Foundations of Algorithms, Fall, 2025
Homework 5 FA25

Certain homework problems marked ***Collaborative Problem*** are to be worked by all members of the group, with each member *participating fully* and substantially contributing to the solution concept. **Each student must do the final write-up of his/her answers independently; do not copy either text or images from another group member into your own homework submission.** Individual participation will be evaluated via assessments submitted by each group member when the homework is due. Remember that all homework must include a personal statement of academic integrity, as noted in the Getting Started module.

1. [30 points total] Suppose we have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u and obtain a tree isomorphic to T. Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.)

2. [30 points total] Suppose you and your friend Alanis live together with $n - 2$ other people at a popular "co-operative" apartment. Over the next n nights, each of you is supposed to cook dinner for the co-op exactly once, so some one cooks on each of the nights. To make things interesting, everyone has scheduling conflicts with some of the nights (e.g., exams, deadlines at work, basketball games, etc.), so deciding who should cook on which night becomes a tricky task. For concreteness, let's label the people $\{p_1, ..., p_n\}$ and the nights $\{d_1, ..., d_n\}$. Then for person p_i, associate a set of nights $S_i \subset \{d_1, ..., d_n\}$ when there are *not* available to cook. A *feasible dinner schedule* is defined to be an assignment of each person in the co-op to a different night such that each person cooks on exactly one night, there is there is someone to cook on each night, and if p_i cooks on night d_j, then $d_j \notin S_i$.

   (a) [15 points] Describe a bipartite graph G such that G has a perfect matching if and only if there is a feasible dinner schedule for the co-op.

   (b) [15 points] Your friend Alanis takes on the task of trying to construct a feasible dinner schedule. After great effort, she constructs what she claims is a feasible schedule and then heads off to work for the day. Unfortunately, when you look at the schedule she created, you notice a big problem– $n - 2$ of the people at the co-op are assigned to different nights on which they are available (no problem there), but for the other two people p_i and p_j, and the other two days d_k and d_l, you discover she has accidentally assigned both p_i and p_j to cook on night d_k and no one to cook on night d_l. You want to fix this schedule but without having to recompute everything from scratch. Show that it is possible, using her "almost correct" schedule, to decide in only $O(n^2)$ time whether there exists a feasible dinner schedule for the co-op. If one exists, your algorithm should also provide that schedule.

3. [40 points total] ***Collaborative Problem:*** We define the *Escape Problem* as follows. We are given a directed graph $G = (V, E)$ (picture a network of roads.) A certain collection of vertices $X \subset V$ are designated as *populated vertices*, and a certain other collection $S \subset V$ are designated as *safe vertices*. (Assume that X and S are disjoint.) In case of an emergency, we want evacuation routes from the populated vertices to the safe vertices. A set of evacuation routes is defined as a set of paths in G such that (i) each vertex in X is the tail of one path, (ii) the last vertex on each path lies in S, and (iii) the paths do not share any edges. Such a set of paths gives a way for the occupants of the populated vertices to "escape" to S without overly congesting any edge in G.

   (a) [20 points] Given G,X, and S, show how to decide in polynomial time whether a set of evacuation routes exists.

   (b) [20 points] Suppose we have exactly the same problem as in (a), but we want to enforce an even stronger version of the "no congestion" condition (iii). Thus, we change (iii) to say, "the paths do not share any *vertices*." With this new condition, show how to decide in polynomial time whether such a set of evacuation routes exists. Also provide an example with the same G, X, and S in which the answer is "yes" to the question in (a) but "no" to the question in (b).