Midterm  
Introduction to Algorithms  
601.433/633

Tuesday, October 23rd, 1:30-2:45pm

Name:

Ethics Statement
I agree to complete this exam without unauthorized assistance from any person, materials, or device.

Signature: Date:
1 Problem 1 (25 points): Multiple Choice

Indicate which of the following are true. You do not need to give a proof, nor do you need to give counterexamples.

(a) Let \( T(n) = 3T(n-2) \), with \( T(1) = T(2) = 1 \). Circle whichever of the following are true (there may be more than one).

\[
T(n) = \Theta(3^{n/2}) \quad T(n) = \Theta(3^{n/3}) \quad T(n) = \Theta(3^n) \quad T(n) = \Theta(2^n)
\]

(b) \( (\log(n^3))^5 = O(2^5 \log \log n) \)

true  false

(c) On every input, the expected running time of randomized quicksort is \( O(n \log n) \)

true  false

(d) The median-of-medians algorithm (BPFRT) runs in \( O(n) \) time for median-finding, but \( \omega(n) \) time for selecting the \( k \)-th smallest element.

true  false

(e) The sorting lower bound implies that no sorting algorithm can sort \( n \) integers between 1 and \( n^{10} \) in \( o(n \log n) \) time (in the worst case).

true  false

(f) The best list-based Union-Find data structure we discussed in class has an amortized cost in \( O(\log n) \) for Make-Set, \( O(1) \) for Union, and \( O(1) \) for find.

true  false

(g) If we use trees with union-by-rank and path compression, the amortized cost of both Union and Find is \( O(\log^* n) \).

true  false

(h) Let \( H \) be a universal family of hash functions from \( U \) to \([M]\). Then for any sequence of \( L \) insert, delete, and lookup operation, the expectation of the total running time (when \( h \) is drawn uniformly from \( H \)) is \( O(L) \).

true  false

(i) Let \( H \) be a universal family of hash functions. Then for every \( h \in H \), if we choose two elements \( x, y \in U \) uniformly at random, the probability that \( h(x) = h(y) \) is at most \( 1/M \)

true  false

(j) If we build a B-tree with \( t = \Theta(\log n) \), then the running time of Insert is \( O(\log n) \)

true  false
2 Problem 2 (25 points): Data Structures

(a) Draw all possible 2-3-4 trees on the keys \{1, 2, 3, 4, 5\}.
(b) For each of the trees in the previous part, draw a corresponding red-black tree (use a dashed line for red and a solid line for black).
(c) Draw all possible binary min-heaps on the keys \{1, 2, 3, 4, 5\}. 
(d) How many different binomial heaps are there on the keys \{1, 2, 3, 4, 5\}? Justify your answer. Hint: think about the structure of a binomial heap. How many ways are there to fill in that structure legally?
3 Problem 3 (25 points): Amortized Analysis

In this problem we have two stacks $A$ and $B$. In what follows, we will use $n$ to denote the number of elements in stack $A$ and use $m$ to denote the number of elements in stack $B$. Suppose that we use these stacks to implement the following operations:

- **PushA($x$)**: Push element $x$ onto $A$.
- **PushB($x$)**: Push element $x$ onto $B$.
- **MultipopA($k$)**: Pop min($n$, $k$) elements from $A$.
- **MultipopB($k$)**: Pop min($m$, $k$) elements from $B$.

We are using the stacks as a black box. You may assume that **PushA**, **PushB**, **MultipopA(1)**, and **MultipopB(1)** each take exactly one unit of time (i.e. it takes one time step to push or pop a single element).

Justify all of your answers.

(a) Consider the potential function $\Phi = n + m$. Under this potential function, what is the amortized cost of **MultipopA($k$)**, and what is the amortized cost of **MultipopB($k$)**?
Now suppose we add a new operation \textsc{Transfer}(k): Repeatedly pop one element from $A$ and push it into $B$, until either $k$ elements have been moved or $A$ is empty.

(b) What is the amortized cost of \textsc{Transfer}(k) as a function of $n, m, k$ under our potential function $\Phi = n + m$?
(c) Now consider the potential function $\Phi = 3n + m$. What is the amortized cost of $\text{TRANSFER}(k)$ under this potential function?
4 Problem 4 (25 points): Dynamic Programming

A palindrome is a string which is equal to its own reversal, i.e., it is the same whether read backwards or forwards. Given a string $X = (x_1, x_2, \ldots, x_n)$, a string $Z = (z_1, z_2, \ldots, z_k)$ is a palindromic subsequence of $X$ if $Z$ is a palindrome and if $Z$ is a subsequence of $X$ (i.e., there are values $i_1, i_2, \ldots, i_k \in [n]$ such that $i_j < i_{j+1}$ for all $j \in [k-1]$ and $z_j = x_{i_j}$ for all $j \in [k]$). In this problem we will consider algorithms to find the longest palindromic subsequence of $X$.

For $1 \leq i \leq j \leq n$, let $S(i, j)$ be the length of the longest palindromic subsequence of $(x_i, x_{i+1}, \ldots, x_j)$. So $S(1, n)$ is the length of the longest palindromic subsequence of $X$.

(a) Give a recursive formula (not algorithm) for $S(i, j)$. Justify your answer.
(b) Consider the recursive algorithm \textit{without memoization} (not the dynamic programming algorithm) defined by your formula from the previous part. What would the running time of this algorithm be? Justify your answer.
(c) Now consider the dynamic programming algorithm (either top-down or bottom-up) defined by your answer to part (a). What is its running time? Justify your answer.