Michael A. Bender

CSE 373 – Analysis of Algorithms, Spring 2005

Assignment #1
Part 1 (1-8) Due Tuesday, 15th February, 2005
Part 2 (9-14) Due Tuesday, 22nd February, 2005

Each of the problems should be solved on a separate sheet of paper to facilitate grading. Please don't wait until the last minute to look at the problems.

*Part*1

Problem 1
In this problem, we will learn some techniques for summing series.
(1) Show how to compute the sum $S = 1 + 2 + 2^2 + 2^3 + 2^4 + \ldots + 2^n$.
Hint: What is $(S + 1) - 1$?
(2) Now show how to compute the sum $T = 1 + r + r^2 + r^3 + r^4 + \ldots + r^n$.
Hint: Consider the two quantities T and $r \cdot T$.
(3) Suppose $0 < r < 1$. What is T as $n \to \infty$?

Problem 2
In this problem, we will learn two techniques for summing the following series:
$Z = 1 + 2r + 3r^2 + 4r^3 + 5r^4 + \ldots + nr^{n-1}$.
(a) Compute the sum by using the fact that $Z = \frac{\partial T}{\partial r}$ That is, we obtain Z by differentiating series T (from previous problem) with respect to r.
(b) Now evaluate Z by computing $Z - r \cdot Z$. What happens ?
(c) Suppose $0 < r < 1$. What is Z as $n \to \infty$?

Problem 3
In this problem, you should approximate the sum $L = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$.
Hint: Use the fact that $\sum_{i=2}^{n} \frac{1}{i} \approx \int_{x=2}^{n} \frac{1}{x} dx$.
(a) What is L ?
(b) What is L as $n \to \infty$?

Problem 4
(a) Let $\lfloor\lfloor n \rfloor\rfloor$ to be the largest power of 2 smaller than n. Prove that $1+2+4+8+\cdots+\lfloor\lfloor n \rfloor\rfloor = O(n)$.
Note: REMEMBER THIS PROBLEM!!! You will need it in the rest of your semester.
(b) Let $\lfloor\lfloor n \rfloor\rfloor_3$ to be the largest power of 3 smaller than n. Prove that $1+3+9+27+\cdots+\lfloor\lfloor n \rfloor\rfloor_3 = O(n)$.
(c) Prove that $1 + 2 + 3 + \cdots + n = O(n^2)$.

Problem 5

Prove or find a counterexample for the following. Assume that $f(n)$ and $g(n)$ are monotonically increasing functions that are always larger than 1.

(1) $f(n) = o(g(n))$ implies $\log(f(n)) = o(\log(g(n)))$

(2) $f(n) = O(g(n))$ implies $\log(f(n)) = O(\log(g(n)))$

(3) $f(n) = o(g(n))$ implies $2^{f(n)} = o(2^{g(n)})$

(4) $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$

Problem 6 (From Brassard and Bratley)

Prove that

a. $\log_a(xy) = \log_a x + \log_a y$

b. $\log_a x^y = y \log_a x$

c. $\log_a x = \frac{\log_b x}{\log_b a}$

d. $x^{\log_b y} = y^{\log_b x}$.

Problem 7 (From Brassard and Bratley)

Explain why

$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$ for every real number x.

Problem 8 (Extra Credit)

n people wish to cross a bridge at night. A group of at most two people may cross at any time, and each group must have a flashlight. Only one flashlight is available among the n people, so some sort of shuttle arrangement must be arranged in order to return the flashlight so that more people may cross. Each person has a different crossing speed; the speed of a group is determined by the speed of the slower member (or either if they are of the same speed.) Your job is to determine a strategy that gets all n people across the bridge in the minimum time. Provide the total number of seconds it takes to get everyone across, as well as the groups that cross, in order. (There may be multiple optimal solutions, in that case, any of them are fine.)

The speed of a person is given in the number of seconds it would take him to cross. (If he is the slower member of the group.)

Speeds:

1: 1

2: 2

3: 5

4: 10

Part2

Problem 9

For the following C codes, find the time-complexity by first finding the recurrence relation and then simplifying the recurrence.

(1)
```
int blah(int n)
{
int sum = 0;
int i, j;
if (n == 0)
    return 1;
for(i=0; i<=n-1; i++)
    for(j=0; j<=log(n); j++)
    {
          ⋮
       loop-body
          ⋮
    }
sum = blah(n/2);
sum += blah(n/2);
return sum;
}
```

(2)
```
int blah(int n)
{
int sum = 0;
int i, j;
if (n == 0)
    return 7;
for(i=0; i<=n-1; i++)
    sum += blah(i);
return sum;
}
```

Problem 10

Suppose that you have an array of N elements, where each element is at most a distance of K from its final sorted position. Show that you can sort these numbers in $O(N \log K)$ time.

Problem 11

Consider the following algorithm for computing a^x given a and x:

function power (a, x)
    if ($x == 0$) then return (1)

if (x is even) then return $(power(a, \lfloor x/2 \rfloor))^2$
else return $(a * (power(a, \lfloor x/2 \rfloor))^2)$

a. What is the running time of this algorithm? ____________________
b. What is the recurrence relation for the running time? ____________________

Problem 12
Sort the following functions in increasing orders of growth using $o(f(n))$.

Example: $n^2 = \Theta(2n^2) = o(n^3) = \Theta(n^3 + 2n^2)$

$f_1(n) = n$
$f_2(n) = n^2 \lg(n)$
$f_3(n) = 3^n$
$f_4(n) = \lg(n)$
$f_5(n) = n^3$
$f_6(n) = \lg^{10}(n)$
$f_7(n) = n^2$
$f_8(n) = n^n$
$f_9(n) = n \lg(n)$
$f_{10}(n) = n^2 \lg^{100}(n)$
$f_{11}(n) = n^{0.01}$
$f_{12}(n) = n^{2.01}$
$f_{13}(n) = n!$
$f_{14}(n) = 2^n$

Problem 13 (Extra Credit)
Show that the product $(a + bi)(c + di)$ of two complex numbers can be evaluated using just three real-number multiplications. You may use a few extra additions.

Problem 14 (Extra Credit)
You have been charged with finding the 3 fastest horses out of a group of 25. You also need to know which is first, second, and third. You know that any given horse always finishes a race in exactly the same amount of time. You also know that the horses' times are unique (i.e., no two horses ever tie.) You have a racetrack in which you can race up to 5 horses per race.

The problem: You are charged 25 dollars per race, and you do not have a stopwatch. You are only able to record the relative positions for each race. (Which horse came in first, second, third, fourth, or fifth.) Find an arrangement of races in which the cost out of your pocket is minimized, and where you *always* succeed in finding the first, second, and third fastest horses.