tools or anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Typesetting Solutions should be typed and submitted as a PDF file on Gradescope. You may use any program you like to type your solutions. LaTeX, or "Latex", is commonly used for technical writing (`overleaf.com` is a free web-based platform for writing in Latex) since it handles math very well. Word, Google Docs, Markdown or other software are also fine.

Solution guidelines For problems that require you to provide an algorithm, you must provide:

1. pseudocode and, if helpful, a precise description of the algorithm in English. As always, pseudocode should include
   - A clear description of the inputs and outputs
   - Any assumptions you are making about the input (format, for example)
   - Instructions that are clear enough that a classmate who hasn't thought about the problem yet would understand how to turn them into working code. Inputs and outputs of any subroutines should be clear, data structures should be explained, etc.

   If the algorithm is not clear enough for graders to understand easily, it may not be graded.
2. a proof of correctness
3. an analysis of running time and space

You may use algorithms from class as subroutines. You may also use facts that we proved in class.

You should be as clear and concise as possible in your write-up of solutions. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand.

Problem 1 *Polling (10 points)*

As the elections are coming near new polls are published every day. As part of a research group at BU you are conducting experiments on how belonging to a social group influences ones' vote. For this you will select some known groups and survey every one of its members. To get unbiased results you decide that none of the groups you select can share any members. From the Registrar at BU you get a list of all sanctioned groups within BU along with their member lists. Design an approximation algorithm to select the largest number of groups.

1. Design an approximation algorithm for this problem. (*For proof you have to show that the output of the algorithm indeed yields a valid solution for the problem. Don't forget to analyze the running time.*)
2. Clearly state and prove the approximation ratio of your algorithm. (*Any approximation ratio correlated with one of the relevant algorithms we learned in class will get full credit.*)

Problem 2 *Catering (10 points)*

You are a caterer and you've been asked to pick meal choices. You have a list of entrees and drinks (fancy steak, expensive wine, etc) and a list of valid entree+drink pairings from your sommelier. You also have a list of guest preferences for entrees and drinks (list of acceptable choices), but the preferences are just for entrees and just for drinks, not for the pairs.

(For some reason you don't care about the price of items, nor potential waste of food or drinks...)

1. Design an approximation algorithm to pick a minimal set of pairs of entrees and drinks, such that each pair is valid, and each guest is happy to eat at least one of the pairs. (*Prove that your algorithm indeed yields a valid solution, but you can omit the running time analysis.*)
2. Clearly state and prove the approximation ratio of your algorithm.
3. Now suppose that as a caterer you still have to deliver entrees+drinks in pairs. However, guests can mix-and-match; they may take an entree from one pair with a drink from another pair as long as both are on their preference lists. Design an approximation algorithm to cater this scenario with the fewest possible number of pairs selected. (*Prove that your algorithm indeed yields a valid solution, but you can omit the running time analysis.*)
4. Clearly state and prove the approximation ratio of your second algorithm.