Data Structures
Assignment-5

1
a. In the program i is initialized to n. above first loop and end of the loop i is iterated by $i = i/2$ and the condition is $i > 1$

so, $i = n$ $i > 1$
$i = n/2$
$i = n/4$
⋮

For a certain value of i, i will be less than or equal to 1

so $i \leq 1$
$n/2^k \leq 1$
$n \leq 2^k$
$k \geq \log_2 n$

so, first while loop will run $\log_2 n$ times so order of first while loop is $O(\log n)$
Similarly, second loop will run $\log_2 n$ times, so order of second while loop is $O(\log n)$
Now, for third loop, as k is initialized to zero & k is iterated by $k = k+2$ and the condition is $k < n$
so, order of third while loop is $O(n)$
so final order of whole program is $(\log n)(\log n)(n/2)$
So, order is $O(n(\log n)^2)$.

1b. In the program i & s are initialized to 1 & the condition is $s <= n$ & i is iterated by $i = i+1$ and s is iterated by $s = s+i$
so, the loop will run for the following values of s.
1, 3, 6, 10, 15 —— n → r terms

So. $\frac{r^2+r}{2} = n$

where $\frac{r^2+r}{2}$ represent r^{th} term

$r^2+r = 2n$

$r^2+r-2n=0$

$r = \frac{-1 \pm \sqrt{1+8n}}{2}$

$r = \frac{-1+\sqrt{1+8n}}{2}$

So, complexity of program will be $(1+3+6 \ldots n)c_1$

where $c_1 \rightarrow$ Time taken by statements inside loop.

Sum of series is $\frac{r(r+1)(r+2)}{6}$

$$\frac{\left(\frac{\sqrt{1+8n}-1}{2}\right)\left(\frac{\sqrt{1+8n}-1+2}{2}\right)\left(\frac{\sqrt{1+8n}-1+4}{2}\right)}{6}$$

$$= \frac{\left(\frac{\cancel{1+8n}-1}{4}\right)\left(\frac{\sqrt{1+8n}+3}{2}\right)}{6} = n/_6\left(\sqrt{1+8n}+3\right)$$

$$= \sqrt{n^2+8n^3}+n/_2 \simeq n^{3/2}$$

So, order of the program is $O(n^{3/2})$.

1c. In the program i is initialized to zero & condition is $i^2 <= n$ & iterated as i++.

So, for some value of i i.e $i=k$, $i^2=n$

$\Rightarrow k^2=n$

$k=\sqrt{n}$

So, loop will run until i value will be $\sqrt{n}$ as i is initialized to zero.

Loop will run $\sqrt{n}$++ times.

So, order of program is $O(\sqrt{n})$ i.e $O(n^{1/2})$.

(d) As the break statement is included, the inner loop won't run as loop as the break statement is included.
So, the message in the inner loop prints (n+1) times as the condition is i=0 & i<=n, so the outer loop will run (n+1) times

∴ Complexity of whole program is $O(n)$.

(e)

for i=1 inner loop runs n times

for i=2 " " " $n/2$ times (for j=1, j=3, j=5, 7, 9)

for i=3 " " " $n/3$ times (for j=1, j=4, j=7 ...)

$n/4$ " "

$n/5$ " "

So, total complexity is $(n+n/2+n/3+n/4+n/5 - - -)\, c_1$

c_1 is time taken by statement inside inner loop.

$n(1+1/2+1/3+ - - +1/n)$

Sum of series will be $\log n$

So order is $O(n\log n)$

(f)

$T(n) = 8T(n/2)+n^3$

$T(n) = 8\{8T(n/4)+(n/2)^3\}+n^3$

$T(n) = (8)^2 T(n/4)+n^3+n^3$

$T(n) = (8)^2\{8T(n/8)+(n/4)^3\}+n^3+n^3$

$T(n) = (8)^3 T(n/2^3)+n^3+n^3+n^3$

$T(n) = 8^k + T(n/2^k)+n^3+n^3+n^3+ \dots$ k times

$n/2^k = 1 \Rightarrow n=2^k \Rightarrow \boxed{k=\log_2 n}$

Order will be $\boxed{O(n^3\log_2 n)}$

Scanned by CamScanner

Ans 3

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

$$T(n) = \begin{cases} \text{constant} & n <= 1 \\ aT(n-b) + f(n) & n > 1 \end{cases}$$

