SOLUTIONS

COMP3270-002 Algorithms, First Midterm

Fall Semester, 2014

Directions The test is open book and open notes, but NOT open phone or open computer (e-reader or computer used solely as e-reader is ok). For each problem, show your work completely. Give reasons for all answers – this is how I give partial credit. **Each part of each question is worth 10 points, 100 total points)**

1. Suppose an algorithm similar to mergesort is developed, where the sorting works by making two recursive calls on arrays of size $2n/3$, but then requires only $O(\sqrt{n})$ work to combine the two sorted arrays. Give a recurrence relation and as tight a bound as possible on running time of that algorithm.

$$T(n) = 2T(2n/3) + \Theta(n^{0.5})$$

Use Master Method $a=2,\ b=3/2,\ f(n)=n^{0.5}$

$n^{\log_b a} = n^{\log_{3/2} 2} = n^{(\ln 2/\ln 1.5)} = n^{1.71}$

case 1: $f(n) = O(n^{\log_b a - \varepsilon})$ for $\varepsilon < 1.21$

$$\boxed{T(n) = \Theta(n^{\log_{3/2} 2})}$$

2. For each of the following recurrence relations, give as tight a bound as possible on the running time of an algorithm whose running time $T(n)$ is described by the recurrence relations. Use the Master Method.

   (a) $T(n) = 2T(n/4) + O(1)$

$a=2,\ b=4,\ f(n)=1$

$f(n) = 1 = O(n^{\log_b a}) = O(n^{\log_4 2}) = O(n^{0.5})$ (at $\varepsilon < 0.5$)

case 1: $\boxed{T(n) = \Theta(n^{0.5})}$

$a=5 \quad b=2 \quad f(n)=n\lg n$

$n^{\log_b a} = n^{\log_2 5} = n^{\ln 5/\ln 2} = n^{2.32}$

(b) $T(n) = 5T(n/2) + O(nlgn)$

$f(n) = n\lg n = O(n^{2.32-\varepsilon})$ for $\varepsilon < 0.32$

$\Rightarrow$ case 1

$\boxed{T(n) = O(n^{\log_2 5})}$

(c) $T(n) = 3T(n/3) + O(n)$

$a=3, \ b=3 \quad f(n)=n$

$n^{\log_b a} = n^{\log_3 3} = n$: Hence $f(n) = \Theta(n^{\log_b a})$ (case 2)

$\boxed{T(n) = \Theta(n\lg n)}$

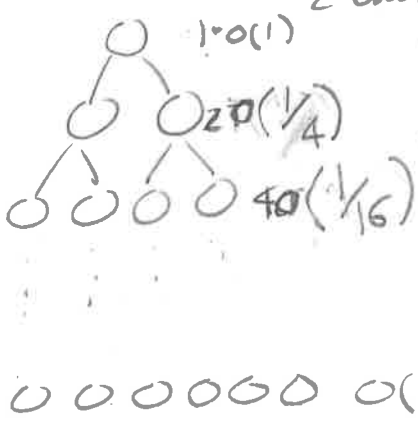
3. For part a in the previous problem, solve the recurrence using the Recursion Tree method.

2 children each level since $a=2$

level 0: $1 \cdot O(1)$

level 1: $2O(1/4)$

level 2: $4O(1/16)$



wlog let $n=4^k$

at level i you have $(1/2)^i$ work

level k: $O(1)$ ← # of leaves is 2^k

$\frac{n}{4^k} = 1 \Rightarrow n = 4^k \quad \log_4 n = k$

$\boxed{T(n) = n^{0.5}}$

$2^k = 2^{\log_4 n} \rightarrow$ take $\log_4 = \log_4 2 \log_4 n$

$n^{0.5} = \log_4 n^{0.5} = 0.5 \log_4 n = \log_4 2 \log_4 n$

same

Midterm #1, Problem #6

To show $\max(f(n), g(n)) = \Theta(f(n) + g(n))$

First upper bound, i.e. $\exists\, c > 0, n_0 \geq 0 \ni n > n_0$
$\Rightarrow$
$$0 \leq \max(f(n), g(n)) \leq c(f(n) + g(n))$$

Take $c = 1$ $n_0 = 1$

$\max(f(n), g(n))$ is always $<$ the sum since either

at any n $\max(f(n), g(n)) = f(n) \leq f(n) + g(n)$
or $\max(f(n), g(n)) = g(n) \leq f(n) + g(n)$

Now lower bound, to show $\exists\, c > 0, n_0 \geq 0 \ni$
$n > n_0 \Rightarrow c\,(f(n) + g(n)) \leq \max(f(n), g(n))$

let $c = 1/2$ we know $f(n) + g(n) \leq 2g(n)$
or $2(f(n))$ at any n
(not necessarily both)

suppose for given n $f(n) \geq g(n)$

$$\tfrac{1}{2}(f(n) + g(n)) \leq f(n) = \max(f(n), g(n))$$

suppose $g(n) \geq f(n)$
$$\tfrac{1}{2}(f(n) + g(n)) \leq g(n) = \max(f(n), g(n))$$

Hence $\max(f(n), g(n)) = \Theta(f(n) + g(n))$

4. For the following bit of pseudocode, give a worst-case big-theta running time boundary

```
FIND(A,x)
  index = 1
  while (index <= A.length)
       if A[index] == x
             return index
       else
            index = index + 1
  return -1
```

Worst case is A[A.length] == x

so you must do n iterations

(n = A.length)

$\boxed{\Theta(n)}$

5. For the previous question, give tight bounds for the average case and best case running time.

Average # of iterations $= \frac{1}{n}\sum_{i=1}^{n} i = \frac{1}{n}\frac{(n)(n+1)}{2}$

$\boxed{= \Theta(n)}$

Best case you find it first-try

$\boxed{O(1)}$

6. Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. Using the basic definition of Θ-notation, prove that $max(f(n), g(n)) = \Theta(f(n) + g(n)$ Note that $max(x)$ is defined as $f(x)$ if $f(x) > g(x)$ at that particular x, otherwise $max(x) = g(x)$ at that value of x.

see back of page

7. True or false, $n^2 - 3n + 2 = o(n^2)$. Justify your answer.

$\boxed{\text{False}}$. Since n^2-3n+2 is $\Omega(n^2)$

find $c \ni 0 \le n^2-3n+2 \le cn^2$

$$1 - 3/n + 2 \le c$$

let $n_0 = 100$

$$1 - 3/100 + 2 \le c$$

let $c = 2.97$

8. Write pseudo-code for a routine to add two matrices A and B. You can assume both matrices are the same size $n \times n$, and that you can access that size with attributes $A.rows$, $A.columns$.

```
let C be a new n×n matrix
n = A.rows
for i = 1 to n
    for j = 1 to n
        C[i][j] = a[i][j] + B[i][j]
return C
```