

# CIS 121 - Fall 2009

## Midterm Review 1

### 1

Order the following functions by asymptotic growth rate indicating when two or more are big-Theta of each other:  $n^2\sqrt{n}$ ,  $2^{2n-3}$ ,  $\sqrt{n^3}$ ,  $4n \log n + 2n$ ,  $3^{n+3}$ ,  $1 + 2^2 + \dots + n^2$ ,  $\log n^n$ ,  $\frac{\log n^2}{\log n}$

#### Solution:

$$\frac{\log n^2}{\log n} = 2$$

$$\log n^n = n \log n$$

$$4n \log n + 2n \text{ is } \Theta(n \log n)$$

$$\sqrt{n^3} = n\sqrt{n}$$

$$n^2\sqrt{n}$$

$$1 + 2^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6} \text{ is } \Theta(n^3)$$

$$3^{n+3} \text{ is } \Theta(3^n)$$

$$2^{2n-3} = 4^n/8 \text{ is } \Theta(4^n)$$

### 2

For each statement below, decide whether it is true or false. In each case attach a *very brief* explanation of your answer.

1. Suppose that the worst-case of method `qq` is  $\Theta(n \log n)$  and the worst case running time of method `uu` is  $\Theta(n^2)$ . Then there is no input for which `uu` runs faster than `qq`, true or false?

**Solution:** FALSE. There may be inputs which are worst-case for `qq` but not worst-case for `uu`. For such inputs, `qq` runs in  $\Theta(n \log n)$  but `uu` may run much faster, for example in constant time.

ALTERNATIVE JUSTIFICATION: Since the Big-Theta relation only refers to “big enough”, there may be a small input for which `uu` runs faster than `qq`.

2. Suppose that  $f(n)$  is  $O(1)$ . Then, there exists a constant  $c'$  such that  $f(n) \leq c'$  for all  $n \geq 1$ , true or false?

**Solution:** There was a missing piece of information in this problem. We need to assume that  $f$  is only defined on the integers. In that case, the answer is TRUE, because we can take

$$c' = \max(f(1), \dots, f(N-1), c)$$

where  $N, c$  are given by the fact that  $f(n)$  is  $O(1)$ . (That is, there exist  $N, c > 0$  such that  $f(n) \leq c \cdot 1$  for all  $n \leq N$ )

If  $f$  is a function defined on positive reals, then it is possible that in the interval  $[1, N)$  the function is unboundedly large and the answer would be FALSE.

3. Suppose we decide to change our model of computation (step-counting) by counting instructions that involve the creation of an array `new int[n]` as taking  $10n$  steps. In this new model, a program can have a different asymptotic complexity than in the original model, true or false?

**Solution:** TRUE. For example a program that just creates an array of the same size as the input and does nothing else (not even an array initialization) will run in time  $O(n)$  with this new model and time  $O(1)$  with our usual model.

### 3

In this problem you are NOT allowed to use the theorems about Big-Oh stated in the lecture notes. Your proof should follow just from the definition of Big-Oh.

Prove that  $n\sqrt{n}$  is not  $O(n)$ .

**Solution:** Proof by contradiction. Suppose that  $n\sqrt{n}$  is  $O(n)$ . Then, there exist  $c, n_0 > 0$  such that  $n\sqrt{n} \leq cn$  for all  $n \geq n_0$ . Dividing by  $n$  and squaring we get  $n \leq c^2$  for all  $n \geq n_0$ . But this is impossible. Take, for example,  $n = \max(n_0, c^2 + 1)$ .

### 4

Let  $A$  be an integer array of length  $n \geq 3$ . Give an  $O(n)$  algorithm to compute another array  $B$  of

length  $n - 1$  such that for  $i = 0, \dots, n - 2$  we have  $B[i] = \max(\sum_{j=0}^i A[j], \sum_{j=i+1}^{n-1} A[j])$ .

You can use pseudocode to describe the algorithm. (You do *not* need to justify that your algorithm works, just give it.)

**Solution:** Here is the algorithm:

1. Construct an array  $C$  of length  $n - 1$ , such that  $C[i] = \sum_{j=0}^i A[j]$  for  $i = 0, \dots, n - 2$ . Do this as follows:
  - 1.1 Initialize a variable  $L = 0$
  - 1.1 For each  $i = 0, \dots, n - 2$  add  $A[i]$  to  $L$  and store the result in  $C[i]$ .
2. Construct an array  $D$  of length  $n - 1$  such that  $D[i] = \sum_{j=i+1}^{n-1} A[j]$  for  $i = 0, \dots, n - 2$ . Do this as follows:
  - 1.1 Initialize a variable  $U = 0$

1.1 For each  $i = n - 2, \dots, 0$  add  $A[i + 1]$  to  $U$  and store the result in  $D[i]$ .

3. Construct the desired array  $B$  by storing in  $B[i]$  the largest of the two values  $C[i]$  and  $D[i]$ , for each  $i = 0, \dots, n - 2$ .

