Homework 4
Due: 11am September 28, 2016
Professor Lila Fontes

Directions:

• This homework is due 11AM Wednesday September 28. Write your solution using Latex. Submit this homework using as file hw4.tex using github.

• Try to make your answers as clear and concise as possible. When you describe an algorithm (in words or pseudocode), make sure to also include a clear, informal, high-level description of your algorithm in English.

• For this homework, you will work with a partner. This week I have assigned you partners. It’s ok to discuss approaches with others at a high level, but most of your discussions should just be with your homework partner. The only exception to this rule is work you’ve done with a lab partner while in lab. In this case, note who you’ve worked with and what parts were solved during lab.

If there are questions about academic integrity, please visit the section on Academic Integrity on the course website.

0. Before final submission, make sure to fill out the README file.

1. Analysis. Let \( f(n) = 4n^{5/3} \) and \( g(n) = n^{5/4}(\log n)^7 \). Prove that \( g(n) = O(f(n)) \). You may use techniques and facts from class and the textbook; your proof should be formal and complete.

2. All-Pairs Shortest Paths. Design and analyze a polynomial-time algorithm that takes a directed graph \( G = (V, E) \) and for all \( u, v \in V \) computes the length of the shortest \( u \leadsto v \) path or determines that no such path exists.

3. Enemies on the Move. Alice and Bob are very active students at University of Pennsylvania. They used to be best friends but now despise each other. Alice and Bob can’t stand to be in the same room, or even nearby. However, they each take many classes and are active in several clubs. Is it even possible to avoid each other?

This can be modeled as a graph problem. The input consists of:

• a directed graph \( G = (V, E) \),

• an integer \( k \leq n \),

• start vertices \( s_A \) and \( s_B \in V \), and

• end vertices \( t_A \) and \( t_B \in V \).

In this problem, Alice starts at \( s_A \) and wants to travel to \( t_A \), while Bob starts at \( s_B \) and wants to travel to \( t_B \). At each time step, either Alice or Bob moves along a single edge. (You can assume they move separately.) At all times, Alice and Bob must be at least \( k \) edges apart.
Design and analyze a polynomial-time algorithm that determines if Alice and Bob can get where they want to go while maintaining distance.

(Hint: It might be helpful to use your solution to problem (2) as a subroutine.)

4. Ethnographers. (Kleinberg and Tardos, 3.12) You’re helping a group of ethnographers analyze some oral history data they’ve collected by interviewing members of a village to learn about the lives of people who have lived there over the past two hundred years.

From these interviews, they’ve learned about a set of \( n \) people (all now deceased), whom we’ll denote \( P_1, P_2, \ldots, P_n \). They’ve also collected facts about when these people lived relative to one another. Each fact has one of the following two forms:

- for some \( i \) and \( j \), person \( P_i \) died before person \( P_j \) was born; or
- for some \( i \) and \( j \), the lifespans of \( P_i \) and \( P_j \) overlapped at least partially.

Naturally, the ethnographers are not sure that all these facts are correct; memories are not very good, and a lot of this was passed down by word of mouth. So what they’d like you to determine is whether the data they’ve learned simultaneously hold.

Give an efficient algorithm to do this: either it should propose dates of birth and death for each of the \( n \) people so that all the facts hold true, or it should report (correctly) that no such dates can exist—that is, the facts collected by the ethnographers are not internally consistent.

5. Extra credit. In class we saw an algorithm for testing bipartiteness which used BFS to color the vertices. It should be possible to use DFS to test bipartiteness to color the vertices. Give an algorithm (in pseudocode) which uses DFS to test bipartiteness. Rigorously prove that your algorithm works.