Homework #4
CS270 Spring 2016

Due Wednesday, April 6, within 10 minutes of the start of lecture

General Instructions. The following assignment is meant to be challenging, so please allow yourself plenty of time to work on it. Please provide a formal mathematical proof for all your claims, and present runtime guarantees for your algorithms using asymptotic (big-O) notation, unless stated otherwise. You may assume that all basic arithmetic operations (multiplication, subtraction, division, comparison, etc) take constant time. In problems involving graphs, we typically use \( n \) and \( m \) to denote the number of nodes and edges, respectively, unless otherwise stated.

Collaboration. Consistent with the tradition in theoretical CS (and other mathematical disciplines), I will allow collaboration and discussion among students in this class for the homework, with a couple of caveats. First, if you meet in a group to work on the homework, your group must include no more than 5 individuals, and you must acknowledge everyone in your group on your submitted homework. Second, you are not allowed to take any written material with you out of the meeting. Third, you must wait at least 30 minutes after any discussion with fellow students, and then write up your solutions independently. The second and third rules hold even if said discussions occur during office hours.

Consulting outside sources. I expect that you will not seek out outside sources pertaining to particular homework problems. In other words, consulting solution manuals, seeking help on the homework from students not enrolled in the class, or googling for solutions to similar problems, are all absolutely considered cheating. That said, you are allowed (and encouraged) to consult general sources (e.g. Wikipedia, books, or course notes) pertaining to course content in order to broaden your perspective, so long as you do not specifically seek out material related to a particular homework problem.

Submission. Homework will be due in-class. Remember to write your name, USC ID number, names of your collaborators, and which lecture you are signed up for (3:30 vs 5:30) on your submission. If you must submit the homework early or late, please leave it in the dedicated drop box #8 on the first floor of PHE, and immediately send email to Kevin Lei (yinghanl@usc.edu) informing him of the time of submission. Recall that you have 4 late days overall in the class, to be used in integer amounts.
Recommended practice problems (do not hand in): KT Problems 6.2, 6.4, 6.6, 6.8, 6.11, 6.16, 6.20, 6.24

Problem 1. (10 points)
After years as a computer scientist, you decide to get into currency trading. There are \( n \) currencies \( M = \{1, \ldots, n\} \) of interest to you, and on a particular day there is a fixed exchange rate which determines how much of one currency you can exchange for a unit of the other. Specifically, for each ordered pair of currencies \( i, j \in M \), you are given a number \( \alpha_{ij} \geq 0 \) equal to the units of currency \( j \) you would receive in exchange for a single unit of currency \( i \). Note that we do not assume any relationship between \( \alpha_{ij} \) and \( \alpha_{ji} \). Also note that \( \alpha_{ij} \) might equal 0, which can be interpreted as disallowing that particular trade.

You notice that on some days, you can trade currencies so that starting with a single US dollar, you end up with strictly more than one US dollar. For example, if the exchange rate is such that one US Dollar (USD) buys you 0.5 Euros (EUR), one Euro buys you 100 Japanese Yens (JPY), and one Yen buys you 3 US Cents (0.03 USD), then trading along this cycle turns one US Dollar into \( 0.5 \times 100 \times 0.03 = 1.5 \) USD. Given the \( n \) currencies and exchange rates \( \{\alpha_{ij} : i, j \in M\} \), design a polynomial-time algorithm for determining whether there is a sequence of trades with positive return on a single US Dollar. Note that the sequence of trades need not be a simple cycle; i.e., you may buy or sell a currency more than once. Also note that you do not need to return the sequence of trades, but simply return a Boolean value on whether or not such a sequence exists.

(Hint: Read chapter 6.10 before trying to answer this question.)

Problem 2. (10 points)
A certain species of fern thrives in lush rainy regions, where it typically rains almost every day. However, a drought is expected over the next \( n \) days, and a team of botanists is concerned about the survival of the species through the drought. Specifically, the team is convinced of the following hypothesis: the fern population will survive if and only if it rains on at least \( \lceil \frac{n}{2} \rceil \) days during the \( n \)-day drought. In other words, for the species to survive there must be at least as many rainy days as non-rainy days.

Local weather experts predict that the probability that it rains on a day \( i \in \{1, \ldots, n\} \) is \( p_i \in [0, 1] \), and that these \( n \) random events are independent. Assuming both the botanists and weather experts are correct, show how to compute the probability that the ferns survive the drought. Your algorithm should run in time \( O(n^2) \).

Problem 3. (10 points)
At a garage sale one day, you stumble upon an old school video game. In this video game, your character must take a journey along an \( n \times n \) grid, collecting rewards along the way. Specifically, there is an \( n \times n \) matrix \( A \) with nonnegative entries, and your character collects a reward equal to \( A_{ij} \) if he visits the cell \((i, j)\) of the grid. Your objective is to maximize the sum of rewards collected by your character.

(a) [4 points]. The rules of level one of the game are as follows. Your character starts at the top-left corner — i.e., cell \((1, 1)\) — of the grid, and must travel to the the bottom-right corner — i.e., cell \((n, n)\) — in sequence of steps. At each step, your character is allowed to move either one cell to the right or one cell down in the grid; stepping upwards, to the left, or diagonally is not
allowed. Show how to compute the optimal journey in $O(n^2)$ time.

(a) [6 points]. Level two is more tricky, and the rules are as follows. Your character starts at the top-left corner, and must travel to the bottom-right corner and then return to the top-left corner. In the first stage of the journey — from the top-left corner to the bottom-right corner — each step must move your character one cell either to the right or down, as in part (a). In the second stage of the journey — from the bottom-right corner back to the top-left corner — each step must move your character one cell either to the left or up. Each reward can be collected at most once: If you visit the cell $(i, j)$ twice, once in the first stage and once in the second stage, then you collect only $A_{ij}$ and not $2A_{ij}$ from that cell. Show how to compute the optimal journey in $O(n^4)$ time.

**Problem 4. (10 points)**
You run a garment company. You sell $n$ different garments, the $i$th of which requires one rectangular piece of cloth which is $a_i$ inches long and $b_i$ inches wide, and sells for $v_i$ dollars. A large piece of cloth comes to you on a roll of length $L$ inches and width $W$ inches. Assume that $L$, $W$, $a_i$, and $b_i$ are integers. Also assume that the machine you use to cut cloth can only cut a rectangular piece of cloth horizontally or vertically into two (not necessarily equal) rectangular pieces. You will use this machine to cut up the $L \times W$ piece of cloth in order to make a set of garments maximizing your total revenue in dollars. Note that you may make multiple copies of any one garment. Compute, in time polynomial in $n$, $L$, and $W$, the maximum revenue possible using your cutting machine. Note that length and width are interchangeable — i.e., a garment of length $a$ and width $b$ is identical to a garment of length $b$ and width $a$. 

3