$a = 1 \quad b = 1 \;;\; k = 1$ (order of $f(n)$)

So, order is $\underline{O(n^{k+1}) = O(n^2)}$

(b) $T(n) = T(n/3) + n/2$

Comparing with $T(n) = a(n/b) + \Theta(n^k \log^p n)$

Here $a = 1 \;;\; b = 3 \;;\; k = 1 \;;\; p = 0$

and $a < b^k$ and $p = 0$

$T(n) = \Theta(n^1 \log^0 n)$

$T(n) = O(n)$

order of $\boxed{O(n)}$

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

Comparing with $T(n) = aT(n/b) + O(n^k \log^p n)$

Here $a = 2 \;;\; b = 4 \;;\; k = 1/2 \;;\; p = 0$

& $a = b^k$ and $p > -1$

Order of $\underline{T(n) = \Theta(n^2 \log n)}$

(d) $T(n) = T(n-2) + n^3$

Comparing with $T(n) = aT(n-b) + f(n)$ if $n > 1$

& $T(n) = c$ if $n <= 1$

Here $a = 1, b = 2 \;;\; k = 3$

Order of $\underline{T(n) = O(n^{k+1}) = O(n^4)}$

(e) $T(n) = 7T(n/2) + n^2 \log n$

comparing with $T(n) = aT(n/b) + \Theta(n^k \log^p n)$

Here $a = 7, \; b = 2, \; k = 2$ & $p = 1$

Here $a > b^k$

Order of $T(n) = \Theta(n^{\log_b a}) = \Theta(n^3)$

(f) = $T(n) = T\left(\frac{7n}{10}\right) + n$

Comparing with $aT(n/b) + \theta(n^k \log^p n)$

$a = 1$; $b = 10/7$ $k = 1$ $p = 0$

So, order of $T(n) = \boxed{\theta(n)} = \theta(n^k \log^p n)$

3.

$T(n) = 2T(n^{1/2}) + \log n$

$T(n) = 2\left[2T(n^{1/4}) + \log n^{1/2}\right] + \log n$

$T(n) = 4T(n^{1/4}) + \log n + \log n$

$T(n) = 4\left[2T(n^{1/8}) + \log (n)^{1/4}\right] + 2\log n$

$T(n) = 8T(n)^{1/8} + 3\log n$

at some k. $(n)^{1/2^k} = 2$

$\log_2 n = 2^k$

$\boxed{k = \log_2(\log_2 n)}$

$T(n) = 2^k T(n^{1/2^k}) + k\log n$

$T(n) = 2^k T(2) + (\log_2 n)(\log_2 n)$

So, complexity will be $\boxed{O(\log n \cdot \log(\log n))}$

4a Given,

$T(n) = T(n/2) + n\log n$

$= T(n/4) + n\log n + n/2 \log n/2$

$= T(n/8) + n\log n + n/2 \log n/2 + n/4 \log n/4$

we can write,

$T(n) = T(1) + (n\log n + n/2 \log n/2 + n/4 \log n/4 + \ldots + \log n \text{ terms})$

we can assume that,

it takes k terms to reach $T(1)$

so, $n/2^k = 1$

$n = 2^k$

$\boxed{k = \log n}$

$$T(n) = T(1) + n[\log n + \log n^{1/2} + \log n^{1/4} + \dots\dots]$$

$$T(n) = T(1) + n[\log n \times n^{1/2} \times n^{1/4} \dots \dots \log n \text{ terms}]$$

