Ph.D. Qualifying Exam: Analysis of Algorithms

This is a closed book exam. The total score is 100 points. Please answer all questions.

(20 points) 1. (This problem tries to establish that polynomial running time is asymptotically bounded from above by exponential running time.) Prove $n^{2.1} = o(15^n)$. Hint: You may take advantage of the fact that $\ln x \le x - 1$ for any $x > 0$. Also note that $\ln 15 \approx 2.7$.

(35 points) 2. An array $A[1\ldots n]$ is said to have a *majority element* if more than half of its entries are the same. The elements of the array are not comparable, i.e., the truth of $A[i] > A[j]$ is not defined. But the truth of $A[i] = A[j]$ or $A[i] \neq A[j]$ are uniquely defined. For example, the array can be

$$A = (\spadesuit, \clubsuit, \heartsuit, \spadesuit, \clubsuit, \spadesuit, \diamondsuit, \spadesuit, \spadesuit)$$

In this case, "$\spadesuit$" would be the majority element. We also define the majority element of a singleton array is just that singleton element. The problem is to detect a majority element from the array A of size n if it exists or simply report the array does not have a majority element.

You will not be able to apply any hash function or indexing on the elements in A.

You may find the Master Theorem useful in deriving the running time.

> **Theorem 1** (Master Theorem). *Let $a \ge 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence*
>
> $$T(n) = aT(n/b) + f(n),$$
>
> *where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows.*
>
> 1. *If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.*
> 2. *If $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \lg n)$.*
> 3. *If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.*

Show how to solve this problem in $O(n \log n)$ time. (Hint: Split the array A into two arrays A_1 and A_2 of half the size. Observe the relations of the majority elements among A, A_1 and A_2.) You must justify your algorithm and analyze the running time.

3. For two strings $X = \langle x_1, \ldots, x_n \rangle$ and $S = \langle s_1, \ldots, s_m \rangle$, if for some $1 \le i_1 < i_2 < \ldots < i_n \le m$,

$$s_{i_1} s_{i_2} \ldots s_{i_n} = x_1 \ldots x_n$$

then S is a *super-sequence* of X, and X is a *subsequence* of S. For example, *abecbdaed* is a super-sequence of *abcbae*, but *baecbdaed* is not.

(10 points) (a) Please determine a shortest common super-sequence of

acbd and *dccab*

(35 points) (b) Given two strings X and Y, of length m and n respectively, devise a dynamic programming algorithm to find the shortest common super-sequence for X and Y. Please

(1) define the recurrence used by your dynamic programming strategy,

(2) give the pseudo-code of the dynamic programming algorithm to find the length of a shortest common super-sequence, and

(3) give the pseudo-code of a backtrack algorithm to print out a shortest common super-sequence.