Sample Midterm

This is a sample midterm based on problems that were assigned in previous years.

Problem 1. (36 points)

Give a short answer to each of the following questions. Briefly justify your answers.

(a) Is \( \log^2 n = \Omega(n) \)?

(b) Is \( f(n^2) = O((f(n))^2) \) true for all functions \( f(n) \) that are positive and monotonically increasing?

(c) We know that the largest element in a heap \( A \) of \( n \) distinct elements is \( A[1] \) and, hence, can be determined in \( O(1) \) time. Can the second largest element in a heap be determined in \( O(1) \) time?

(d) Is the following statement true? If the worst-case running time of an algorithm \( A \) is \( \Theta(n) \), then there exist positive constants \( c_1, c_2, \) and \( n_0 \), such that the running time of \( A \) on every input of size \( n \) is at least \( c_1 n \) and at most \( c_2 n \) for all \( n \geq n_0 \).

(e) True or false: The height of any binary tree with \( n \) nodes is \( \Omega(\log n) \).

(f) For which of the three open-addressing schemes, namely, linear probing, quadratic probing, and double hashing, is the following statement true? For any two distinct keys \( k_1 \) and \( k_2 \), if \( h(k_1) = h(k_2) \), then the probe sequence for \( k_1 \) is the same as that for \( k_2 \). Here, \( h \) is the primary hash function.

Problem 2. (20 points)

(a) Solve the following recurrence relation. Assume that \( T(n) \) is \( \Theta(1) \) for \( n \leq 2 \).

\[
T(n) = 4T(n/2) + 2n^2.
\]

(b) Consider the following implementation of Quicksort.

```
NEW-QUICKSORT(A, p, r) (to sort A[p..r])
1. If p < r then
3.   Partition A into two parts A[p..q] and A[q+1..r] such that every element
     in A[p..q] is at most m and every element in A[q+1..r] is greater than m.
4.   NEW-QUICKSORT(A, p, q)
5.   NEW-QUICKSORT(A, q + 1, r)
```
Derive and solve a recurrence for the worst-case running time of NEW-QUICKSORT on an array of \( n \) elements. Assume that all the elements in the input array are distinct.

**Problem 3. (15 points)**

A *mode* of a list of elements is an element that appears in the set most frequently. For example, the list \( \{2, 45, 53, 2, 45, 53, 2, 81\} \) has two modes, 2 and 53, since each of them appears more times than any other element in the set.

Design an efficient algorithm to determine a mode of a list of \( n \) elements. (If the list has more than one mode, then your algorithm may return any one of the modes.) Analyze the worst-case running time of your algorithm. The more efficient your algorithm is in terms of its worst-case running time, the more credit you will get.

**Problem 4. (14 points)**

Give an optimal Huffman code for an \( n \)-character data file in which the frequency of the \( i \)th character is \( 2^i \). (You may simply state the answer without any additional justification.)

**Problem 5. (15 points)**

A group of \( n \) men and \( n \) women are attending a dance class. The instructor wants to pair each man with a woman in such a way that the sum of the absolute value of the height differences between partners is minimized. Give a greedy algorithm for computing an optimal pairing. Briefly justify the correctness of your algorithm. Analyze the running-time of your algorithm.