$$T(n) = T(1) + n[\log n^{1+1/2+1/4} \dots]$$

$$T(n) = T(1) + n\left[\log n^{\frac{1(1-(1/2)^{\log n})}{(1-1/2)}}\right]$$

$$T(n) = T(1) + n\left[\log n^{\left(\frac{n-1}{2n}\right)}\right]$$

$$T(n) = T(1) + n(\log n)$$

so, order of program is $\boxed{O(n \log n)}$

(4b)

$$T(n) = T(n-1) + \log n$$

$$\cancel{T(1)} - T(0) = \log 1$$

$$\cancel{T(2)} - \cancel{T(1)} = \log 2$$

$$\cancel{T(3)} - \cancel{T(2)} = \log 3$$

$$\vdots \qquad \vdots$$

$$\cancel{T(n)} - \cancel{T(n-1)} = \log n$$

$$T(n) = T(0) + \log 1 + \log 2 + \log 3 + \dots \log n$$

$$T(n) = C + \log(1 \times 2 \times 3 \dots \times n)$$

$$\boxed{T(n) = C + \log(n!)}$$

Complexity will be $\log(n!)$

(4c) $T(n) = T(n-2) + 1/\log n$

$$T(n) = T(n-4) + \frac{1}{\log(n-2)} + \frac{1}{\log n}$$

$$T(n) = T(n-6) + \frac{1}{\log(n-4)} + \frac{1}{\log(n-2)} + \frac{1}{\log n}$$

$$T(n) = T(0) + \underbrace{\frac{1}{\log 2} + \frac{1}{\log 4} + \frac{1}{\log 6} + \dots + \frac{1}{\log n}}_{n/2 \text{ terms}}$$

$n/2 \times$ least value $<=$ sum of series $<= n/2 \times$ max value

$n/2 \times 1/\log n <=$ sum of series $<= n/2 \times \frac{1}{\log 2}$

so, time complexity is order of $\underline{O(n)}$.

(4d) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

$T(n) = T(n/4) + T(n/8) + T(n/16) + T(n/8) + T(n/16) + T(n/32) + T(n/16) +$

$T(n/32) + T(n/64) + n + n/2 + n/4 + n/8$

⋮

Similarly,

$T(n) = T(0) + n + \frac{7n}{8} + (7/8)^2 n + (7/8)^3 n + \ldots + (7/8)^{\log n} n$

$T(n) = c + n\underbrace{[1 + (7/8) + (7/8)^2 + \ldots + (7/8)^k]}_{\text{Decreasing G.P}}$

$k = \log n$

So, complexity will be $\boxed{O(n)}$

4e

$T(n) = T(n-a) + T(a) + cn$

$T(n) = T(n-2a) + T(a) + c(n-a) + T(a) + cn$

$T(n) = T(n-3a) + T(a) + c(n-2a) + T(a) + c(n-a) + T(a) + cn$

⋮

$T(n)$

at some, $k \Rightarrow n - ka = 0$

$n = ka$

$k = n/a$

$T(n) = (n/a)\,T(a) + (c/a)\,n^2 - a\,(1 + 2 + 3 + \ldots (n/a)^{th}\text{ term})$

$= (n/a)\,T(a) + (c/a)\,n^2 - a(n/a)\left(\frac{n+a}{2a}\right)$

Order of $T(n)$ will be $\boxed{O(n^2)}$

4f. $T(n) = T(n-1) + T(n/2) + n$

Assume $A(n) = T(n/2) + n \Rightarrow T(n) = T(n-1) + A(n)$

$\Rightarrow T(n) = T(n-2) + A(n-1) + A(n)$

$\vdots$

$T(n) = A(1) + A(2) + \ldots + A(n) \Rightarrow T(n) = \sum_{i=1}^{n} T(i/2) + \sum_{i=1}^{n} i$

Assuming $n/2$ is integer division, $T(n/2) = T(\frac{n+1}{2})$

So, $T(n) = 2 * \sum_{i=1}^{n/2} T(n) + \frac{n*(n+1)}{2}$

Since, $T(n) \leq T(n/2)$ for every $m \leq n/2 \Rightarrow T(n) \leq n * T(n/2)$

