Midterm 1

CS 141 - Intermediate Data Structures and Algorithms

October 23, 2013

By taking this exam, I affirm that all work is entirely my own. I understand what constitutes cheating, and that if I cheat I may be expelled from UC Riverside.

Signature:

Printed Name:

---

1. (5pt) For each problem below, circle the expression that is asymptotically larger (for example, using Θ notation). If they are the same, then circle equal. **Each correct answer results in +1 point, each incorrect answer in -1 point, and each blank answer in 0 points.**

(i)	$n \log^2 n$	$n \log^3 n$	equal
(ii)	$n!$	2^n	equal
(iii)	$(\log n)^{\log n} n$	$\frac{n}{\log n}$	equal
(iv)	$n^{0.5}$	$5^{\log_2 n}$	equal
(v)	$\sum_{i=1}^{n} i^k$	n^{k+1}	equal

2. (5pt) For each statement below, circle whether it is true or false. **Each correct answer results in +1 point, each incorrect answer in -1 point, and each blank answer in 0 points.**

(i)	True	False	Strassen's algorithm is the asymptotically fastest known method for matrix multiplication.
(ii)	True	False	If a and b are even integers, then $\gcd(a, b) = 2\gcd(a/2, b/2)$
(iii)	True	False	If a and b are relatively prime, then $\gcd(a, b) = 1$.
(iv)	True	False	0 divides every number.
(v)	True	False	a mod b can be calulated in time $O(n^4)$, where n is the number of bits in both a and b.

3. (10pt) For each function below, give a recurrence equation for the number $T(n)$ of letters it prints and the asymptotic value (using big-O notation) of $T(n)$. **You do not need to show your work.**

```
function PrintXs (n : integer)
 // assume n is a power of 2
 if n > 1
    for i ← 1 to 3n do print("X")
    PrintXs(n/2)
    PrintXs(n/2)
```

Recurrence equation:

$T(n) =$

Solution:

$T(n) =$

```
function PrintYs (n : integer)
 // assume n is a power of 2
 if n > 1
    for i ← 1 to 3n do print("Y")
    PrintYs(n/2)
```

Recurrence equation:

$T(n) =$

Solution:

$T(n) =$

```
function PrintZs (n : integer)
 // assume n is a power of 2
 if n > 1
    for i ← 1 to 3n do print("Z")
    PrintZs(n/2)
    PrintZs(n/2)
    PrintZs(n/2)
```

Recurrence equation:

$T(n) =$

Solution:

$T(n) =$

4. (10pt) Give pseudocode for the fast modular exponentiation algorithm.

   modexp (x, y, N):

5. (15 pt) In each row of the table below you are given three parameters of the RSA crypto-system: p, q, and e. For each row, determine whether these parameters are correct. If they are correct, in the last two columns choose the correct values of the public key (N, e) and secret key d. If they are not correct, indicate why.

To discourage guessing, if you select the wrong public key or secret key, then -1 points will be given. If you do not answer, then 0 points will be given.

p	q	e	Correct? If not, why?	Public key	Secret Key
7	13	5		$(91, 5)$ $(72, 5)$ $(91, 7)$	73 23 29
11	11	7		$(121, 7)$ $(100, 7)$ $(11, 7)$	7 43 3
5	11	7		$(55, 11)$ $(55, 7)$ $(40, 7)$	23 11 8
5	21	11		$(105, 3)$ $(105, 11)$ $(80, 11)$	86 51 17
5	13	3		$(115, 3)$ $(115, 13)$ $(48, 13)$	77 31 37

6. (5pt) For his private RSA key, Professor Bo Zo chooses two primes p and q, where p has 1000 bits, but q has only 20 bits. Given just his public key (N, e), describe how you could compute Bo Zo's private key d in just a few minutes.

7. (5pt) Professor Bo Zo chooses new RSA keys by the standard method and reveals his public key (N, e). But then he teases his class by revealing the sum $p + q$ of the two prime factors of N. Describe an efficient algorithm for calculating Bo Zo's private key d.

8. (15pt) Suppose you have k sorted arrays, each with n elements. Give an efficient divide and conquer algorithm for merging these arrays into a new sorted array with nk elements.

   Note: If you use any helper functions, you must explicitly define them.

```
function MERGEK(list of vectors L)
    // base case
    if number of vectors = 1 then
        return L[0]
    end if
    // recursion
    let L1 = first half of L
    let L2 = second half of L
    return merge2(mergek(L1), mergek(L2))
end function
```

```
function MERGE2(arrays A1, A2)
    Let i,j,k = 0
    Let R = array of size A1 + size A2
    while i < size of A1 and j < size of A2 do
        if A1[i] < A2[i] then
            R[k] = A1[i]
            i++
        else
            R[k] = A2[j]
            j++
        end if
        k++
    end while
    while i < size of A1 do
        R[k]=A1[i]
        i++
    end while
    while j < size of A2 do
        R[k]=A2[j]
        j++
    end while
    return R
end function
```

9. (15pt) You are given two sorted vectors A and B. Write an algorithm that computes the kth largest element in the union of A and B. For example, if $A = [1, 2, 3, 4, 5, 6, 7, 8]$, $B = [0, 3, 9]$, and $k = 2$, then your algorithm should return the number 1. It must run in time $O(\log |A| + \log |B|)$.

   Note: If you use any helper functions, you must explicitly define them.

```
function FindK(vectors A,B; integer k)
    // base case
    if A[k/2] = B[k/2] then
        return A[k/2]
    end if
    // recursion
    if A[k/2] < B[k/2] then
        let A' = A[k/2..k]
        let B' = B[0..k/2]
        return FindK(A',B')
    else
        let A' = A[0..k/2]
        let B' = B[k/2..k]
        return FindK(A',B')
    end if
end function
```

10. (15pt) You are given a monotonic increasing function f. This means that f obeys the property:

$$x_1 > x_2 \implies f(x_1) \geq f(x_2)$$

At some point a, $f(a) = 10$. Write an efficient algorithm that calculates $\lfloor a \rfloor$. For example, if the input function is $f(x) = 4x$, then $f(2.5) = 10$, so your function should return the number 2.

Hint: First figure out how you could solve the problem if you already knew numbers x and y such that $x \leq a \leq y$. Then, figure out how to find the bounds x and y.

Note: If you use any helper functions, you must explicitly define them.

```
function FindBounds(f)
    x ← −1
    while f(x) > 10 do
        x ← 2x
    end while
    y ← 1
    while f(y) < 10 do
        y ← 2y
    end while
    return (x, y)
end function

function BinarySearch(f,x,y)
    // base case
    if ⌊x⌋ = ⌊y⌋ then
        return ⌊x⌋
    end if
    // recursion
    let m = (x+y)/2
    if f(m) < 10 then
        return BinarySearch(f, x, m)
    else
        return BinarySearch(f, m, y)
    end if
end function

function FindA(f)
    let (x, y) = FindBounds(f)
    return BinarySearch(f,x,y)
end function
```