**Practice Final Exam—From Fall 2004**

**Problem 1. Recurrences** (4 parts) [8 points]

For each of the recurrences below, do the following:

- Give the solution using \( \Theta \)-notation. You need not provide a proof or other justification.
- Name a recursive algorithm we’ve seen during the term whose running time is described by that recurrence.

(a) \( T(n) = T(n/2) + \Theta(1) \)
(b) \( T(n) = 2T(n/2) + \Theta(n) \)
(c) \( T(n) = T(n/5) + T(7n/10) + \Theta(n) \)
(d) \( T(n) = 7T(n/2) + \Theta(n^2) \)

**Problem 2. Design Techniques and Data Structures** (5 parts) [10 points]

For each of the following design techniques and data structures, name an algorithm covered this term that uses it.

(a) Divide and conquer:
(b) Dynamic programming:
(c) Greedy:
(d) Binary search tree:
(e) FIFO queue:

**Problem 3. True or False, and Justify** (12 parts) [84 points]

Circle T or F for each of the following statements to indicate whether the statement is true or false, respectively. If the statement is correct, briefly state why. If the statement is wrong, explain why. The more content you provide in your justification, the higher your grade, but be brief. Your justification is worth more points than your true-or-false designation.

(a) T F If \( f(n) \) is asymptotically positive, then \( f(n) + o(f(n)) = \Theta(f(n)) \).
(b) T F An adversary can provide an input to randomized quicksort that will elicit\(^1\) its \( \Theta(n^2) \) worst-case running time.

\(^1\)elicit  transitive verb 1 : to draw forth or bring out (something latent or potential) (hypnotism elicited his hidden fears) 2 : to call forth or draw out (as information or a response) (her performance elicited wild applause) — Merriam-Webster’s Collegiate Dictionary, Tenth Edition, 1993.
(c) T F Any comparison sort of 5 elements requires at least 7 comparisons in the worst case.

(d) T F Consider a sequence of $n$ operations on an initially empty dynamic set. Suppose that the amortized running time of each operation is $O(1)$. Then, the $n$ operations take $O(n)$ time in the worst case.

(e) T F A good heuristic for a simple dynamic set implemented as a linked list is to move an item to the front of the list whenever it is accessed.

(f) T F Prim’s algorithm, Dijkstra’s algorithm, and the Bellman-Ford algorithm are all examples of greedy algorithms.

(g) T F For the all-pairs shortest-paths problem on an edge-weighted graph $G = (V, E)$ with $E = \Theta(V^{3/2})$, the Floyd-Warshall algorithm is asymptotically at least as fast as Johnson’s algorithm.

(h) T F Suppose that the constraint graph $G = (V, E)$ of a linear-programming system of difference constraints is acyclic. Then, a solution always exists and can be found in $O(V + E)$ time.

(i) T F Let $G = (V, E)$ be an edge-weighted digraph, where edge weights are given by the function $w : E \to \mathbb{R}$. Define another edge-weight function $w' : E \to \mathbb{R}$ by

$$w'(u, v) = w(u, v) - \text{out-degree}(u) + \text{out-degree}(v).$$

Then, $G$ contains a negative-weight cycle under $w$ if and only if $G$ contains a negative-weight cycle under $w'$.

(j) T F Suppose that all edge capacities in a flow network are integer multiples of 3, but that the value of a flow between the source $s$ and the sink $t$ is not a multiple of 3. Then, an augmenting path from $s$ to $t$ exists.

(k) T F Given a maximum flow $f$ on a flow graph $G = (V, E)$ with source $s$ and sink $t$, a minimum cut separating $s$ from $t$ can be found in $O(V + E)$ time.

(l) T F The Karp-Rabin algorithm always reports a match of a pattern in a text string if one exists.

**Problem 4. Set Equality** [15 points]

Let $S$ and $T$ be two sets of numbers represented as unordered lists of distinct numbers. All you have are pointers to the heads of the lists, but you do not know the list lengths. Describe an $O(\min \{|S|, |T|\})$-expected-time algorithm to determine whether $S = T$. You may assume that any operation on one or two numbers can be performed in constant time.

**Problem 5. Minimum Spanning Tree** [15 points]

Let $G = (V, E)$ be a connected undirected graph with edge-weight function $w : E \to \mathbb{R}$. Consider the following algorithm:
1 while there exists a cycle \( C \) in \( G \)
2 do find an edge \( e \in C \) such that \( w(e) = \max_{e' \in C} \{w(e')\} \)
3 \( G \leftarrow (V, E - \{e\}) \)

Prove that when this algorithm terminates, \( G \) forms a minimum spanning tree of the original input graph.

**Problem 6. Woody the Woodcutter** (3 parts) [15 points]

Given a log of wood of length \( k \), Woody the woodcutter will cut it once, in any place you choose, for the price of \( k \) dollars. Suppose you have a log of length \( L \), marked to be cut in \( n \) different locations labeled 1, 2, \ldots, \( n \). For simplicity, let indices 0 and \( n + 1 \) denote the left and right endpoints of the original log of length \( L \). Let the distance of mark \( i \) from the left end of the log be \( d_i \), and assume that \( 0 = d_0 < d_1 < d_2 < \cdots < d_n < d_{n+1} = L \). The **wood-cutting problem** is the problem of determining the sequence of cuts to the log that will (1) cut the log at all the marked places, and (2) minimize your total payment to Woody.

(a) Give a small example illustrating that two different sequences of cuts to the same marked log can result in two different costs.

Let \( c(i, j) \) be the minimum cost of cutting a log with left endpoint \( i \) and right endpoint \( j \) at all its marked locations.

(b) Complete the following recursive definition, and briefly justify your answer:

\[
c(i, j) = \min_{i < k < j} \left\{ \right. \]

(c) Using part (b), describe an efficient algorithm to solve the wood-cutting problem. What is the running time of your algorithm?

**Problem 7. Edge Covering** [15 points]

Given an undirected graph \( G = (V, E) \) with no isolated vertices (vertices with degree 0), an **edge cover** is a set \( C \subseteq E \) of edges such that for all \( u \in V \), there exists a \( v \in V \) such that \( (u, v) \in C \). The **edge-covering problem** is the problem of finding an edge cover of minimum cardinality.

Describe an \( O(1) \)-approximation algorithm for the edge-covering problem. Analyze your algorithm’s running time and its ratio bound (by what factor worse than the optimal is the approximation your algorithm produces?).