Practice Final Exam

Problem 1. Recurrences (4 parts)

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. Algorithms and running times (5 parts)

Match each algorithm below with the tightest asymptotic upper bound for its worst-case running time by inserting one of the letters A, B, . . ., into the corresponding box. Some running times may be used multiple times or not at all.

You need not justify your answers.

- Dijkstra’s implemented using Fibonacci heap
- Dijkstra’s implemented using binary heap
- Building a BST
- BUILD-HEAP
- Viterbi decoding in HMM with $n^{1/2}$ states and $n$ emissions

A: $O(E + V \lg V)$
B: $O((V + E) \lg V)$
C: $O(n)$
D: $O(n \lg n)$
E: $O(n^3)$
F: $O(n^2)$
Problem 3. **Design Techniques and Data Structures** (5 parts)

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 4. **True or False, and Justify** (12 parts)

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 its $\Theta(n^2)$ worst-case running time.
(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 In an HMM, let $x_j$ be the emission observed at time $j$. Given a series of observed emissions $x_1, x_2, \ldots, x_n$, the most likely state at time $i$ is independent of emissions $x_{i+1}, x_{i+2}, \ldots, x_n$.
(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.

---

1elicit 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.
(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 5. Set Equality
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 6. Minimum Spanning Tree
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. \hspace{1em} do find an edge \( e \in C \) such that \( w(e) = \max_{e' \in C} \{w(e')\} \)
3. \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} G \leftarrow (V, E - \{e\})

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

Problem 7. Woody the Woodcutter (3 parts)
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) Using part (b), describe an efficient algorithm to solve the wood-cutting problem. What is the running time of your algorithm?

**Problem 8. Edge Covering**

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?).

**Problem 9. Radix Sort**

We have seen that COUNTING-SORT requires \( O(n+k) \) time to sort \( n \) numbers in the range \( 1, \ldots, k \), while RADIX-SORT requires \( O(nd) \) time to sort \( n \) number of \( d \) digits each. By a judicious combination of these algorithms, we can get a linear running time, as in COUNTING-SORT, when operating on elements from a wider range, as in RADIX-SORT.

(a) Given a number \( x \), show how to get the \( i \)th digit of its base-\( r \) representation in \( O(1) \) time. (The 1st digit is the least-significant one.)

(b) What is the running time of RADIX-SORT on an array of \( n \) numbers in the range \( 0, \ldots, n^5 - 1 \), when using base-10 representations?

(c) Give an algorithm to sort an array of \( n \) numbers in the range \( 0, \ldots, n^5 - 1 \) in only \( O(n) \) time. Does your technique extend to wider ranges of elements? Explain.