CS 610-101, Fall 2015, Midterm Exam
Prof. David Nassimi

Exam #: _____
(Time: 2 Hours)

Name: Solution
First Last

High = 100
MEDIAN = 67

	GRADE	
1		/ 10
2		/ 15
3		/ 15
4		/ 15
5		/ 15
6		/ 15
7		/ 15
SUM		/100

DIST:

90's	5
80's	6
70's	5
60's	9
50's	4
40's	3
30's	3
<30	3

1. Prove the following sum is $\Theta(n^2 \log n)$.

$\Sigma_{i=1}^{n}(i \log i)$

(a) Prove $O(n^2 \log n)$

$$\sum_{i=1}^{n} i \log i \le \sum_{i=1}^{n} n \log n$$

$$\le n^2 \log n$$

$$= O(n^2 \log n)$$

(b) Prove $\Omega(n^2 \log n)$

$$\sum_{i=1}^{n} i \log i \ge \sum_{i=\lceil \frac{n}{2} \rceil}^{n} i \log i \ge \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \lceil \tfrac{n}{2} \rceil \log \lceil \tfrac{n}{2} \rceil$$

$$\ge \lceil \tfrac{n}{2} \rceil^2 \log \lceil \tfrac{n}{2} \rceil$$

$$\ge \left(\tfrac{n}{2}\right)^2 \left(\log \tfrac{n}{2}\right) \ge \left(\tfrac{n}{2}\right)^2 (\log n - 1)$$

$$= \Omega(n^2 \log n)$$

2. (a) Prove by induction that the solution of the following recurrence equation is $T(n) = An^2+Bn+C$, and determine the constants A, B, C. Assume n is a power of 2.

$$T(n) = \begin{cases} 4T(n/2)+n+6, & n \geq 2 \\ 2, & n = 1. \end{cases}$$

Base, $n=1$: $T(1) = 2 = A+B+C$

Hypo: For $n \geq 2$, suppose

$$T\left(\frac{n}{2}\right) = A\left(\frac{n}{2}\right)^2 + B\left(\frac{n}{2}\right) + C$$

Then,

$$T(n) = 4\left[A\frac{n^2}{4} + B\frac{n}{2} + C\right] + n + 6$$

$$= An^2 + (2B+1)n + (4C+6)$$

$$= An^2 + Bn + C$$

Need $2B+1 = B$, $4C+6 = C$

$B=-1$ $C=-2$ $A=5$ $\therefore T(n) = 5n^2 - n - 2$

(b) Use master theorem to find the exact solution of the following recurrence. Assume n is a power of 2.

$$T(n) = \begin{cases} 8T(n/2)+n, & n \geq 2 \\ 1, & n = 1. \end{cases}$$

$a=8$, $b=2$, $\alpha=1$

$h = \lg_2 8 = 3$

$$\therefore \boxed{T(n) = An^3 + Bn}$$

$n=1$: $T(1) = 1 = A+B$

$n=2$: $T(2) = 8T(1)+2 = 10 = 8A+2B$

$A = 4/3$, $B = -1/3$

$$\therefore T(n) = \frac{4}{3}n^3 - \frac{1}{3}n$$

3. Find the solution of the following recurrence equation by repeated substitution, assuming $n = 2^k$ for some integer k.

$$T(n) = \begin{cases} 4T(n/2) + n, & n \geq 2 \\ 1, & n = 1. \end{cases}$$

$$T(n) = n + 4T\left(\frac{n}{2}\right)$$

$$= n + 4\left[\frac{n}{2} + 4T\left(\frac{n}{4}\right)\right]$$

$$= n + 2n + 4^2 T\left(\frac{n}{2^2}\right)$$

$$= n + 2n + 4^2\left[\frac{n}{2^2} + 4T\left(\frac{n}{2^3}\right)\right]$$

$$= n + 2^1 n + 2^2 n + 4^3 T\left(\frac{n}{2^3}\right)$$

$$\vdots$$

$$= n\underbrace{\left(1 + 2 + 4 + \cdots + 2^{k-1}\right)}_{\substack{2^k - 1 \\ n-1}} + 4^k \cancel{T\left(\frac{n}{2^k}\right)}$$

$T(1) = 1$

$(2^k)^2 = n^2$

$$T(n) = n(n-1) + n^2$$

$$= n^2 - n + n^2$$

$$T(n) = 2n^2 - n$$

4. Given a sorted array of n elements. We want to determine if there is a pair of elements $A[i]$ and $A[j]$, $i \neq j$, such that $A[i] = -A[j]$. (We want a Boolean function which returns a TRUE/FALSE answer.)

