

## CSCI 303 Homework 2

### Problem 1 (4.3-2):

The recurrence  $T(n) = 7T(n/2) + n^2$  describes the running time of an algorithm  $A$ . A competing algorithm  $A'$  has a running time of  $T'(n) = aT'(n/4) + n^2$ . What is the largest integer value for  $a$  such that  $A'$  is asymptotically faster than  $A$ ?

### Problem 2 (Derived from 4-1 and 4-4):

Give asymptotic upper and lower bounds for  $T(n)$  in each of the following recurrences. Make your bounds as tight as possible, and justify your answers.

a.  $T(n) = 2T(n/2) + n^3$ .

b.  $T(n) = 16T(n/4) + n^2$ .

c.  $T(n) = 7T(n/3) + n^2$ .

d.  $T(n) = 7T(n/2) + n^2$ .

e.  $T(n) = 2T(n/4) + \sqrt{n}$ .

f.  $T(n) = 3T(n/2) + n \lg n$ .

g.  $T(n) = 4T(n/2) + n^2 \sqrt{n}$ .

h.  $T(n) = T(9n/10) + n$ .

### Problem 3 (Derived from 4-1 and 4-4):

Give asymptotic upper and lower bounds for  $T(n)$  in each of the following recurrences. Make your bounds as tight as possible, and justify your answers. You may assume that  $T(1)$  is a constant.

a.  $T(n) = T(n - 1) + n$ .

b.  $T(n) = T(n - 1) + 1/n$ .

c.  $T(n) = T(n - 1) + \lg n$ .

d.  $T(n) = 2T(n/2) + n \lg n$ .

e.  $T(n) = 5T(n/5) + n/\lg n$ .

### Problem 4 (Not in book):

The following algorithm uses a divide-and-conquer strategy to search an unsorted list of numbers. Given a list of numbers  $A$  and a target number  $t$ , the algorithm returns 1 if  $t$  is in the list, and 0 otherwise.

```
UNSORTED-SEARCH( $A, t, p, q$ )
if  $q - p < 1$ 
    if  $A[p] = t$  return 1 else return 0
if UNSORTED-SEARCH( $A, t, p, \lfloor \frac{p+q}{2} \rfloor$ ) = 1 return 1
if UNSORTED-SEARCH( $A, t, \lfloor \frac{p+q}{2} \rfloor + 1, q$ ) = 1 return 1
return 0
```

Analyze this algorithm with respect to worst-case asymptotic complexity, and give the worst-case running time in terms of  $\Theta$  notation. How does this algorithm compare to the naive algorithm that simply iterates through the list to look for the target?

**Problem 5 (28.2-4):**

V. Pan has discovered a way of multiplying  $68 \times 68$  matrices using 132,464 multiplications, a way of multiplying  $70 \times 70$  matrices using 143,640 multiplications, and a way of multiplying  $72 \times 72$  matrices using 155,424 multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare to Strassen's algorithm?

**Problem 6 (28.2-6):**

Show how to multiply the complex numbers  $a + bi$  and  $c + di$  using only three real multiplications. The algorithm should take  $a$ ,  $b$ ,  $c$ , and  $d$  as input and produce the real component  $ac - bd$  and the imaginary component  $ad + bc$  separately.