Since $T(n/2) \geq n/2$

$T(n) \leq n * T(n/2) + (n+1) T(n/2) \leq 2(n+1) T(n/2)$

So, $n = 2^k$

$U(k) = 2 * 2^{k+1} \cdot U(k-1)$

Order of $T(n) = \boxed{O(2^{(\log n)^2})}$

5Ans. Given,

$f(n) = 3n^{\sqrt{n}}$ & $g(n) = 2^{\sqrt{n}\log_2 n}$

consider $g(n) = 2^{\sqrt{n}\log_2 n}$

$g(n) = 2^{\log n(\sqrt{n})}$ $[\because a\log n = \log n^a]$

$g(n) = n^{\sqrt{n}}$

So, $\boxed{f(n) = O(g(n))}$

7A. Complexity of Binary Search:

In binary search a sorted array is searched by repeated dividing the search interval in half. Begin with interval covering the whole array. If the value of search key is less than the item in the middle of interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or interval is empty.

Sorted Array: 1, 3, 6, 9, 14, 17, 25, 36, 57, 79, 98

Now search 25:

Scanned by CamScanner

Iteration 1:

We will consider the middle element i.e 17,

Since 25 is greater than 17, so, we divide the array into two halves & consider the sub-array element 17.

Now this sub array with the element 17 will be taken into next iteration.

Iteration 2: 25, 36, 51, 79, 98.

→ Middle Element is 51.

→ Since, 25<51, so we divide the array into two halves and consider the sub-array before 51.

→ It is taken into next iteration

Similarly it goes till it find the number (If no. is present in the sorted array) otherwise it will terminate.

So, for 1st iteration length of array is n

2nd " " $n/2$

3rd " " $n/2^2$

kth " " $n/2^k$

After k division the length of array becomes 1

$$n/2^k = 1 \Rightarrow n = 2^k$$

$$\boxed{k = \log_2 n}$$

∴ Complexity will be order of $O(\log_2 n)$

(7.5) Complexity of Bubbled Sort:

In bubble sort each element is compared with the next element & it is sorted as required.

main condition in bubble sort is

```
for (i=n-1; i>=0; i--)
{ for (j=0; j<i; j++)
```

{
    statements
    }

}

so best case is when elements are in sorted order

worst case - elements in reverse order

Average case - Random order

$i = n-1$ ——— $n-1$ times

$i = n-2$ ——— $n-2$ times

⋮

$i = 0$ ——— 1 time

so complexity will be $(1+2+3+ \dots + (n-2)+(n-1))$

$$\frac{n(n-1)}{2} = O(n^2)$$

(7c) Complexity of fibonacci number:

If $(n<=1)$
  return n;
else
  return (fib(n-1) + fib(n-2));

so, $T(n) = T(n-1) + T(n-2) + C$

$T(n-2) \sim T(n-1)$

$T(n) = 2T(n-1) + c$

$T(n) = 2[2T(n-2)+c] = 4T(n-2) + 2c + c = 4T(n-2) + 3C$

$T(n) = 8T(n-3) + 7C$

$$\boxed{T(n) = 2^k T(n-k) + (2^k - 1)\, C}$$

for some k value $n-k=0 \Rightarrow n=k$

$T(n) = 2^n T(0) + (2^n - 1) c$

$T(n) = 2^n (1+c) - c$

Complexity is $\boxed{O(2^n)}$

(8a) Run time analysis:

Run time analysis is a theoretical classification that estimates and anticipates the increase in running time of an alogorithm as its input size increases.

(8b) Rate of Growth

The rate at which running time increases as a function of input is called rate of growth

(8c) Best case: Best case is the function which performs the minimum number of steps on input data of n elements

Worst case: It is the function which performs the maximum no. of steps on input data of sizen.

Averagen: It is the function which performs an average no. of steps on input of n elements.