5

(a) Outline how this problem may be solved by using the binary-search algorithm, $BS(A, left, right, key)$. (Do not give the code for BS. It is a given function, which you call.) Analyze the time complexity of this approach.

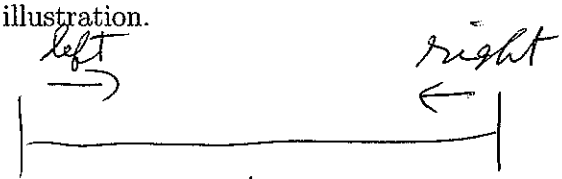
```
For i = 0 to n-2
  { BS(A, i+1, n-1, -A[i])
    if found return (TRUE)
  }
return (False)
```

complexity: $O(n \log n)$

10

(b) Describe a more efficient $O(n)$ algorithm to solve this problem. Give the pseudocode. Explain how the algorithm works, and provide a numerical illustration.

We do inward scan.




```
If (A[left] + A[right] > 0)  right--
else if (A[left] + A[right] < 0)  left++;
```

```
l = 0;  r = n-1
While (l < r)
  { if (A[l] + A[r]) > 0)  r = r-1;
    else if (A[l] + A[r] < 0)  l = l+1;
    else return (True)
  }
return (false)
```

CS 610-101, Fall 2015, Midterm Exam Name: Solution

5. This problem deals with computing x^n, where x is *real* and n is an integer, by repeated multiplications.

(a) It is possible to perform this computation with $O(\log n)$ multiplications. Write an efficient recursive algorithm (pseudocode), POWER(x, n), to compute x^n.

10

```
real POWER (real x, int n)
{ if (n==1) return x;
  T = POWER(x, ⌊n/2⌋);
  T = T * T;
  if (n is ODD) T = T * x;
  return (T) }
```

(b) Let $f(n)$ be the worst-case number of multiplications used by your algorithm. Write a recurrence for $f(n)$ and find the solution.

5

$$f(n) = \begin{cases} 0 & \text{if } n = 1 \\ f(\lfloor \frac{n}{2} \rfloor) + 2, & n \geq 2 \end{cases}$$

$$f(n) = 2 \lfloor \log n \rfloor$$

Easy to prove by Induction.

6. Consider the algorithm BUILD-HEAP, which starts with an array $A[1..n]$ of n random elements and builds a valid min-heap of n elements.

(a) The following algorithm is an inefficient way of building the heap, by a series of $n-1$ insert operations. Analyze the time complexity of of this algorithm.

5

for $k = 2$ to n
Insert $A[k]$ into the heap $A[1..k-1]$ and make a heap with one more element $A[1..k]$.

Complexity: $\sum_{k=2}^{n} \log k = \Theta(n \log n)$

(b) Now give a more efficient recursive algorithm (pseudocode) for BUILD-HEAP based on divide-and-conquer approach, using PUSHDOWN.

5

```
Build (Type A[], int r (RecT), int n)
{ if (2r > n) return;   // It is LEAF or NIL
  Build (A, 2r, n);  // Build left subtree
  Build (A, 2r+1, n); // Build right subtree
  Pushdown (A, r, n);
}
```

(c) What is the time compleity of this efficient BUILD-HEAP? Use a recurrence equation for the analysis. Use master theorem as a guide to guess the solution form. Don't bother to prove the correctness of it. Let $f(n)$ = worst case number of key cmpr

5

$$f(n) = \begin{cases} 2f\left(\frac{n}{2}\right) + 2\log n, & n > 1 \\ 0, & n = 1 \end{cases}$$

$$\boxed{f(n) = An + B\log n + C}$$

7. Starting with an empty AVL tree, insert the following sequence of elements into it. Show the result (which must be balanced) after each insertion.

$$50, 60, 70, 65, 80, 90, 62, 69$$