

CIS 121 - Fall 2009

Midterm 1

NAME:

LAB SESSION:

1	20	
2	20	
3	20	
4	15	
5	10	
6	15	
Sum	100	

For code fragments whose running time is a constant, do not give a specific number of steps, use symbolic constants like c_1 , c_2 , etc. instead.

$$1 + q + q^2 + q^3 + \dots + q^{n-1} = \frac{q^n - 1}{q - 1}.$$

1 (20 Points)

Find the big-Theta bound for the worst running time of the following code

```
void frodo(int n) {
    int sum = 0;
    for (int j = 0; j < n*n; j++)
        if (j % n == 0)
            for (int k = 0; k < j; k++)
                sum++;
}
```

Solution: The outer for+ loop will run a total of n^2 times. The inner for+ loop will execute $\sum_{i=0}^{n^2} i = \frac{n^2(n^2+1)}{2}$ times, but since it is only executed one out of n times, we have a runtime of $\Theta(n^3)$

2 (20 Points)

The concatenation of strings ($s1 + s2$) takes $\Theta(N)$ time, everything else (constructors, substring) constant time. To find a Θ -bound for the method below, write the recurrence relation and solve it for $N = 2^m$. Write the result in terms of N (not m).

```
public static String reverse(String s) {
    int N = s.length();
    if (N <= 1) return s;
    String left = s.substring(0, N/2);
    String right = s.substring(N/2, N);
    String revright = reverse(right);
    String revleft = reverse(left);
    return (revright+revleft);
}
```

Solution: The function with string of length N calls itself two times with input $N/2$. Before returning the value it concatenates the strings returned from the recursive calls which are additional N operations. The cost of the remaining steps is constant. Hence the recurrence (up to constant factor) has the form:

$$T(N) = 2T(N/2) + N$$

For $N = 2^m$ it is $T(2^m) = 2T(2^{(m-1)}) + 2^m$. It can be solved by writing it as

$$\begin{aligned} T(2^m) &= 2T(2^{(m-1)}) + 2^m \\ 2T(2^{(m-1)}) &= 2^2T(2^{(m-2)}) + 2 \cdot 2^{(m-1)} \\ 4T(2^{(m-2)}) &= 2^3T(2^{(m-3)}) + 2^2 \cdot 2^{(m-2)} \\ &\vdots \\ 2^m T(2^0) &= 2^m T(2^0) + 2^m \cdot 2^0 \end{aligned}$$

Adding the above equations results in

$$T(2^m) = 2^m T(1) + (2^m + 2 \cdot 2^{(m-1)} + 2^2 \cdot 2^{(m-2)} + \dots + 2^m \cdot 2^0) = (T(1) + m)2^m$$

Hence $T(N)$ is $O(N \log N)$.

3 (20 Points)

Solve the recurrence relation

$$T(n) = T(n/2) + \log n$$

and provide a big-Theta bound. No need to apply the formal definition. Assume $T(1) = c$.

Solution: Substitute $n = 2^k$, recognize $\log(2^k) = k$.

$$T(n) = T(n/2) + \log n$$

$$T(2^k) = T(2^{k-1}) + k$$

$$T(2^{k-1}) = T(2^{k-2}) + k - 1$$

$$T(2^{k-2}) = T(2^{k-3}) + k - 2$$

⋮

$$T(2) = T(1) + 1$$

Then,

$$T(2^k) = c + 1 + 2 + \cdots + k$$

Which is $\Theta(k^2)$ and since we know $k = \log(n)$.

So $\Theta(\log^2(n))$

4 (15 Points)

Sort the following running times so that $f(n)$ is left from $g(n)$ is $f(n)$ is $O(g(n))$:

$$n!, (\log n)^2, 7n^{500} + 2^n, n \log n, (\sqrt{2})^n, \log(\log n), 3^n$$

Solution: $\log(\log n), (\log n)^2, n \log n, (\sqrt{2})^n, 7n^{500} + 2^n, 3^n, n!$

5 (10 Points)

10^n is $O(n!)$. True or False? Prove your answer given the definition of big-O.

Solution: True. We show 10^n is $O(n!)$ by showing there exists constant N and C such that

$$\forall n \geq N. 10^n \leq cn! \quad (1)$$

which is equivalent to

$$\forall n \geq N. \log(10^n) \leq \log(cn!) \quad (2)$$

$$n \log(10) \leq \log c + \log(n!) \quad (3)$$

$$(n \log(10) - \sum_{k=1}^n \log(k)) \leq \log c \quad (4)$$

$$\sum_{k=1}^n (\log(10) - \log(k)) \leq \log(c) \quad (5)$$

Note that the left side of the constraint is decreasing, therefore, simply pick $c = 1$, and N the first number makes left side negative.

6 (15 Points)

a. Prove that $\log(n!)$ is $O(n \log n)$.

Solution: Observe that

$$\log(n!) = \sum_{k=1}^n \log(k) \leq \sum_{k=1}^n \log(n) = n \log n$$

Hence simply pick $c = 1$ and $N = 1$ we would have big-O definition. b. We know that

$$\left(1 + \frac{1}{n}\right)^n < 3.$$

Prove using induction that

$$n! > \left(\frac{n}{3}\right)^n.$$

Solution: Prove by induction:

base case: $1! = 1 > 1/3 = (\frac{1}{3})$

induction step: assuming $k! > (\frac{k}{3})^k$, for $k + 1$ we have:

$$\frac{(k+1)!}{k!} = (k+1)$$

that is

$$(k+1)! = k!(k+1) \tag{6}$$

and

$$\begin{aligned} \frac{\left(\frac{k+1}{3}\right)^{k+1}}{\left(\frac{k}{3}\right)^k} &= \left(\frac{k+1}{k}\right)^k \left(\frac{k+1}{3}\right) \\ &= \left(\frac{k+1}{k}\right)^k \left(\frac{k+1}{3}\right) \\ &= \left(1 + \frac{1}{k}\right)^k \left(\frac{k+1}{3}\right) \\ &< 3 \left(\frac{k+1}{3}\right) \\ &= k+1 \end{aligned}$$

which is

$$\left(\frac{k+1}{3}\right)^{k+1} < \left(\frac{k}{3}\right)^k (k+1) \tag{7}$$

from induction step we can then conclude

$$\left(\frac{k+1}{3}\right)^{k+1} < \left(\frac{k}{3}\right)^k (k+1) < k!(k+1) = (k+1)! \tag{8}$$

c. Prove that $\log(n!)$ is $\Omega(n \log n)$.

Solution:

If 6b holds then $\log(n!) > n(\log n - \log 3) > c \log n$ for $c = 1/2$ and $n \geq 10$