CS3230 Tutorial 2 (Analysis of Algorithms) Sample Solutions

February 2, 2017

1 Question 1

The general procedure for solving this kind of questions: For any eventually positive functions $f(n)$ and $g(n)$, if we would like to compare the order of growth, we compute:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \text{ has a lower order of growth than } g(n) \\ c > 0 & f(n) \text{ has the same order of growth as } g(n) \\ \infty & f(n) \text{ has a higher order of growth than } g(n) \end{cases}$$

Thus the final answer for this question: the former versus the latter:

1. $n(n+5,000) \approx n^2$ has the **same** order of growth (quadratic) as $2,000n^2$ to within a constant multiple.
2. $10^{1,000}n^2$ (quadratic) has a **lower** order of growth than $10^{-1,000}n^3$ (cubic).
3. $\log_2 n^{1,000} = 1,000 \cdot \log_2 n$. Since changing a logarithm's base can be done by the formula $\log_a n = \log_a b \cdot \log_b n$, these logarithmic functions have the **same** order of growth to within a constant multiple.
4. $(\log_2 n)^2 = \log_2 n \cdot \log_2 n$ and $\log_2 n^2 = 2\log n$. Hence $(\log_2 n)^2$ has a **higher** order of growth than $\log_2 n^2$.
5. $2^{n-1} = \frac{1}{2}2^n$ has the **same** order of growth as 2^n to within a constant multiple.
6. $(n-1)!$ has a **lower** order of growth than $n! = (n-1)!n$.

Note that L'hopital's rules can be useful in computing the limits.

2 Question 2

There are generally two methods for solving this kind of questions:

- Find function $g(n)$, such that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0$$

  then $f(n) \in \Theta(g(n))$.

- Find functions $a(n) \in \Theta(g(n))$ and $b(n) \in \Theta(g(n))$, such that $a(n) \leq f(n) \leq b(n)$ for all n that are sufficiently large; then $f(n) \in \Theta(g(n))$.

1. n^{18}
2. n^2
3. $n^2 \log n$
4. 3^n
5. $\log n \left(Hint : \log(n+1) - 1 < \lceil \log(n+1) \rceil < \log(n+1) + 1 \right)$

6. $\log n$ $\left(Hint: 1 + \frac{k}{2} < \sum_{i=1}^{2^k} i^{-1} < 1 + k.\right)$

7. $n \log n$ $\Big(Hint: (\frac{n}{3})^n < n! < (\frac{n}{2})^n$, for $n \geq 6$; Sterling's approximation $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$. Proof by induction.$\Big)$

3 Question 3

1. The algorithm returns "true" if its input matrix is symmetric and "false" if it is not.
2. Comparison of two matrix elements.
3. $$C_{worst}(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n-1}(n-i) = (n-1) + \ldots + 1 = \frac{(n-1)n}{2}$$
4. $O(n^2)$.

4 Question 4

The key idea here is to walk intermittently right and left going each time exponentially farther from the initial position, where the motivation is given in tutorial class. A simple implementation of this idea is to do the following until the door is reached:

For $i = 0, 1, \ldots$, make 2^i steps to the left, return to the initial position if the door is not reached, make 2^i steps to the right, and return to the initial position again. Let $2^{k-1} < n \leq 2^k$. The number of steps this algorithm will need to find the door can be estimated above as follows (in the worst case):

$$\sum_{i=0}^{k-1} 4 \cdot 2^i + 3 \cdot 2^k = 4(2^k - 1) + 3 \cdot 2^k < 7 \cdot 2^k = 14 \cdot 2^{k-1} < 14n.$$

Hence the number of steps made by the algorithm is in $O(n)$. (Note: the shown summation is not the only solution to this problem. There exist many other series that also give $O(n)$ complexity).

5 Question 5

Observe that for every move of the ith disk, the algorithm first moves the tower of all the disks smaller than it to another peg (this requires one move of the $(i+1)$st disk) and then, after the move of the ith disk, this smaller tower is moved on the top of it (this again requires one move of the $(i+1)$st disk). Thus, for each move of the ith disk, the algorithm moves the $(i+1)$st disk exactly twice. Since for $i = 1$, the number of moves is equal to 1, we have the following recurrence for the number of moves made by the ith disk:

$$M(i+1) = 2M(i) \text{ for } 1 \leq i < n, M(1) = 1.$$

Its solution is $M(i) = 2^{i-1}$ for $i = 1, 2, \ldots, n$.

6 Question 6

Let $W(n)$ be the number of different ways to climb an n-rung ladder. $W(n-1)$ of them start with a one-rung climb and $W(n-2)$ of them start with a two-rung climb. Thus,

$$W(n) = W(n-1) + W(n-2) \text{ for } n \geq 3. \;\; W(1) = 1, W(2) = 2.$$

Solving this recurrence either "from scratch" or better yet noticing that the solution runs one step ahead of the canonical Fibonacci sequence $F(n)$, we obtain $W(n) = F(n+1)$ for $n \geq 1$.

Any bugs and typos, please report to Roger Zimmermann (rogerz@comp.nus.edu.sg).