Quiz #1
Introduction to Algorithms/Algorithms 1
600.363/463

Tuesday, February 25th, 9:00-10:15am

Ethics Statement

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

Signature Date

1 Problem 1 (10 points: each subproblem is 2 points)

For each statement below, indicate whether it is true or false. You do not need to provide proofs or counterexamples.

1. Let f, g, h be three positive functions. If $f = \Theta(g)$ and $h = \Theta(g)$ then $f + h = \Theta(g)$.

   true false

2. Let f, g be two positive functions. If $f = \Omega(g)$ then $g = o(f)$.

   true false

3. $n^2 = O(n^{\log(n)})$

   true false

4. $n \log n = \omega\left((\log(n))^{10}\right)$

   true false

5. $n^{\log n} = \Theta\left(\log(n^n)\right)$

   true false

2 Problem 2 (20 points; each subproblem is 10 points)

Give asymptotic upper bounds for the following recurrences. You may use the Master theorem when it is applicable. You may assume that $n = a^k$ for some a, k. Assume that $T(0) = T(1) = 1$.

1. $T(n) = 25T(n/5) + n + 1$

2. $T(n) = T(n-3) + 5$ (assume that $T(2) = 1$.)

3 Problem 3 (60 points)

Given k arrays $S_1, S_2, \ldots, S_k$, each a sorted array of n numbers, devise a divide-and-conquer algorithm which combines these k sorted arrays into a single sorted kn-length array in $O(kn \log k)$ time.

30 points will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem in $O(k^2 n)$ time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details.

Full credit will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem in $O(kn \log k)$ time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

If you present more than one solution, we will grade the best of your solutions. You may use and assume the correctness of any algorithms from class.

4 Problem 4 (60 points)

Given two sorted arrays A and B, each containing n numbers (with all $2n$ numbers in the two arrays distinct), we call a pair of integers (i, j) with $1 \leq i \leq n$ and $1 \leq j \leq n$, an inversion if $A[i] > B[j]$. Devise an algorithm for counting the number of inversions for a pair of sorted arrays A and B in $O(n)$ time. Prove the correctness of your algorithm and give and prove its runtime.

25 points will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem in $O(n \log n)$ time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details.

Full credit will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem in $O(n)$ time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

If you present more than one solution, we will grade the best of your solutions.