Quiz 2 Practice

This take-home quiz contains 6 problems worth 25 points each, for a total of 150 points. Your quiz solutions are due between 4:00 and 6:00 P.M. on Friday, November 07, 2008, in the Stata G5 lounge. Make sure to hand your quiz to one of the TAs. Late quizzes will not be accepted unless you obtain a Dean’s Excuse or make prior arrangements with your recitation instructor. You must hand in your own quiz solutions in person.

The quiz should take you about 12 hours to do, but you have three days in which to do it. Plan your time wisely. Do not overwork, and get enough sleep. Ample partial credit will be given for good solutions, especially if they are well written. Of course, the better your asymptotic bounds, the higher your score. If the bounds are the same, a worst-case bound will receive more points than an expected bound. Bonus points will be awarded for exceptionally efficient or elegant solutions.

Policy on academic honesty: The rules for this take-home quiz are like those for an in-class quiz, except that you may take the quiz home with you. As during an in-class quiz, you may not communicate with any person except members of the 6.046 staff about any aspect of the quiz during the exam period, even if you have already handed in your quiz solutions. If at any point any student or staff member initiates a conversation with you about the exam, even to just ask you how it’s going, you can be friendly, but you should reply “I am not talking about the exam”. You should similarly not talk to any of your friends, acquaintances, or anyone else about the problems in the exam. The official end of the exam period is 6:00 P.M. on Friday, November 07.

This take-home quiz is “limited open book.” You may use your course notes, the CLRS textbook, basic reference materials such as dictionaries, and any of the materials posted on the course web page, but no other sources whatsoever may be consulted. For example, you may not use notes or solutions to problem sets, exams, etc. from other times that this course or other related courses have been taught. You may not use other materials on the World-Wide Web. These materials will not help you and may sidetrack you, but you may not use them anyhow. Implementing your algorithms is acceptable but cannot be used as a proof of correctness or running time for your solutions.

If at any time you feel that you may have violated this policy, it is imperative that you contact the course staff immediately. If you have any questions about what resources may or may not be used during the quiz, please send email to 6046-staff@mit.edu.

Write-ups: Begin each problem on a new sheet of paper. Staple together the pages of each individual problem. Each problem must be written or typed neatly. On the top of each page, make sure to write your name, 6.046J/18.410J, the problem number, your recitation time, and your TA’s name.

Your write-up for a problem should start with a topic paragraph that provides an executive summary of your solution. This executive summary should describe

1. the problem you are solving,
2. the techniques you use to solve it,
3. any important assumptions you make,
4. the asymptotic bounds on the running time, space, or other relevant complexity measures your algorithm achieves, including whether they are worst-case, expected, or amortized.

Write your solutions cleanly and concisely to maximize the chance that we understand them. When describing an algorithm, give an English description of the main idea of the algorithm. Adopt suitable notation. Use pseudocode if necessary to clarify your solution. Give examples, draw figures, and state invariants. A long-winded description of an algorithm’s execution should not replace a succinct description of the algorithm itself.

Provide short and convincing arguments for the correctness of your solutions. Do not regurgitate material presented in class. Cite algorithms and theorems from CLRS, lecture, and recitation to simplify your solutions. Do not waste effort proving facts that can simply be cited.

Be explicit about running times and algorithms. For example, don’t just say that you sort \( n \) numbers, state that you are using heapsort, which sorts the \( n \) numbers in \( O(n \lg n) \) time in the worst case. If the problem contains multiple variables, analyze your algorithm in terms of all the variables, to the extent possible.

Part of the goal of this quiz is to test your engineering common sense. If you find that a question is unclear or ambiguous, make reasonable assumptions in order to solve the problem, and state clearly in your write-up what assumptions you have made. Be careful what you assume, however, because you will receive little credit if you make a strong assumption that renders a problem trivial.

**Bugs, etc.:** If you think that you’ve found a bug, send email to 6046-staff@mit.edu. Corrections and clarifications will be sent to the class via email and posted on the class website. Check your email and the class website daily to avoid missing potentially important announcements. If you did not receive an email reminding you about Quiz 2, then you are not on the class email list and you should let your recitation instructor know immediately.

**Survey:** With this quiz is a survey on your experiences with the quiz, especially as they relate to academic honesty. Please fill it out and hand it in when you hand in your quiz solutions. The information you provide will be anonymous. No attempt will be made to identify individuals from their comments. This information will be used to gauge the usefulness of the quiz. Summary statistics and quoted responses will be provided to this and future classes.

**PLEASE REREAD THESE INSTRUCTIONS ONCE A DAY DURING THE EXAM.**

GOOD LUCK, AND HAVE FUN!
Problem 1. Trip Planning with a Gas Tank

You want to drive from some location \( s \) to some location \( t \) without running out of gas while spending as little money as possible. You are given a map of the road system designated by a set of locations \( V \) and a set of directed roads \( E \), where \((u, v) \in E\) means that there is a road going from \( u \in V \) to \( v \in V \). You know \( w(u, v) \), the length of the road from \( u \) to \( v \). You are also told which locations \( S \subseteq V \) have gas stations. When you come to a node \( u \) with a gas station, you have the option to fill up the tank of your car by paying \( c(u) \) dollars (the gas is free - you only pay for the parking).

