CENG218 HW #1

Sude Nur Gevrek 270201041
Osman Topal 230201048

Question 1 :

a. $f(n) = 3n^6 + \log_2 n^8$

$\log_2 n = \lg n$
$\log_{10} n = \log n$

I. Let's consider $g(n) = 3n^6$, $h(n) = \lg n^8$; so $f(n) = g(n) + h(n)$

II. Also assume that $g(n) = O(t(n))$, $h(n) = O(z(n))$

III. $g(n) \le c \cdot t(n)$ $\forall n \ge n_0$, $\exists c > 0$ $\qquad$ $h(n) \le d \cdot z(n)$ $\forall n \ge n_0$, $\exists d > 0$

$3n^6 \le c \cdot n^6$ $\qquad$ $\log_2 n^8 \le d \cdot z(n)$

$c = 4$ $n_0 = 1$ $\qquad$ $8 \lg n \le d \cdot \lg n$

$\boxed{t(n) = n^6}$ $\qquad$ $d = 9$ $n_0 = 1$

$\boxed{z(n) = \lg n}$

IV. Now combine these expressions by properties of big-O notation.

$f(n) = O(n^6) + O(\lg n)$

$f(n) = O(n^6 + \lg n) \xrightarrow[\text{property}]{\text{use the}} O(g_1 + g_2) = O(\max(g_1, g_2))$

$f(n) = O(\max(n^6, \lg n))$ → since n^6 grows faster than $\log n$ at some point.

$\boxed{f(n) \text{ is } O(n^6)}$ //

b. $f(n) = \log_2 n^3 + n \log_{10} n^2$

I. Let's consider $g(n) = \log_2 n^3$, $h(n) = n \cdot \log_{10} n^2$; so $f(n) = g(n) + h(n)$

II. Assume that $g(n) = O(t(n))$, $h(n) = O(z(n))$

III. $g(n) \le c \cdot t(n)$ $\forall n \ge n_0$, $\exists c > 0$ $\qquad$ $h(n) \le d \cdot z(n)$

$\log_2 n^3 \le c \cdot t(n)$ $\qquad$ $n \log_{10} n^2 \le d \cdot z(n)$

$3 \log_2 n \le c \cdot \log_2 n$ $\qquad$ $t(n) = \log_2 n = \lg n$ $\qquad$ $2n \log_{10} n \le d \cdot n \log_{10} n$

$c = 4$ $n_0 = 2$ $\qquad$ $d = 3$ $n_0 = 1$ ✓

$3 \log_2 n \le 4 \log_2 n$ $\qquad$ $z(n) = n \cdot \log_{10} n$

$3 \le 4$ ✓

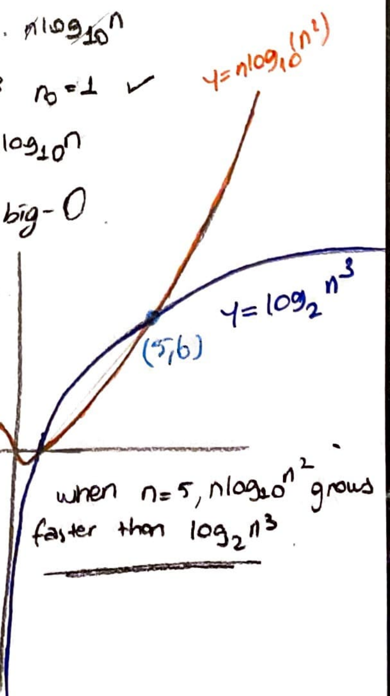
IV. Now combine these expressions by properties of big-O

$f(n) = \underbrace{O(\log_2 n)}_{g(n)} + \underbrace{O(n \log_{10} n)}_{h(n)}$ $\qquad$ * when it is $\log n$, bases are not important

$= O(\log_2 n + n \log_{10} n)$

$= O(\max(\log_2 n, n \log n))$

$= O(n \lg n)$



when $n = 5$, $n \log_{10} n^2$ grows faster than $\log_2 n^3$

Question 2.

a.

$x \leftarrow 1;$	$\longrightarrow c_1$	1	constant time required
while $x < 3*n$ do	$\longrightarrow c_2$	$\lfloor \frac{3n-1}{2} \rfloor + 1$	How many times the loop is executed? To find out that, we need to get the number of terms that is between x and $3n$
print(x);	$\longrightarrow c_3$	1	
$x \leftarrow x+2;$	$\longrightarrow c_4$	1	
end			

The time complexity denoted as $T(n)$

$$T(n) = (c_1 + c_2 + c_3 + c_4) \cdot 1 + c_2 \cdot \left\lfloor \frac{3n-1}{2} \right\rfloor$$

$$T(n) = O(1) + O(n)$$

* NOTE: big O of constant is $O(1)$

Formula of # of terms

$$\frac{\text{last term} - \text{first term}}{\text{accurement}} + 1 = \left\lfloor \frac{3n-1}{2} \right\rfloor + 1$$

$x \leftarrow 1$

$x \leftarrow x+2$

$$T(n) = O(1+n)$$

$$= O(n)$$

---

b.

