600.363/463 Algorithms - Fall 2013
Solution to Assignment 2

(90 points+20 bonus points)

I (30 points)

1 $a = 25, b = 5, \log_b a = \log_5 25 = 2$. Since $f(n) = n^{2.1} = n^{2+0.1} = \Omega(n^{2+0.1})$, by the master theorem part 3, $T(n) = \Theta(n^{2.1})$

2 $a = 25, b = 5, \log_b a = \log_5 25 = 2$. Since $f(n) = n^{1.5} = n^{2-0.5} = O(n^{2-0.5})$, by the master theorem part 1, $T(n) = \Theta(n^2)$

3 $a = 25, b = 5, \log_b a = \log_5 25 = 2$. Since $f(n) = n^2 = \Theta(n^2)$, by the master theorem part 2, $T(n) = \Theta(n^2 \log n)$

II (20 points)

$$\begin{aligned}
T(n) &\leq 25T\left(\frac{n}{5}\right) + n^2 \log n \\
&= 25\left(25T\left(\frac{n}{5^2}\right) + \left(\frac{n}{5}\right)^2 \log \frac{n}{5}\right) + n^2 \log n \\
&= 25^2T\left(\frac{n}{5^2}\right) + 25\left(\frac{n^2}{5^2}\right) \log \frac{n}{5} + n^2 \log n \\
&= 25^2T\left(\frac{n}{5^2}\right) + n^2 \log \frac{n}{5} + n^2 \log n \\
&= 25^2T\left(\frac{n}{5^2}\right) + 2n^2 \log n - n^2 \log 5 \\
&= 25^2\left(25T\left(\frac{n}{5^3}\right) + \left(\frac{n}{5^2}\right)^2 \log \frac{n}{5^2}\right) + 2n^2 \log n - n^2 \log 5 \\
&= 25^3T\left(\frac{n}{5^3}\right) + 3n^2 \log n - (1+2)n^2 \log 5 \\
&= \cdots \\
&= 25^kT\left(\frac{n}{5^k}\right) + kn^2 \log n - (1+2+\cdots+k-1)n^2 \log 5 \\
&= 25^kT\left(\frac{n}{5^k}\right) + kn^2 \log n - \frac{k(k-1)\log 5}{2}n^2
\end{aligned}$$

Let $n/5^k = 1 \Rightarrow k = \log_5 n = \log n / \log 5$, then

$$\begin{aligned}
T(n) &\leq 25^{\log_5 n}T(1) + \frac{\log n}{\log 5}n^2 \log n - \frac{\log n(\log n - \log 5)}{2\log 5}n^2 \\
&= n^2O(1) + O(n^2 \log^2 n) - O(n^2 \log^2 n) \\
&= O(n^2 \log^2 n)
\end{aligned}$$

Alternatively, proof by induction also get the full points.

III (20 points)

Algorithm 1: Element-Distinctness

```
  Input:  One array A of length n
  Output:  True if A has duplicate elements. False otherwise.
1 Sort A by an O(n log n) algorithm;
2 flag ← False;
3 for i ← 1..n − 1 do
4 |   if A[i+1]==A[i] then
5 |   |   flag← True;
6 |   |   break;
7 |   end
8 end
9 return flag;
```

Sorting takes $O(n \log n)$ time, and the for loop in lines 3-8 takes $O(n)$ time at the worst case, therefore the algorithm takes $O(n \log n)$ time.

IV (20 points)

1 3-elements-grouping

Let x be the pivot element, then the number of elements greater than x is at least $2(1/2\lceil n/3 \rceil) \geq n/3$, thus at most $2n/3$ elements are considered in the next recursion. Therefore

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$$

for some positive constant c.

Claim that $T(n) \leq dn \log n$ for d sufficiently large,

$$\begin{aligned} T(n) &\leq d\left(\frac{n}{3}\right)\log\left(\frac{n}{3}\right) + d\left(\frac{2n}{3}\right)\log\left(\frac{2n}{3}\right) + cn \\ &= dn\log n + \left(\frac{2}{3}\right) dn \log 2 - dn \log 3 + cn \\ &= dnlogn - ((\log 3 - 2/3)d - c)n \\ &\leq dn \log n \end{aligned}$$

for $d \geq c/(\log 3 - 2/3)$.

Therefore $T(n) = O(n \log n)$.

2 7-elements-grouping

Let x be the pivot element, then the number of elements greater than x is at least $4(1/2\lceil n/7 \rceil) \geq 2n/7$, thus at most $5n/7$ elements are considered in the next recursion. Therefore

$$T(n) = T\left(\frac{n}{7}\right) + T\left(\frac{5n}{7}\right) + cn$$

for some positive constant c.
Claim that $T(n) \le dn$ for d sufficiently large,

$$\begin{aligned} T(n) &\le d\left(\frac{n}{7}\right) + d\left(\frac{5n}{7}\right) + cn \\ &= \left(\frac{6}{7}d + a\right)n \\ &= dn - \left(\frac{1}{7}d - c\right)n \\ &\le dn \end{aligned}$$

when $d \ge 7c$.
Therefore $T(n) = O(n)$.

V (BONUS 20 points) For the element distinctness problem derive a lower bound of $\Omega(n \log n)$. (Hint: As we already know that the lower bound for comparison-based sorting is $\Omega(n \log n)$, we just need to prove that the number of comparisons involved in *element-distinctness* problem is no less than the number of comparisons in comparison-based sorting problem.)
Proof:
Without loss of generality, let $a_1 < a_2 < \cdots < a_n$ be any n distinct elements.
Claim: for any permutation of $a_1, a_2, \cdots, a_n$ given as input to the element-distinctness algorithm, before it responds that the element are distinct, it must performed enough comparisons so that it can output the sorted data.
(Proof by contradiction) If the claim is not true, suppose for some i, the order that $a_i < a_{i+1}$ is not known, that is, the element-distinctness algorithm has not compare a_i and a_{i+1} yet, then change a_{i+1} to a_i and this change would not result in any change to the response. So the element-distinctness algorithm would output that the new set of elements are distinct, which is obviously incorrect.
Hence the element-distinctness algorithm performs no less comparisons than sorting algorithms, therefore it has the lower bound of $\Omega(n \log n)$.