(Although this is not required as part of the solution we note that this algorithm runs in time  $O(n)$  because each of the 3 phases runs in  $O(n)$ . Indeed each phase consists of an iteration of  $O(n)$  steps with constant amount of work done at each iteration.)

## 5

Consider the following statement:

“if  $f(n)$  and  $g(n)$  are positive functions and  $f(n)$  is  $O(g(n))$  then  $n^{f(n)}$  is  $O(n^{g(n)})$ ”.

If the statement is true, prove it. If the statement is false, give a counterexample.

**Solution:** *Answer:* The statement is FALSE. Here is a counterexample:

$$f(n) = 2 \quad g(n) = 1$$

$2$  is  $O(1)$  but  $n^2$  is not  $O(n)$ .

## 6

By adding to the code fragment below, describe in Java a method for multiplying an  $n \times m$  matrix  $A$  and an  $m \times p$  matrix  $B$ . Then, give a Big-oh upper bound on the running time of your method in terms of  $n, m, p$ .

```
double[] matmult(int n, int m, int p, double A[][], double B[][]) {  
    ...  
}
```

**Solution:** The running time is  $O(npm)$ .

## 7

Define the function  $f(n)$  as follows:

$$f(n) = 2^n \text{ when } n \text{ is odd}$$
$$f(n) = 2^{n/2} \text{ when } n \text{ is even}$$

$2^n$  is  $O(f(n))$ , true or false. Prove your answer.

**Solution:** The answer is: FALSE. We prove it by contradiction. Assume that there is a constant  $c$  and a number  $N$  such that

$$2^n \leq cf(n) \text{ for all } n \geq N$$

Choose an  $n_0$  that is an even number and is greater than  $N$  and is also greater than  $2 \log c$ . From  $n_0 > 2 \log c$  it follows that  $c < 2^{\frac{n_0}{2}}$ . Then  $cf(n_0) = c2^{\frac{n_0}{2}} < 2^{\frac{n_0}{2}} \cdot 2^{\frac{n_0}{2}} = 2^{n_0}$ . Which is a contradiction.

## 8

For each statement below, decide whether it is **true** or **false**. Attach a *very brief* explanation.

1. Suppose program  $A$  runs in time  $\Theta(n^2)$  and program  $B$  runs in time  $\Theta(n^3)$ . Then there is no input for which program  $B$  runs faster than program  $A$ , true or false?

**Solution:** Answer: FALSE. The inequalities in the asymptotic bounds only hold for large enough input size. There may be an input of smaller size for which  $B$  runs faster than  $A$ .

2. Suppose we decide to make our JAVA model of computation more realistic by dividing JAVA instructions into *easy* and *hard* instructions where the easy ones cost 1 step and the hard ones cost 10 steps. A program can have a different asymptotic complexity in this new model of computation from its complexity in the original model, true or false?

**Solution:** FALSE. This only changes the running time by some constant factor.

3.  $\gcd(x, y) = \gcd(x - y, y)$  for all pairs of integers  $x$  and  $y$ , true or false?

**Solution:** TRUE. A number  $d$  that divides both  $a$  and  $b$  also divides  $a - b$  and  $a + b$ . Hence the set of all common divisors of  $x$  and  $y$  equals the set of all common divisors of  $x - y$  and  $y$ . When  $x = y = 0$ , both gcds are undefined.

4. Theoretical analysis of running time is useful in distinguishing between programs that are similar in their efficiency while empirical measurements can be used to distinguish between programs with widely different running times, true or false?

**Solution:** FALSE. It is the opposite that is true since the theoretical analysis ignores the distinctions caused by constant factors.

## 9

Prove that if  $f(n)$  is  $O(g(n))$  then  $1000f(n)$  is  $O(0.001g(n))$ .

**Solution:** We are given that there exists a  $c > 0$  and an  $N$  such that for all  $n \geq N$

$$f(n) \leq cg(n)$$

Then

$$1000f(n) \leq c'(0.001g(n))$$

so we can take  $c' = 1000c/0.001 = 1000000c$  and  $N' = N$ .

## 10

Consider the following Java code fragment

```
static void swap( int[] a, int i, int j) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

```
static int foo( int j, int i) {
    int r = 1;
    for (int k = j; k < i; k++)
        r = (r * k) % 100 ;
    return r;
}
```

```
static void bar(int n) {
    int[] a = new int[n];
    for (int i = 0; i < n; i++)
        a[ i ] = i+1;
    for (int i = 1; i < n; i++ )
        swap( a, i, foo( 0, i) );
}
```

Analyze the running time of `bar` in Big-Oh notation.

**Solution:** The running time of `foo( j, i)` is  $O(i-j)$  if  $i > j$  and  $O(1)$  (constant time) if  $i \leq j$ . Swap is  $O(1)$ . Therefore there exist constants  $c_1 > 0$  and  $c_2$  such that the running time of the second loop is less than

$$\sum_{i=1}^n (c_1 i + c_2)$$

After calculating the sum we conclude that the second loop runs in time  $O(n^2)$ . The first loop runs in time  $O(n)$  and the rest is constant time. Therefore the running time of `bar( n)` is  $O(n^2)$ .