Introduction to Algorithms December 10, 2003
Massachusetts Institute of Technology 6.046J/18.410J
Professors Shafi Goldwasser and Silvio Micali Handout 34

Practice Final

- Do not open this exam until you are directed to do so. Read all the instructions first.
- When the exam begins, write your name on every page of this exam booklet.
- The exam contains 8 multi-part problems. You have 180 minutes to earn 160 points.
- This exam booklet contains 14 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your exam.
- This exam is closed book. You may use two handwritten A4 or $8\frac{1}{2}'' \times 11''$ crib sheets. No calculators or programmable devices are permitted.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Points	Grade	Initials
1	12		
2	18		
3	27		
4	20		
5	16		
6	21		
7	26		
8	20		
Total	160		

Name: _____

Circle the name of your recitation instructor:

Problem -1. Recurrences [12 points]

For each of the following recurrences, do the following:

- Give the solution in $\Theta(\cdot)$ notation.
- Name a method that can be used to solve the recurrence. **(Do not give a proof.)**
- Mention a recursive algorithm we've seen in class whose running time is described by that recurrence.

(a) $T(n) = T(n/2) + \Theta(1)$

T(n) = $\Theta($ $)$

Method:

Algorithm:

(b) $T(n) = T(n/5) + T(7n/10) + \Theta(n)$

T(n) = $\Theta($ $)$

Method:

Algorithm:

(c) $T(n) = 7T(n/2) + \Theta(n^2)$

T(n) = $\Theta($ $)$

Method:

Algorithm:

Problem -2. Design Decisions [18 points]

The use of algorithms often requires choices. What is efficient in one context may be suboptimal in another. For each of the following pairs, give *one* reason, circumstance, or application for which Choice 1 would be preferable to Choice 2, and vice-versa. Be succinct.

EXAMPLE: (1) insertion sort (2) merge sort

(1) Insert sort is preferrable on small arrays, or when space is a limited resource.
(2) Merge sort is preferable on large arrays, because of its asymptotically optimal $O(n \log n)$ *running time.*

(a) (1) randomized quicksort (2) bucket sort

(b) (1) randomized select (2) worst-case $O(n)$-time select

(c) (1) red-black tree (2) hash table

Problem -3. Short Answer [27 points]

Give *brief*, but complete, answers to the following questions.

(a) Suppose you are given an unsorted array A of n integers, some of which may be duplicates. Explain how you could "uniquify" the array (that is, output another array containing each unique element of A exactly once) in $O(n)$ expected time.

(b) Given a list of distinct real numbers $z_0, z_1, \ldots, z_{n-1}$, show how to find the coefficients of the degree-n polynomial $P(x)$ that evaluates to zero only at $z_0, z_1, \ldots, z_{n-1}$. Your procedure should run in time $O(n \log^2 n)$. (*Hint:* The polynomial $P(x)$ has a zero at z_j if and only if $P(x)$ is a multiple of $(x - z_j)$.)

(c) Consider a generalization of the max-flow problem, in which the network G may have many sources and many sinks. Explain how to reduce this problem to the conventional max-flow problem. Specifically, describe how you would convert the generalized network G to a conventional network G', and how you would translate a maximum flow in G' back to a maximum flow in G.

Problem -4. True or False, and Justify [20 points]

Circle **T** or **F** for each of the following statements, and briefly explain why. The better your argument, the higher your grade, but be brief. Your justification is worth more points than your true-or-false designation.

(a) **T** **F** If a problem L is in NP and L reduces to $3SAT$ ($L \leq_P 3SAT$), then L is NP-complete. (You may assume $P \neq NP$.)

(b) **T** **F** If a graph has negative-weight edges, there exists a constant C such that adding C to every edge weight and running Dijkstra's algorithm produces shortest paths under the original edge weights.

(c) **T** **F** It is possible to compute the convolution of two vectors, each with n entries, in $O(n \lg n)$ time.

(d) **T** **F** The following procedure produces a minimum spanning tree of n given points in the plane.

- Sort the points by x-coordinate. (Assume that all x-coordinates are distinct.)
- Connect each point to its closest neighbor among all points with smaller x coordinate.
  That is, if $p_1, p_2, \ldots, p_n$ denotes the sorted order of points, then for $2 \leq i \leq n$, connect point p_i to its closest neighbor among $p_1, p_2, \ldots, p_{i-1}$.