$x \leftarrow 0;$	$\longrightarrow c_1$	1	
while $x < n$ do	$\longrightarrow c_2$	$\lfloor \frac{n}{3} \rfloor + 1$	times executed
print(x);	$\longrightarrow c_3$	1	
$x \leftarrow x+3;$	$\longrightarrow c_4$	1	
$y \leftarrow -1;$	$\longrightarrow c_5$	1	
while $y < m$ do	$\longrightarrow c_6$	$\lfloor m+1 \rfloor + 1$	times executed
print(y);	$\longrightarrow c_7$	1	
$y \leftarrow y+1;$	$\longrightarrow c_8$	1	
end			
end			

used # of terms formula $\left\lfloor \frac{n-0}{3} \right\rfloor + 1$ $x=0$

these are 1 since they're asked for on the question explanation.

used # of terms formula $\left\lfloor \frac{m+1}{1} \right\rfloor + 1$ $m-(-1)$ $y \leftarrow -1$ increment y by 1.

$$T(n) = 1 \cdot \sum_{i=1}^{8} c_i + c_2 \cdot \left\lfloor \frac{n}{3} \right\rfloor \cdot c_6 \lfloor m+1 \rfloor$$

constant

they are multiplied since there exists two while loop and one of the while loop (2^{nd}) is executed once more whenever outer while loop's condition is true.

$$T(n) = O(1) + O(n) \cdot O(m)$$

$$T(n) = O(1 + nm) = O(\max(1, nm))$$

$$= O(nm)$$

Because constants don't grow in time but nm is changing.

c. $x \leftarrow 3$ $\longrightarrow c_1$ 1

while $x+1 < n*n - 2*n$ do $\longrightarrow c_2$

print(x); $\longrightarrow c_3$ 1

$x \leftarrow 2*x$; $\longrightarrow c_4$ 1

end

while $x+1 < n^2 - 2n \longrightarrow \overbrace{4, 7, 13, 25, 49, \ldots n^2-2n}^{x+1}$

$x \leftarrow 2*x \longrightarrow \underbrace{6, 12, 24, 48 \ldots n^2-2n-1}_{x}$

if we found the number of terms, we can find out the time complexity of while loop. We realized that;

$6, 12, 24, 48, \ldots n^2-2n-1$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

$3.2^1 \; 3.2^2 \; 3.2^3 \; 3.2^4 \ldots 3.2^{n^2-2n-1}$

The multiplier 3 is common so we can ignore it

As you see the number of terms depends on $\log_2(n^2-2n-1)$

so its asymptotic upper bound is $O(\lg n)$

So

$$T(n) = \underbrace{1 . \sum_{i=1}^{3} c_i}_{\text{constant}} + c_2 . \lg n)$$

* ignore constants *

$= O(1) + O(\lg n)$

$= O(1 + \lg)$

$= O(\max(1, \lg))$

$= O(\lg n)$

since $\lg n$ grows faster than a constant (does not grow).

Question 3

required time of the function

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \text{ is minimal size} \\ aT(n/b) + D(n) + C(n) & \text{if } n > 1 \end{cases}$$

This is the general recurrence relationship for recursive functions.
a is the number of subproblems, $1/b$ is the size of these subproblems.
$D(n)$ is the required time to divide problem.
$C(n)$ is the required time to combine the solutions.

So the recurrence relation for our question is;

a)

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \text{ is minimal size } (n=1) \\ 4T(\lfloor n/2 \rfloor) + \underbrace{D(n) + C(n)} & \text{if } n > 1 \end{cases}$$

$\Theta(n)$ is given in the question by saying that " This function needs $\Theta(n)$ time in order to determine these subarrays."

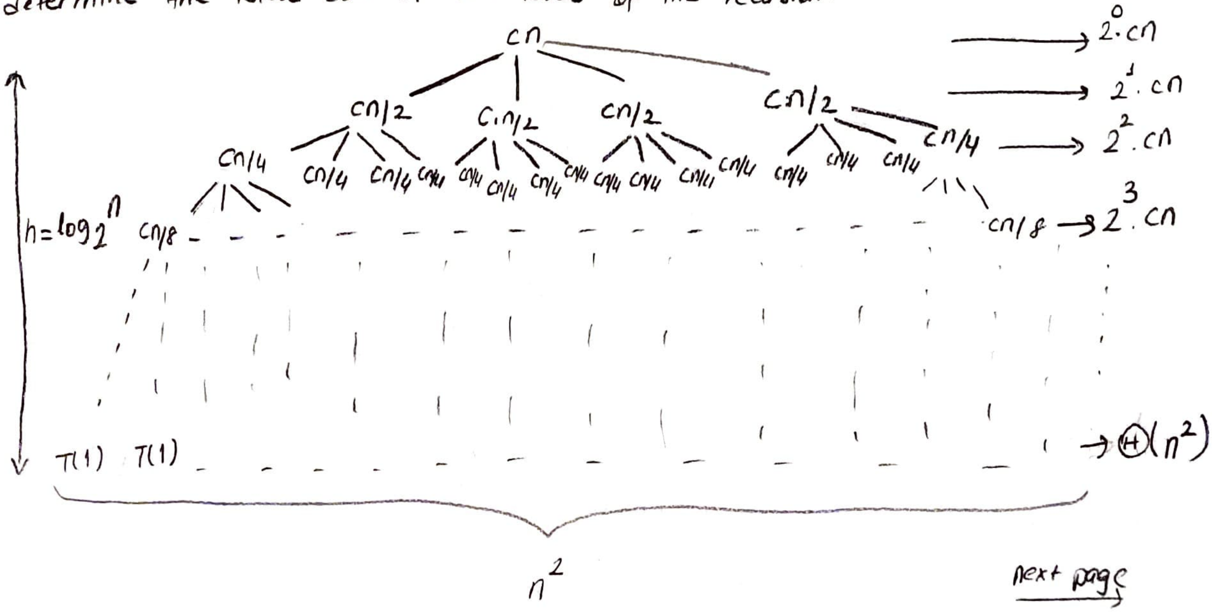
$$T(n) = \begin{cases} \Theta(1) & \text{if } n \text{ is minimal size } (n=1) \\ 4T(\lfloor n/2 \rfloor) + \Theta(n) & \text{if } n > 1 \end{cases}$$

