

# CSE 259 Algorithms and Complexity

## Fall 2006; Exam I; Solutions

- $n^n = 2^{n \log n}$ .  $2^{2^{\log^2 n}} = 2^{n^{\log n}}$ . Since  $n \log n = o(n^{\log n})$ , it follows that  $n^n = O(2^{2^{\log^2 n}})$ .
  - This statement need not always be true. As an example, if  $f(n) = n^3 + n^2$  and  $g(n) = n^3 + n$ ,  $f(n) - g(n) = n^2 - n \neq \Theta(n^3)$ .

2. Consider the following algorithm:

**for**  $i := 1$  **to**  $\alpha n^{2/3} \log_e n$  **do**

    Pick a random  $j \in [1, n]$  and pick a random  $k \in [1, n]$ ; If  $j \neq k$  and  $a[j] = a[k]$   
    then output: "Type II" and quit;

Output: "Type I";

**Analysis:** Note that if the array is of type I, the above algorithm will never give an incorrect answer. Thus assume that the array is of type II. We'll calculate the probability of an incorrect answer as follows.

Probability of coming up with the correct answer in one iteration of the for loop is  $\frac{n^{2/3}(n^{2/3}-1)}{n^2} \approx \frac{1}{n^{2/3}}$ . Thus, probability of failure in any iteration is  $1 - \frac{1}{n^{2/3}}$ . As a consequence, probability of failure in  $q$  successive iterations is  $(1 - \frac{1}{n^{2/3}})^q \leq \exp(-q/n^{2/3})$  (using the fact that  $(1 - 1/x)^x \leq 1/e$  for any  $x > 0$ ). This probability will be  $\leq n^{-\alpha}$  when  $q \geq \alpha n^{2/3} \log_e n$ .

Thus the output of this algorithm is correct with high probability.

3. Keep only one variable called  $min$ . INSERT( $x$ ): if  $(x < min)$   $min = x$ ; MIN( $x$ ): return  $min$ ;
4. There are five calls made to **Heapify**. The tree takes the following shape after these five calls:  
calls:           23, 12, 5, 6, 34, 17, 14, 8, 2, 11, 21;           23, 12, 5, 8, 34, 17, 14, 6, 2, 11, 21;  
23, 12, 17, 8, 34, 5, 14, 6, 2, 11, 21;           23, 34, 17, 8, 21, 5, 14, 6, 2, 11, 12;           and  
34, 23, 17, 8, 21, 5, 14, 6, 2, 11, 12.
5. Recurrence relation for the run time of  $\mathcal{A}$  is:  $T(n) = 8T(n/2) + n^2$ . Here  $a = 8, b = 2, f(n) = n^2$ .  $n^{\log_b a} = n^3$ . Case 1 of Master theorem applies. Thus,  $T(n) = \Theta(n^3)$ .

Recurrence relation for the run time of  $\mathcal{B}$  is:  $T(n) = 64T(n/8) + n^{2.8}$ . Here  $a = 64, b = 8, f(n) = n^{2.8}$ .  $n^{\log_b a} = n^2$ . Case 3 of Master theorem applies implying that  $T(n) = \Theta(n^{2.8})$ .

Therefore, algorithm  $\mathcal{B}$  is preferable.

6. Let the array be  $a[1 : n]$ . Invoke  $\text{FindTransition}(a, 1, n)$ .

$\text{FindTransition}(a, l, r)$

(1)  $m = \lceil (l + r)/2 \rceil$ .

(2) if  $a[m - 1] < a[m]$  and  $a[m] > a[m + 1]$  return  $m$ ;

(3) if  $a[m - 1] < a[m] < a[m + 1]$  return  $\text{FindTransition}(a, m + 1, r)$ ;

(4) if  $a[m - 1] > a[m] > a[m + 1]$  return  $\text{FindTransition}(a, l, m - 1)$ ;