Maggie Johnson

Handout #16a
CS109

Problem Set #4 Solutions

Rosen, Section 1.8

#8

a) O(x^4): n=3 is too small since log x grows without bound as x increases. But it does grow smaller than x so n=4 is the best answer.
b) n = 5
c) For large x, this fraction gets close to 1 (divide numerator and denominator by x^4). so n = 0 and we get O(1).
d) Similar to c, but this time n = -1 since for large x, f(x) is approximately = to 1/x.

#32 Give a big-oh estimate of the product of the first n odd positive integers.

The nth odd positive integer = 2n-1. Thus each of the first n odd positive integers is at most 2n. So their product is at most $(2n)^n$ giving us $O((2n)^n)$.

We also accepted any other *legal* tighter bounds as answers.

#38 The definition of "f(x) is Θ(g(x))" is that f(x) is both O(g(x)) and Ω(g(x)). That means that there exists positive constants C1, k1, C2, k2 such that |f(x)| <= C2 |g(x)| for all x > k2 and |f(x)| >= C1 |g(x)| for all x > k1. Similarly, we have the positive constants C1', k1', C2', k2' such that |g(x)| <= C2' |h(x)| for all x > k2' and |g(x)| >= C1' |h(x)| for all x > k1'. We can combine these inequalities to obtain |f(x)| <= C2 C2' |h(x)| for all x > max(k2, k2'), and |f(x)| >= C1 C1' |h(x)| for all x > max(k1, k1'). This means that f(x) is Θ(h(x)).

1) In pseudocode:

```
i = m;                                  1
j = n;                                  1
pair = (0,0);                           1
repeat until i = 0 or j = 0 {           m+n (worst case)
        if ui + dj = C then             m+n
                pair = (i, j);
                exit;
        else if ui + dj < C then        m+n
                j = j - 1;              worst case: m times
        else if ui + dj > C then        m+n
                i = i - 1;              worst case: n times
```

// Indices of the summands for C will be in "pair" or (0,0) if C is not obtainable.

Note: The conditions of the if-statement will execute each time through the loop, but only one set of statements inside will execute each time, with worst case being m+n (if C is not obtainable): 5(m+n) + 3, which is O(m+n) and is linear.

2) $n^3 + n^2(n+1) + n(n+1) + (n+1) = O(n^3)$

3) Note that the value of bar(n,n) = $(n^2 + 3n)/2$. The function bar takes O(n) time and line (2) of foo takes O(n) time. The for-loop of lines (1)-(2) is iterated $(n^2 + 3n)/2$. The evaluation of bar(n,n) at line (1) takes O(n) time. Thus, the procedure foo now takes $O(n^3)$ time (using the product rule). The running time of main is dominated by the running time of foo and thus main also takes $O(n^3)$ time.

4) P(n): $s_n =$

$2^{((n+1)/2)}$	if n is odd
$2^{(n/2)}$	if n is even

proof by strong induction:

P(n) denotes that: $s_n =$

$2^{((n+1)/2)}$	if n is odd
$2^{(n/2)}$	if n is even

i) base case: Prove P(0) and P(1) are true: if n = 0, $2^0 = 1$; if n = 1, $2^{(2/2)} = 2$
ii) induction: assume that the formula holds all positive integers i < k;
show that it holds for k.

$s_i =$	$2^{((i+1)/2)}$	if n is odd
	$2^{(i/2)}$	if n is even

$s_k =$	$2 * s_{k-2}$		given
	$2 * 2^{(((k-2)+1)/2)}$	if k-2 is odd	substitution of inductive hypothesis:
	$2 * 2^{((k-2)/2)}$	if k-2 is even	k-2 is less than k
	$2^{((k-1)/2 + 1)}$	if k-2 is odd	definition of exponents & algebra
	$2^{((k-2)/2 + 1)}$	if k-2 is even	
	$2^{((k-1)/2 + 1)}$	if k is odd	definition of odd/even; if k is odd so is
	$2^{((k-2)/2 + 1)}$	if k is even	k-2; if k is even so is k-2
	$2^{((k+1)/2)}$	if k is odd	algebra
	$2^{(k/2)}$	if k is even	

By the principle of strong mathematical induction, P(n) is true for all n >= 0.

5) Discover and prove a formula for

$$\sum_{k=1}^{n} k^2$$

$(2n^3 + 3n^2 + n) / 6$ Taken from Handout #9

P(n): A closed form formula for the summation above is $(2n^3 + 3n^2 + n) / 6$

base case: $1^2 = (2 * 1 + 3 * 1 + 1) / 6 = 1$
inductive hypothesis: Assume P(n) is true, prove it is true for P(n+1), i.e.,

$$\sum_{k=1}^{n+1} k^2 = (2(n+1)^3 + 3(n+1)^2 + (n+1)) / 6$$

Proof:

$$\sum_{k=1}^{n} k^2 + (n+1)^2 = (2n^3 + 3n^2 + n) / 6 + (n+1)^2$$

$$\begin{aligned}\sum_{k=1}^{n+1} k^2 &= (2n^3 + 3n^2 + n) / 6 + (n+1)^2 \\ &= (2n^3 + 3n^2 + n) / 6 + 6(n^2 + 2n + 1) / 6 \\ &= (2n^3 + 9n^2 + 13n + 6) / 6 \\ &= (2(n+1)^3 + 3(n+1)^2 + (n+1)) / 6\end{aligned}$$

By the principle of mathematical induction, P(n) is true for n.

6) $u_n = 1 + 3 + 3^3 + 3^4 + \ldots + 3^n = (3^{(n+1)} - 1) / 2$ (geometric series)

7a) We assume that i > j in the following function:

```
int gcd(int i, int j) {
        int r;

        r = i % j;
        if (r != 0)
                gcd(j, r);
        else
                return j;
}
```

7b) Let T(i) be the running time of gcd(i,j). Suppose gcd(i,j) calls gcd(j,m) which calls gcd(m,n). We shall show that m <=i/2. There are two cases:

1) j <= i/2 then m <= j <= i/2.
2) j > i/2, then m = i % j = i-j < i/2

Thus, we conclude that after every two calls to gcd, the first argument is reduced by at least half. If we substitute the text of gcd for one invocation of the recursive call, we can model the running time of gcd by the recurrence:

T(i) <= O(1) + T(i/2)

The solution to this recurrence is $O(\log i)$.