CS 141, Fall 2004 — Warm-up questions

Here are some problems to warm up for quiz 1.

1. Order the following functions according to their asymptotic growth rate. Indicate which functions belong to the same complexity class.

$$4n\log n, 2^n\log n, n^2+8n\log n, 2^n, (n+4)(n-6), \sqrt{n}, 2^{n-4}, 2^{n/2}$$

2. What is the definition of "worst case time-complexity" of an algorithm?
3. What is meant by the statement "the algorithm A runs in $O(1)$ time".
4. What is meant by the statement "the algorithm A runs in $\Theta(1)$ time".
5. What is meant by the statement "the algorithm A runs in $\Omega(1)$ time".
6. Mark by true or false each of the following (no need to prove)

$5n \in O(n\log n)$ — ☐ True ☐ False

$12n^2 \in O(n\log n)$ — ☐ True ☐ False

$\sqrt{n} \in O(n\log n)$ — ☐ True ☐ False

$\frac{n}{\log n} \in \Theta(n)$ — ☐ True ☐ False

$3^n \in O(2^n)$ — ☐ True ☐ False

$\log 3^n \in O(\log 2^n)$ — ☐ True ☐ False

$3^n \in \Omega(2^n)$ — ☐ True ☐ False

$\log 3^n \in \Omega(\log 2^n)$ — ☐ True ☐ False

7. • Derive an *exact* (without using the "big-oh" notation) recurrence relation for the worst-case time complexity of the following method

```
Algorithm RECURSIVE ($A$ : array, $n$ : integer)
    if $n = 1$ then return
    else
        $B_1 \leftarrow$ RECURSIVE ($A, n/2$)
        $B_2 \leftarrow$ RECURSIVE ($A, n/2$)
        for $i \leftarrow 0$ to $n - 1$ do
            for $j \leftarrow 0$ to $n - 1$ do
                "do something" with $B_1$ and $B_2$
```

assuming that the statement **return** costs 1 elementary operation and "do something" costs c elementary operations

- solve *exactly* (without using the "big-oh" notation) the recurrence relation (assume for simplicity that n is a power of two).
- prove by induction the correctness of your solution

8. For each of the following recurrence relations determine whether the Master Theorem can be *directly* applied. If it can, give the asymptotic bound and show which one of the three cases holds and why. If it cannot, show why all cases fail. Assume that $T(1) = 1$.

$$\begin{aligned} T(n) &= 4T(n/4) + 6 \\ T(n) &= 4T(n/3) + n^2 \\ T(n) &= 2T(n/2) + n/\log n \\ T(n) &= 6T(n/6) + 3n - 5 \\ T(n) &= T(\sqrt{n}) + \log n \end{aligned}$$

9. Assume you are given an array A of size n containing real numbers. Describe and analyze an efficient algorithm to determine $n/4$ number **not** in array A.

10. Assume you are given an array A of size n and an array B of size m, $n \geq m$. The elements in each array are integers in arbitrary order. Describe and analyze an efficient algorithm to determine whether the elements in the two arrays are disjoint (i.e., there are no duplicates). State your running time in terms of m and n.

11. Let $S = \{a, b, c, d, e, f, g\}$ be a collection of objects with the following benefit-weight values: a:(12,4), b:(10,6), c:(8,5), d:(11,7), e:(14,3), f:(7,1), g:(9,6). What is an optimal solution to the fractional knapsack problem for S assuming we have a sack that can hold objects with total weight 18? Show your work.

12. Suppose we are given a set of tasks specified by pairs of the start times and finish times as

$$T = \{(1,2), (1,3), (1,4), (2,5), (3,7), (4,9), (5,6), (6,8), (7,9)\}$$

Solve the task scheduling problem for this set of tasks.

13. Use the divide-and-conquer algorithm (Karatsuba algorithm) to compute 10110011×10111010 in binary. Show your work.

14. Use Strassen's matrix multiplication algorithm to multiply the following matrices

$$X = \begin{pmatrix} 3 & 2 \\ 4 & 8 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 & 5 \\ 9 & 6 \end{pmatrix}$$

15. Draw the frequency array and Huffman tree for the following string: `"dogs do not spot hot pots or cats"`.

16. Let A be an $n \times n$ array of 0's and 1's. Design an $O(n^2)$ time algorithm for finding the largest square block of A which contains 1's only.

17. Design a divide-and-conquer algorithm for finding the minimum and the maximum element of n numbers using no more than $3n/2$ comparisons.

Some solutions.

1.
$$\sqrt{n} < 4n \log n < \{n^2 + 8n \log n, (n+4)(n-6)\} < 2^{n/2} < \{2^n, 2^{n-4}\} < 2^n \log n$$

2.3.4.5. See slides

6. T, F, T, F, F, T, T, T

7.
$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + cn^2 & n > 1 \end{cases}$$

and $T(n) = n + 2cn^2 - 2cn$

8.

$$\begin{array}{rc} T(n) = 4T(n/4) + 6 & \text{case (1) where } \epsilon = 1 \\ T(n) = 4T(n/3) + n^2 & \text{case (3) where } \epsilon \leq 2 - \log_3 4 \text{ and } \delta \geq 2 - \log_3 4 \\ T(n) = 2T(n/2) + n/\log n & \text{all fail} \\ T(n) = 6T(n/6) + 3n - 5 & \text{case (2) where } k = 0 \\ T(n) = T(\sqrt{n}) + \log n & \text{does not match the template} \end{array}$$

9. Can be done in $O(n)$ time by first finding the max and then printing the following numbers $max + 1$, $max + 2$, $\ldots$, $max + n/4$. Finding the min would also work.

10. Sort B in $O(m \log m)$. For each element in A search it in B using binary search. The overall time is $O(m \log m + n \log m)$.

16. By dynamic programming. Define $l(i, j)$ to be the length of the side of the largest square block of 1's whose bottom-right corner is $A(i, j)$. Write $l(i, j)$ as a function of $l(i-1, j-1)$, $l(i-1, j)$ and $l(i, j-1)$.

17. First note that a simple linear scan would take $2n - 2$ comparisons. In order to get to $3n/2$, compare all pairs of adjacent values in A. The smaller goes in another array B, the larger goes into an array C. This requires $n/2$ comparisons. Then, we find the minimum in B using $n/2$ comparisons, and we find the maximum in C using $n/2$ comparisons.