(8d) Big O notations: Big-oh is an upper asymptotic bound for an algorithm, useful for elevating best case performance

mathematically $f(n) = O g(n)$

If $f(n) \leq g(n) \; \forall n \in n_0$

Big Omega: Big Omega is a lower asymptotic bound for an algorithm, useful for elevating best case performance.

mathematically, $f(n) = \Omega g(n)$

If $f(n) \geq g(n) \quad \forall n \geq n_0$

Big theta: Big theta is most accurate asymptotic bound for an algorithm

Mathematically, $f(n) = \Theta g(n)$

If $f(n) = O g(n)$ & $f(n) = \Omega g(n)$

Scanned by CamScanner

6A. **Proof of Master's Theorem:**

Start with recurrence

$$T(n) = aT(n/b) + cn^k$$

$$= a\left[aT(n/b^2) + c(n/b)^k\right] + cn^k$$

$$= a^2T(n/b^2) + cn^k\left[1 + a/b^k\right]$$

$$\vdots$$

$$= a^s T(n/b^s) + cn^k\left[(a/b^k)^s + (a/b^k)^{s-1} + \dots + a/b^k + 1\right]$$

we stop the expanding when we reach the base.

when $n/b^s = P_0$ This occurs after $s = \log_b(n/B)$

$s = \log_b n + \text{constant}$ iterations.

Notice that the expansion is split into two terms. The asymptotic form of $T(n)$ is just a competition between two terms to see which one dominates.

The second term has a geometric sum: using the formula for a geometric sum gives the following equation of $T(n)$

$$T(n) = a^s q_0 + cn^k\left[\frac{1-(a/b^k)^{s+1}}{1-(a/b^k)}\right] \quad (1)$$

In case where $a < b^k \Rightarrow a/b^k < 1$

Here $a/b^k < 1$ and as n grows large the sum of the above geometric is dominated by the constant terms.

$$\frac{1}{1-a/b^k} = \Theta(1)$$

So, $T(n) = \Theta(a^s) + \Theta(n^k)$ using our expression as for s:

$$a^s = \Theta(a^{\log_b n}) \cdot \Theta(n^{\log_b a}) = \Theta(n^k)$$

Since $a < b^k$ ie: $\log_b a < k$.

$$\therefore T(n) = \Theta(n^k) + \Theta(n^k) = \Theta(n^k)$$

In case where $a > b^k \Rightarrow a/b^k > 1$

Geometric sum equ(1) is dominated by

$$\frac{(a/b^k)^{s+1}}{a/b^k - 1} = \theta\left((a/b^k)^s\right)$$

2nd term of T(n) is

$$cn^k \theta\left((a/b^k)^{\log_b n}\right) = cn^k \theta\left(\frac{n^{\log_b a}}{n^k}\right)$$

$$= \theta(n^{\log_b a})$$

$$\Rightarrow a^s = \theta(n^{\log_b a})$$

$$T(n) = \theta(n^{\log_b a})$$

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

In case where $a = b^k$

In equation (1) sum is 1

There are $s+1$ terms so 2nd term of T(n) becomes

$$cn^k (s+1) = \theta(n^k \log_b n)$$

$$= \theta(n^k \log n)$$

1st term of $T(n) = \theta(n^{\log_b a}) = \theta(n^k)$

so, the 2nd term dominates and

$$T(n) = \theta(n^k \log n)$$

Qualitatively, if $a > b^k$ the better recursive is the number of recursive calls we have to make. Otherwise its extra work during the call that dominates the run time.

$$T(n) = aq_0 + cn^k \left[\frac{1 - (a/b^k)^{s+1}}{1 - (a/b^k)}\right] \text{ —— (1)}$$

Here T(n) is denominated by 2nd part which contains the terms of a/b^k. Here the ratio of a and b^k plays an important rule as the eq (1) is a geometric sum.

It $r = a/b^k$

Scanned by CamScanner

Then $a/b^k > 1$ increasing G.P
$a/b^k < 1$ decreasing G.P
$a/b^k = 1$ constant G.P

so, by this we can say that a/b^k plays a major role in deriving master theorem. so, the asymptotic nature of $T(n)$ depends on the value of a/b^k. the behaviour of the block diagram also depends on a/b^k.