Problem Set 4 **CS 3851** **Name** Hudson Arney

Submission instructions: Your answers should be submitted to Canvas as a PDF.

1. Practice the "guess and check" substitution method by solving the following recurrence relation:

$T(n) - O(n) = T(n/2)$ $\quad T(n) \leq c\,O(n)$

$2\,T(n) = 2T(n/2) + O(n)$

$T(n) \leq cn^3$

a. Check if $T(n) \in O(n^3)$

a) $2c(n/2)^3 + O(n) \leq cn^3$

$(n^3/4)c + O(n) \leq cn^3$

$O(n) \leq \frac{3cn^3}{4}$

This holds true

b. Check if $T(n) \in O(n^2)$

b) $T(n) \leq cn^2$

$2c(n/2)^2 + O(n) \leq cn^2$

$\frac{cn^2}{2} + O(n) \leq cn^2$

$O(n) \leq c\frac{n^2}{2}$

This holds true

c. Check if $T(n) \in O(n)$

d. Based on your results, what is the tightest upper bound you found for T(n)?

c) $2c(n/2) + O(n) \leq cn$

$2cn + O(n) \leq cn$

$O(n) \leq -cn$ This is not possible

a) $T(n) = 2T(\frac{n}{2}) + O(n)$

$\leq 2 \cdot c \cdot \frac{n^3}{8} + c_1 \cdot n$

$\frac{cn^3}{4} + c_1 n \leq cn^3/n^3$

$\frac{c}{4} + c_1 \cdot \frac{1}{n^2} \leq c$

$\frac{c_1}{n^2} \leq \frac{3}{4}c$

d) $T(n) \in O(n^2)$ is the tightest upper bound from the sub. method

Problem Set 4 **CS 3851** **Name** Hudson Arney

Submission instructions: Your answers should be submitted to Canvas as a PDF.

1. Practice the "guess and check" substitution method by solving the following recurrence relation:

T(n) = 2T(n/2) + O(n)

a. Check if T(n) ∈ O(n^3)

b. Check if T(n) ∈ O(n^2)

c. Check if T(n) ∈ O(n)

d. Based on your results, what is the tightest upper bound you found for T(n)?

Induction Goal: $T(k) \in O(k^3)$; $k < n$

$T(n/2) \le \left(\frac{n}{2}\right)^3$

$T(n/2) \le c \cdot \frac{n^3}{8}$

Sub into eq:

$T(n) \le 2 \cdot c \cdot \frac{n^3}{8} + \underbrace{c_0 n}_{O(n)\text{ term}}$

$2c\frac{n^3}{8} + c_0 n \le cn^3$ for all $n \ge n_0$

$\frac{1}{4}c + c_0 \cdot \frac{1}{n^2} \le c$

$c_0 \frac{1}{n^2} \le \frac{3}{4}c$

$\frac{4}{3} \cdot c_0 \cdot \frac{1}{n^2} \le c$

let $n_0 = 2$:

$c \ge \frac{4}{3}c_0 \cdot \frac{1}{2^2} = \frac{c_0}{3}$

For any $c \ge \frac{c_0}{3}$ where c_0 is the bounding constant for the O(n) term, the induction goal holds for $n_0 = 2$

IG: $T(n) \le cn^2$

IH: $T(\frac{n}{2}) \in O(n^2)$

b) $T(n) = 2T(\frac{n}{2}) + O(n)$

$\le 2(c(\frac{n}{2})^2) + c_1 n \rightarrow$ plug $T(\frac{n}{2})$ for $T(n)$

$= \frac{2(c\frac{n^2}{4}) + c_1 n}{n^2} \le \frac{cn^2}{n^2}$

$= \frac{c_1}{n} \le \frac{c}{2} \Rightarrow \frac{1}{4} \le \frac{1}{2}$

$n_0 = 4$

$c = 1$

$c_0 = 1$

$T(n) \in O(n^2)$

c) $k = \frac{n}{2}$, Induction Hypothesis: $T(\frac{n}{2}) \le c(\frac{n}{2})$ for all $k < n$

$cn + c_1 n \le cn$

$c_1 n \le 0$

but $c_1 > 0$ ∴ breaks rule and $T(n) \notin O(n)$

2. For each of the following recurrence relations, determine which of the three cases of the Master's theorem applies.

a. $T(n) = \underset{a}{2}T(n/\underset{b}{2}) + \underset{f(n)}{\Theta(n)}$

b. $T(n) = 8T(n/2) + \Theta(n^2)$

c. $T(n) = 3T(n/4) + n \log n$

$f(n) = O(n)$
$T(n) = \Theta(n \log n)$

a) $\Theta(n) \equiv n^{\log_2 2} = n$ = Case 2 of master theorem

b) $\Theta(n^2) \equiv n^{\log_2 8} = n^3 \rightarrow n^3$ dominates $f(n) = \Theta(n^2)$ therefore it is Case 1 of the master theorem

$\hookrightarrow \Theta(n^2) \in O(n^{3-1}) \Rightarrow \Theta(n^2) \in O(n^2) \Rightarrow T(n) = \Theta(n^3)$

c) $n \log n \equiv n^{\log_4 3}$
$\downarrow$
$\log_4 3 \log(n)$

$f(n) = n \log n$ dominates $\log_4 3 \log(n)$, therefore it is case 3 of the master theorem

3. For each of the following recurrence relations: (1) find a reasonably tight bound in terms of big-o using the substitution method and (2) check your solution by also solving the recurrence relation with the Master method. Show all work.

