EE360C: Algorithms
The University of Texas at Austin — Exam #3
Dr. Christine Julien — May 10, 2014

Name: **EID:**

Exam #3

Instructions. WRITE LEGIBLY.
No calculators, laptops, or other devices are allowed. This exam is **closed book**, but you are allowed to use a **one-page** reference sheet. Write your answers on the test pages. If you need scratch paper, use the back of the test pages, but indicate where your answers are. Write down your process for solving questions and intermediate answers that **may** earn you partial credit.

If you are unsure of the meaning of a specific test question, write down your assumptions and proceed to answer the question on that basis. **Questions about the meaning of an exam question will not be answered during the test.**

When asked to describe an algorithm, you may describe it in English or in pseudocode. If you choose the latter, make sure the pseudocode is understandable.
You have **180 minutes** to complete the exam. The maximum possible score is 100.

Imagine...

You've graduated from a top US computer engineering program and are ready to embark on an esteemed career in computing. You've been hired by a software startup called ShutterTalk, which specializes in sharing ephemeral photos between groups of friends via smartphones. This is no run of the mill startup. They've got plans. And you're here to implement them.

Problem 1: Loop Invariants [15 points]

On your first day on the job, you catch yourself reflecting on how you aced the interview. In your algorithms class, you'd learned all about specifying loop invariants and were able to impress the technical interviewers with your ability to solve a question about loop invariants posed in the context of their software system. Specifically, they gave you the following problem:

ShutterTalk had recently transitioned to a new back-end database system. In ShutterTalk, each user has a single cover photo, but, before the transition, each cover photo contained a reference to the associated user id, but for some bizarre reasons, each user object was not also associated with the cover photo. ShutterTalk has n users; in the original database, these users' ids were stored in a simple flat list: $\{u_1, u_2, \ldots u_n\}$. The original ShutterTalk back-end stored the n cover photos separately, where each cover photo contained a reference to its user, i.e., as $p_j.user$. In the revised database, ShutterTalk wanted there to be just one data structure $\{user_1, user_2, \ldots, user_n\}$ where each user object contains a reference to the cover photo, i.e., as $user_i.photo$. The developers at ShutterTalk wrote the following code to accomplish this (the code is correct):

```
1  create {user_1, user_2, user_n}
2  for i ← 1 to n
3        do for j ← 1 to n
4              do if u_i equals p_j.user
5                    then user_i.id = u_i and user_i.photo = p_j
6  return {user_1, user_2, ..., user_n}
```

Recreate your responses from your interview that impressed the ShutterTalk interviewers so much.

(a) **[6 points]** Write a loop invariant *for the outer loop* of this algorithm.

(b) **[9 points]** Prove that your loop invariant (*i*) is true initially, (*ii*) is true after each iteration of the loop, and (*i*) tells you something about the correctness of the program when it completes.

Problem 2: Kitchen Patrol [20 points]

ShutterTalk's office space has a shared kitchen facility. Unfortunately, it's early days, and there is not yet sufficient funding to hire staff responsible for keeping the kitchen cleaned up. Therefore, the first task for the newbie (that's you!) is to design an algorithm to assign each of the n members of ShutterTalk's technical staff into a kitchen patrol rotation. Specifically, your job is to create an n day rotation for the days $\{d_1, d_2, \ldots, d_n\}$ in which each staff member is responsible for kitchen patrol exactly once. There's a wrinkle, of course; a given staff member is not available every day, either because of vacation or because of important technical deadlines. That is, for each member of the staff, $\{s_1, s_2, \ldots, s_n\}$, there is a subset of the days when the staff member is *not* available for kitchen patrol. You have these "vacation" days accessible to you before starting your algorithm.

A *feasible schedule* is an assignment of each member of the staff to perform kitchen patrol on one of the n days such that the staff member is available on the scheduled day. Describe a polynomial time algorithm that determines whether such a feasible schedule is possible and, if it is, outputs the schedule. Give the running time of your algorithm.

Problem 3: Saving Photos [25 points]

You've managed to prove yourself through the interview process and your first in-office assignment; the ShutterTalk development team is ready to trust you with a piece of their application's implementation. The goal is to define an intelligent, automated way to store a small history of the user's photos. ShutterTalk and its users assign some values to the photos that help in designing an automated algorithm. First, assume each photo has a size b_i in bytes. ShutterTalk has promised users that the application will not use more than a fixed M amount of bytes for storing photo data. Each photo is also give a *value* (v_i) that is defined as a function of the *quality* (q_i) of the photo, it's *age* (a_i), and it's estimated *social impact* (s_i) (e.g., based on the number of captured faces in the photo or the closeness of the identified friends in the photo). That is, each photo has a value $v_i = value(q_i, a_i, s_i)$. Your goal is to compute the optimal set of photos to keep in ShutterTalk's allocated storage. Specifically, you should keep a subset P of all available photos such that the total size of the photos in P is less than M and the total value of the photos in P is maximized. Derive the running time of your algorithm. Is the algorithm's running time polynomial in the size of the input?

Problem 4: NP or not NP? [40 points]

Your next task for ShutterTalk is more sophisticated. You've been asked to lead the development of a completely new feature: synchronous photo broadcast, in which a single photo is simultaneously delivered to a pre-specified set of receivers.

For simplicity, assume a particular ShutterTalk user can be in only one of two states at any time: each user is either a *sender* or a *receiver*. At any time, there are n senders and m receivers. When a sender wants to perform a synchronous broadcast, he selects a subset of the receivers who should all simultaneously receive the photo. At the instant the photo is delivered, all of the designated receivers should be "assigned" to the given sender such that none of them are receiving any other photos and are instead all guaranteed to be focused on this single, shared experience.

Given that there may be many requests coming at the same time, and the sets of targeted receivers may be overlapping, we phrase the general *Synchronous Sender Broadcast Scheduling* problem thusly: Given sets of senders and receivers, the set of requested receivers for each sender, and a number k, is it possible to assign receivers to senders so that (1) each receiver is assigned to not more than one sender, (2) at least k senders will be active, and (3) every active sender is assigned all of its requested receivers.

For each of the following, either give a polynomial time algorithm or prove that the problem is NP-complete by starting from the *Independent Set* problem. If you give a polynomial time algorithm, you should also give the running time of your algorithm.

The Independent Set Problem. *Given a graph G and a number k, does G contain an independent set of at least k? In a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is* independent *if no two nodes in S are joined by an edge.*

(a) **[10 points]** The general *Synchronous Sender Broadcast Scheduling* problem described above.

(b) **[10 points]** The special case when $k = 2$, i.e., when we want to enable exactly 2 senders.

(c) **[10 points]** The special case when there are two different types of receivers, say men and women, and each sender will broadcast to at most one receiver of each type.

(d) **[10 points]** The special case when each receiver will be targeted by at most two of the senders.

Scratch Page