Assuming you start with a full tank containing \( k \) units of gas, give an efficient algorithm that finds a cheapest route from \( s \) to \( t \) such that you never run out of gas.

Note: it is okay to run out of gas at the same moment you reach a gas station.

Problem 2. Karp and Rabin compute the median

Karp and Rabin each hold a set of \( n \) numbers, denoted by \( K \) and \( R \), respectively. The sets \( K \) and \( R \) are disjoint. They would like to compute the median of the union set \( K \cup R \). However, they live on different coasts, and they can communicate only by exchanging simple messages. Each message can either contain a number from their sets, or an integer in the range \([1, n]\).

Give an efficient communication protocol that Karp and Rabin can use to compute the median of the set \( K \cup R \). The efficiency of the protocol is measured by the number of messages exchanged.

Problem 3. The lightest path

Consider a weighted connected graph \( G \), with \( m \) edges and \( n \) vertices. Each edge weight is an integer in the range \([1, 10n]\). Consider any two nodes \( u, v \in G \), and a path \( P \) between \( u \) and \( v \). The weight of the path \( P \) is defined as the maximum weight of any edge in \( P \).

Give an efficient algorithm that given the graph \( G \) and vertices \( u \) and \( v \), finds the minimum weight path between \( u \) and \( v \) in \( G \).

Problem 4. Updating Max-Flow

Let \( G = (V, E) \) be a flow network with source \( s \), sink \( t \), and integer capacities. Suppose that we are given a maximum flow in \( G \) obtained using the Ford-Fulkerson method. Suppose that the capacity of a single edge \((u, v) \in E\) is increased by 1. Give an \( O(V + E)\)-time algorithm to update the maximum flow.

Problem 5. Search Engine Excerpts

Most popular Internet search engines feature document excerpts for each found document. The goal of search engines is to provide the smallest document excerpt possible that helps the user realize if the found document is relevant to their search. One possible heuristic is to find the smallest excerpt from a found document that includes all of the keywords used in the search.

You are given \( k \) search keywords, and a document that is a sequence of \( N \) words. All \( k \) search keywords are present in the document.

Describe an efficient algorithm that finds a smallest range \([i, j]\) in the document \((i, j \in [1, N])\), that contains at least one instance of every search keyword.
Note: you can assume that each word or search keyword is represented by a number.

**Problem 6. Languages of the new empire**

In the glory days of the Roman empire, Julius Caesar seeks to establish efficient communications between Rome and the myriad provinces it commands in close and remote corners of the empire. One-way broadcast messages are sent frequently from Rome to all of the \( N \) provinces along previously-specified paths, to announce new taxes, new wars, and other urgent matters. The problem is that many different languages are spoken across the empire, and thus a lot of consecutive translations may need to happen along the way to a remote province when a message is sent, causing unwanted delays. Each province speaks only one of the \( K \) total languages in the empire, and Caesar insisted that they receive messages in their own language (his magnanimity knows no limits).

Communication happens by messengers carrying stone tablets, along a graph \( G = (V, E) \) of roads and sailing routes. Since all roads lead to Rome and only three-way intersections are present, the communication graph is a binary tree, with Rome at the root and provinces at the leaves. At each intersection (each internal node) lies an outpost where a single stone tablet arrives, and is then copied and passed on (exactly once for each of the two outgoing paths), in the same language when possible, or translated to a new language when necessary. For every broadcast, a single message tablet is created in Rome, in Latin, and arrives at Rome’s outpost, where it starts its journey throughout the empire. After each intersection, exactly one tablet, in a specified language, leaves down each subsequent path. Since translation of messages takes a long time, you need to select the language of tablets to be used along each segment in a way that minimizes the total cost of needed translations.

In addition to the graph \( G \), and the language spoken at each province, you are given the cost of translating from each language to each other language, represented as a \( K \times K \) matrix \( M \) of positive numbers for the cost \( M[i,j] \) of translating from language \( i \) to language \( j \). Note that translation costs need not be symmetrical, that the cost of translating a language to any other language is always strictly positive, and that all diagonal entries of matrix \( M \) are zero as no translation is necessary from a language to itself. You can also assume that the costs reported in \( M[i,j] \) are optimal, namely you cannot get a lower translation cost from \( i \) to \( j \) by translating through a series of other languages.

Design an efficient algorithm to select the language of tables to be used on each segment (and the translations done at each outpost) in order to minimize the total cost of translations. For partial credit, you can simplify the problem by assuming that any translation has unit cost, i.e. \( M[i,j] = 1 \) for \( i \neq j \).