a) T(n) = 2T(n/2) + n

$2c(n/2)^2 + O(n) \le cn^2$

1) $T(n) \in O(n^2)$

$\frac{cn^2}{2} + O(n) \le cn^2$

$T(n) \le cn^2$

$O(n) \le c\frac{n^2}{2}$ ✓

2) $n \equiv n^{\log_2 2}$

$n = n$, Case 2 M.T. ✓

$T(n) = \theta(n^{\log_2 2} \cdot \log n) = \boxed{\theta(n \log n)}$

b) T(n) = 4T(n/4) + 1

$T(n) \in O(n^2)$

$4(c(n/4)^2) + 1 \le n^2 c$

$T(n) \le n^2 c$

$\frac{n^2}{4}c + 1 \le n^2 c$

$1 \le \frac{3n^2}{4}c$ ✓

$1 \equiv n^{\log_4 4}$

$1 \le n$, Case 1 M.T. ✓

$\boxed{T(n) = \theta(n \log n)}$

c) T (n) = 9T(n/3) + n

$n \equiv n^{\log_3 9}$

$T(n) \in O(n^2)$

$9^{\log_3 n} T\left(\frac{n}{3^{\log_3 n}}\right) + n \log_3 n$

$n \le n^2$, Case 1 M.T.

$9^i T\left(\frac{n}{3^i}\right) + i(n)$

$9(c(n/3)^2) + n \le cn^2$

$f(n) = O(n^{2-\varepsilon})$

$\boxed{T(n) = \theta(n^2)}$

$\log_3 n = i$

$cn^2 + n \le cn^2$

$T(n) = \theta(n^{\log_3 9})$

$c + \frac{1}{n} \le c$

$c \le c - \frac{1}{n}$ ✓

d) T(n) = 3T(n/4) + n log n

$n \log n \equiv n^{\log_4 3}$

$n \log(n) \equiv \log_4 3 \log(n)$

$n \ge \log_4 3$

$T(n) \in O(n \log n)$

$3(c\frac{n}{4}\log(n/4)) + n\log n \le nc\log n$

$\frac{3n}{4}c\log\left(\frac{n}{4}\right) + n\log n \le n c \log n$

$3/4\, n\log(n/4) \le c f(n)$ True, if $c = 3/4$

$\frac{3}{4}nc(\log n - \log 4) + n\log n \le nc\log n$

$3/4\, n\log(n/4) \le c n\log n$

$f(n) = \Omega(n^{\log_4 3 + \varepsilon})$

Case 3, M.T. ✓

$\boxed{T(n) = \theta(n\log(n))}$

constant

$\frac{3}{4}c\log n - (\frac{3}{4}c\log 4) + \log n \le c\log n$

$\frac{3}{4}c + 1 \le c$

$1 \le \frac{1}{4}c$ ✓

4. The pseudocode for binary search is given below. The function takes an array A, a query item key, a starting index i, and an ending index j. (Assume that it is initially called with i = 1 and j = A.length.)

```
BINARY-SEARCH(A, key, i, j)
    if i > j
        return NIL        ] Base case O(1)
    mid=(i+j)/2  ——> Divide  O(1)
    if A[mid] == key
        return mid
    elif key < A[mid]
        return BINARY-SEARCH(A, key, i, mid – 1)     O(log n)
    elif key > A[mid]
        return BINARY-SEARCH(A, key, mid + 1, j)
```

Conquer

a) Specify which steps correspond to the divide and conquer steps in the divide and conquer framework. Note that binary search does not have a combine step.

b) Give the run time for each step in the divide and conquer framework.

c) Write the recurrence relation. $T(n) = \underset{a}{T}(n/\underset{b}{2}) + \underset{f(n)}{O(1)}$

d) Solve the recurrence relation.

e) Why can we describe the run time of binary search using O notation but not Θ notation?

Binary Search can be described using O notation because it provides an upper bound on the runtime of the algorithm. BS. doesn't provide a tight lower bound, which Θ needs.

d) $K \le \log_2(n)$ $\quad 1 \equiv n^{\log_2 1}$ $\quad$ Using the M.M., Binary Search falls under Case 2.

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

$1 = n^0$

$1 = 1$ ✓

Definitions

$\Theta(g(n))$ = { f(n) : there exists positive constants c_1, c_2, and n_0 such that

$0 \leq c_1\, g(n) \leq f(n) \leq c_2\, g(n)$ for all $n \geq n_0$ }

$O(g(n))$ = { f(n) : there exists positive constants c and n_0 such that

$0 \leq f(n) \leq c\, g(n)$ for all $n \geq n_0$ }

$\Omega(g(n))$ = { f(n) : there exist positive constants c and n_0 such that

$0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$ }

Master theorem: Let $a \leq 1$ and $b > 1$ be constants, let f(n) be an asymptotically positive function and let T(n) be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n)$$

Then T(n) has the following asymptotic bounds:

1. If $f(n) \in O(n^{\log b\ a-\varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) \in \Theta(n^{\log b\ a})$.

2. If $f(n) \in \Theta(n^{\log b\ a})$, then $T(n) \in \Theta(n^{\log b\ a} \log n)$.

3. If $f(n) \in \Omega(n^{\log b\ a+\varepsilon})$ for some constant $\varepsilon > 0$, and if $a\, f(n/b) \leq c\, f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) \in \Theta(f(n))$.