b) To solve $T(n)$ by recursion tree we can assume that $c = \Theta(1)$ since it requires constant time. So,

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 4T(\lfloor n/2 \rfloor) + \underbrace{c.n}_{\Theta(1.n) = \underbrace{\Theta(1)}_{c}.\underbrace{\Theta(n)}_{n}} & \text{if } n > 1 \end{cases}$$

where $c > 0$ constant

In a recursion tree each node represents the cost of a single subproblem somewhere in the set of function invocations. We sum the costs within each level of the tree to obtain a set of per-level costs, and then we sum all the per-level costs to determine the total cost of all levels of the recursion.



next page

Since subproblem sizes decrease by a factor of 2 each time we go down one level, and then we eventually reach a boundary condition

The subproblem size for a node at depth i is $n/2^i$. So, the subproblem size hits $n=1$ when $n/2^i = 1$ or $i = \log_2 n$. So the tree has $\log_2 n + 1$ levels $(0, 1, 2, \ldots \log_2 n)$

Now, find the cost of each level. Each level has four times more nodes than the level before, so the # of nodes at depth i is 4^i. So, each level cost $c.\, n/2^i$. $(0, 1, 2, \ldots \log_2 n - 1)$

Total cost over all nodes at depth i for $i = 0, 1, 2, \ldots, \log_2 n - 1$ is $4^i \cdot c.(n/2^i)$

$= \dfrac{4^i}{2^i} c.n = 2^i c.n$

The bottom level at depth $\log_2 n$ has $4^{\log_2 n} = n^{\log_2 4} = n^2$ nodes and each contributes $T(1)$ cost. So the total cost of $n^2 \underbrace{T(1)}_{\text{assume it is a constant}}$ is $\Theta(n^2)$

$$T(n) = \underbrace{cn + 2^1 cn + 2^2 cn + \cdots 2^{\log_2 n - 1}}_{\text{to find that we need to use summation expression}} + \Theta(n^2)$$

$= cn \sum_{i=0}^{\log_2 n - 1} 2^i + \Theta(n^2)$ $\rightarrow$ we have a formula for series $\dfrac{1-x^n}{1-x} = \sum_{i=0}^{n-1} x^i$

$$= cn.\left(\frac{1-2^{\log_2 n}}{1-2}\right) + \Theta(n^2)$$

$$= cn.\ \frac{1-n^{\log_2 2}}{-1} + \Theta(n^2)$$

$$= cn.\ \frac{1-n}{-1} + \Theta(n^2) = cn(n-1) + \Theta(n^2)$$ since "c" is constant ignore it

$$= cn^2 - cn + \Theta(n^2)$$

$$= \Theta(n^2)$$

c) Substitution method requires two steps:

I. Guess the form of the solution

II. Use mathematical induction to find the constants and show the solution works.

SOLUTION

$$T(n) = 4T(\lfloor n/2 \rfloor) + n$$

Guess $O(n^2)$

$T(n) \leq c.n^2 \quad \exists c > 0, \ n_0 \leq n$

$\leq 4c.(n/2)^2 + n$

$= cn^2 + n$

$\leq cn^2$ true for no choice, $c > 0$. we need smt. else

① So we should strengthen the inductive hypothesis by subtracting a low-order term.

New inductive hypothesis:

$T(n) \leq c_1.n^2 - c_2 n$

$\leq 4(c_1(n/2)^2 - c_2(n/2)) + n$

$= c_1 n^2 - 2c_2 n + n$

$= \underbrace{c_1 n^2 - c_2 n}_{\text{desired}} - \underbrace{(c_2 n - n)}_{\text{residual}}$ residual ≥ 0 as long as $c_2 \geq 1$.

$\leq c_1 n^2 - c_2 n$

Choose c_1 big enough to handle the base case

Base: $T(1) \leq c_1 - c_2$, for any $c_1 > c_2$ can be chosen.