EE360C: Algorithms
The University of Texas at Austin — Final Exam Practice Questions
Dr. Pedro Santacruz & Dr. Evdokia Nikolova — Dec. 8, 2014

Name: **EID:**

Final Exam Practice Questions

Instructions. No calculators, laptops, or other devices are allowed. This exam is **closed book**, but you are allowed to use a **two, double-sided** cheat sheets. You must submit your cheat sheet with the exam. 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.**

You have **180 minutes** to complete the exam. The maximum possible score is 100.

Some useful information:

Logarithms and Factorial:

$$\log(n!) = \Theta(n \log n)$$

Arithmetic Series:

$$\sum_{k=1}^{n} k = \frac{1}{2}n(n+1)$$

Sum of Squares:

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Sum of Cubes:

$$\sum_{k=0}^{n} k^3 = \frac{n^2(n+1)^2}{4}$$

Geometric Series:

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$$

Infinite Geometric Series:

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

Practice Problem 1: Saving Photos

You work for a startup company, ShutterTalk, and its 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 $v_i = \mathit{value}(q_i, a_i, s_i)$.

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?

Practice Problem 2: NP or not NP?

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) The general *Synchronous Sender Broadcast Scheduling* problem described above.

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

(c) 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.

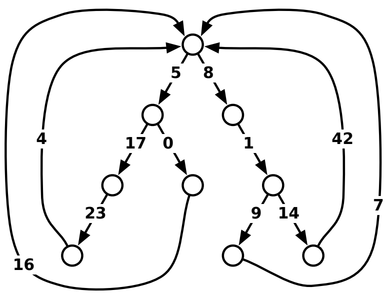
(d) The special case when each receiver will be targeted by at most two of the senders.

Practice Problem 3: Single Source Shortest Paths

Recall Dijkstra's algorithm:

```
DIJKSTRA$(G, w, s)$
1  INITIALIZE-SINGLE-SOURCE$(G, s)$
2  $S \leftarrow \emptyset$
3  $Q \leftarrow G.V$
4  while $Q \neq \emptyset$
5      do $u \leftarrow$ EXTRACT-MIN$(Q)$
6         $S \leftarrow S \cup \{u\}$
7         for each vertex $v \in \text{Adj}[u]$
8             do RELAX$(u, v, w)$
```

A *looped tree* is a weighted directed graph built from a binary tree by adding an edge from every leaf back to the root. Every edge has a non-negative weight. An example of a looped tree is shown below.



A looped tree.

(a) How much time would Dijkstra's algorithm require to compute the shortest path from u to v in a looped tree with n nodes? (Do NOT assume that either u or v is the root of the tree, though one could be.)

(b) Describe and analyze a faster algorithm to find the shortest path from u to v in a looped tree.

Practice Problem 4: Flow Networks

An ad hoc sensor network network is made up of low-cost, low-power devices that are distributed in some physical space. One example of these networks in the literature involves tossing a bunch of these sensors out of an airplane over a forest. The sensors are then randomly distributed and expected to monitor for conditions that are indicative of a forest fire.

The sensor nodes, however, are prone to failure for a variety of reasons. A given sensor s can detect that it is about to fail (in practice, this may not be possible, but let's assume it's true). When s determines it is failing, it needs to send a representation of its current state to some other nearby sensor that can take over for it. Each device has a limited transmission range, d; i.e., a node can communicate with exactly the set of nodes within d meters. (Notice that this relationship is symmetric: if sensor s_1 can communicate with s_2, the reverse is also true.) Because some set of the sensor's neighbors may have already failed, we actually want to make sure that we send the backup copy to some live node; therefore when a sensor s is failing, it should send its state to k other sensors. These k sensors constitute the *back up set* for s.

You're given a set of n sensors with known positions represented by an (x, y) coordinate for each sensor. You must design an algorithm that determines whether it is possible to choose a back up set for each of the n devices (i.e., for each device, find k other sensors with d meters), with the added constraint that a single sensor can serve as a back up for at most b other sensors. You must also output the back up sets for each sensor, given that a solution exists.

To be completely clear, ensure you complete the following steps:

- Formulate the problem as a network flow problem. This means define the vertices, edges, and capacities of the network flow graph *and* relate your rationale for the structure of the network flow graph to the original problem.
- Be sure to represent k and b in your network flow graph.
- Given a max-flow or a min-cut on your network flow graph, describe how you can determine whether it is possible to choose a back up set for each sensor.
- Given a max-flow or a min-cut on your network flow graph, describe how to state the back up sets, given that a feasible solution exists.

Scratch Page