Problem -5. Average Path Lengths in DAGs [16 points]

Instead of shortest paths in a graph, you might be interested in the *average* length of all paths from a vertex s to a vertex t. In order for this to make sense, we assume that the graph $G = (V, E)$ is directed and acyclic. Let $w(u, v)$ denote the weight of edge (u, v). It suffices to compute the total length of all paths from s to t, and the number of such paths. This problem is well-suited to a dynamic programming approach.

(a) [8 points] Define *count*$[x]$ to be the number of distinct paths from x to t. Define *sum*$[x]$ to be the sum of the lengths of all paths from x to t. Give recurrences for *count*$[x]$ and *sum*$[x]$ in terms of the neighbors of x and the edge weights, and give the bases cases as well.

(b) [8 points] Describe a dynamic-programming algorithm to solve the stated problem, and analyze its running time. (*Hint:* In what order should you consider the vertices, so that when considering vertex x, the values of *count*$[y]$ and *sum*$[y]$ have already been computed for all neighbors y of x?)

Problem -6. Amortized 2-3-4 Trees [21 points]

In this problem, we will analyze the amortized number of modifications made to a 2-3-4 tree during an INSERT operation. For the purpose of this problem, assume that DELETE operations do not occur.

(a) [3 points] Give a tight asymptotic bound on the number of nodes created or modified during one call to INSERT, in the worst case.

(b) [6 points] Suppose we insert an element into a 2-3-4 tree, and the INSERT algorithm splits k nodes. Give an *exact* (not big-O) upper bound on the number of nodes in the tree that are created or modified in this case.

(c) [6 points] Let T be a 2-3-4 tree, and define a potential function

$$\phi(T) = 3 \times \text{(number of "full" nodes in } T\text{)}$$

(a node is full if it stores 3 keys).
If a call to INSERT splits k nodes, how does $\phi(T)$ change as a result of this call?

(d) [6 points] Prove that the amortized number of nodes created or modified per INSERT is $O(1)$.

Problem -7. Maximum Bottleneck Path [26 points]

In the ***maximum-bottleneck-path*** problem, you are given a graph G with edge weights, and two vertices s and t, and your goal is to find a path from s to t whose minimum edge weight is maximized. In other words, you want to find a path from s to t in which *no* edge is light.

(a) [8 points] Suppose that all edges have nonnegative weights. How would you modify a shortest-path algorithm that we covered in lecture to solve the maximum-bottleneck-path problem?

(b) [4 points] Does your solution change if the edges have negative weights? What if there are negative-weight cycles?

(c) [6 points] Suppose we do not need a path which maximizes the minimum edge weight, but we only need a path in which every edge has at least a certain weight. Describe an $O(V + E)$-time algorithm for finding a path from s to t in which every edge has least a given minimum weight $w_{\min}$. (Such an algorithm would have been useful in the movie *Speed*, in which every road traversed had to have a speed limit of at least 50 mph.)

(d) [8 points] Describe how you can make $O(\lg E)$ calls to the algorithm in part **(c)** to solve the maximum-bottleneck-path problem in $O((V + E) \lg E)$ time.

Problem -8. Cliquependent Graphs [20 points]

Given a graph $G = (V, E)$ and nonnegative integers c, s, we say that G is (c, s)-***cliquependent*** if *both* of the following are true:

- there exists a subset $C \subseteq V$ such that $|C| = c$ and, for all distinct $i, j \in C$, $(i, j) \in E$, and
- there exists a subset $S \subseteq V$ such that $|S| = s$ and, for all distinct $i, j \in S$, $(i, j) \notin E$.

Given a graph G and a pair (c, s), the ***cliquependence decision problem*** is to determine whether G is (c, s)-cliquependent.

(a) [3 points] Define the set CLIQUEPENDENT which contains all "yes" instances to the cliquependence decision problem.

(b) [6 points] Show that CLIQUEPENDENT $\in$ NP.

(c) [7 points] Show that CLIQUEPENDENT is NP-complete. (*Hint:* Reduce from *either* CLIQUE or INDEPENDENT-SET.

(d) [4 points] Suppose an $O(n^{100})$-time algorithm were found for the cliquependence decision problem. What would be the implications, if any, on the "P $\stackrel{?}{=}$ NP" question?

SCRATCH PAPER — Please detach this page before handing in your exam.

SCRATCH PAPER — Please detach this page before handing in your exam.